Proceedings of the Fifth Dutch-Belgian Information Retrieval Workshop

10-11 January, 2005
Utrecht University
Utrecht, the Netherlands

Proceedings of the Fifth Dutch-Belgian Information Retrieval Workshop

DIR'5

10-11 January, 2005
Utrecht University
Utrecht, the Netherlands

editor:
Roelof van Zwol
Utrecht University
Utrecht, the Netherlands

Sponsors

SIKS: Dutch Research School for Information and Knowledge Systems
Web-site: http://www.siks.nl/

WGI: Dutch Working Community on Information Sciences
Web-site: http://www.ins.cwi.nl/projects/informatiewetenschap/

Publication details

Title:
Proceedings of the Fifth Dutch-Belgian Workshop on Information Retrieval
Edited by:
Roelof van Zwol
Published by:
Center for Content and Knowledge Engineering
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
the Netherlands
Printed by:
Drukstore, Leusden, the Netherlands
Cover design by:
Roelof van Zwol
ISBN:
90-393-0031-3

Subject headings: information retrieval

Preface

Welcome to the fifth Dutch-Belgian workshop on Information Retrieval. The primary aim of the workshop is to provide an international meeting place where researchers, who are working in the domain of information retrieval and related disciplines, can exchange information and present new research developments. This year, there is a special interest in contributions focusing on multimedia- and structured document retrieval.

Mounia Lalmas from Queen Mary University of London has been invited to open the workshop with a presentation on 'XML retrieval and evaluation: where are we?'. As one of the project leaders of the INitiative for the Evaluation of XML Retrieval (INEX), she is the appointed person to give a state-of the-art overview on the developments in structured document retrieval.

All articles accepted for this workshop have been reviewed by two members of the Program Committee. This year the following researchers took place in the Program Committee: Anne Diekema, Djoerd Hiemstra, Theo Huibers, Franciska de Jong, Marie-Francine Moens, Arjen de Vries, and Roelof van Zwol. The articles cover a wide variety of topics within the information retrieval domain, and reflect the different views and ongoing developments in the field. We thank the Program Committee members for their high quality reviews.

Furthermore, we are grateful for the sponsoring supplied by WGI, the Dutch Working Community on Information Sciences, and SIKS, the Dutch Research School for Information and Knowledge Systems. The Institute for Information and Computing Sciences (Utrecht University) provided us with the venue for this workshop, for which we are also grateful.

The workshop programme, included in the proceedings, allows for plenty of discussion about the presentations and arising research questions. We encourage the workshop participants to make their opinions known, and to make this an interactive workshop. Last but not least, a number of articles have been selected for publication in a special issue of the Journal on Digital Information Management (JDIM).

We hope that you will have a fruitful and enjoyable time at the workshop!

Roelof van Zwol (roelof@cs.uu.nl),
Arjen de Vries (arjen@cwi.nl), and
Djoerd Hiemstra (hiemstra@cs.utwente.nl)

Utrecht, January 4, 2005

DIR'5 Program Committee

Anne Diekema (Syracuse University)
Djoerd Hiemstra (University of Twente, co-chair)
Theo Huibers (University of Twente / KPMG)
Franciska de Jong (University of Twente)
Marie-Francine Moens (K.U. Leuven)
Arjen de Vries (CWI Amsterdam, co-chair)
Roelof van Zwol (Utrecht University, co-chair)

Workshop Programme

	Monday January 10, 2005 - van Unnik building - Room: 001
Time slot	***Activity***
12:00 - 13:15	Registration, coffee, and sandwiches
13:15 - 14:15	Opening and keynote session:
	• Mounia Lalmas. *XML retrieval and evaluation: where are we*?
14:15 - 14:30	Coffee break
14:30 - 15:30	Article session:
	• David Ahn, Sisay Fissaha Adafre, Maarten de Rijke. *Extracting Temporal Information from Open Domain Text: A Comparative Exploration.*
	• Sun Yang, Soon Cheol Park. *Generation of Non-redundant Summary Based on Sum of Similarity and Semantic Analysis.*
15:30 - 15:45	Tea break
15:45 - 16:45	Article session:
	• Rachel Tsz-Wai Lo, Ben He, Iadh Ounis. *Automatically Building a Stopword List for an Information Retrieval System.*
	• Loes Braun, Floris Wiesman, Jaap van den Herik. *Towards Automatic Formulation of a Physician's Information Needs.*
16:45 - 17:40	Reception
	Tuesday January 11, 2005 - Went building - Room: Groen (Green)
Time slot	***Activity***
09:00 - 9:45	Registration, and coffee
9:45 - 10:45	Article session:
	• Borkur Sigurbjornsson, Jaap Kamps, Maarten de Rijke. *Building a Cross-Lingual Web Retrieval Collection.*
	• Maria Biryukov, Roxana Angheluta, Marie-Francine Moens. *Multidocument Question Answering Text Summarization using Topic Signatures.*
10:45 - 11:00	Coffee break
11:00 - 12:00	Article session:
	• Matthijs Bomhoff, Theo Huibers, Paul van der Vet. *User Intentions in Information Retrieval.*
	• Dolf Trieschnigg, Wessel Kraaij. *Hierarchical Topic Detection in Large Digital News Archives.*
12:00 - 13:15	Lunch
13:15 - 14:15	Article session:
	• Rainer Typke, Marc den Hoed, Justin de Nooijer, Frans Wiering, Remco Veltkamp. *A Ground Truth For Half A Million Musical Incipits.*
	• Sau Kwan Chan, Ben He, Iadh Ounis. *An In-depth Study of the Automatic Detection and Correction of Spelling Mistakes.*
14:15 - 14:30	Coffee break
14:30 - 15:30	Article session:
	• Ameelie Imafouo, Michel Beigbeder. *An Experimental Methodology to Study Collections Size Impact on Retrieval Effectiveness.*
	• Gilad Mishne, Maarten de Rijke, Valentin Jijkoun. *Using a Reference Corpus as a User Model for Focused Information Retrieval.*
15:30 - 15:35	Closing

Table of Contents

XML Retrieval and Evaluation: Where are we?

Mounia Lalmas

Department of Computer Science
Queen Mary University of London
Mile End Road
London, E1 4NS
http://qmir.dcs.qmw.ac.uk
E-mail: mounia@dcs.qmul.ac.uk

Abstract

Todays content is increasingly a mixture of text, multimedia, and metadata. One way to format this mixed content is according to the adopted W3C standard for information repositories, the so-called eXtensible Markup Language (XML). The increasing use of XML in scientific data repositories, Digital Libraries and on the Web, has brought about an explosion in the development of XML tools, and in particular systems to store and access XML content. Whereas many of todays access tools still treat documents as single large (text) blocks, XML offers the opportunity to exploit the internal structure of documents in order to allow for more precise access thus providing more specific answers. Providing effective access to XML-based content is therefore a key issue.

Providing effective access to XML-based content is what XML retrieval research is about. XML retrieval systems aim to exploit the logical structure of documents, which is explicitly represented by the XML markup, to retrieve document components (the so-called XML elements) instead of whole documents in response to a user's query. Implementing this more focused retrieval paradigm means that an XML retrieval system needs not only to find relevant information in the XML documents, but also determine the appropriate level of component granularity to return to the user. It is this goal that makes the development AND the evaluation of XML retrieval approaches challenging.

INEX, the Evaluation Initiative for XML Retrieval, is an initiative currently supported by DELOS; a network of excellence in digital libraries (http://www.delos.info/). INEX was set up at the beginning of 2002 with the aim to establish infrastructures, XML test-suites and appropriate measurements for evaluating XML retrieval approaches. INEX is responsible for a range of evaluation activities in the field of XML information access. INEX 2004 had five tasks, where each task was designed to test particular aspects of XML retrieval: ad-hoc retrieval, interactivity, natural language querying, relevance feedback, and heterogeneous content. A large number of institutions (The following institutions contributed to the organization of INEX 2004: U Duisburg-Essen (D), QMUL (UK), U Amsterdam (NL), LIP6 (F), U Otago (NZ), CWI (NL), QUT (AUS), IBM (IL), LIS (DK), U Minnesota-Duluth (US)) are strongly involved in the organization of these activities, by contributing to the evaluation methodologies and providing evaluation software and tools. After three years of INEX, we are still needing further research in order to arrive at the correct methodology for evaluating XML retrieval, in particular with respect to returning the appropriate level of component granularity.

This talk will discuss the issues involved in returning the appropriate level of component granularity to the user in the context of INEX, and before INEX, as encountered in some of my previous work. My aim here is to both present the evaluation methodology followed by INEX, and to obtain feedback from the attendees as the issues we are dealing with concern many other areas.

For more details about INEX 2004, see http://inex.is.informatik.uni-duisburg.de:2004/.

Extracting Temporal Information from Open Domain Text: A Comparative Exploration

David Ahn Sisay Fissaha Adafre Maarten de Rijke

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
ahn,sfissaha,mdr@science.uva.nl

ABSTRACT

The utility of data-driven techniques in the end-to-end problem of temporal information extraction is unclear. *Recognition* of temporal expressions yields readily to machine learning, but *normalization* seems to call for a rule-based approach. We explore two aspects of the (potential) utility of data-driven methods in the temporal information extraction task. First, we look at whether improving recognition beyond the rule base used by a normalizer has an effect on normalization performance, comparing normalizer performance when fed by several recognition systems. We also perform an error analysis of our normalizer's performance to uncover aspects of the normalization task that might be amenable to data-driven techniques.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Linguistic processing

General Terms

Temporal information extraction

1. INTRODUCTION

Current Information Retrieval (IR) systems allow us to locate documents that might contain pertinent information, but most of them leave it to the user to extract useful information from a ranked list. This leaves the (often unwilling) user with a relatively large amount of text to consume. There is a need for tools that reduce the amount of text that has to be read to obtain the desired information. To address this need, the IR research community is currently exploring ways of pinpointing highly relevant information. We view information extraction [4] as one of the core technologies to help facilitate highly focused IR. Indeed, recognizing entities and semantically meaningful relations between those entities is key to providing focused information access.

DIR 2005, 5th Dutch-Belgian Information Retrieval Workshop.

Temporal information extraction provides a particularly interesting task in this respect. Temporal expressions (from now on, *timexes*) are natural language phrases that refer directly to time points or intervals. They not only convey temporal information on their own but also serve as anchors for locating events referred to in text. For reasons to be explained below, *recognizing* temporal expressions has become a "do-able" task, even without tremendous knowledge engineering efforts. Moreover, in recent years, the task of *interpreting* temporal expressions has begun to receive attention [17, 21]. And finally, from a user's perspective, temporal aspects of events and entities, and of text snippets and documents, provide a very natural mechanism for organizing information. Specifically, Question Answering (QA) is an area where accurate analysis (both recognition and normalization) of temporal expressions plays an important role. For example, answering questions like *"When was Van Gogh born?"* requires accurate identification of the date of birth of the person under consideration (recognition) and rendering of the answer in some standard format (normalization).

The importance of the proper treatment of timexes is reflected by the relatively large number of NLP evaluation efforts where they play a role. Recognizing timexes, for example, is an intergral part of many information extraction tasks (e.g., MUC-6 and 7 Named Entity Recognition tasks, ACE-2004 Event Recognition task). There are a number of annotation guidelines for timexes [10, 22, 20]. And recently, a timex annotated corpus has been released with the aim of improving the tasks related to timexes [24].

The type of timexes considered in a typical information extraction task are limited to date and time values [19, 7]. In the 2004 Temporal Expression Recognition and Normalization (TERN) task, however, a wide variety of timexes are considered—which makes the task of recognition and normalization more challenging and much more interesting.

This is an exploratory paper, in which our main interest is in the end-to-end timex-recognition-plus-interpretation task. Specifically, we are interested in identifying opportunities for the use of data-driven methods in the interpretation part of the task. On the recognition side, rule-based systems can provide very high precision but require significant human effort in rule development to achieve reasonable recall. As we will see, machine learning can provide excellent results on the recognition task with minimal human intervention, given a tagged corpus. Furthermore, such a data-driven approach may be preferable because of its portability and robustness.

For the interpretation task, however, machine-learning methods based on sequence labeling seem less appropriate. For one thing, there are a potentially unlimited number of classes (i.e., temporal values). Another problem is that a significant proportion of timexes require non-local context for interpretation (i.e., they are anaphoric or deictic). Even more problematic is the fact that many timexes require significant temporal computation with respect to contextually given information—the connection between form and content is mediated by both context and world knowledge.

Nevertheless, we aim to understand what the opportunities are for using robust, "shrink-wrapped" machine-learning tools for the interpretation task. We are interested in determining whether the use of machine learning for recognition affects normalization performance downstream, as well as where within the normalization task a data-driven approach might provide some leverage. To these ends, we present several experiments comparing recognition performance of machine-learning and rule-based systems and also comparing the performance of a rule-based normalizer on the output of various recognition systems. We analyze the performance of the rule-based normalizer, looking for places to apply machine learning.

In §2, we outline the TERN evaluation tasks of identification and normalization. In §3, we describe our identification experiments, comparing two rule-based systems and two runs of a machine learning system with different feature sets. Our normalization system is described in §4. Finally, we conclude in §5.

2. TASK DESCRIPTION

We participated in the 2004 edition of TERN, the Temporal Expression Recognition and Normalization Evaluation (`http://timex2.mitre.org/tern.html`). The TERN evaluation is organized under the auspices of the Automatic Content Extraction (ACE, `http://www.nist.gov/speech/tests/ace/`) program, whose objective is to develop natural language processing technology to support automatic understanding of textual data. We submitted runs from one of the rule-based recognition systems and from the rule-based normalization system described below. The TERN evaluation provided specific guidelines for the identification and normalization of timexes, as well as tagged corpora for training and testing and evaluation software. We followed these guidelines and used these resources for the experiments described below.

The TERN evaluation consisted of two distinct tasks: recognition and normalization. Timex recognition involves correctly detecting and delimiting timexes in text. Normalization involves assigning recognized timexes a fully qualified temporal value. Both tasks are defined, for human annotators, in the TIDES TIMEX2 annotation guidelines [10].

The TIDES guidelines introduce a type of SGML or XML element, TIMEX2, to mark timexes in text [10]. TIMEX2 elements may contain one or more of the following attributes: VAL, ANCHOR_DIR, ANCHOR_VAL, MOD, SET, NON_SPECIFIC. The VAL attribute indicates the reference of the TIMEX2; its range of values are an extension of the ISO 8601 standard for representing time [13]. The ANCHOR_DIR and ANCHOR_VAL attributes are for temporal expressions (primarily durations and fuzzy references) that are anchored to a reference time. At present, the MOD, SET, and NON_SPECIFIC attributes are normalization attributes that our system does not handle.

The recognition and normalization tasks are performed with respect to corpora of transcribed broadcast news speech and news wire texts from ACE 2002, ACE 2003, and ACE 2004, marked up in SGML format and, for the training set, hand-annotated for TIMEX2s.

An official scorer that evaluates both recognition and normalization performance is provided as part of the TERN evaluation. For recognition results, the scorer computes precision, recall, and F-measure both for TIMEX2 tags (in other words, for *overlap* with a gold standard TIMEX2 element) and for *extent* of TIMEX2 elements (in other words, exact match of entire timexes). For normalization, the scorer computes precision, recall, and F-measure for each of the normalization attributes listed above. Recall (and thus, F-measure), however, is computed not with respect to all possible TIMEX2 elements in the gold standard but only with respect to the TIMEX2 elements tagged by the system. Because we are interested in the end-to-end task, we report our results in this paper with recall (and F-measure) computed with respect to all possible TIMEX2s.

3. RECOGNITION

The recognition task is to identify phrases that refer to time points. The TIDES guidelines limit the set of markable timexes (which they indicate with the TIMEX2 tag) to those phrases headed by a temporal trigger word. The latter seem to fall into several categories. Some refer to time units of definite duration (minute, afternoon, day, night, weekend, month, summer, season, quarter, year, decade, century, millennium, era, semester). Others refer to definite points in time (January, Monday, New Year's Eve, Washington's Birthday, yesterday, today, tomorrow, midnight). Still others indicate repetition with respect to a definite period (daily, monthly, biannual, semiannual, hourly, daily, monthly, ago). And some refer to temporal concepts that can at least be oriented on a timeline with respect to some definite time point (future, past, time, period, point, recent, former, current, ago, currently, lately).

Syntactically, TIMEX2s must be one of the following: noun, noun phrase, adjective, adverb, adjective phrase, or adverb phrase. All premodifiers and postmodifiers of the timex must be included in the extent of the TIMEX2 tag, e.g.,

- Premodifiers: *8* winters, *the past* week, *four bad* years, *about 15* minutes, *less than a* week
- Postmodifiers: *Nearly* three years *later*, the week *before last*, two years *ago*, three years *in prison*, Only days *after his father was assassinated*, months *of Israeli-Palestinian bloodshed and Israeli blockades*

Recognition can be thought of either as a partial parsing task or as a labeling task, and rule-based methods or machine-learning systems can be deployed, according to the perspective taken. In §3.1, we describe a machine-learning approach to timex identification, which is based on Conditional Random Fields, and then in §3.2, we present our rule-based approach.

3.1 Machine learning methods

Timexes are a type of entity commonly dealt with in named entity recognition tasks in which machine learning techniques have been shown to provide good results. As mentioned before, though, the type of timexes considered in TERN are quite a bit more diverse than the ones considered in typical information extraction tasks. Nevertheless, these timexes still have some desirable properties from the perspective of machine learning. In particular, the core vocabulary used for building them is restricted, so the use of lexical features does not significantly increase the feature set used for machine learning.

A machine learning technique that has recently been introduced to tackle the problem of labeling and segmenting sequence data is Conditional Random Fields (CRFs, [16]). Unlike Hidden Markov Models [16], CRFs are based on exponential models in which probabilities are computed based on the values of a set of features induced from both the observation and label sequences. This enables the incorporation of overlapping and interacting features into the model. CRFs have been shown to perform well in a number of natural language processing applications, such as POS tagging [16], shallow parsing or noun chunking [23], and named entity recognition [18]. Their characteristics make CRFs ideally suited for the specific task of recognizing timexes as they provide us with a framework for combining evidence from different sources to maximize performance.

We used the implementation of CRFs from the minorThird toolkit for extracting timexes from text [8]. The training material is a tagged corpus in which the timexes are marked by XML tags. The task, then, is to learn from these training instances of timexes rules or patterns to recognize new instances. In this framework, phrase identification tasks are reduced to word labeling tasks by assigning each word one of the labels (B)egin, (I)nside, (O)utside, according to whether it begins a phrase of interest (in our case, a timex), continues such a phrase, or is not part of such a phrase [14].

3.1.1 *Experiments*

The training data consists of 511 files from ACE 2002, ACE 2003 and ACE 2004; the test data consists of 192 files. As mentioned earlier, the temporal expressions in the training files, are marked with XML tags. The minorThird system automatically converts from XML format to B-I-O format. A temporal expression enclosed by `<TIMEX2>` tags constitutes a span. The features in the training instances are generated by looking at the surface forms of the tokens in the spans and their surrounding contexts.

In order to simplify our implementation and achieve reasonably high generalization, we use simple lexical and character features which can easily be derived from the surface forms of words or phrases. We used a context window of three words to the left and right. One set of features is the lowercase form of all the tokens in the span, with each token contributing a separate feature. The tokens in the context window constitute another set of features. These feature sets capture the lexical content and context of timexes. Additionally, character types and character type patterns of tokens in the spans are used to capture the character patterns exhibited by some of the tokens in temporal expressions. The patterns are defined using the symbols, A, a, X, x, and 9. A and a are used to define character type features; X and x are used to express the character type pattern features; and 9 is used for representing numeric tokens. For example, the token *Monday* is represented by the character type 'Aaaaaa' and character type pattern 'Xx+'. Character features are extracted not only from tokens within the spans but also from the context windows. The first and the last tokens of a span and their corresponding character types and character type patterns are also part of the feature set. The number of tokens in a span is yet another feature.

3.1.2 *Results*

We train the system with the surface features listed previously. The recognition results for the CRF-based system, scored using the official TERN scorer, are given in Tables 1 and 2, row 1.

	TIMEX2		
	Precision	Recall	F
CRF	0.980	0.842	0.906
CRF + extra features	0.979	0.856	0.914
POS-only	0.979	0.633	0.769
POS and chunk	0.989	0.537	0.696
POS-only (w/years)	0.978	0.713	0.825
POS and chunk (w/years)	0.987	0.617	0.759

Table 1: Recognition results: TIMEX2 (overlap).

Table 1 indicates the system performance on partial-match identification of TIMEX2s, where credit is given for tagging any part of an actual TIMEX2. Table 2 indicates the system performance on exact-match identification, in which a system only gets credit for the identifying the exact extent of a TIMEX2. Unsurprisingly, the scores for the extent measure are lower than those for the TIMEX2 measure.

	TEXT		
	Precision	Recall	F
CRF	0.798	0.685	0.737
CRF + extra features	0.855	0.748	0.798
POS-only	0.795	0.514	0.625
POS and chunk	0.830	0.450	0.584
POS-only (w/years)	0.811	0.591	0.684
POS and chunk (w/years)	0.843	0.527	0.648

Table 2: Recognition results: TEXT (extent).

3.1.3 *Error Analysis*

An analysis of the output of the CRF system reveals errors which can be eliminated with additional features. One set of errors relates to spans that are too short or too long. To address this problem, we created a dictionary of core lexical items in timexes and restricted context information to these lexical items, which should improve detection of timex boundaries. The list was generated from the Penn TreeBank by extracting the most frequent lexical items occurring in the constituents labeled -TMP. Furthermore, the default character pattern for numeric tokens is the simple pattern 9+; it assigns the same form for all numeric values, which misses an important fact about the form of year expressions. A simple pattern which takes into account the form of a year expression enabled the system to extract year values more precisely. We also added a list of names for days of the week and months of the year as separate features. These simple additions to the default features resulted in significant performance improvements (Tables 1 and 2, row 2).

Although the use of additional features, such as lists of names of days, helps to improve the scores, there seems to be a limit to what can be achieved using such shallow methods given the fact that the training material is fixed. A deeper level of linguistic analysis needs to be made to achieve significant improvement. Analysis of some of the errors suggest that some knowledge of constituency, e.g., noun chunking or dependency parsing, might help in identifying timex boundaries. In general, the CRF-based recognizer has a tendency to limit extents to common temporal elements, such as, for example, *Orthodox Christmas Eve* $\rightarrow$ *Christmas*.

3.2 Rule-based methods

Finite-state automata have been employed extensively in partial parsing, or chunking, as well as in morphological analysis and other natural language processing tasks [2, 15, 5]. In partial parsing, chunks form the basic level of constituency, the non-recursive "core" of a major phrase [1]. Chunks can be identified by regular expression patterns over part-of-speech tags, and, in turn, regular expression patterns over chunks can be used to form higher levels of constituency, such as simplex clauses.

One way of looking at the timex detection task is as a partial parsing task in which timexes form the level of interest. Apart from date and time expressions with fixed forms (which lend themselves readily to regular expression patterns), timexes are constrained to be natural language phrases headed by one of a small number of trigger words as discussed in §2. Chunks can be used to approximate these phrases and can, if needed, be joined by appropriate patterns to form timexes.

We experimented with using two levels of prior linguistic analysis to feed our rules: part-of-speech (POS) tags and chunks. In one experiment, we used patterns over text tagged only with POS tags; in the other, the patterns range over chunked and POS-tagged text.

Our development efforts concentrated on the patterns involving chunks and POS tags. We first built up patterns for word classes: deictics (*today*, *yesterday*, etc.), units (*hour*, *day*, etc.), days (*Monday*, *Tuesday*, etc.), months, temporal adjectives, temporal adverbs, and so on. We also built up complex expressions for times (e.g., 12:24:45.69GMT) and dates (e.g., 10/31/1999). Then, in principle, our identification patterns would simply look for chunks headed by words from the appropriate classes. We followed this principle for units and deictics, but for other classes, we wrote patterns that look at more of the relevant internal structure of chunks, expecting that this might help normalization. For example, one of our complex date patterns is:

```
<NC>[ Month Date Comma? Year_full ]
```

where `Month` refers to the month class, `Date` refers to the date class (i.e., 1–30), and `Year_full` refers to years from 1900–2099. In developing rules for normalization, as described in §4, this pattern can be interpreted directly.

For our experiment without chunks, we used the same word classes and complex expressions for times and dates, as well as the patterns that looked at the internal structure of chunks. We then wrote simple noun and preposition chunk patterns to allow for head-based chunk patterns. Thus, for noun chunks headed by units, we used the following pattern:

```
(DT|CD|PP\$)? \
  (CD|JJ[RS]?|RB[RS]?|VVG|VVN)* NN? Unit
```

where `(DT|CD|PP$)` matches specifiers (determiners, cardinal numbers, and possessive pronouns), `(CD|JJ[RS]?|...)` matches modifiers, including adjectives, adverbs, and participles, `NN` matches prenominals, and `Unit` matches words from the unit class.

3.2.1 Experiments

We built two rule-based systems for timex identification: the first uses only POS tags while the second makes use of chunks and POS tags. The first system is based on hand-built patterns over POS tags and lexical items. For POS-tagging, we use TreeTagger [25] and convert the output into XML. The intuition behind the use of POS tags is that they provide some level of generalization for pattern writing, as well as resolving some word sense ambiguities (e.g., the ordinal number sense of *second* can be distinguished from other senses (including the time unit sense) simply by virtue of being tagged as an adjective[1]). While the generalization provided by POS-tagging is merely a convenience for closed-class categories, such as determiners, for which patterns could exhaustively list *a*, *the*, *this*, etc., it is absolutely necessary for writing patterns over open-class categories, such as adjectives and nouns.

To compile our regular expression patterns into finite-state automata, we use flex [11], a standard scanner generator that does character-level regular expression matching. The overall system is written in CDuce, a higher-order functional programming language that allows pattern matching over XML structures [6]. Since the TERN data is accompanied by a DTD indicating which document elements may contain TIMEX2s, a CDuce program whose type structure is derived directly from the DTD determines which elements need to be processed. It sends the text of these elements to be POS-tagged by TreeTagger and then TIMEX2-tagged by the flex-generated scanner. Finally, extraneous tags (i.e., TreeTagger output) are stripped off, and the output is aligned with the initial input to restore formatting.

The second system (which was essentially the recognition component of our submission to TERN) is based on the same hand-built patterns as our first rule-based system but takes chunks into account, as well. The intuition behind the use of chunks is that they provide ready-made approximations to phrases that just as crucially keep *out* words that do not belong as include words that do. Then, as we explain above, we can simply write patterns to look for chunks headed by timex trigger words and, if necessary, join a timex-headed chunk with a required complement chunk. For example, the timex *24 years after his death* consists of a noun chunk *24 years* and its complement preposition chunk *after his death*.

As before, we use TreeTagger, but here it serves double duty, POS-tagging and chunking the input; again, TreeTagger output is converted into XML. Now, regular expressions for matching sequences can be compiled into finite-state automata by a variety of tools (such as flex), but the patterns we use for matching the hierarchically structured TreeTagger output require compilation into tree automata. This is where CDuce comes in: it compiles regular expressions over hierarchical structures (in particular, XML elements) into tree automata, using pattern ordering and a greedy matching policy [12] to resolve conflicts. This system works exactly as the other one, except that instead of sending the TreeTag-

[1]Of course, the various nominal senses of *second* cannot be distinguished in this way.

ger output to a flex-generated scanner for TIMEX2-tagging, the CDuce program itself does the tagging.

3.2.2 Results

We ran each of the two systems described above on the TERN test corpus. The results, again scored using the official TERN scorer, are given in Tables 1 and 2, in the rows labeled *POS-only* and *POS and chunk*.

After the TERN evaluation, a preliminary error analysis revealed that both rule-based systems lack a rule to identify years standing alone. Adding a simple rule to identify years between 1900 and 2019 increases recall significantly for both tag and extent, with only a very small precision penalty. The results of adding this rule are shown in Tables 1 and 2 in rows *POS-only (w/years)* and *POS and chunk (w/years)*.

3.2.3 Error Analysis

Both rule-based systems very rarely misidentify something as a TIMEX2. Most of these spurious TIMEX2s are non-temporal occurrences of the words *past*, *present*, and *now*, which are anyhow only weakly normalizable. The real problem with the rule-based systems is in recall, which is largely a matter of insufficient patterns. For example, as previously mentioned, one pattern that was inexplicably left out of both rule-based systems is the simple year pattern.

Both rule-based systems do make a number of extent errors, though. The vast majority of these errors (90.5% for TERN, 93.1% for flex) result from the systems tagging too little of an expression, missing premodifiers and arguments. In some cases, it is clear that deeper syntactic information is needed to get the full extent—for example, chunking is not enough to distinguish *24 years after his death* from *Two years after the crash site*. The first phrase is a complete timex, since the phrase *after his death* is a PP, but the second expression is not—*after the crash site* is the beginning of a complement clause (*... was discovered*).

One caveat, though, regarding the use of deeper syntactic analysis: more complicated syntactic analysis is generally less reliable. Even the chunker, which performs relatively shallow analysis, introduces errors. Some of the extent errors peculiar to the chunk-based system include phrases such as *we're talking months*, *congress last week*, *Palestinian leader Yasser Arafat last week*, *secretly indicted nine months ago*. In each of these cases, the chunker included too much material in a noun chunk, which our patterns take as the boundary for timexes with unit triggers, such as *months*.

Overall, both rule-based systems exhibit high precision and low recall, a symptom of the development methodology and the amount of time devoted to actual rule development. The differences between the systems—greater precision for the chunk-based system and greater recall for the POS-tag-based system—are not particularly surprising. The chunk-based system requires more pieces of information to be in place for an expression to qualify as a TIMEX2 (e.g., a date expression needs not only to contain a Month and a Date, but it must also have been marked as a noun chunk), so false positives are less likely. But since its source for this additional information is less reliable (i.e., chunking is more difficult than tagging), it is more likely to reject well-formed expressions that have been POS-tagged correctly but chunked incorrectly. Interestingly, the use of chunks provides only a small boost to precision at a relatively high price for recall. Since chunking does not seem to be the right level of linguistic analysis to solve the extent problems, it is not clear that the additional processing they require is warranted, something that must be kept in mind when considering how to improve the CRF-based recognizers, as well.

3.3 Recognition: Upshot

To put these results in context, the top-performing system at the 2004 TERN evaluation—also a machine learning-based system—scored 0.981, 0.909, 0.944 (precision, recall, F-measure) for partial-match (TIMEX2) and 0.906, 0.840, 0.872 for exact-match (TEXT) [9]. The results of the CRF run with tuned features approaches this level of performance (and compares well to the other TERN systems). What is more, even the results of the CRF system using the default feature set are considerably better than either rule-based system. Recall is vastly improved, with only a very small precision penalty. Despite the differences in performance in the various systems, they shared several features: good precision, approaching good recall on tags, and problems with extent largely stemming from not having enough syntactic information to get arguments and modifiers right.

4. NORMALIZATION

Given the ease with which timex recognition can be performed by an off-the-shelf machine learning system with little human intervention, it is natural to question why one should bother with a rule-based system at all. One possible answer is normalization. Remember that recognition is only a part of the larger task of temporal information extraction. For the kind of downstream application we are interested in, it is important to interpret recognized expressions and render them in a standard format. The most natural way to approach this normalization problem is with a rule-based system,[2] and it seems that if effort is going to be invested in rule development for normalization, it is only natural to use those rules for recognition, as well. After all, for a downstream application that relies on normalized output, recall of unnormalizable timexes is superfluous.

The questions we would like to address, then, involve the proper role of machine learning in an end-to-end temporal information extraction system. Is it really the case that better recognition performance is wasted on rule-based normalization? Are there places within the normalization task itself where data-driven approaches can be applied?

We describe a rule-based normalization system we have built and report on several runs of the system on the output of different timex recognizers. A detailed error analysis of these runs reveals several points within the normalization pipeline where using a data-driven approach might improve performance. Furthermore, comparison of the results indicates that, at least for a rule-based system that relies heavily on imperfect upstream linguistic analysis components, such as ours, better recognition performance may, in fact, improve normalization.

4.1 Normalization method

Timex normalization is the problem of assigning a value to a recognized timex. As we discuss in §2, the TIDES guidelines distinguish several kinds of values; our normalization system focuses on time points, durations, and the

[2]Consider the TERN 2004 evaluation, where all recognition-only systems were machine learning-based, while all the systems participating in the full task were rule-based!

past, present, and future reference tokens. Time points are expressed by three different kinds of timexes: fully qualified, deictic, and anaphoric. Fully qualified timexes, such as *March 15, 2001*, can be normalized without reference to any other temporal entities. Deictic and anaphoric timexes, however, must be interpreted relative to another temporal entity. Deictic timexes, such as *today*, *yesterday*, *three weeks ago*, *last Thursday*, *next month*, and so on, are interpreted with respect to the time of utterance—for our corpus, the document time stamp. Anaphoric timexes, such as *March 15*, *the next week*, *Saturday*, are interpreted with respect to some reference time—a salient time point previously evoked in the text that may shift as the text progresses. Moreover, some anaphoric timexes (those without an explicit direction indicator such as *next* or *previous*) also depend on the tense of the verb they modify: with past tense, they indicate the most recent time point prior to the reference time that matches the timex description; with present and future tense, they indicate the opposite.

For our normalization system, fully qualified and deictic timexes are straightforward to normalize, particularly since the documents it processes are time-stamped. Our system finesses anaphoric timexes in two ways. First, it simply treats them as deictics, instead of keeping track of introduced temporal entities and trying to determine their relative salience. Since the TERN corpus consists largely of short news stories focused on the immediate present, this heuristic seems reasonable. Secondly, instead of using the tense of the verb on which an anaphoric timex is dependent, we take the tense of the first (tensed) verb in the same sentence as an anaphoric timex. Determining dependency relations would require additional (more fragile) syntactic analysis, while finding tensed verbs is straightforward given the syntactic analysis already performed.

Durations are more difficult. Even fully qualified timexes expressing durations, i.e., those in which both the quantity and the unit are specified, such as *six months*, *800 years*, *three long days*, are systematically ambiguous between a duration and a point reading. Furthermore, many durations are anchored, either explicitly, as in *He will be barred from teaching for five years after that*, or implicitly. Our normalizer only normalizes fully qualified duration-like timexes, making no attempt to anchor them. Also, in the absence of an explicit directional indicator, such as *next* or *ago*, it assumes a duration reference.

Our normalizer also normalizes timexes that refer generally to the past, present, or future, such as *now*, *recently*, *future*. Such timexes receive a token value, PRESENT_REF, PAST_REF, or FUTURE_REF, as appropriate, as well as values for ANCHOR_DIR (AS_OF, BEFORE, or AFTER) and ANCHOR_VAL (the document timestamp).

4.2 Normalization system

In this section, we give some specific examples of normalization rules, as well as a technical overview of our system. Our normalization rules are mostly recognition rules augmented with pattern-matching variables to extract elements of the expression necessary to compute normalized values and with functions to perform the computation. For example, consider the pattern from §3.2 for matching fully qualified dates, augmented with pattern-matching variables:

```
<NC>[ (Month & m) (Date & d) Comma? (Year & y) ]
```

The function associated with this rule simply converts `m` to an integer and concatenates `y`, `m`, and `d` to produce the value.

For an example involving computation with respect to the reference time, consider:

```
<NC>[ (CD & cd) _* (Unit & un) ] <ADVC>[ Ago ]
```

where _ is a wildcard. The function associated with this recognition rule subtracts `cd un` units from the reference time and takes truncates resulting value to the appropriate granularity.

The most involved computation, requiring the examination of verb tense, is for some of the simplest expressions:

```
<NC>[ (Day & d) ]
```

where `Day` matches *Monday*, *Tuesday*, etc. The interpretation computation converts `d` into an integer; then, depending on the tense of the sentence, computes the date for the closest `d` in the past or in the future, and finally, produces a value from the year, month, and date of the result.

Finally, the rule for fully qualified durations is:

```
<NC>[ (CD & cd) _* (Unit & un) ]
```

where normalized versions of `cd` and `un` are concatenated (with a leading `P`) to produce a duration value. Note that this rule must be ordered *after* the *ago*-rule above and similar such rules in order to avoid spurious matching.

Much of the work done in rule development for identification paid off in development of the normalization system. Furthermore, on the assumption that for normalization, the ambiguity reduction and generalization benefits of POS tags are unnecessary, we were able to liberalize our normalization patterns so that they ignore POS tags. Our strategy of identifying duration timexes, however, using patterns matching noun chunks that simply have an appropriate head, meant that new patterns, such as the one above, had to be developed for interpreting such timexes.

The normalizer is written in Perl and takes as input the output of our TERN timex tagger, including not just newly added TIMEX2 tags, but also the XML elements indicating chunks and words. XPath queries are used to extract the document date stamp and sentences containing TIMEX2s and to determine the tense of these sentences (from the firstverb chunk). TIMEX2s are matched against regular expression patterns that extract the necessary information for normalization. Relative expressions are evaluated with respect to the document date stamp and the tense of the sentence; the Time::Piece Perl module is used to perform any necessary temporal arithmetic. The output of the normalizer is simply its input with the addition of the appropriate attributes for normalized TIMEX2s.[3]

4.3 Results

We ran our normalizer on the output of two different recognition systems (the chunk-based finite-state system, with the added year pattern, and the optimized CRF system from §3), as well as on the gold standard for recognition output, in order to see the effects of recognition performance on normalization performance. As we mention above, our initial expectation was that since the rules for normalization are more or less those of the rule-based recognizer, any

[3]We note that the normalization system we report on in this paper differs slightly from our TERN entry in that two simple bugs involving output format have been fixed.

	Corr	Incor	P	R	F
VAL	782	143	0.845	0.449	0.586
ANCHOR_DIR	63	11	0.851	0.150	0.255
ANCHOR_VAL	62	12	0.838	0.148	0.252

(a) Rule-based recognizer.

	Corr	Incor	P	R	F
VAL	885	231	0.793	0.501	0.614
ANCHOR_DIR	123	13	0.904	0.294	0.444
ANCHOR_VAL	121	15	0.890	0.289	0.436

(b) CRF recognizer.

	Corr	Incor	P	R	F
VAL	955	219	0.812	0.549	0.655
ANCHOR_DIR	137	14	0.901	0.327	0.480
ANCHOR_VAL	137	14	0.901	0.327	0.480

(c) Gold standard recognition.

Table 3: Normalization results

additional timexes identified by better recognition systems would not be normalized anyhow.

In Table 3(a), (b), and (c) we give the results of these three runs on the TERN test corpus. Remember that our recall and F-measure scores differ from the official TERN scores; they are computed with respect to all possible timexes rather than just those identified by the recognizer (1741 for VAL, and 419 for both ANCHOR_DIR and ANCHOR_VAL). According to the official TERN scorer, the recall scores (and correspondingly, the F-meastures) are higher for both the rule-based and CRF runs (0.699 and 0.583, respectively).

To put these results in context, the best performing system at TERN 2004 had scores of 0.875, 0.784, and 0.827 for VAL; 0.833, 0.585, 0.687 for ANCHOR_DIR, and 0.683, 0.649, 0.666 for ANCHOR_VAL (again, these scores are computed differently than the official TERN scores).

4.3.1 Error analysis and discussion

Looking at the results across runs, it seems that, in fact, better recognition actually does help normalization. We performed a preliminary comparison of the gold standard and rule-based recognizer runs to try to determine why this might be the case. Of course, the gold standard simply presents more actual timexes to the normalizer than the rule-based recognizer; the question is why, given that the normalizer patterns are the same as the rule-based recognizer patterns, the normalizer actually attempts to normalize any of these additional timexes.

What we found is that, for the most part, the upstream components are to blame. In order to produce input to the normalizer for the gold standard and CRF runs, we simply sent the recognized output through TreeTagger. As it turns out, though, the additional presence of TIMEX2 tags affects the tagging and chunking performance in two ways. First of all, it improves tokenization, especially of punctuation signs, allowing more normalization patterns to match. Chunking is also improved by the presence of TIMEX2 tags in the input. TreeTagger has an SGML mode in which it ignores, but does not respect, markup. Thus, it often produces chunks that cross TIMEX2 boundaries. We had to write a post-processor to close and re-open such chunks in the output of TreeTagger on the gold standard and CRF recognition output in order for our normalizer to run at all (it relies on well-formed XML). This post-process, however, creates "good" (matching) chunks in places where no such chunks exist in the output of TreeTagger on untagged input.

The other notable source of additional matches in the gold standard run is our liberalization of patterns in the normalization system to ignore POS tags. As it turns out, the lack of capitalization in the broadcast news documents results in a large number of mistagged day and month names which are thus missed by the rule-based recognizer.

We have not yet compared the CRF run with the rule-based run, but we expect that similar considerations will apply. We do note that the overall performance of recognition + normalization using the CRF recognizer is better than the strictly rule-based combination, so it looks as though any significant improvements in recognition performance, even if still imperfect, helps normalization.

We did perform a detailed error analysis of the run using the output of the rule-based recognizer and found a number of sources of (precision) error. The single largest source of error (42%) is an interesting property of the corpus—unlike in ordinary conversation, many of the documents in the corpus refer to the current day by name rather than as *today*.[4] Our finesse of anaphoric timexes is to blame for 17% of the errors—only one, though, actually requires interpretation with respect to a reference time distinct from the document time stamp; the remainder fail because of our strategy of choosing the first tensed verb in the sentence to decide directionality rather than looking more closely at the actual dependency relations. Errors of misclassification account for another 15% of the errors: the largest number of these are the result of normalizing a point reference as a duration; the other major misclassification error is with respect to uses of *today* to mean the present rather than the current day. A simple pattern-ordering bug results in 9% of the VAL errors and also explains all of the ANCHOR_DIR and ANCHOR_VAL errors. The remaining errors stem from problems with the recognizer or the tagger and chunker.

4.4 Normalization: Upshot

We take two lessons away from these experiments. The first is that, even from the perspective of the end-to-end task of temporal information extraction, it seems to be worthwhile to optimize recognition and normalization of timexes independently. The second is that data-driven approaches can help reduce the reliance on brittle upstream linguistic processing required by rule-based approaches. A caveat, though: as we point out in §3, it does seem that some sort of deeper syntactic analysis may be necessary to improve extent recognition performance, even for machine learning.

Additionally, our error analysis points to some aspects of the normalization task that may be amenable to a data-driven approach. Almost half of the errors our normalizer makes are the result of incorrectly guessing whether undirected anaphoric expressions such as *Tuesday* refer to the past, the present, or the future. This decision, though, can be easily stated as a classification problem for machine learning, with somewhat local relevant features (such as parent verb tense). Additionally, given appropriate data sources, even peculiarities of the data with regard to this decision, such as the use of *Tuesday* on a Tuesday to refer to the

[4]Most of these documents seem to be newswire items time-stamped late in the day, so the problem may be that they were intended for release on the following day.

current day rather than the previous Tuesday, can be automatically learned. The other major source of error is another sort of misclassification: confusing points, durations, and fuzzy reference. This decision, again, can be stated as a simple three-way classification problem, and, again, many of the relevant features (such as governing prepositions) are local. Thus, although it is not clear how to turn the general problem of normalization into a labeling task for machine learning, there do seem to be sub-problems that may be amenable to a data-driven approach.

5. CONCLUSION

Timex recognition is a task for which machine learning is well suited. An off-the-shelf system can produce respectable results straight out of the box, while a little bit of informed feature selection results in near state-of-the-art performance. There is room for improvement, though, particularly with respect to extent, where some sort of syntactic analysis seems to be required.

Rule-based systems are good at identifying timexes with high precision, while recall depends directly on effort invested. Our experiments indicate, though, that chunking is too shallow a level of syntactic analysis to help much with extent problems, and that, at the same time, it is inaccurate enough to introduce problems of its own.

With respect to normalization, we see, first of all, that better recognition performance does seem to improve rule-based normalization, even when the performance goes beyond that of a recognizer using the same set of rules as the normalizer. This suggests that independent optimization of recognition and normalization is a reasonable strategy for optimizing the end-to-end temporal information extraction task. Secondly, we have also uncovered some parts of the normalization task that may be amenable to a machine-learning approach. As we continue development of our normalizer, we proceed with ideas on how to use data-driven techniques, as well as expert linguistic knowledge, to improve performance.

6. ACKNOWLEDGMENTS

David Ahn was supported by the Netherlands Organization for Scientific Research (NWO) under project number 612.066.302. Sisay Fissaha Adafre was supported by NWO under project number 220-80-001. Maarten de Rijke was supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 220-80-001, 612.000.106, 612.000.207, 612.066.302, 264-70-050, and 017.001.190.

7. REFERENCES

[1] S. Abney. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, 1991.

[2] S. Abney. Partial parsing via finite-state cascades. *Natural Language Engineering*, 1996.

[3] D. Ahn, V. Jijkoun, J. Kamps, G. Mishne, K. Müller, M. de Rijke, and S. Schlobach. The University of Amsterdam at TREC 2004. In *TREC 2004 Conference Notebook*, Gaithersburg, Maryland USA, 2004.

[4] D. Appelt and D. Israel. Introduction to information extraction technology: IJCAI-99 tutorial, 1999. `http://www.ai.sri.com/~appelt/ie-tutorial/`.

[5] K. Beesley and L. Karttunen. *Finite-State Morphology*. CSLI Publications, 2003.

[6] V. Benzaken, G. Castagna, and A. Frisch. CDuce: An XML-centric general-purpose language. In *Proceedings of the ACM International Conference on Functional Programming*, 2003.

[7] N. Chinchor. MUC-7 named entity task definition, September 1997. `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/ne_task.html`.

[8] W. Cohen. Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data, 2004. `http://minorthird.sourceforge.net`.

[9] L. Ferro. Annotating the tern corpus, 2004. `http://timex2.mitre.org/tern_2004/ferro2_TERN2004_annotation_sanitized.pdf`.

[10] L. Ferro, L. Gerber, I. Mani, and G. Wilson. *TIDES 2003 Standard for the Annotation of Temporal Expressions*. MITRE, April 2004.

[11] Flex. `http://www.gnu.org/software/flex`.

[12] A. Frisch and L. Cardelli. Greedy regular expression matching. In *Proceedings of ICALP*, 2004.

[13] ISO 8601: Information interchange – representation of dates and times, 1997.

[14] D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice-Hall, 2000.

[15] L. Karttunen, J.-P. Chanod, G. Grefenstette, and A. Schiller. Regular expressions for language engineering. *Natural Language Engineering*, 1997.

[16] J. Lafferty, F. Pereira, and A. McCallum. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[17] I. Mani and G. Wilson. Robust temporal processing of news. In *Proceedings of the 38th ACL*, 2000.

[18] A. McCallum and W. Li. Early results for Named Entity Recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th CoNLL*, 2003.

[19] MUC-6 named entity task definition, May 1995. `http://www.cs.nyu.edu/cs/faculty/grishman/NEtask20.book_1.html`.

[20] J. Pustejovsky, J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz. TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the AAAI Spring Symposium*, 2003.

[21] F. Schilder and C. Habel. From temporal expressions to temporal information: Semantic tagging of news messages. In *Proceedings of the ACL-2001 Workshop on Temporal and Spatial Information Processing*, 2001.

[22] A. Setzer and R. Gaizauskas. Annotating events and temporal information in newswire texts. In *Proceedings of LREC2000*, 2000.

[23] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*, 2003.

[24] TimeBank. `http://www.cs.brandeis.edu/~jamesp/arda/time/timebank.html`.

[25] TreeTagger. `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`.

Generation of Non-redundant Summary Based on Sum of Similarity and Semantic Analysis

Sun Yang
Internet Lab.
Chonbuk National University
Division of Electronics & Information Engineering
82-63-279-6684 Chonju 561-756 Korea
supermary@internet.chonbuk.ac.kr

Soon Cheol Park
Internet Lab
Chonbuk National University
Division of Electronics & Information Engineering
82-63-270-2467 Chonju 561-756 Korea
scpark@chonbuk.ac.kr

ABSTRACT

With tons of information pouring in everyday, text summaries are becoming more and more essential. This paper proposes our summarization system: an efficient method to extract the most important and non-redundant “n” (related to the compression rate) sentence segments based on sum of similarity and semantic analysis. The new characteristics of our method are listed as follows: 1) we use preprocessing to delete the additional information (comma parenthesis) that won't turn up in the summarization; 2) get sentence segment by syntax; 3) redesign the vector similarity between a pair of sentences by using sum of similarity; 4) in order to maximize topic diversity, we use a strikingly different redundancy reducing ways other than MMR. Because of all of above properties, experimental results show that our approach compares favorably with some other summary systems.

Categories and Subject Descriptors

H.3.1 **[Information and Retrieval]**: Content Analysis and Indexing–*Abstracting methods*.

General Terms

Algorithms, Performance, Experimentation, Theory.

Keywords

Text summarizations, aggregate similarity, reduce redundancy.

1. INTRODUCTION

Document summarization has been a research topic since the 1950s; however, it became active in the second half of the 1990s. Nowadays, with the development of the Internet, it is becoming more and more important. There are two major types of text summary: extract and abstract. Extract summarization means the summarized text is extracted from the original text on a statistical basis or by using heuristic methods or a combination of both. Abstract summarization means the summarized text is an interpretation of the original text. The quality of an extract summary might not be as good as an abstract summary, but it is considered well for a reader to understand the main ideas of a document, so many works in this area are based on extraction.

Important seminal papers and recent advanced papers on text summarization can be found in Mani and Maybury's book on text summarization [10]. We briefly review here those based on the extraction approach. Luhn proposed a simple but effective approach by using term frequencies and their related positions to weight sentences that are extracted to form a summary [6]. Subsequent works have demonstrated the success of Luhn’s approach [11] [1]. Edmunson proposed that use of other features such as title words, sentence locations, and bonus words to improve sentence extraction [7]. Goldstein et al. presented an extraction technique that assigns weighted scores for both statistical and linguistic features in the sentence [9]. Recently, Salton et al. have developed a model for representing a document by using undirected graphs [5].

In addition to extracts and abstracts, summaries may differ in several other ways. Some of the major types of summary that have been identified include indicative (keywords indicating topics) vs. informative (content-laden); generic (author's perspective) vs. query-oriented (user-specific); background vs. just-the-news; single- document vs. multi-document; neutral vs. evaluative. A full understanding of the major dimensions of variation, and the types of reasoning required to produce each of them, is still a matter of investigation. This makes the study of automated text summarization an exciting area in which to work.

This paper is structured as follows. Section 2 presents some of the previous work in text summarization from which we developed our ideas. Section 3 describes each module in detail in our summarization system. Section 4 provides the results of experiments designed to evaluate the performance of our approach. Section 5 concludes the paper and discussions of some future research.

2. RELATED WORK

Almost all of this work focused on “summarization by text-span extraction”, with sentences as the most common type of text-span, because the readability of a list of keywords is typically low while paragraphs are unlikely to cover the information content of a document given summary space constraints.

2.1 Summary Foundation

In general, there are three major problems in creating extract summaries:

(1) Selecting the most important sentences.

(2) Generating coherent summaries.

(3) Repetitive information (redundancy) in the summary.

We also generate the summary by considering above problems. Salton’s SMARTIR is a very famous summarization system [11].

Salton et al have developed a model for representing a document by using undirected graphs. The basic idea is to consider vertices as paragraphs and edges as the similarities between two paragraphs. They suggested that the most important paragraphs should be the highly bushed node linked to many other paragraphs, which are likely to discuss the topic covered in those paragraphs. Consequently, such nodes are high likely to be included in the summary.

We don't apply this way directly with the text span as sentence level, because of the experiment by Canasai Kruengkrai and Chuleerat Jaruskulchai [2]. They found that use only this way is inadequate to measure the information of sentences in some cases, and the score of nodes in the text relation map tends to be slightly different on some document lengths. Especially, when nodes are used to represent text spans at the sentence level, the text relationship map is produced with the high redundancy in the sentence scores. If we immediately extract these nodes of sentences, many sentences with the same score are also included in the summary, which is a not good summary. In this paper, we overcome this disadvantage by use a measure called sum of similarity, we will show the detail of our approach in the section3.

2.2 Reduce Summary Redundancy

An ideal text summary includes the relevant information, for which the user is looking, and excludes extraneous and redundant information while providing background to suit the user's profile. Maximal Marginal Relevance (MMR) is a technique to reduce redundancy [8]. MMR reranks retrieval query results based on a document's dissimilarity to other relevant documents. The method is extended to summarize single-documents by re-ranking salient sentences instead of documents. Our ways of reducing redundancy is very different from MMR.

2.3 Rhetorical Structure Theory

For many natural language processing (NLP) tasks, identifying sentence boundaries is one of the most important prerequisites. And for our summarization system, sentence segmentation can affect the performance of the system. As an attempt to exploit the structural aspect of text, Marcu provided a corpus analysis when he created a tree, known as the Rhetorical Structure Theory (RST) Tree, for all the segments in the text. After the tree is created for each document, segments are laid out in a manner in which more important segments occupy the upper level of the tree nodes, whereas less important segments reside deeper in the tree. As a result, summarization can be carried out by performing cuts at various depths of the tree, producing summaries of various lengths.

But his approach also has disadvantages: (1) suffers from the complexity issue; (2) Every time a new document needs to be considered, the process for constructing its RST tree is very costly; (3) Furthermore, when the text does not contain many rhetorical relations, there is little telling which segment is more important than the others. Because of this, our approach doesn't use the whole RST tree, we only use the criterion of segmentation by a cue phrases. The basic idea about this is to separate out units that possibly convey independent meanings.

3. WHOLE ALGORITHM

Our summarization system is to employ IR techniques after preprocessing as far as they can take us, and then to augment them with semantic and statistical methods. At last we use a new way to reduce the redundancy. We can see the whole frame shown as Figure 1. Then we will describe each module in detail.

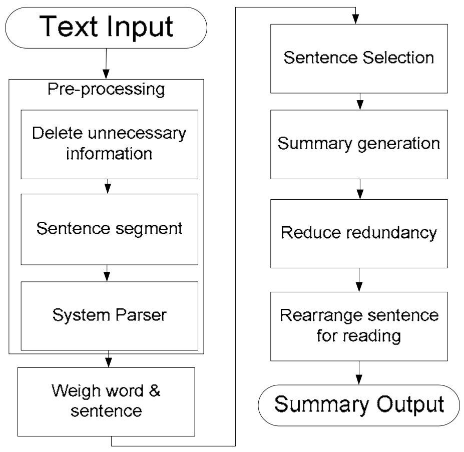


Figure 1. Overview of whole system.

3.1 Pre-Processing

3.1.1 Comma Parenthesis.

Words enclosed by curly braces ")" are called comma parenthesis, which are considered as additional information inside a segment, and won't turn up in summarization. So we use the preprocessing to delete the content between "(" and ")", then let the remain text as the input text.

3.1.2 Sentence Segmentation

(A) By using syntax.

Most of the experiments show that the sentence segmentation can affects the performance of the summarization system. In our system, we first divide the text to sentence by sentence. We don't only use a short list of end-of-sentence punctuation marks, such as periods, question mark, and exclamation point, colon, semicolon etc, since these forms of punctuation are not used exclusively to mark sentence breaks, sentence boundaries are ambiguous. For example a period can be used in an abbreviation, as a decimal point, in e-mail addresses and to indicate ellipsis. Some examples are shown below.

(a) She needs her car by 5 p.m. on Saturday evening.

(b) At 5 p.m. I had to go to the bank.

(c) "I have a dream!" is very famous.

(d) Decimal, e-mail addresses, abbreviations, initials in names, honorific titles.

(1) So we correct the condition to detect the sentence: When meet the punctuation, must detect the next character is whether space, if yes, may be this punctuation is sentence boundary, but if have no space, this isn't the end of the sentence. By this way we can solve above (c) and (d), such as the middle period in a.m. / p.m. / A.C / B.D.

(2) Every complete sentence starts with an uppercase letter, so after (1) we check the next character is whether uppercase, now we can solve the problem like this above (a), but we can't still solve the problem (b) by use this way. So we decide to use more semantic ways to solve in future.

Till now we can get the number of the sentence and sentence information in the input text.

(B) By using the feature comes from Rhetorical

Structure Theory (RST).

In a complex sentence, with two clauses, the main segment is called a nucleus, and its subordinate segment is called a satellite and is connected to the main segment by some kind of rhetorical relation. There are many such signaled by different cue phrases (e.g. because, but, if, however ...). Generally, when a rhetorical relation occurs, the nucleus is considered as a more important segment and has more chance of being in the summary than its satellite counterpart.

A sentence is segmented by a cue phrase. (See [3] for detailed descriptions of the cue phrases and the segmentation algorithm) The basic idea behind this is to separate out units (i.e. sentence segments) that possibly convey independent meanings. For instance, we can generate two sentence segments "I love playing tennis" and "because it is exciting" from the topic sentence "I love playing basketball, because it is exciting". The cue phrase used here is because which connects the two segments (i.e. main and subordinate clauses) with a cause relationship.

The purpose of segmentation is to use sentence segments as a basic unit for summarization.

3.2 Parser of System

In our parser, we use Porter Stemmer algorithms [12]. The Porter Stemmer is based on the idea that the suffixes in the English language are mostly made up of a combination of smaller and simpler suffixes. Porter Stemmer is most widely used in IR research, and can be thought of as a lexicon-free stemmer because it uses the cascaded rewrite rules that run very quickly and do not require the use of a lexicon. For the parser, the input is text; the output is the number of the term and whole term contents. After that we can count the frequency of the occurrence tf for each term in the text, and we can get the term-id (term identification number). These are the contents included in the future term table.

3.3 The Weight of the Word

Our summary is meant to use as simple and efficient term weighting algorithms as possible. "Word features" employed by many other systems which may be helpful in a specific text domain, but they would have to be redesigned whenever it were ported to a new domain. We also find that words in the title/topic and/or appearing in the first/last few sentences can be given more weight by means of an alterable parameter. At the same time, we find only the capital letter feature can show important information in the text.

3.3.1. Capital Letter In the Text.

Our system reads each sentence and gives value to the capitalized words, we define the value P_{upcase} , if the word is capital letter, and it receives a score grater than 1, otherwise 1.

3.3.2. TF×IDF.

The tf×idf method proved itself better than all the other methods of weight computation. Term frequency of the K_{th} term in a document, tf_k, times the logarithm of the number of documents in a collection N, divided by the number of documents where this term occurs n_k : $tf_k \times \log(N/n_k)$. This formula measures how frequently the term in a sentence occurs relatively to their occurrence in the entire document.

3.3.3. Term Weight.

We define $tf_k \times \log(N/n_k) \times P_{upcase}$ as term weight w_k .Till now we can update our term-table. The table includes *term-id, term, tf, idf.*

3.4 Sum of Similarity vs. Bushy Path

We use the concept of the text relationship map depicted by Salton et al [5]. Given a document D, we can represent it by an undirected graph G = (V,E), where V=$\{s_1,...s_n\}$is the set of sentence in that document. An edge (s_i, s_j) is the similarity between sentences S_i and S_j . A sentence s_i is considered to be a *t*-dimensional vector. This vector is a set of terms in the sentence $\{w_{s_i,1}, w_{s_i,2}, ..., w_{s_i,t}\}$, *t* is the number of terms in sentence S_i , and $W_{s_i,t}$ means the weight of the term *t* in sentence S_i . The cosine similarity between two sentences can be calculated as:

$$sim(s_i, s_j) = \frac{\sum_{k=1}^{t} w_{s_{i,k}} w_{s_{j,k}}}{\sqrt{\sum_{k=1}^{t} w_{s_{i,k}}^2} \sqrt{\sum_{k=1}^{t} w_{s_{j,k}}^2}}$$

From section 2, we know that Salton's way has some disadvantage, and they use bushy path (the number of links connecting it to other nodes on the text relationship map) as the importance of paragraphs or sentences. The link appears between two nodes when the similarity between two paragraphs or sentences is sufficiently large. The similarity threshold is adjusted as one of model parameters. So we overcome shortage by using the sum of similarity as shown in Figure 2. We measure the weight of sentence s_i, W_{s_i}, as an aggregate similarity (as shown in Figure 3), of which a node representing a sentence on the map is defined as the sum of weights on the links connecting it to other nodes on the map. $W_{s_i} = \sum_{j=1, j\neq i}^{n} sim(i, j)$ Because the important node is linked to many other nodes, so the sum of the similarity is relatively high comparing with other nodes. Then we sort the whole sentence according to their weights in decreasing order. We finally extract *n* top-ranked sentences as our initial summarization.

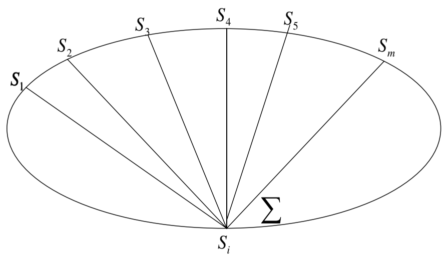


Figure 2. Sum of similarity.

And then we deal with this initial summarization to reduce redundancy, and then generate our final summarization.

$$\begin{array}{ccccc}
S_{11} & S_{12} & \ldots\ldots & S_{1n} & \sum_{\substack{i \neq 1 \\ i=2}}^{n} S_{1i} \\
S_{21} & S_{22} & \ldots\ldots & S_{2n} & \sum_{\substack{i=1 \\ i \neq 2}}^{n} S_{2i} \\
S_{j1} & S_{j2} & \ldots\ldots & S_{jn} & \sum_{\substack{i=1 \\ i \neq j}}^{n} S_{ji} \\
S_{n1} & S_{n2} & \ldots\ldots & S_{nn} & \sum_{\substack{i=1 \\ i \neq n}}^{n-1} S_{ni}
\end{array}$$

Figure 3. Weight of each sentence

3.5 Reduce redundancy

Our way highly improves the diversity of information coverage of summary, it is robust and transparent. To minimize summary redundancy, we do not consider sentence dissimilarity like MMR; we remove terms that have already been included in other important sentence. Figure 4 is the algorithm of our way to reduce redundancy.

```
Input: Initial summary
Output: non-redundant summary
1: Baseline (B) $\leftarrow$ Initial summary
2: Repeat
3: $S^* \leftarrow S_i$ with max sim($B$,S$_i$), ($1 \le i \le n$, but we initialize $i$=1)
4: Output $\leftarrow$ $Output \cup S^*$
5: Remove from B all terms occurring in $S^*$
6: until B not changed
7: Output non-redundant summary
```

Figure 4. Algorithm of reducing redundancy.

After generating of initial summarization, we extract the n top-ranked sentences. We use the whole summary as a baseline against any individual sentence, then we choose first sentence in initial summary because we think the first sentence in initial summary is the most important sentence. Then terms that appear in the first sentence are stricken out of the baseline, so as to remove content from the document. The process repeats until no more term can be stricken out from the baseline string. Figure 5 is the example. (We only simply use three sentences here) Because we have the term-table, so we use the term-id to represent the term directly.

Initial summary:

"1 5 3 8 9 4" = S_1

"2 5 19" = S_2

"3 9 4 1" = S_3

1: Baseline (B) = "1 5 3 8 9 4 2 5 19 3 9 4 1" $\rightarrow$ S_1

2: B= "$\cancel{1}$ $\cancel{5}$ $\cancel{3}$ $\cancel{8}$ $\cancel{9}$ $\cancel{4}$ 2 $\cancel{5}$ 19 $\cancel{3}$ $\cancel{9}$ $\cancel{4}$ $\cancel{1}$"

3: B= "2 19" $\rightarrow$ S_2

4: B= "$\cancel{2}$ $\cancel{19}$"

5: B= " " stop !

6: Output non-redundant summary is : $\{S_1, S_2\}$

Figure 5. Example of reducing redundancy.

4. EXPERIMENT RESULTS

4.1 Data Set (DUC Collection)

The National Institute of Standards and Technology (NIST) with support from the Defense Advanced Research Projects Agency (DARPA) is conducting a series of evaluations in the area of text summarization, the Document Understanding conference, (DUC), providing the appropriate framework for system independent evaluation of text summarization systems. In DUC 2002, Task #1 was single-document summarization-the goal being to generate 100 word summaries (although this limit is not strictly enforced) of 533 different domains documents from the TREC collection. In our system, we use some of these documents as the input, and then compare our output results with the initial summaries that DUC supply.

4.2 Results and Discussions

4.2.1 Evaluation Rules

Unlike document information retrieval, text summarization evaluation has not extensively addressed the performance of different methodologies by evaluating the contributions of each component. Here we evaluate the results of summarization by the classic precision and recall [14]. "S" means summaries. Finally, F is a combination of precision and recall can be calculated as follows: $F = (2 \times P \times R)/(P + R)$.

$$Precision = \frac{\text{\#of sentences extracted by the system which are in the target S}}{\text{total \# of sentences extracted by the system}}$$

$$Recall = \frac{\text{\#of sentences extracted by the system which are in the target S}}{\text{total \# of sentences in the target summaries}}$$

4.2.2 Performance Evaluation

Then we provide experimental evidence that our algorithm gives reasonable performance. The number of sentences in summary can be defined by user. We also examined our algorithm by using a way without reducing redundancy and extracting sentence randomly. Then we compare our method with commercial Copernic Summarizer [13].

Figure 6 P-R curves give us a detailed picture of how our method compares in performance to the other three ways of summary:

1) Random summaries perform poorly overall.
2) Very famous commercial summarizer.
3) Use our way without reducing redundancy.
4) Use our way of reducing redundancy.

From Figure 6 we can say that our method is the practical and best way when comparing with other three ways. We can see Random summaries perform poorly overall. And we can find use of our way to summary, the result is much better than random ways. But only use the initial way to summarize is not enough. So we continue to evaluate. We can clearly find if we use our way of reducing redundancy the result is much better than the initial way. And this way is much easier than MMR. We will compare the MMR with our method by evaluation in future work. Till now our system can achieve up to 72.6% precision.

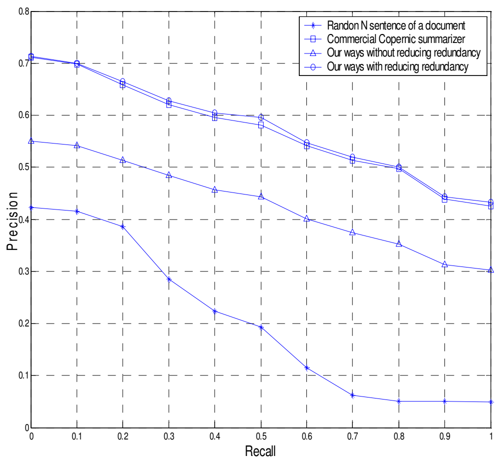


Figure 6. P-R curves.

So our way proved to be effective. Despite slightly outperform when comparing with copernic summarizer, we were able to observe some encouraging responses. We believe that we can train our system as more effective as possible because our way is promising.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a practical automatic text summarizer based on sentence extraction. Our method adds pre-processing to delete additional information that should not be included in the summarization before system parser, and change famous Salton's model by using the sum of similarity. The most important point is that our algorithm also deals with the initial summary to reduce the redundancy effectively by using the method very different from and easier than MMR. In future work, we intend to conduct experiments with larger and different document genres to improve our system. We also interested in the document abstracts based on Hidden Markov model (HMM). We plan to construct markov model with sentence structure. From this model we can get new sentence. Then get the abstract from the initial document, we think this way is promising.

6. REFERENCES

[1] A. M. Lam-Adesina and G. J. F. Jones. Applying summarization techniques for term selection in relevance feedback. Proceedings of the 24th ACM SIGIR, pages 1–9, 2001.

[2] Canasai Kruengkrai and Chuleerat Jaruskulchai. Generic Text Summarization Using Local and Global Properties of Sentences. Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference. Pages:201 – 206. (2003)

[3] D. Marcu. The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1997.

[4] G. Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. In I. Mani, M. Maybury (Eds.), Advances in automatic text summarization. MIT Press, 1999.

[5] H. P. Luhn. The automatic creation of literature abstracts. IBM Journal of Research and Development, pages 159–165, 1959.

[6] H. P. Edmundson. New methods in automatic extraction. Journal of the ACM, 16(2):264–285, 1969.

[7] I. Mani and M. Maybury. Advances in Automatic Text Summarization. MIT Press, 1999.

[8] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval, pages 335-336,1998.

[9] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. Proceedings of the 22nd ACM SIGIR, pages 121–128, 1999.

[10] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Seeing the whole in parts: Text summarization for web browsing on handheld devices. WWW10, May 2-5 2001.

[11] Salton, G., Singhal, A., Mitra, M., and Buckley, C. Automatic Text Structuring and Summarization. Advance in Automatic Text Summarization. Cambridge Massachusetts. MIT press, 341-355,1997 .

[12] http://www.tartarus.org/~martin/PorterStemmer/

[13] http://www.copernic.com/en/products/summarizer/

[14]http://www.hsl.creighton.edu/hsl/Searching/Recall-Precision.html

Automatically Building a Stopword List for an Information Retrieval System

Rachel Tsz-Wai Lo, Ben He, Iadh Ounis
Department of Computing Science
University of Glasgow
17 Lilybank Gardens
Glasgow, UK
lotr|ben|ounis@dcs.gla.ac.uk

ABSTRACT

Words in a document that are frequently occurring but meaningless in terms of Information Retrieval (IR) are called stopwords. It is repeatedly claimed that stopwords do not contribute towards the context or information of the documents and they should be removed during indexing as well as before querying by an IR system. However, the use of a single fixed stopword list across different document collections could be detrimental to the retrieval effectiveness. This paper presents different methods in deriving a stopword list automatically for a given collection and evaluates the results using four different standard TREC collections. In particular, a new approach, called term-based random sampling, is introduced based on the Kullback-Leibler divergence measure. This approach determines how informative a term is and hence enables us to derive a stopword list automatically. This new approach is then compared to various classical approaches based on Zipf's law, which we used as our baselines here. Results show that the stopword lists derived by the methods inspired by Zipf's law are reliable but very expensive to carry out. On the other hand, the computational effort taken to derive the stopword lists using the new approach was minimal compared to the baseline approaches, while achieving a comparable performance. Finally, we show that a more effective stopword list can be obtained by merging the classical stopword list with the stopword lists generated by either the baselines or the new proposed approach.

Keywords

Information Retrieval, information theory, stopword

1. INTRODUCTION

Two related facts were noticed in the early days of information retrieval by Luhn [13]. First of all, a relatively small number of words account for a very significant fraction of all text's size. Words like IT, AND and TO can be found in virtually every sentence in English-based documents. Secondly, these words make very poor index terms [3]. Users are indeed unlikely to ask for documents with these terms. Moreover, these words make up a large fraction of the text of most documents. According to Francis and Kucera [8], the ten most frequently occurring words in English typically account for 20 to 30 percent of the tokens in a document.

These words are said to have a very low discrimination value [16] when it comes to IR and they are known as stopwords or sometimes as noise words or the negative dictionary. In other words, the amount of information carried by these words is negligible. Consequently, it is usually worthwhile to ignore all stopword terms when indexing the documents and processing the queries.

By analysing the Brown corpus, Fox [7] derived a list of stopwords which contains all the obvious stopwords like THE, FOR, IS, AND, IT etc. This stopword list is referred to as the *classical stopword list* in this paper. However, each collection of documents is unique. It is therefore sensible to automatically fashion a different stopword list for different collections in order to maximise the performance of an IR system. Moreover, it is possible that the pattern of word occurrences has changed over the last 20 years, especially in the context of the Web. Therefore, the current classical stopword list might need to be updated.

In this paper, we introduce a new approach, called term-based sampling, inspired by the query expansion technique [19]. Using the Kullback-Leibler divergence measure [5], the new approach determines the amount of information a word contains [15]. The less information a word has, the more likely it is going to be a stopword. We evaluate our new term-based random sampling approach using various TREC collections. In addition, we compare the new term-based random sampling approach to four approaches inspired by Zipf's law [20], which are used here as our baselines.

The remainder of this paper is organised as follows: Section 2 introduces the baseline approaches. Section 3 discusses the new proposed approach, term-based sampling approach, in details. Section 4 gives the experimental setup and the

5th Dutch-Belgium Information Retrieval Workshop (DIR) '05 Utrecht, the Netherlands

evaluation approach. Section 5 discusses and analyses the results. Finally, Section 6 concludes the work and provides some possible future work, based on the findings of our experiments.

2. OUR BASELINE APPROACHES

George Kingsley Zipf (1902-1950) observed that the term's *rank-frequency* distribution can be fitted very closely by the relation:

$$F(r) = \frac{C}{r^{\alpha}} \tag{1}$$

where $\alpha \approx 1$ and C is ≈ 0.1. The above equation (1) is known as *Zipf's law* [20]. Our four baseline approaches are inspired by Zipf's law. Each collection is indexed, stemmed but no tokens are removed. This allows us to determine the best possible stopword list for a given collection. We use the general method, given in Table 1, to derive a stopword list based on Zipf's law.

Using the algorithm in Table 1, by substituting 'term frequency' with one of the following four refinements, different sets of stopword lists can be computed. Four refinements are used for this experiment because we do not know for certain which refinement would produce a better set of stopwords.

- Term frequency (TF) of the terms in the corpus: In other words, the number of times a certain term appears throughout a specific collection.
- Normalised term frequency: Normalising the term frequency (TF) by the total number of tokens in the collection (i.e. the size of the lexicon file). The calculation is straightforward and can be achieved using the following formula:

  $$TF_{Norm} = -log\left(\frac{TF}{v}\right) \tag{2}$$

  where TF is the term frequency of a particular term and v is the total number of tokens in the lexicon file.
- Inverse Document Frequency (IDF) [12]: Using the term frequency distribution in the collection itself where the IDF value of a given term k is given by:

  $$idf_k = log\left(\frac{NDoc}{D_k}\right) \tag{3}$$

  where $NDoc$ is the total number of documents in the corpus and D_k is the number of documents containing term k.

  In other words, infrequently occurring terms have a greater probability of occurring in relevant documents and should be considered as more informative and therefore of more importance in these documents.
- Normalised IDF: The most common form of IDF weighting is the one used by Robertson and Sparck Jones [14], which normalises with respect to the number of documents not containing the term $(NDoc - D_k)$ and adds a constant of 0.5 to both numerator and denominator to moderate extreme values:

  $$idf_{k\ Norm} = log\left(\frac{(NDoc - D_k) + 0.5}{D_k + 0.5}\right) \tag{4}$$

  where $NDoc$ is the total number of documents in the collection and D_k is the number of documents containing term k.

As shown in Table 1, a threshold needs to be determined. The aim is to find a threshold which allows generating a set of stopword list that would produce the best average precision. Choosing the threshold at random is not appropriate and is, therefore, not recommended. To choose a threshold, it is essential to investigate the frequencies difference between two consecutive ranks, namely $F(r)$ and $F(r+1)$. This is because if the difference between $F(r)$ and $F(r+1)$ is very high i.e. the term with frequency $F(r)$ occurs more often than the term with frequency $F(r+1)$, we could then choose that value as a threshold. In other words, any terms with frequencies $\geq F(r)$ could be used as a stopword.

The choice of the most appropriate threshold for a given collection is very important. If one too many words is considered to be a stopword, then there is a possibility that a relatively informative word has been omitted from the retrieval process, resulting in a lower retrieval effectiveness. On the other hand, if not enough words are considered to be stopwords, large amount of documents that are less specific to a given query would be retrieved and the average precision can also decrease. Thus, for a given collection, the general method of Table 1 requires finding the optimal associated threshold. In Section 4, we show how the thresholds could be set.

3. THE TERM-BASED RANDOM SAMPLING APPROACH

We introduce the term-based random sampling approach, which is based on how informative a particular term is. We could determine whether a specific term is a stopword based on its importance i.e. the less important a term is, the more likely it is a stopword. The importance of a term can be assessed using the *Kullback-Leibler divergence measure* [5].

Intuitively, the proposed approach is similar to the idea of query expansion [19] in IR in which we expand the given query based on a particular query term. The idea behind query expansion is to find terms that complement the initially chosen query term (or terms) motivated by the realisation that one term per concept might sometimes be inadequate to express a concept accurately - so we add terms (original query term(s) plus expanded terms). Our approach is based on query expansion with one difference. Instead of finding terms that have the same or similar meaning to a given term, we find all the documents containing this term and we use these documents as our sample. Then, we extract the least informative terms from the sample by measuring the divergence of a given term distribution within

```
Generate a list of term frequencies vs terms based on the corpus.
Sort the term frequencies listings in descending order, i.e. the terms with the highest term frequencies will be at the top.
        Rank the terms according to their term frequencies. The one with the highest term frequencies would have rank = 1 and the next most frequent term would have rank = 2, etc.
        Draw a graph of term frequencies vs ranks. This should obey Zipf's Law.
Choose a threshold and any words that appear above the particular threshold are treated as stopwords.
Run the querying with the above said stopword list, all the stopwords in the queries would be removed.
Evaluate the system after querying and note down the average precision.
```

Table 1: Algorithm for baseline approaches

the sampled document set from its distribution in the collection background. We then use the Kullback-Leibler (KL) divergence measure in order to determine each term's importance.

Similar to the baseline approaches, once stemmed, all tokens are kept in the lexicon file. This would once again enable us to deduce all possible stopwords for a given collection. Using the KL measure, the weight of a term t in the sampled document set is given by:

$$w(t) = P_x \cdot log_2 \frac{P_x}{P_c} \tag{5}$$

In the above formula, $P_x = tf_x/l_x$ and $P_c = \frac{F}{token_c}$, where tf_x is the frequency of the query term in the sampled document set, l_x is the sum of the length of the sampled document set, F is the term frequency of the query term in the whole collection and $token_c$ is the total number of tokens in the whole collection. The steps for the term-based random sampling approach is given in Table 2.

Selecting a random term in the lexicon file has the possibility of finding only one document containing that term which would result in a relatively small sample. This problem can be overcome by repeating the selection step Y times. Theoretically, if we run the selection step repeatedly, a better sample can be achieved, creating a better overview of the terms distribution and their importance. Note that this approach is simpler to implement than the baseline approaches, introduced in Section 2, even though the algorithm looks quite complex. This is because everything can be done automatically and therefore is less expensive to carry out. For the baseline approaches, we need to go through the $F(r) - F(r + 1)$ listing one by one, and we also need to fit the tf/idf vs rank graph to Zipf's law. Furthermore, there is no need to monitor the process unlike the baseline approaches, where the thresholds are required to be selected by going through the $F(r) - F(r + 1)$ listing.

The term-based random sampling approach is an alternate approach in determining stopwords. This approach is based on how informative a term is and since stopwords are terms that contain no information, this approach could enable us to determine a different set of stopwords, compared to the baseline approaches. Moreover, the term-based random sampling approach selects its first term randomly, hence enables us to have a better coverage of the whole collection. Since we have no idea what term it would select as its first term, for all we know, it could be a pre-defined stopword or it could be a term that is truly informative, for example, a term that occurs very infrequently in the collection.

4. EXPERIMENTAL SETUP

We experiment with four TREC collections[1], as shown in Table 3. Each TREC collection comes with a set of queries (See Table 4). Each query consists of three fields, namely Title, Description and Narrative. For all reported experiments, we used all three fields. The reason is that this would maximise our chances of using the generated stopwords, and hence enable us to assess the effectiveness of both the baselines and the proposed approaches.

Terrier[2] retrieval platform was used to conduct all of our experiments. Terrier is based on a divergence from randomness (DFR) framework for deriving non-parametric probabilistic models for IR. The PL2 model was used for all of the experiments. PL2 is one of the DFR document weighting models [1]. Using the PL2 model, the relevance score of a document d for query Q is given by:

[1]See http://trec.nist.gov/data/docs_eng.html for further information on the TREC collections
[2]Terrier is developed by the Information Retrieval Group at the University of Glasgow. Further information about Terrier can be found at: http://ir.dcs.gla.a.uk/terrier/

```
Repeat $Y$ times, where $Y$ is a parameter:
    Randomly choose a term in the lexicon file, we shall call it $\omega_{random}$
    Retrieve all the documents in the corpus that contains $\omega_{random}$
    Use the refined Kullback-Leibler divergence measure to assign a weight
    to every term in the retrieved documents. The assigned weight will give
    us some indication of how important the term is.
    Divide each term's weight by the maximum weight of all terms. As
    a result, all the weights are controlled within [0,1]. In other words,
    normalise each weighted term by the maximum weight.
    Rank the weighted terms by their associated weight in ascending order.
    Since the less informative a term is, the less useful a term is and hence,
    the more likely it is a stopword.
    Extract the top $X$ top-ranked (i.e. least weighted), where $X$ is a param-
    eter.
You now have an array of length $X * Y$. Each element in the array is associated to a weight.
Shrink the array by merging the elements containing the same term and take the average of the
term's associated weights. For example, if the term "retrieval" appears three times in the array
and its weights are 0.5, 0.4 and 0.3 respectively, we merge these three elements together into one
single one and the weight of the term "retrieval" will become
    $\frac{(0.5 + 0.4 + 0.3)}{3} = 0.4$
Rank the shrunk array in increasing order depending on the term's weight. In other words, sort
the array in ascending order.
Extract the $L$ top-ranked terms as stopword list for the collection. $L$ is a parameter. Therefore, it
is often a good idea to use trial and error.
```

Table 2: Algorithm for Term-based sampling approach

Collection	Size	Number of documents	Number of terms in the lexicon	c value
disk45 [17]	2GB	528155	801397	2.13
WT2G [10]	2GB	247491	1020277	2.75
WT10G [9]	10GB	1692096	3206346	2.43
DOTGOV [18]	18GB uncompressed	1247753	2821821	2.00

Table 3: Collections used for the experiment

Collection	Query Sets	Number of queries
disk45	TREC7 and TREC8 of ad-hoc tasks	100
WT2G	TREC8	50
WT10G	TREC10	50
DOTGOV	TREC11 and TREC12 merged	100

Table 4: Query sets used for the experiment

$$\begin{aligned} score(d,Q) &= \sum_{t\in Q} w(t,d) \\ &= \sum_{t\in Q} \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{tfn}{\lambda} + \\ &\quad (\lambda + \frac{1}{12 \cdot tfn} - tfn) \cdot \\ &\quad \log_2 e + 0.5 \cdot \log_2(2\pi \cdot tfn)\big) \qquad (6) \end{aligned}$$

where λ is the mean and variance of a Poisson distribution. $w(t,d)$ is the weight of document d for query term t.

The normalised term frequency tfn is given by the *normalisation 2*[2]:

$$tfn = tf \cdot \log_2(1 + c \cdot \frac{avg\_l}{l}), (c > 0) \qquad (7)$$

where l is the document length and avg_l is the average document length in the whole collection. tf is the original term frequency. c is the free parameter of the normalisation method, which can be different for different collections and is automatically estimated [11]. The c value used for each collection can be found in Table 3.

The control factor for this experiment is Fox's classical stopword list. Using both the baselines and the term-based sampling approach, we compared the derived stopword list to the classical stopword list, using average precision. For the baseline approaches, we varied the threshold values, hoping to find one particular set of stopword list that would produce a better average precision than the classical stopword list. Over 50 different sets of stopword lists were generated for each of the variants.

Since the term-based random sampling approach takes into account how informative a term is, the proposed algorithm is able to automatically formulate a list of stopwords that are likely to increase the average precision. The parameter X was set to 200, Y was set to 1000 while L was set to 400. Empirically, this setting was shown to achieve the optimal results, while the computational effort was kept within a reasonable limit.

Finally, we produced another new stopword list by merging Fox's classical stopword list with the best stopword list generated using either the baselines or the term-based random sampling approach. The merging process consists of adding the newly defined stopwords to the existing classical stopword list, removing any duplicates in order to ensure each term will occur once and once only in the new merged stopword list. The idea is that this merged stopword list might even be stronger in terms of effectiveness. This idea of merging follows directly from a classical IR technique of combining evidences, as discussed in [6]. In the next section, we compare the term-based random sampling approach to our baselines, and evaluate the effectiveness of the merged stopword list.

5. RESULTS AND ANALYSIS

In the following sections, we present the findings of our experiments. Section 5.1 discusses the obtained results using the variant baseline approaches. Section 5.2 presents the results using the new proposed approach. Finally, we present a further method for generating a more effective stopword list in Section 5.3. All of the statistical testing were done using the Wilcoxon Matched-Pairs Signed-Ranks Test.

5.1 Baseline Approaches - Overall Results

We generated over 50 stopword lists for each of the variants described in Section 2. The best obtained results are shown in Table 5. As indicated in this table, *normalised IDF* is the best variant of the approaches inspired by Zipf's law. This was expected, since IDF is more reliable than TF alone and normalised IDF is a refinement to IDF.

Table 5 also shows that in the case of the WT2G collection, the results obtained are the best, with the average precision improving by about 5% using normalised IDF. This is perhaps due to the fact that this collection is the smallest out of the four that we have used and the number of tokens in this collection is relatively small compared to the other collections. Results obtained using WT10G collection were also good. In WT10G, using the normalised IDF method results in an improvement of 4% average precision. The average precision for the DOTGOV and disk45 collection did very marginally improve using normalised IDF. Finally, it is worth mentioning some of the newly generated stopwords found using the baseline approaches for each collection (See Table 6). Notice that the terms *'http', 'html', 'web'* appear in every Web collection. As a result, we have successfully updated the stopword list with these terms.

5.2 Term-Based Random Sampling Approach - Overall Results

The overall results obtained using the proposed term-based random sampling approach can be seen in Table 7. The results obtained were not as positive as the baseline approaches nor as positive as we have hoped, especially with the DOTGOV and the WT10G collections. However, the computational effort is reduced significantly when compared to the baseline approaches. After investigation, we noticed that very few of the newly generated stopwords (the ones that were not already contained in the classical stopword list) occurred in the queries. However, as we will show in Section 5.3.2, if we merge the new generated stopword list with the classical stopword list, we can produce a more efficient stopword list for the system. Selected terms generated using the new proposed approach can be found in Table 8.

5.3 Refinement - Merging

We have shown that the classical stopword list was indeed very effective on its own, despite its old age. Moreover, most terms in the newly defined stopword lists were also in the classical one. Therefore, one can assume that if we merge the two lists together, then this should provide us with an even more effective stopword list for a given collection. In Section 5.3.1 we merge the classical stopword list with the stopword lists produced using normalised IDF while in Section 5.3.2, we merge the classical stopword list with the best

Collection	Classical	TF	Normalised TF	IDF	Normalised IDF	p-value
disk45	0.2123	0.2130	0.2123	0.2113	**0.2130**	0.8845
WT2G	0.2569	0.2650	0.2676	0.2682	**0.2700**	0.001508*
WT10G	0.2000	0.2049	0.2076	0.2079	**0.2079**	0.1231
DOTGOV	0.1223	0.1212	0.1208	0.1227	**0.1227**	0.55255

Table 5: Best average precision produced by each method for different collection, using baseline approaches, where Classical is Fox's stopword list. * indicates a significant difference between the stopword lists generated by normalised IDF and Fox's classical stopword list at 0.05 level.

disk45	WT2G	WT10G	DOTGOV
financial	html	able	content
company	http	copyright	gov
president	htm	ok	define
people	internet	http	year
market	web	html	administrate
london	today	january	http
national	policy	history	web
structure	content	facil	economic
january	web	html	year

Table 6: Selected terms derived using the baseline approaches for each collection used

Collection	Classical	Best Obtained	p-value
disk45	**0.2123**	**0.2129**	0.868
WT2G	**0.2569**	**0.2668**	0.07544
WT10G	**0.2000**	**0.1900**	0.4493
DOTGOV	**0.1223**	**0.1180**	0.002555*

Table 7: Average Precision for best results produced using term-based random sampling approach, where Classical is Fox's stopword list and Best Obtained is the best obtained stopword list using the parameters mentioned in Section 4. * indicates a significant difference between the stopword lists generated by the new proposed term-based random sampling approach and Fox's classical stopword list at 0.05 level.

disk45	WT2G	WT10G	DOTGOV
column	advance	copyright	server
general	beach	friend	modify
califonia	company	memory	length
industry	environment	mountain	content
month	garden	problem	accept
director	industry	science	inform
desk	material	special	connect
economic	pollution	internet	gov
business	school	document	byte

Table 8: Selected terms generated for each collection using the new proposed term-based random sampling approach

stopword list produced using the new approach, the term-based random sampling approach. As mentioned in Section 4, the merged stopword list was produced by adding all the terms from the newly defined stopword lists onto the classical stopword list and then removing all the duplicate terms so that each term appears once in the newly merged stopword list.

5.3.1 Baseline Approaches

A shown in Table 9, by merging the best obtained stopword lists with the classical stopword list, we have managed to provide the system with an even more effective set of stopwords, hence improving the average precision. In particular, we have managed to improve the system by almost 6% in the WT2G collection, 5.45% for the WT10G collection, 1.5% for the DOTGOV collection, respectively.

5.3.2 Term-Based Random Sampling Approach

Table 10 shows that by merging the best obtained stopword list with the classical stopword list, we also obtain some reasonable improvement. Comparing the results obtained using both the normalised IDF baseline and the new proposed approach, the results were more desirable when using our baseline approaches. Looking at the TREC queries, we noticed that these queries were relatively conservative and as a result, not all the stopwords found using the new proposed approach were in the queries. However, the computation effort required to obtain the stopword list using the proposed approach was minimal compared to our baseline approaches, suggesting that it is more easily used in an operational setting. Its balance between cost and effectiveness seems to be reasonably good compared to the baselines.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new method for automatically generating a stopword list for a given collection. The approach, called term-based random sampling approach, is based on how informative a given term is. We investigated the effectiveness and the robustness of this new approach using various standard collections. The new approach was compared to four variant baselines approaches inspired by Zipf's law. The results show that the proposed novel approach achieves a comparable performance to the baseline approaches, while having a lower computational overhead. In addition, using the proposed approach, the optimal threshold setting is easier to obtain. Moreover, the experimental results demonstrate that a more effective stopword list could be derived by merging Fox's classical stopword list with the stopword list produced by either the baselines or the proposed approach.

The KL divergence measure has been used to assess the informative amount of a given term in this paper. We plan to investigate other metrics such as the Poisson-based approach in [15], where documents are assumed to be independent rather than disjoint, or the divergence from randomness approaches proposed in [1]. Another area that requires further investigation is whether a verb or a noun are equally informative [4]. For example, consider the phrase *'I can open a can of tuna with a can opener'*, there are three occurrences of the term *'can'*. However, the first *'can'* is a verb whereas the latter two are nouns. Should we remove all the frequently occurring nouns and verbs as stopwords, the precision of the IR system would decrease considerably. The ideal Information Retrieval system stopword list would be able to take into accounts both verb and noun and their appropriate usage during indexing a collection.

7. ACKNOWLEDGMENTS

This work is funded by a UK Engineering and Physical Sciences Research Council (EPSRC) project grant, number GR/R90543/01. The project funds the development of the Terrier IR framework (http://ir.dcs.gla.ac.uk/terrier).

8. REFERENCES

[1] G. Amati. *Probabilistic Models for Information Retrieval based on Divergence from Randomness.* PhD thesis, Department of Computing Science, University of Glasgow, 2003.

[2] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389, 2002.

[3] R. K. Belew. *Finding Out About.* Cambridge University Press, 2000.

[4] S. Charkrabarti. *Mining the Web: Discovering knowledge from hypertext.* Morgan Kaufmann, 2003.

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley, 1991.

[6] W. B. Croft. Combining approaches to information retrieval. In *Advances in Information Retrieval - Recent Research from the Center for Intelligent Information*, pages 1–28. Kluwer Academic Publishers, 2000.

[7] C. Fox. Lexical analysis and stoplists. In *Information Retrieval - Data Structures & Algorithms*, pages 102–130. Prentice-Hall, 1992.

[8] W. Francis. *Frequency Analysis of English Usage: Lexicon and Grammar.* Houghton Mifflin, 1982.

[9] D. Hawking. Overview of the TREC2002. In *Proceedings of the Nineth Text REtrieval Conference (TREC 9)*, pages 87–94, Gaithersburg, MD, 2000.

[10] D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the TREC-8 Web Track. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 131–150, Gaithersburg, MD, 1999.

[11] B. He and I. Ounis. A study of parameter tuning for term frequency normalization. In *Proceedings of the Twelfth ACM CIKM International Conference on Information and Knowledge Management*, pages 10–16, New Orleans, LA, USA, 2003.

[12] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[13] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

Collection	Classical	Normalised IDF	Merged Stopword List	p-value
disk45	0.2123	0.2130	**0.2130**	0.8845
WT2G	0.2569	0.2700	**0.2712**	0.000746*
WT10G	0.2000	0.2079	**0.2109**	0.03854*
DOTGOV	0.1223	0.1227	**0.1241**	0.6775

Table 9: Average Precision for Merged Stopword List using baseline approaches, where Classical is Fox's stopword list. * indicates a significant difference between the stopword lists generated by the merged stopword list using normalised IDF and Fox's classical stopword list at 0.05 level.

Collection	Classical	Best Obtained	Merged Stopword List	p-value
disk45	0.2123	0.2129	**0.2129**	0.868
WT2G	0.2569	0.2668	**0.2703**	0.008547*
WT10G	0.2000	0.1900	**0.2066**	0.4451
DOTGOV	0.1223	0.1180	**0.1228**	0.5085

Table 10: Average Precision for Merged Stopword List using term-based random sampling approach, where Classical is Fox's stopword list and Best Obtained is the best obtained stopword list using the parameters mentioned in Section 4. * indicates a significant difference between the stopword lists generated by the merged stopword list using term-based random sampling approach and Fox's classical stopword list at 0.05 level.

[14] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[15] T. Roelleke. A frequency-based and a Poisson-based definition of the probability of being informative. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 227–234, Toronto, Canada, 2003.

[16] C. J. van Rijsbergen. *Information Retrieval, 2nd edition.* Butterworth-Heinemann, 1979.

[17] E. Voorhees and D. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 1–23, Gaithersburg, MD, 1999.

[18] E. M. Voorhees. Overview of TREC2002. In *Proceedings of the Eleventh Text REtrieval Conference (TREC2002)*, pages 1–16, Gaithersburg, MD, 2002.

[19] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, Zurich, Switzerland, 1996.

[20] H. Zipf. *Human Behaviours and the Principle of Least Effort.* Addison-Wesley, Cambridge, MA, 1949.

Towards Automatic Formulation of a Physician's Information Needs [*]

Loes Braun
Institute for Knowledge and Agent Technology
P.O.Box 616
6200 MD Maastricht,
The Netherlands
L.Braun@cs.unimaas.nl

Floris Wiesman
Academic Medical Center Amsterdam
P.O. Box 22700
1100 DE Amsterdam,
The Netherlands
F.J.Wiesman@amc.uva.nl

Jaap van den Herik
Institute for Knowledge and Agent Technology
P.O.Box 616
6200 MD Maastricht,
The Netherlands
Herik@cs.unimaas.nl

ABSTRACT
The goal of this paper is to contribute to the improvement of the quality of care by providing physicians with patient-related literature. For physicians, it is a problem that they are often not aware of gaps in their knowledge and the corresponding information needs. Our research aim is to resolve this problem by formulating information needs automatically. Based on these information needs, patient-related literature can be retrieved. In this paper, we investigate how to model a physician's information needs. Thereafter, we design and analyze an approach to instantiate the model with patient data, resulting in information-need templates that are able to represent patient-related information needs. Since the patient data are in Dutch, we developed a translation mechanism for translating Dutch terms into the standard English terminology. From our experiments it is clear that a physician's information needs can be modelled adequately and can be substantiated into patient-related information needs. As an aside, we have shown that the automatic translation mechanism is not sufficiently effective in comparison with a manual translation mechanism. The usability of our information-need formulation approach is demonstrated by performing a literature retrieval that based on the formulated information needs. Since the number of formulated information needs is rather high, methods have to be developed that restrict the set of automatically formulated information needs to a more specialized set.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process*; J.3 [**Life and Medical Sciences**]: Medical Information Systems

[*]Additional authors: Arie Hasman (Academic Medical Center Amsterdam) and Erik Korsten (Catharina-ziekenhuis Eindhoven)

General Terms
Languages, standardization

Keywords
Information retrieval, information need, electronic patient record

1. INTRODUCTION
In our research, we investigate how to support physicians in the information-retrieval (IR) process, so as to improve the quality of care. We start with an example that precisely illustrates the need for patient-related literature.

An 84-year-old woman was brought into the emergency department of a hospital, suffering from dyspnea and loss of consciousness. Five days earlier she had visited her general practitioner who diagnosed her with suspected respiratory tract infection and prescribed a drug called Clarithromycin. However, instead of improving, her condition worsened. In the hospital the diagnosis pneumonia was considered and she was treated accordingly, but without any effect. Upon her family's request, the patient was not admitted to the intensive care unit and she died one day after she was admitted to the hospital. Surprisingly, an autopsy revealed that the cause of death was not pneumonia, but a case of severe acute pancreatitis. The autopsy also revealed that the most plausible cause for the pancreatitis was the use of Clarithromycin, since pancreatitis is a (rare) side effect of the use of Clarithromycin [22].

Since the incidence of Clarithromycin-induced pancreatitis is quite low, it is understandable (but still undesirable) that the physician in the example above was not aware of this possible side effect. If the physician had performed a literature search in Medline on the side effects of Clarithromycin, he[1] probably would have found an article by Leibovitch, Levy, and Shoenfeld [11], in which another case of Clarithromycin-induced pancreatitis is discussed. If he had read this article, he probably would have ordered additional diagnostic tests to exclude pancreatitis (e.g., blood amylase) and he could have started the appropriate treatment immediately.

[1]For brevity we will use the pronoun 'he' ('his') where 'he or she' ('his or her') is meant.

Table 1: Number of information needs identified by a literature survey and interviews.

Identification method	Source	Identification domain	#INs identified
Literature survey	[2]	Outpatient care, inpatient care, internal medicine	16
	[3]	General practice, cardiology, pulmonology, allergology	77
	[5]	Family care	10
	[7]	Primary care	16
	[8]	Various	32
	[10]	Various	10
	[21]	Surgical care	2
	[23]	Primary care	8
Interviews		Anesthesiology	2
		Cardiology	1
		Neurology	0
		Pulmonology	3
		Surgery	3

The reason why the physician did not perform a literature search is twofold. First, the physician was not aware of the fact that he needed information on the side effects of Clarithromycin. So, he had no incentive to search for information on the topic. Hence, we may conclude that in this case there was an implicit information need. Second, even if the physician had been aware of his information need, he probably would not have had the time to perform a proper search action. The rapid growth of medical information sources and the complexity of the information spaces would impose too large a burden on a physician to retrieve information relevant to the specific patient.

The example above clearly illustrates that the retrieval of relevant, patient-related literature is vital to the quality of care (cf. [6]). Various articles discuss IR systems that provide such literature (e.g., [12, 20, 25]); our research concurs with these articles. However, in our opinion the overall shortcoming of the systems mentioned in the articles above is that the degree of necessary interaction with the systems is too high. This is especially true in the area of making information needs explicit. Therefore, our main research objectives are (1) to investigate to what extent a physician's implicit information needs can be made explicit automatically, and (2) to implement our approach into a computer system that supports a physician in his daily work by providing him with patient-related literature based on his information needs. We consider literature as patient-related if it is relevant to a physician with respect to (a) the specific patient he is currently treating and (b) his current activities concerning the patient.

The article describes our approach in making a physician's implicit information needs explicit automatically. In section 2 we describe how we determine a physician's information needs and how we model these needs. Section 3 presents our approach to formulate information needs regarding a specific patient, based on the patient data in the electronic patient record (EPR). In section 4 we discuss two translation mechanisms and choose one of them for the translation of medical terms. Section 5 demonstrates the usefulness of the information-need formulation approach by retrieving literature based on the formulated information needs. Section 6 provides our conclusions.

2. MODELLING INFORMATION NEEDS

Our approach to make a physician's information needs explicit resides in anticipating them. As a starting point for this process, we need a set of a physician's potential information needs. However, such a set can never be complete, since it is impossible to capture all of a physician's information needs. Moreover, a physician generates new information needs over time, which should regularly be added to the set. This is hard to facilitate.

One solution is to build a *model* of a physician's information needs. As long as the model represents information needs on a more abstract level it can be considered complete, meanwhile anticipating future information needs. Modelling a physician's information needs involves two steps described below: (1) identifying a physician's information needs (subsection 2.1) and (2) abstracting the identified information needs (subsection 2.2).

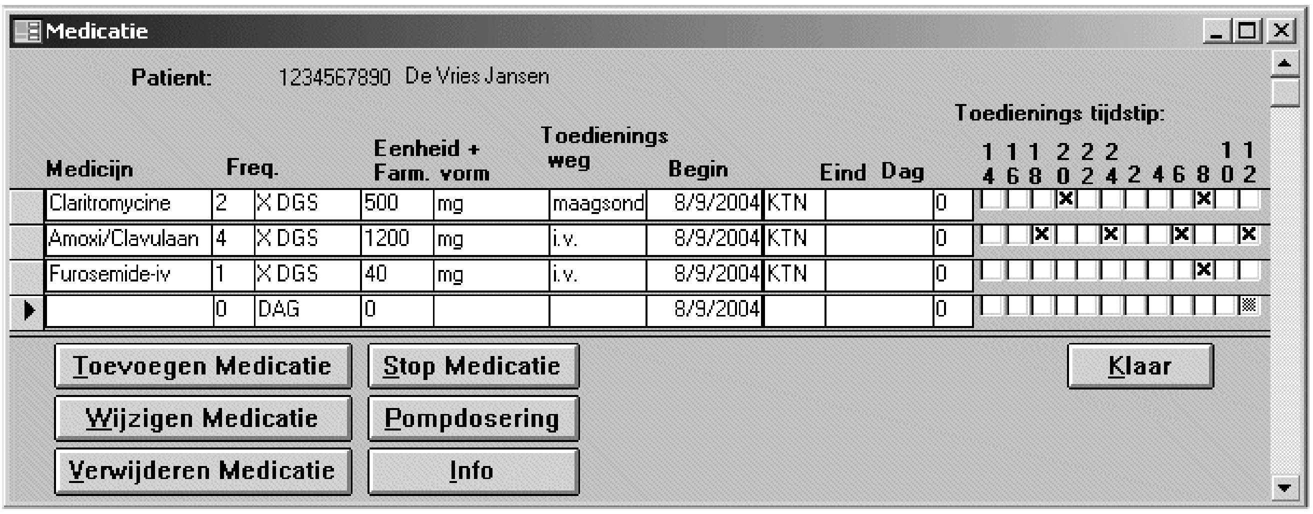
2.1 Identifying Information Needs

To identify a physician's information needs, we used two methods, viz. (1) a literature survey and (2) interviews. Both identification methods are briefly described below. Table 1 summarizes the sources, the identification domains, and the number of information needs identified.

In our literature survey, we searched for articles presenting information needs that are general, i.e., not specific for a particular group of physicians or for a particular geographical area. We found only eight such articles. This set of articles covered a large number of medical domains from which the information needs were identified. In total we arrived at 171 information needs.

To obtain a set of information needs that is as diverse as possible, we succeeded in interviewing five physicians in five different medical specialisms: (1) anaesthesiology, (2) cardiology, (3) neurology, (4) pulmonology, and (5) surgery. The physicians were interrogated by means of an interview scheme composed in advance. This led to 9 information needs.[2]

[2]Since we have to search English literature and several information needs were in Dutch, we had to translate the Dutch



Patient: 1234567890 De Vries Jansen

Medicijn	Freq.		Eenheid + Farm. vorm		Toedienings weg	Begin		Eind	Dag	14	16	18	20	22	24	2	4	6	8	10	12
Claritromycine	2	X DGS	500	mg	maagsond	8/9/2004	KTN		0	☐	☐	☐	☒	☐	☐	☐	☐	☐	☒	☐	☐
Amoxi/Clavulaan	4	X DGS	1200	mg	i.v.	8/9/2004	KTN		0	☐	☐	☒	☐	☐	☒	☐	☐	☒	☐	☐	☒
Furosemide-iv	1	X DGS	40	mg	i.v.	8/9/2004	KTN		0	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒	☐	☐
▶	0	DAG	0			8/9/2004			0	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	▨

Figure 1: Interface of the Intensive Care Information System, showing a patient's medication.

2.2 Abstracting Identified Information Needs

The identified information needs are highly context-dependent, which may render them useless in another (different) context. To reduce context dependency, we abstracted the information needs, so as to make them context independent. For the abstraction we used an approach similar to the one used by Ely, Osheroff, and Ebell [5]. Hence, we replaced each medical concept in the information needs by its semantic type, which is a high-level description of the medical concept (e.g., the concept 'Pneumonia' has the semantic type DISEASE OR SYNDROME). We obtained the semantic types of the concepts from the Semantic Network of the Unified Medical Language System (UMLS) that comprises 135 types [18].

The abstraction resulted in a general class of information needs, called information-need templates, which are abstract, natural-language representations of information needs containing data slots which can be filled with medical concepts (e.g., What are the side effects of [DISEASE OR SYNDROME]?). Some information needs resulted in the same information-need template. For example, Does Norpace cause fatigue? and Does Clarithromycin cause high blood pressure? both resulted in the information-need template Does [CHEMICAL] cause [SIGN OR SYMPTOM]? To obtain a proper set of information-need templates, we removed all doubles. Currently, the set comprises 167 information-need templates.

3. CONVERTING TEMPLATES

To determine a physician's information needs, we have to convert information-need templates into information needs. Since we want to determine a physician's information needs with respect to a specific patient, the patient's medical data play a crucial role in the process. In this section we describe how we convert information-need templates into patient-related information needs (subsection 3.1). Thereafter, subsection 3.2 describes experiments and presents the results. In subsection 3.3 the experimental results are discussed.

information needs into English.

3.1 Conversion Methodology

To formulate a patient-related information need, an information-need template has to be instantiated with the patient's medical data. The data are acquired from the EPR of the specific patient. An EPR stores all medical data of a patient digitally (e.g., diagnosis, medication, and treatment). In our research we used the Intensive Care Information System (ICIS), used at the Intensive Care Unit of the Catharina-ziekenhuis in Eindhoven.[3] As can be seen from figure 1, we have maintained the Dutch terminology since it is used in practice.

Our approach of converting an information-need template into a patient-related information need comprises three steps described below.

1. Select EPR-queries for retrieving the appropriate patient data[4] in the EPR,
2. Execute the selected EPR-queries: the desired data are retrieved from the EPR, and
3. Instantiate the information-need template with the retrieved patient data.

In the first step, selecting the suitable EPR-queries, we start determining which semantic types occur in the information-need template. To convert the information-need template into an information need, each of these semantic types has to be instantiated with patient data. Consequently, an EPR-query has to be selected for each semantic type in the information-need template. The EPR-queries are selected from a list of predefined EPR-queries. Each EPR-query in this list specifies how to find the patient data associated with

[3]Intensive Care Informatie Systeem, Version 2.8. INAD Computers & Software B.V. Eindhoven, Werkgroep ICIS Afd. Intensive Care, Dienst Informatie Voorziening, Catharina-ziekenhuis Eindhoven.

[4]All patient data used in the examples of section 3 are fictitious for reasons of privacy.

the corresponding semantic type. Some high-level semantic types have subtypes, which should also be instantiated. We use the hierarchy within the UMLS Semantic Network to determine the subtypes of the high-level semantic types. For them queries are selected as well. Assume we have the following template:

What are the side effects of [CHEMICAL]?

Based on (1) the semantic type CHEMICAL, (2) the information structure of our EPR, and (3) the patient number of the specific patient, the EPR-query below is selected (the names of the database tables are in Dutch). Since the semantic type CHEMICAL has many subtypes (e.g., ORGANIC CHEMICAL and PHARMACOLOGIC SUBSTANCE), more EPR-queries are selected (viz. one query for each subtype), but for brevity, they are not presented here.

```
SELECT Medicijn FROM Medicatie WHERE PatientNummer=
1234567890
```

The list of EPR-queries is only applicable to our specific EPR, since EPRs from other EPR-systems have different information structures. To adapt our approach to another EPR-system, new EPR-queries have to be formulated. To facilitate easy adaptation, all potential EPR-queries for a specific EPR-system are specified in a model, which is runtime consulted by the system and can be easily reformulated.

The second step is to execute the selected queries to extract the desired patient data from the EPR. The actual query-execution process is handled by the database itself. Each result that an EPR-query returns for a semantic type is called an active concept of that specific semantic type. Assume that our patient is taking three different medications (see figure 1). Then, our EPR-query has three results and consequently, the semantic type CHEMICAL has three active concepts, viz. (1) Clarithromycine, (2) Amoxi/Clavulaan, and (3) Furosemide-iv. There are two difficulties. First, the terms resulting from the EPR-query are in Dutch, whereas the information needs to be formulated have to be in English, because our system searches English literature databases. Consequently, the terms have to be translated. Second, our EPR does not solely use standardized terms, making translation of the terms more difficult. To overcome these difficulties, the UMLS Metathesaurus [15] was used to achieve translation of the patient data (see section 4).

The third step is to instantiate the information-need template with the data obtained from the EPR (the active concepts of the semantic types). We call an information-need template applicable (i.e., it can be instantiated with patient data) if each semantic type within the information-need template has one or more active concepts. The template is instantiated by systematically replacing each semantic type by one of its active concepts, for each possible combination. The total number of resulting information needs is the product of the numbers of instances of all semantic types in the template. For three instances (which are currently in Dutch) of the semantic type CHEMICAL, our information-need template is instantiated three times.

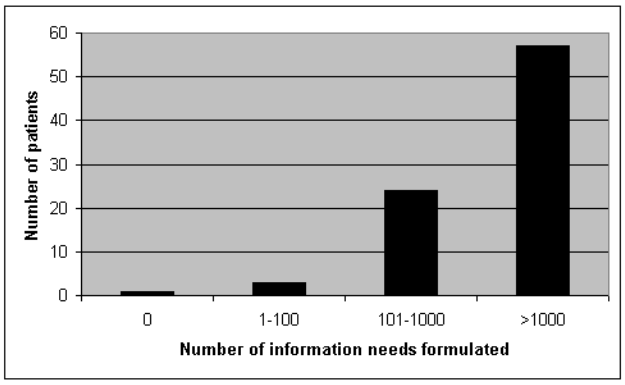


Figure 2: Number of patients for which a specific number of information needs is formulated.

- What are the side effects of Clarithromycine?
- What are the side effects of Amoxi/Clavulaan?
- What are the side effects of Furosemide-iv?

If a literature search were conducted based on the above information needs, patient-related literature would be found. If not all semantic types have instances, the information-need template is not applicable and consequently it is not instantiated.

3.2 Experiments and Results

To establish the feasibility of our approach for instantiating information-need templates, we let our system formulate information needs based on the EPRs of 85 patients. Each EPR contained information about all hospital admissions (in the hospital under consideration) of a specific patient. We used our complete set of 167 information-need templates. The list of predefined EPR-queries comprised 8 queries. For each patient, our experiment resulted in a set of information needs. We divided the number of formulated information needs into four categories, viz. (i) no information needs, (ii) a manageable number of information needs, (iii) a hardly manageable number of information needs, and (iv) and unmanageable number of information needs. We placed each patient in one of these categories, based on the number of formulated information needs associated with the patient (see figure 2).

As can be seen in figure 2, the total number of information needs is quite high for most patients. If a literature search were conducted for all these information needs, the set of retrieved literature would be unmanageably high. Since we do not want to overload physicians with literature, the number of information needs for which literature is retrieved should be restricted. In the ideal situation, all patients would be in the 1-100 information needs category.

3.3 Discussion

From subsection 3.2 it is clear that the number of automatically formulated information needs per patient is still (too) high. Since a high number of information needs will lead to a large amount of retrieved literature, physicians would be overloaded with information. So, our approach is not yet sufficiently adequate. To improve the approach, the number of information needs for which literature is retrieved should be reduced.

In practice, information needs will not be formulated by considering all patient data anew. They will be continuously reformulated after every change in the EPR. Only information-need templates which are related to the changed entry will be instantiated. Consequently, the number of information needs which is formulated at one time will be reduced considerably.

A way to reduce the total number of formulated information needs any further is by taking two additional parameters into account. The first parameter is the *usefulness of the patient data.* In our experiment, data from all hospital admissions (in the hospital under consideration) were used to generate information needs, whereas only patient data relevant to the patient's current problem are important. Therefore, it is essential to distinguish between useful and non-useful data. In general, the current admission will provide the most useful data. However, not all data from the current admission might be useful. For example, when a physician has already selected chemotherapy as the appropriate treatment for a lung-cancer patient, he may be assumed not to have information needs concerning the selection of a treatment, e.g., what is the treatment for lung cancer for this patient? Yet, he might have information needs concerning the execution of the selected treatment, e.g., how high is the dose of chemotherapy for lung cancer for this patient? The second parameter is the *specialism of the physician.* Since a physician's information needs are probably connected to his specialism, we might ignore several information needs, because they are not linked to the physician's specialism. Ignoring the information needs is solely based on the patient data with which the corresponding information-need templates were instantiated. As information-need templates contain no patient data, templates cannot be ignored in advance. A series of future experiments may clarify whether the two parameters are appropriate filters for the number of information needs.

4. DATA TRANSLATION

As mentioned in subsection 3.1, the EPR data have to be translated and standardized to obtain proper information needs. Two approaches for translating our EPR are envisaged, viz. manual and automatic translation. For both approaches, the use of the UMLS Metathesaurus (subsection 4.1) is essential. It is important that the translation approach is both effective and scalable. Manual translation is highly effective, since it will produce many correctly classified terms. However, this approach seems not to be scalable; for instance, a translation which is developed specifically for our EPR will probably not be useful in other EPRs. In comparison with manual translation, automatic translation is less effective for small sets; it will classify fewer terms correctly. However, an automatic translation approach is more scalable than a manual approach, since it might be useful in other domains as well. To select a translation approach, we had to choose between effectiveness and scalability. In our opinion, effectiveness is more important than scalability, but if an automatic translation mechanism could be developed which is sufficiently effective, automatic translation might be preferred, since the approach is scalable as well.

To make a comparative assessment, we constructed two translation mechanisms, one based on the manual approach and the other based on the automatic approach. Both mechanisms are presented in subsection 4.2. Subsection 4.3 describes our experiments with both translation mechanisms and chooses the best-performing one as the translation mechanism to be used. In subsection 4.4 we discuss how the translation mechanism is incorporated into the process of automatic formulation of information needs.

4.1 The UMLS Metathesaurus

The Unified Medical Language System (UMLS) [15], which is developed at the National Library of Medicine (NLM), facilitates the development of software uses the language of biomedicine and health. The UMLS comprises Knowledge Sources and software tools, which can be used in medical information system design or research [19]. The UMLS Knowledge Sources comprise (a) the Metathesaurus, (b) the Semantic Network, and (c) the SPECIALIST lexicon. For our purpose, the UMLS Metathesaurus is most important.

The UMLS Metathesaurus[5] provides information about over 1 million biomedical concepts and 4.3 million concepts names. These concepts and concept names are taken from more than 100 controlled vocabularies and classifications (in 15 languages) used in the biomedical domain. The Metathesaurus also provides mappings between these concepts [17].

4.2 Translation Mechanisms

The manual translation mechanism consisted of manually mapping all terms from the EPR-system (a collection of 1583 different terms) onto UMLS concepts. We were able to map 72.5% of the terms. The remaining 27.5% of the terms (e.g., *geen plaats afd* = no free bed in unit) could not be mapped because they had no corresponding UMLS concepts. The mapping was stored in a database. To translate a Dutch term, the corresponding UMLS concept is looked up in the database and translated into English via the UMLS Metathesaurus.

Our automatic translation mechanism was based on Tran *et al.* [24] and Deville [4]. Tran *et al.* discuss a five-step method for preprocessing French terms so that they can be translated more easily by means of a medical thesaurus (e.g., the UMLS Metathesaurus). Deville presents rules for translating suffixes of medical terms into English to improve translation performance.

We combined the methods by Tran *et al.* and Deville into a three-step procedure for preprocessing.

1. Normalization
2. Lexical transformation
3. Morphosyntactical transformation

[5]We used release 2004AB.

These three steps have to be performed in the order in which they are described.

Normalization comprises the removal of punctuation marks and the convertion of uppercase characters to lower case. For example, the term *Clarithromycine-po* is transformed into *clarithromycine po*.

Lexical transformation comprises the application of several domain-specific heuristics. These heuristics remove unnecessary extensions and writes abbreviations in their complete form. For example, *clarithromycine po* is transformed into *clarithromycine per os* (*per os* = oral). In total we constructed 14 lexical heuristics.

Morphosyntactical transformation comprises the application of domain-specific heuristics to transform suffixes of terms (we have used a similar approach in a different domain [1]). For example, *clarithromycine per os* is transformed into *clarithromycin per os*. The construction of our set of morphosyntactical heuristics is based on Jensen [9]. In total we constructed 128 morphological heuristics.

After the medical terms are preprocessed by the abovementioned procedure, they are translated by the UMLS Metathesaurus.

4.3 Experiments and Results

To compare the manual and the automatic translation mechanisms we extracted a collection of 3,349 terms from the EPRs of 85 patients. All terms had one of the three semantic types which are most used in our information-need templates, viz. [CHEMICAL], [THERAPEUTIC OR PREVENTIVE PROCEDURE], and [DISEASE OR SYNDROME]. Since many terms occurred in the collection more than once, the doubles were removed to obtain a proper set of terms. The resulting set consisted of 347 different terms. One half of the set, the *tuning set* (173 terms), was used for tuning the automatic translation mechanism, whereas the other half, the *test set* (174 terms), was used to compare the two mechanisms.

We tuned the automatic translation mechanism by assessing its performance in four different configurations, to assess the contribution of each consecutive step in combination with UMLS.

(a) UMLS (no preprocessing methods)

(b) Normalization + UMLS

(c) Normalization + lexical transformation + UMLS

(d) Normalization + lexical transformation + morphosyntactical transformation + UMLS

We applied all four configurations on three different subsets of the UMLS Metathesaurus, viz. (i) the Dutch Metathesaurus, (ii) the English Metathesaurus, and (iii) the complete (multi-language) Metathesaurus. This resulted in 12 test runs.

We used two different matching methods incorporated in the UMLS software tools, viz. *ExactMatch* and *ApproxMatch*.

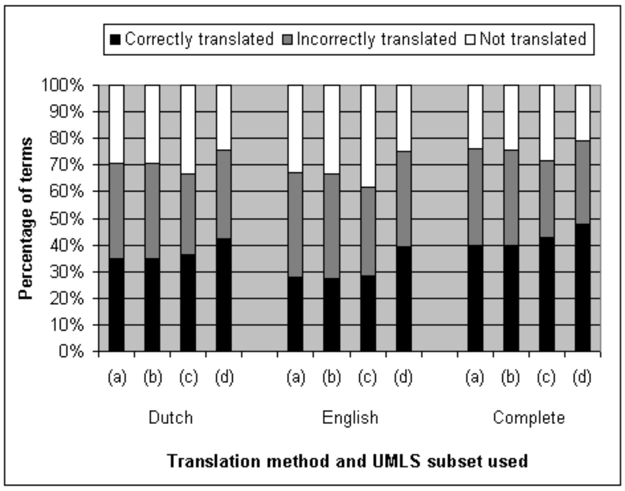


Figure 3: Percentage of correctly, incorrectly, and non-translated terms with respect to four different translation mechanisms (characters refer to the description in the text) and three different UMLS subsets.

During each test run we first attempted to match each term (from the tuning set) exactly with a term from the Metathesaurus, by using *ExactMatch*. The terms which were not matched exactly were thereafter matched approximately, by using *ApproxMatch*. The resulting translations were checked manually for correctness.

The results of the experiments are presented in figure 3. The figure shows that for each of the four types of configurations (a, b, c, and d), the specific configuration which was applied to the complete UMLS Metathesaurus outperforms its two single-language counterparts which were applied to the single-language subsets of the UMLS Metathesaurus. When comparing the four configurations which were applied to the complete UMLS Metathesaurus among each other, the figure shows that configuration (d) has the highest percentage of correctly classified terms. Moreover, configuration (c) has the lowest percentage of incorrectly classified terms. Furthermore, configuration (d) has the lowest percentage of non-classified terms. Since configuration (d) outperforms configuration (c) on two of the three performance criteria, we chose to use configuration (d) as our mechanism for automatic translation of medical terms.

To determine whether the automatic translation mechanism was sufficiently effective in comparison to the manual translation mechanism, we executed both mechanisms on the test set. As mentioned above, we used all three preprocessing procedures applied to the complete UMLS Metathesaurus as our automatic translation mechanisms. The manual translation mechanism simply consisted of looking up UMLS concepts in the database containing our manually constructed mapping and translating the concepts to English via UMLS.

The results of the experiment are shown in table 2. As can be seen from this table, the automatic translation mechanism translated 47.1% correctly, whereas the mechanism

Table 2: Evaluation results of the automatic and manual translation approach.

Evaluation criterion	Automatic translation	Manual translation
Correctly translated terms	47.1%	73.6%
Incorrectly translated terms	33.9%	NR
Non-translated terms	19.0%	26.4%

Note: NR stands for *not relevant.*

using our manually constructed translation obtained 73.6% correctly translated terms. Moreover, the automatic translation mechanism translated 33.9% of the terms incorrectly. On the contrary, our manual translation mechanism never translated terms incorrectly, since we only mapped terms to UMLS concepts when we were certain of the correctness of the mapping. Furthermore, the automatic translation mechanism was not able to translate 19.0% of the terms, whereas our manual translation mechanism was not able to translate 26.4% of the terms.

The experiments showed that the manual mechanism is most effective on two of the three evaluation criteria (correctly and incorrectly translated terms). The automatic translation approach is most effective with respect to only one evaluation criterion: non-translated terms. So, the automatic mechanism was not sufficiently effective in comparison with our manual approach. Despite the superior scalablity of the automatic translation mechanism we chose to use our manual translation mechanism to map terms from the EPD-system to UMLS concepts, since we consider effectiveness more important than scalability.

4.4 Incorporating a Translation Mechanism

To use our translation methodology, we have to incorporate it into the information-need translation process. In subsection 3.1 we explained that the process comprises three steps. We incorporated the term translation between step (2) and (3) of the process. Consequently, the complete information-need formulation process now comprises four steps described below.

1. Select EPR-queries for retrieving the appropriate patient data in the EPR,
2. Execute the selected EPR-queries: the desired patient data are retrieved from the EPR,
3. Translate the retrieved patient data, and
4. Instantiate the information-need templates with the translated patient data.

By means of this four-step process, English information needs can be formulated automatically from Dutch patient data.

5. LITERATURE RETRIEVAL

To make automatic formulation of information needs useful for physicians, the formulated information needs have to be used as a starting point for literature retrieval. In our research we used PubMed [16] as our information source. PubMed is a medical database, developed by the National Center for Biotechnology Information (NCBI) at the National Library of Medicin (NLM). PubMed provides access to various information sources [14]. The most important source for our purpose is Medline, which is a database containing references to over 12 million biomedical journal articles. These articles are taken from over 4,600 journals in 30 different languages (from 1966 until present) [13]. We chose PubMed as our information source since it is extensively used by physicians.

Literature retrieval from PudMed is normally conducted via the PubMed website [16]. However, in our approach, we wanted to retrieve literature via our own application, without the use of the website. To achieve this task, we used NLM's Entrez Programming Utilities. These tools provide access to all NLM data sources without the use of the regular web interface. To retrieve literature from PubMed, it suffices to pass the formulated information need as a parameter to the Entrez Programming Utilities, which reformulates the information need into a straightforward query. Documents retrieved with respect to the information needs are returned automatically. Future experiments with physicians will be conducted to assess the relevance of the retrieved literature and compare it to the relevance of the documents manually found by physicians.

6. CONCLUSIONS

The cooperation with physicians, described in subsection 2.1, showed that we succeeded in identifying a physician's potential information needs and in modelling them into 167 information-need templates by using 135 semantic types (subsection 2.2).

In section 3 we designed an approach to convert information-need templates into patient-related information needs by taking patient data into account. From the experiments, we may conclude that our approach is adequate and can be generalized to other EPR-systems, as long as they use a clear information structure. The number of automatically formulated information needs per patient is still high (subsection 3.2), but may be reduced by taking the *usefulness of the patient data* and the *specialism of the physician* into account.

In section 4.2 two translation mechanisms are described. From section 4.3 we may conclude that in our case, manual translation of the patient data is superior to automatic translation. Subsection 4.4 showed that the translation step can be incorporated quite easily into the process of information-need formulation.

Section 5 demonstrates that the information needs formulated by our approach can be used as a starting point for a literature search, providing a physician with relevant and patient-related medical literature. Therefore, we may conclude that our approach for automatically determining a physician's information needs and retrieving documents based on these information needs is useful.

7. ACKNOWLEDGEMENTS

This research is part of the MIA project (Medical Information Agent), which belongs to the research program ToKeN

(Toegankelijkheid en Kennisontsluiting in Nederland). The MIA project (as well as the complete ToKeN program) is funded by NWO (grant number 634.000.021).

8. REFERENCES

[1] L. Braun, F. Wiesman, and I. Sprinkhuizen-Kuyper. Information retrieval from historical corpora. In D. Hiemstra, W. Kraaij, and M.-F. Moens, editors, *Proceedings of the 3rd Dutch Information Retrieval Workshop (DIR2002)*, pages 106–112, 2002. Leuven, Belgium.

[2] R. J. Cucina, M. K. Shah, D. C. Berrios, and L. M. Fagan. Empirical formulation of a generic query set for clinical information retrieval systems. In V. Patel, R. Rogers, and R. Haux, editors, *Proceedings of the MedInfo2001, London, United Kingdom*, pages 181–185, Amsterdam, The Netherlands, 2001. IOS Press.

[3] P. F. de Vries Robbé, W. P. A. Beckers, and P. E. Zanstra. *MEDES. Het prototype*. Onderzoeksgroep Medische Informatie- en Besliskunde, Academische Ziekenhuis Groningen, Groningen, The Netherlands, 1988.

[4] G. Deville. Corpus-based sublanguage modelling for NLP applications: a tentative methodology. In *Proceedings of the International Colloquium on Trends in Special Language & Language Technology*, 2001.

[5] J. W. Ely, J. A. Osheroff, and M. H. Ebell. Analysis of questions asked by family doctors regarding patient care. *British Medical Journal*, 319(7206):358–361, 1999.

[6] S. Gamble. Hospital libraries enhance patient care and save money. *Journal of the Alberta Association of Library Technicians*, 23(2):10–12, 1996.

[7] P. N. Gorman. Information needs of physicians. *Journal of the American Society for Information Science*, 46(10):729–736, 1995.

[8] H. G. L. M. Grundmeijer, K. Reenders, and G. E. H. M. Rutten. *Het Geneeskundig Proces. Van Klacht naar Therapie*. Elsevier Gezondheidszorg, Maarssen, The Netherlands, 1999.

[9] M. Jensen. Medical terminology, 2004. http://www.gen.umn.edu/faculty_staff/jensen/1135/med_term_activites/default_big_list.html.

[10] R. N. Jerome, N. B. Giuse, K. W. Gish, N. A. Sathe, and M. S. Dietrich. Information needs of clinical teams: analysis of questions received by the clinical informatics consult service. *Bulletin of the Medical Library Association*, 89(2):177–185, 2001.

[11] L. Leibovitch, Y. Levy, and Y. Shoenfeld. Pancreatitis induced by Clarithromycin. *Annals of Internal Medicine*, 125(8):701, October 1996.

[12] R. A. Miller, L. Jamnback, N. B. Giuse, and F. E. Masarie. Extending the capabilities of diagnostic decision support programs through links to bibliographic searching: Addition of 'canned MeSH logic' to the Quick Medical Reference (QMR) program for use with Grateful Med. In P. Clayton, editor, *Proceedings of the (SCAMC), Washington, DC*, pages 150–155, New York, NY, 1991. McGraw-Hill, Inc.

[13] National Center of Biotechnology Information. Medline. Fact Sheet, 2002. http://www.nlm.nih.gov/pubs/factsheets/medline.html.

[14] National Center of Biotechnology Information. Pubmed: Medline retrieval on the world wide web. Fact Sheet, 2002. http://www.nlm.nih.gov/pubs/factsheets/pubmed.html.

[15] National Center of Biotechnology Information. Unified Medical Language System. Bethesda, MD, 2003. http://www.nlm.nih.gov/research/umls/.

[16] National Center of Biotechnology Information. Pubmed, 2004. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi? CMD=search&DB=pubmed.

[17] National Center of Biotechnology Information. UMLS metathesaurus. Fact Sheet, 2004. http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html.

[18] National Center of Biotechnology Information. UMLS semantic network. Fact Sheet, 2004. http://www.nlm.nih.gov/pubs/factsheets/umlssemn.html.

[19] National Center of Biotechnology Information. Unified Medical Language System. Fact Sheet, 2004. http://www.nlm.nih.gov/pubs/factsheets/umls.html.

[20] R. Rada, J. Barlow, D. Bijstra, J. Potharst, P. de Vries Robbé, and P. Zanstra. OAR: Open architecture for reasoning applied to connection patient records to medical literature. In J. Noothoven van Goor and J. P. Christensen, editors, *Advances in Medical Informatics*, pages 287–294, Amsterdam, The Netherlands, 1992. IOS Press.

[21] M. C. Reddy, W. Pratt, P. Dourish, and M. M. Shabot. Asking questions: Information needs in a surgical intensive care unit. In I. S. Kohane, editor, *Proceedings of the Annual AMIA Symposium (AMIA'02), San Antonio, Texas*, pages 647–651, 2002.

[22] B. J. J. W. Schouwenberg and J. Deinum. Acute pancreatitis after a course of clarithromycin. *The Netherlands Journal of Medicine*, 61(7):266–267, 2003.

[23] R. Smith. What clinical information do doctors need? *British Medical Journal*, 313(7064):1062–1068, 1996.

[24] T. D. Tran, N. Gracelon, A. Burgun, and P. Le Beux. Experiments in cross-language medical information retrieval using a mixing translation module. In M. Fieschi, E. Coiera, and Y. C. Li, editors, *Proceedings of the MedInfo2004, San Fransisco, CA, USA*, pages 946–949. IOS Press, 2004.

[25] E. M. van Mulligen. UMLS-based access to CPR data. *International Journal of Medical Informatics*, 53(2–3):125–131, 1999.

Blueprint of a Cross-Lingual Web Retrieval Collection

Börkur Sigurbjörnsson Jaap Kamps* Maarten de Rijke

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{borkur,kamps,mdr}@science.uva.nl

ABSTRACT

The world wide web is a natural setting for cross-lingual information retrieval; web content is essentially multilingual, and web searchers are often polyglots. Even though English has emerged as the *lingua franca* of the web, planning for a business trip or holiday usually involves digesting pages in a foreign language. The same holds for searching information about European culture, sports, economy, or politics. This paper discusses the blue-print of the WebCLEF track, a new evaluation activity addressing cross-lingual web retrieval within the *Cross-Language Evaluation Forum* in 2005.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries

General Terms

Measurement, Performance, Experimentation

Keywords

Information Retrieval, Cross-Language Information Retrieval

1. INTRODUCTION

The world wide web is a natural setting for cross-lingual information retrieval. This is particularly true in Europe: many European searches are essentially cross-lingual. For instance, when organizing to travel abroad for a business trip or a holiday, planning and booking usually involves digesting pages in foreign languages. Similarly, looking for information about European culture, sports, economy, or politics, usually requires making sense of web pages in several languages. A case in point is the current European Union, which has no less than 20 official languages.

*Currently at Archives and Information Studies, Faculty of Humanities, University of Amsterdam.

DIR 2005, 5th Dutch-Belgian Information Retrieval Workshop.

The linguistic diversity of European content is "matched" by the fact that European searchers tend to be multilingual. Some Europeans are native speakers of multiple languages. Many Europeans have a broad knowledge of several foreign languages, and especially English functions as the *lingua franca* of the world wide web. Moreover, many Europeans have a passive understanding of even more languages.

The challenges of cross-lingual web retrieval will be addressed in WebCLEF [17], a new track at the *Cross-Language Evaluation Forum* [3, CLEF] in 2005. In this paper we provide a preliminary overview, discussing our view of the cross-lingual web retrieval task, the document collection used, EuroGOV, and the overall set-up of the WebCLEF track.

The remainder of the paper is organized as follows. In Section 2, we describe cross-lingual aspects of web retrieval in the European context. Section 3 discusses the problems involved, and outlines a test-suite for cross-lingual web retrieval. Then, in Section 4, we provide details of EuroGOV, a new web collection for cross-lingual web retrieval. Section 5 details how this collection will be used within the setting of the WebCLEF track at CLEF. Finally, in Section 6, we discuss our findings and draw some conclusions.

2. CROSS-LINGUAL WEB RETRIEVAL

In this section we discuss why the web is a natural habitat for cross-lingual information retrieval.

2.1 Multilingual Web Content and Users

The web is essentially multilingual. Although reliable statistics on web content and web usage are hard to come by, it is evident that the web is increasingly reflecting the linguistic diversity of the world's population. Let us first look at the web's content. Some indicative figures on the distribution of web content over languages are shown in Table 1.[1] On the one hand, it is clear that English still functions as the *lingua franca* of the web. English is by far the most frequently used language. On the other hand, it is also clear that there is a substantial amount of non-English content on the web. The total amount of non-English pages is approaching that of pages in English. European languages

[1]Estimates are based on pages in the index of search engine `http://alltheweb.com` in 2002 [for details, see 9]. See also `http://www.cia.gov/cia/publications/factbook/rankorder/2184rank.html` for recent data on number of Internet hosts per country.

Table 1: Global Internet Content Statistics. Based on estimated figures from `http://www.netz-tipp.de/languages.html`, 2002.

Web content by language.		
Language	Internet Pages	% Web Content
English	1142.5	56.4
Non-English	2024.7	43.6
European (non-English)	536.9	26.5
Dutch	38.8	1.9
French	113.1	5.6
German	156.2	7.7
Italian	41.1	2.0
Polish	14.8	0.7
Portuguese	29.4	1.5
Russian	33.7	1.7
Scandinavian (total)	17.4	1.3
Spanish	59.9	3.0

Table 2: Global Internet User Statistics. Based on estimated figures from `http://global-reach.biz/globstats`, September 2004.

On-line population by language.		
Language	Internet Access	% On-line Population
English	295.4	35.2
Non-English	544.5	64.8
European (non-English)	285.5	35.7
Dutch	14.0	1.7
French	33.9	4.2
German	55.3	6.9
Italian	30.4	3.3
Polish	9.6	1.2
Portuguese	24.4	3.1
Russian	6.5	0.8
Scandinavian (total)	12.8	1.6
Spanish	72.0	9.0

other than English account for over a quarter of the global web content.

Let us now turn to the web's users. Table 2 gives, again, some indicative figures on the distribution of web users over languages.[2] Here, the situation is even more striking. Nearly two-thirds of the user population has a primary language other than English. Also, the European users excluding native English speakers account for one-third of the whole online user population.

The multilingual nature of the web has prompted many organizations to engage themselves in web globalization efforts [18]. This typically involves localization of web sites tailored to particular markets and users, and is proving inevitable for organizations that get their revenues from web activities, such as e-commerce. Apart from straightforward machine translation, specific cross-lingual retrieval tools and techniques have not yet been adopted by industry [5].

[2]Estimates are based on a variety of sources, e.g., on home access of Internet users [for details, see 6]. See also `http://www.cia.gov/cia/publications/factbook/rankorder/2153rank.html` for recent data on Internet users per country.

2.2 Cross-Lingual Information Retrieval

In 2002, road-map for cross-lingual information retrieval research was suggested by [5]. Gey et al. [5, p.73] list three challenges for cross-lingual information retrieval:

1. Where to get resources for resource-poor languages?
2. Who do we not have a sizable web corpus in multiple languages?
3. Why aren't search engines using our research?

The second challenge is addressed head-on in this paper. Gey et al. [5] also point out a potential problem for the evaluation methodology if English is the dominating language of web pages in a collection. Consider a set of ad hoc retrieval topics for which there are many relevant pages in English. A system focusing exclusively on English will yield very good performance, which is in contrast with the intentions of cross-lingual retrieval. There is a need for a multilingual web collection that is not dominated by one particular language. The obvious candidate is a collection based on European web content.

Cross-lingual information retrieval has been high on the agenda ever since the early years of the web. There has been an interesting shift in focus over the years. Early studies of cross-lingual retrieval, such as [10], focused on monolingual users wanting to search a collection of documents that they cannot read. Recent studies, such as [14], focus on polylingual users wanting to search documents in the languages that they can understand. One of the earliest studies taking into account users understanding multiple languages is [2]. Capstick et al. [2] investigate a system in which users can express their query in their native tongue, while retrieving documents in several languages of which the user has, at least, a passive understanding. In a series of publications, Petrelli et al. [12, 13, 14] have argued convincingly that bilingualism or polylingualism is the rule for many potential users of cross-lingual retrieval systems. As the authors put it, "it is not unusual to find people who are fluent in 4 or 5 languages." Petrelli et al. [11] highlight that many people use different languages in their everyday work, think of journalists, business analysts, professional translators, information professionals, and, of course, scientists. Their varying degrees of knowledge of the languages to search, their generic search expertise, and the final task to perform (e.g., search-only versus search-and-use) create different user classes with different information needs.

3. TOWARD A TEST COLLECTION

In this section we discuss some of the main challenges in building a cross-lingual web retrieval collection, and outline how a such a test collection could be set-up.

3.1 Requirements

As pointed out by Gey et al. [5, p.73], it is non-trivial "to define suitable criteria for the construction of a valid multilingual Web corpus for R&D." Based on our discussion of cross-lingual web retrieval in Section 2, we draft a tentative list of requirements we would like a cross-lingual web retrieval test suite to satisfy.

Ideally, a cross-lingual web retrieval test collection

- should cater for users that are polyglots;
- should address user tasks that are essentially multilingual;
- should have documents in a wide variety of languages;
- should not be dominated by a particular language, i.e., English;
- should be of sufficient size; and
- should be a natural domain for multilingual search.

3.2 Challenges and Solutions

Of course, not all languages are equally acceptable as a vehicle for conveying information to a particular user. It would be natural and attractive to conduct retrieval experiments in a setting where users store profiles in which they list the languages they can read. This brings us directly to one of the main challenges involved in building a truly cross-lingual web retrieval collection: just as users are not able to read all languages, so will individual assessors be unable to provide relevancy judgments for pages in each and every language in a cross-lingual web collection (let alone be a native speakers of each language). At the same time, making relevance judgments based on topical relevance would require the assessors to judge the content of a page regardless of the language in which it is expressed [15, 16].

This is a fundamental challenge that we cannot, and will not be able to resolve, but we can try to minimize the extent to which it affects the cross-lingual web retrieval test collection. Our strategy is based on two key ingredients.

Known-item Search We will focus on *known-item search* exclusively, that is, on tracking down pages known to exist in the collection. This will imply that the (original) target page is fairly unique, although identical mirrors of the page, or translations of the page's content into other languages may occur. Known-item search is a natural task in a web environment, and has some obvious further advantages in the limited assessment effort needed to create the test collection.

Monolingual Topics Our test collection will be built from sets of monolingual topics targeting a particular language or domain of the collection. This implies that topics should have a national focus, making them unlikely to occur in other languages/domains. The original topics will be translated into English and, potentially, other languages, allowing for bilingual and multilingual retrieval against the original monolingual judgments.

To sum up, we face the challenge that users and assessors are polyglots, but not "omniglots." By focusing on monolingual known-item search, assessors should primarily judge pages in their native tongue. If the site provide translations of the target page, these are generally easy to identify. The occurrence of an English version of an originally non-English page is frequent, and sometimes there are translations to a whole range of languages. Can we exclude that relevant pages occur in an unexpected language or domain? No, we cannot. Think of a foreign embassy hosting a content-wise identical page in a different language and a different domain. In this sense the recall base may be incomplete, but we expect that this will not affect the quality of the test-suite.

Note that our focus on known-item search also avoid us falling victim to the concerns of [5]: English pages will not dominate the set of relevant pages for such topics. This does not imply that we are not interested in general ad hoc topics, just that we want to start with known-item search topics. For assessing general ad hoc information needs, the problem of having to assess pages in all the collection languages is unavoidable. The pragmatic solution would be to include a limited set of target languages, i.e., those languages that the topic creator can read, in the topic statement. Note that this may lead, again, to the dominating language problem if one particular language, i.e., English, can be read by all topic creators.

3.3 Outline of a test-suite

Based on our discussion above, we envision the following sets of known-item search tasks.

Monolingual Tasks a set of 50 known-item search topics in a single language, targeting pages in the same language.

Mixed Monolingual Task a set of 200–500 known-item search topics in multiple languages in which the language of the topic statement is typically the language of the target page.

Bilingual Tasks a set of 50 known-item search topics in English, targeting pages in a single other (i.e., non-English) language or domain.

Multilingual Task a set of 200–500 known-item search topics in English, targeting pages in any other (i.e., non-English) language or domain. This may require the extraction of language cues, for example, a topic like "Danish minister of ..." is likely to target pages in Danish, or from the `.dk` domain.

The mono- and bilingual tasks can be organized out as subtasks of the mixed monolingual and multilingual tasks, respectively.

More complex mixtures are possible, also revealing more information:

Language Identification Model the use of language identification tools: *What language do pages in the collection have? What language does the topic have?*

Language Cue Extraction Model the use of language cue extraction: *What language does the targeted page have?*

Search Intentions We could also model user interaction by revealing part of the searcher's intentions: *What language or domain does the targeted page have?*

4. EUROGOV COLLECTION

Cross-lingual web retrieval requires a new document collection to be constructed, containing web content in many languages. Of course there are many options for creating such a collection. Multi-lingual documents are abundant on the web. We have chosen to focus on pages of European government-related sites, where collection building is less restricted by intellectual property rights. We baptize

Table 3: Main domains in the EuroGOV collection, and the dominant languages (based on preliminary page counts).

EuroGOV Collection.		
Domain	Predominant language	# of Pages
.cz	Czech	690,673
.de	German	887,260
.es	Spanish	735,310
.eu.int	Mixed	3,710,000
.fi	Finnish	868,100
.fr	French	1,399,653
.hu	Hungarian	230,830
.it	Italian	570,506
.nl	Dutch	388,470
.pt	Portuguese	186,783
.ru	Russian	185,000
.se	Swedish	312,000
.uk	English	829,740

this collection EuroGOV. We think of this collection as an European counterpart of the .GOV collection.

Our initial plan was to obtain a focused crawl from the European Union seed `.eu.int`. However, restricting a crawler to government-related sites proved highly non-trivial. The collection we want to crawl is fairly heterogeneous, for example in the number of document languages. For some governments the crawling is smooth and we can easily filter out governmental pages (notable examples include `.gov.uk` and `.regeringen.se`). Most governmental sites, however, have more complex structures, and we could only focus the crawl by providing an explicit list of domains. As an example, we crawled 13 different domains to gather pages from the Finnish government. As the following domain list shows there is no easy way of identifying Finnish governmental domains:

> `defmin.fi`, `formin.finland.fi`, `intermin.fi`, `ktm.fi`, `minedu.fi`, `mintc.fi`, `mmm.fi`, `mol.fi`, `om.fi`, `stm.fi`, `vm.fi`, `vnk.fi`, and `ymparisto.fi`

These differences in domain naming traditions will make it difficult to guarantee completeness of some governments. As a result, EuroGOV will contain, as a minimal requirement, a fairly complete content of

- main government portals
- main ministries

4.1 EuroGOV Collection Characteristics

The EuroGOV collection will contain over 10 million pages; this is an indicative figure, based on our current set of seeds and the coverage of these domains by Internet search engine `http://google.com/`. For practical reasons, we will only release a 3 million page subset of the full EuroGOV collection for the WebCLEF 2005 evaluation campaign. The countries and domains included for EuroGOV are chosen in accordance with current CLEF interests and plans. Table 3 gives the preliminary page counts for each of the main domains in the collection. The distribution of the main domains is visualized in Figure 1. Further countries from whose government portals are being considered for inclusion include

- `at`, `be`, `cy`, `dk`, `ee`, `gr`, `ie`, `lu`, `lv`, `mt`, `pl`, and `sk`.

Note that pages in the languages of these domains will 'creep in' anyway. For example, the `eu.int` domain have ample pages in all the 20 official languages of the European Union.

The EuroGOV collection will feature more languages and countries than used in WebCLEF 2005 tasks. We made a deliberate choice to go for this extended list of countries and domains. On the one hand, this will facilitate future task extensions for cross-lingual web retrieval, or re-use of the collection for other purposes. On the other hand, we feel that this reflects the natural situation when building a 'European' search engine. Of course, participating teams at WebCLEF will be free to select only parts of the collection to index for a specific task.

4.2 EuroGOV Availability

The EuroGOV Collection will be made available in January 2005 [17]. The crawled pages will have been cleaned-up and put in a uniform format. The resulting pages will be bundled and compressed in manageable sizes. The collection will be distributed over the Internet; if a participant's local band-width is not sufficient, DVDs can be shipped on request. The collection will be distributed under a license restricted to research use only.

5. WEBCLEF TRACK AT CLEF

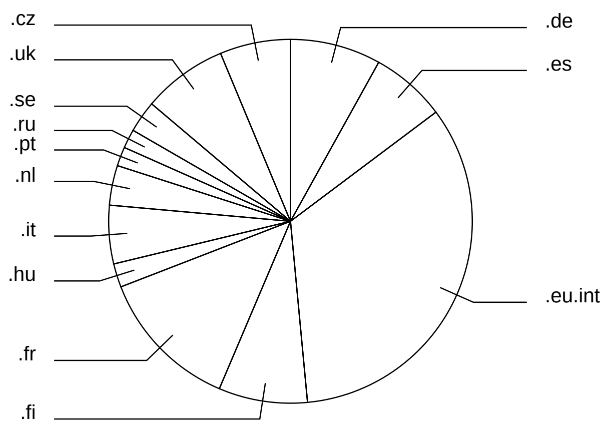
The precise WebCLEF Track guidelines will be determined in early 2005. We intend to involve track participants actively during topic creation and peer-assessments.

5.1 Topic Creation

Topic creators will be asked to create monolingual topics, targeting pages in the topic's language. We ask participants to

- focus on a particular domain/language combination,
- create ± 50 monolingual known-item search topics, and

Figure 1: Composition of the EuroGOV collection (based on preliminary counts).



- provide English translations of the topics.

The monolingual topics form the core of the monolingual tasks, as well as for the mixed monolingual task. The translated topics will be used for the bilingual and multilingual tasks. Depending on interest and available language expertise, further translations may be provided to accommodate other language combinations.

Bilingual topics will be the result of the translation of the original monolingual topics. This will require to make make default assumption explicit, e.g., consider the Dutch monolingual topic

- "*minister van buitenlandse zaken.*"

A literal translation of this topic into English would be

- "minister of foreign affairs."

However, the implicit assumption underlying the Dutch topic, because it is formulated in Dutch, is to find information about the

- "**Dutch** minister of foreign affairs."

By making these default assumptions explicit in the translated topic, the relevance judgments on the original Dutch topic statement will carry over to the translated version.

5.2 Assessment and Evaluation

There will be a rather limited assessment stage, in which we verify that the original target page is unique. Topic creators will be asked to assess their own topics, and identify whether similar or identical content occurs on other pages, possibly in a different language. The results will be evaluated by the familiar measures for early precision, including mean reciprocal rank of the first found relevant page, and success at 1, 5, and 10.

5.3 WebCLEF 2005 Tasks

In the first incarnation of the WebCLEF track, we will focus on a small number of core tasks. Building the WebCLEF test collection will be a community effort, and the specific languages for which topic and judgments are available will depend on the available language expertise. We expect to provide, at least, topics and judgments in the following eight target languages: Dutch, English, French, German, Italian, Portuguese, Russian, and Spanish.

At the time of writing, we consider focusing on two main tasks.

Mixed monolingual Using 50 topics per target language, for at least the eight languages mentioned above. (This is a natural task when building a 'European' search engine.)

Multilingual Using the topic language English, based on the translations of the monolingual topics provided by the topic authors. (This is a natural task when catering for the information needs of a polyglot.)

As a side-product we will also evaluate on the individual monolingual and bilingual retrieval results for each of the target languages.

The topics will contain additional topic fields revealing the topic language, and the language and domain of the intended target page.

6. DISCUSSION AND CONCLUSIONS

The history of building web retrieval test collections has been very well documented in [8]. Our decision to focus on known-item search is largely based on experiences during the web tracks at the Text REtrieval Conference (TREC). An important difference with earlier web retrieval test collections is that we decided to crawl the fairly complete content of a number of sites, rather than letting the crawler navigate freely on the web. This may have interesting effects on the link-structure of the resulting collection. Earlier efforts put much stress on replicating a natural link structure in a web collection [1, 7]. In our collection, we anticipate that each individual site will exhibit a fairly dense network of navigational links, but that the network linking individual sites will be much less dense. Hence, the resulting collection could be viewed a heterogeneous collection of sites.

The EuroGOV web retrieval collection discussed in this paper is distinct from the collections used at TREC by its focus on cross-lingual retrieval. However, web retrieval collections have also been constructed outside the TREC framework. Of particular interest is the web task at NTCIR-3 [4]. Here, a crawl of the Japanese `.jp` domain is used, containing mostly Japanese (90%) and English (8.3%) pages. Our proposal for a cross-lingual web retrieval collection substantially extends the number of languages in that collection, and the planned cross-lingual web track will enable the evaluation of a variety of monolingual, bilingual, and multilingual web retrieval tasks.

Summarizing, in this paper we discussed a wide range of facets of cross-lingual web retrieval. We analyzed the diversity of languages of pages on the web, as well as the native languages of web users. We distinguished two user types for cross-lingual retrieval: on the one hand an essentially monolingual user who searches pages in languages that she cannot read, and, on the other hand, a polyglot who searches in languages that they have some level of proficiency in. We decided to focus on the second type of user, and outlined a blueprint for a cross-lingual web retrieval test collection. The proposed test suite has a document collection, baptized EuroGOV, containing documents from various European governmental sites. This will avoid the dominance of a single language, i.e., English, and provides a natural setting for multilingual web search. We highlighted the problem that users and assessors may be polyglots but no "omniglots," i.e., they will not be able to read all languages in the collection. As a result, we plan to build the test collection around monolingual, known item search topics. By providing translations of the topics, we will create a true bilingual and multilingual test sets.

In its first incarnation in 2005, WebCLEF will focus on two main tasks, *mixed monolingual retrieval* and *multilingual retrieval* using the English topic set. We view this as a stepping stone toward realizing the full potential of cross-lingual web retrieval. Building a test-collection for cross-lingual web retrieval is essentially a community effort: we depend on participants providing language expertise, natural cross-lingual information needs, and relevance judgements. Future editions will address, amongst other things, resource-poor languages, more natural cross-lingual search scenarios, and search engine efficiency.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 612-13-001, 612.000.106, 612.000.207, 612.066.302, and 612.069.-006.

REFERENCES

[1] P. Bailey, N. Craswell, and D. Hawking. Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing and Management*, 39:853–871, 2003.

[2] J. Capstick, A. K. Diagne, G. Erbach, H. Uszkoreit, A. Leisenberg, and M. Leisenberg. A system for supporting cross-lingual information retrieval. *Information Processing and Management*, 36:275–289, 2000.

[3] CLEF. Cross language evaluation forum, 2004. `http://www.clef-campaign.org/`.

[4] K. Eguchi, K. Oyama, E. Ishida, N. Kando, and K. Kuriyama. An evaluation of the web retrieval task at the third NTCIR workshop. *SIGIR Forum*, 38:39–44, 2004.

[5] F. C. Gey, N. Kando, and C. Peters. Cross language information retrieval: a research roadmap. *SIGIR Forum*, 36(2): 72–80, 2002.

[6] Global Reach. Global internet statistics by language, 2004. `http://global-reach.biz/globstats`.

[7] C. Gurrin and A. F. Smeaton. Replicating web structure in small-scale test collections. *Information Retrieval*, 7:239–263, 2004.

[8] D. Hawking and N. Craswell. Very large scale retrieval and web search. In E. Voorhees and D. Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[9] Netz Tipp. Internet statistics: Distribution of languages on the internet, 2004. `http://netz-tipp.de/languages.html`.

[10] D. W. Oard. Serving users in many languages: Cross-language information retrieval for digital libraries. *D-Lib Magazine*, December, 1997.

[11] D. Petrelli, M. Beaulieu, and M. Sanderson. User participation in CLIR research. In F. C. Gey, N. Kando, and C. Peters, editors, *Cross Language Information Retrieval: A Research Roadmap*, pages 43–47, 2002.

[12] D. Petrelli, M. Beaulieu, M. Sanderson, G. Demetriou, P. Herring, and P. Hansen. Observing users, designing Clarity: A case study on the user-centered design of a cross-language information retrieval system. *Journal of the American Society for Information Science and Technology*, 55: 923–934, 2004.

[13] D. Petrelli, P. Hansen, M. Beaulieu, and M. Sanderson. User requirement elicitation for cross-language information retrieval. *The New Review of Information Behaviour Research*, 3, 2002.

[14] D. Petrelli, S. Levin, M. Beaulieu, and M. Sanderson. Which user interaction for cross-language IR? Design issues and reflection. *Journal of the American Society for Information Science and Technology*, 56, 2005.

[15] T. Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science*, 26:321–343, 1975.

[16] E. M. Voorhees. The philosophy of information retrieval evaluation. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross-Language Information Retrieval Systems, CLEF 2001*, volume 2406 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2002.

[17] WebCLEF. Cross-lingual web retrieval, 2004. `http://ilps.science.uva.nl/webclef/`.

[18] J. Yunker. *Beyond Borders: Web Globalization Strategies*. New Riders Publishing, 2002.

Multidocument Question Answering Text Summarization Using Topic Signatures[*]

Maria Biryukov[†] Roxana Angheluta[‡] Marie-Francine Moens[§]

ABSTRACT

In this paper we describe the process of the multidocument question answering summarization based on the topic signatures. We show that summaries produced using the topic signatures have coverage which is comparable to that of the best systems competing at the Document Understanding Conference [5].

General Terms

Topic signatures, Text summarization, Question answering

1. INTRODUCTION

Many automated text processing tasks require the step of topic identification. It is the first stage in the text summarization process and the goal state in the text categorization task. Construction of *topic signatures*, introduced by Chin-Yew Lin [14] is one of the methods of topic identification. Simply stated, a *topic signature* is a pair consisting of the topic word(s), and set of semantically related to it words, that *together* uniquely identify the topic. The process of the topic signature generation exploits the natural tendency of the semantically related words to co-occur more often than by chance in the same context. For example, words "cigarette", "tobacco", "smoking" are very likely to occur in the text whose topic is "cigarette consumption". As such they are strong candidates to be among the "cigarette consumption" topic signature words.

Topic signatures have a variety of applications. When trained on the preclassified documents of specific topic they can be used to identify the presence of the learned topic in the previously unseen documents [15]. Another way to use the trained topic signatures is to apply them to text categorization [10]. Topic signatures that have not been previously trained can be exploited in the processes of the query expansion [28, 13] and text summarization. So far this approach has been taken to successfully produce single and multidocument summaries [15, 16].

[*]This paper is an extended abstract of the thesis submitted in the partial fulfillment of the requirements for the degree of Master of Artificial Intelligence. The full version of the thesis is available at `http://www.esat.kuleuven.ac.be/~abiryuko/Masha`.

[†]Katholieke Universiteit Leuven, Belgium, barmaliska@yahoo.com

[‡]Katholieke Universiteit Leuven, Belgium, anghelutar@yahoo.com

[§]Katholieke Universiteit Leuven, Belgium, marie-france.moens@law.kuleuven.ac.be

Motivated by the variety of the topic signature applications we took on this theme to explore it further. In this work we aimed to test the topic signature approach in a new task, specifically in producing multidocument *question answering* summaries. Based on a given set of documents, our summaries had to answer the question *Who is X*, where X was the name of the person. The topic signatures that provided the basis for the summarization procedure have been constructed around the person name and identified words that highly correlated with it. To form a question answering summary, our system selected those sentences that contained the topic signature words. While in our task construction of the topic signatures was considered an intermediate step, these "personal" topic signatures could be used on their own merit. For example in building profiles for respective persons, or used in more general way, profiles for professions or social roles with further application in the information extraction or text categorization tasks. From the technical point of view, we tested also the different methods of the topic signature generation: two methods that identify the word co-occurrence patterns: the *likelihood-ratio* and the *chi-square* tests, and one method that identifies the most important words in the text: the *term frequency-inverse document frequency weighting scheme, $tf.idf$*.

This paper is organized as follows: Section 2 describes the previous work in text summarization and topic signatures. Section 3 describes the corpus that was available to us, and our methods of the topic signature generation and summarization. Section 4 presents and discusses our experimental results. Section 5 gives conclusions and briefly discusses possible improvements and future research.

2. SUMMARIZATION: RELATED WORK

In this section we define the notion of a text summary and briefly introduce its different types. We review various approaches to the text content identification that set the ground for the topic signature proposal.

2.1 Summarization: Basics

The growing amounts of information available electronically require tools for fast assessing the extent and content of the information resources. A summary may be thought of as such a tool. The goal of the text summarization system is to produce a concise representation of a document or set of documents that have been submitted to it. Depending on the future use one may request a *generic* summary which gives an overall sense of the document(s), or *query-relevant* summary - one that highlights the content which is most closely corresponds to the search query or user question. Summaries differ also by their "fidelity" to the original text. One identifies *extracts* when the summary represents excerpts from the original text, and *abstracts* which consist of novel phrasings describing the main content of the original. Abstract creation may involve sentence compression to represent material in a more compact and coherent way [22, 11], and, importantly, the fusion of the related topics into more general ones [10].

Both, extraction and abstraction, processes start from the identification of what the text is about, i.e. topics of the original texts. Since the late 1950's when the first attempts to produce automated summaries have been made, various approaches to the topic identification have been taken. In the rest of the section we will briefly present those of them that contributed to the invention of the topic signatures.

2.2 Classical Approaches

In early work on automated summarization it was assumed that the most frequently occurring context bearing words reflect the most important text content [18, 7]. This approach was extended with a number of heuristics (such as *Cue phrases*, *Title words*, and *Location*) that aimed to find signals in the text that point at the important content [7]. Though applied to certain extent in the modern systems [12, 27, 9, 14, 10], these old approaches have their limitations. The most serious is style dependency. For example taking the first or last sentences of a paragraph or text (*Location* method), may work with news-wires and scientific texts, but not necessarily with other genres. The same observation is true for the employment of the various lexical cues: discourse markers change significantly from "National Geographic" to "Computational Linguistics". In contrast, word frequency is a good indicator for words that represent important concepts. This fact is considered to be true for the majority of texts, independently of their style [2]. The weak point of the method is that it does not assign any importance to the semantic connections between the words.

2.3 Modern Approaches

Modern approaches can roughly be broken down into the three main categories: *Knowledge-based*, *Statistical*, and *Discourse-based*.

- *Knowledge-based approach* tends to capture the semantic inter-relations within the text and determines the crucial information based on these relations. On the one hand summaries produced by the systems like SUMMONS [21] are considered a big achievement. On the other hand, these systems are knowledge-intensive and domain-dependent. Moreover, prespecified templates on which such systems operate, cause the summaries to be focused on the aspects that the analyst found important. The method is not adequate if one wants to know what the whole text is about.

- *Statistical approaches* seem more robust because they are domain-independent. *tf.idf* weighting schemes (as an extension of the simple word frequency counting suggested by Luhn, it keeps track of both - term frequency within the document as well as its distribution throughout the whole collection) are often used in the IR and summarization tasks [24, 3] to identify for each word the level of its importance. When summarizing, text units that contain the most important ("topic") words, sometimes in combination with an heuristic, such as presence of these words in the title [3], are extracted for the summary. While more robust than a simple word counting, these weighting schemes still operate on the word level.

- In the attempt to take the content analysis beyond the word level, *discourse analysis* has been suggested. It often relies on the lexical cues [20] or exploits the lexical knowledge bases such as WordNet[1] in order to detect the cohesion of a text as signaled by lexical items [23], which allows to identify the most salient portions of a text [2]. The main gain of the discourse-based methods is their ability to capture the dependency relations between the text segments and spot the concepts rather than its word components. The more subtle points are: the possible absence or ambiguity of the lexical cues [23], and the incompleteness of the underlying knowledge bases. The former may eventually lead to the distortion of the information in the summary; the latter bounds the system performance to the level of the knowledge source completeness.

2.4 Topic Signatures

The methods for the topic identification we have mentioned so far suffered either from the inability to capture the information on the level deeper than word, or depended on the accuracy of the semantic analysis or the completeness of the knowledge base. A topic signature [14] is a statistical method that aims to extend the topic identification to the concept level while not relying on the knowledge bases or complex semantic parsers. The basic idea behind the approach is that semantically related terms (words or groups of words) tend to co-occur in the same context. This means that if there is an idea that plays an important role in a text (e.g. main or sub-topic), a set of terms related to it occurs in an expected manner. This set describes a topic in a unique way. A topic signature is thus a pair that consists of a topic term and a set of terms that are related to it. Instead of learning the topically related terms from a knowledge base, one can create them from the texts on a given topic using statistical methods. The *tf.idf* weighting scheme, χ^2, and *likelihood ratio for binomial distribution* tests that we have used are among the possibilities.

In the perspective of text summarization, topic signatures can be used for both extracts [15, 16] and abstracts [15]. To our knowledge they have not been used yet to produce the query-relevant (or question answering) multidocument extracts, which constitutes the task of this work.

3. OUR METHODS

In this section we describe our approach to the generation of the topic signatures and summaries.

Construction of the topic signatures consists in finding words that co-occur with the topic word more often than by chance. All the three methods we have used allow to

[1]WordNet is a lexical reference database for the English language. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. More information about the WordNet is available at `http://www.cogsci.princeton.edu/~wn/`.

determine such words. But the assigned probability of the co-occurrence may vary depending on the way of computation. As a result, topic signatures constructed with the different techniques might not be identical as might not be the summaries for which they provide a basis. Consequently we were interested to compare the extracts created by using different sets of the topic signatures. The extracts had to satisfy a length constraint and be ≤ 665 bytes[2].

3.1 The Corpus

The training data used in this work consisted of 50 clusters of documents provided by DUC'2004 [5]. The clusters represented material concerning a person X, with X different for each cluster. Each cluster was composed of 10 excerpts drawn from the news-wires or newspapers. The length of the passages varied from 50 to 900 words approximately. The passages contributed a different amount of information about a person: from the marginal reference to the main focus of the text. During the preprocessing the corpus was tagged with the LT-POS tagger[3], and formatted to contain one sentence per line. In addition we used two more versions of the same corpus. One version was modified in such a way that if the original sentence contained a personal pronoun in place of the person X's name, this pronoun was substituted with the person X's name itself. In the second version the same kind of modification was performed but regarded only those cases where the personal pronoun appeared in the subject position. This type of the data treatment is called *co-reference resolution [1].* We also used a manually built stop word list to exclude from the analysis the most frequent words such as articles, auxiliary verbs, prepositions etc.

3.2 Topic Signature Creation

3.2.1 Topic identification

As we mentioned in the previous section, it was not necessarily the case that all the documents within a cluster concerned the person X. But since we had eventually to produce a query-oriented summary and the query was formulated around the person's name, we defined the texts' topic to be a person's X family name (further in the text "person's name"). Consequently, for each person X, the topic signature was of the form:

$$TS = \{X, <(t_1, w_1), ..., (t_n, w_n)>\} \quad (1)$$

where X denotes a person X's family name, each t_i denotes an i-th term with its associated weight w_i.

3.2.2 Choice of a term unit

The topic signatures we created consisted of uni-grams. With each technique we constructed 4 sets of topic signatures for each person name.

- First we considered only nouns to be the topic signature terms. This choice was motivated by the assumption that nouns are the most informative units for the topic characterization.

[2]With respect to the type of the summary and its length limit we have replicated the task 5 of the Document Understanding Conference 2004 (DUC'2004) [5]. DUC is a conference which aims to evaluate the research in the area of automatic summarization

[3]Information on the LT-POS tagger is available at: `http://www.ltg.ed.ac.uk/~mikheev/`.

- On the other hand the original way to create topic signature, as described in [10, 14, 15, 16], is to take all non-stop words into account regardless of their part of speech class. In the attempt to increase the coverage, we pursued this approach too, and created the following three sets of topic signatures:
  - the signature terms could be nouns and adjectives;
  - the signature terms could be nouns, adjectives and verbs;
  - the signature terms could be nouns, adjectives and verbs as before, but in this case we created topic signatures after converting all words into their root form using the Porter stemmer algorithm[4]. We decided to perform stemming in order to avoid appearance of the terms with the same stem and, consequently, with the similar meaning, as individual entries in the topic signature.

3.2.3 Choice of a text unit

As we explained in the previous sections, the concept of topic signature is built on the assumption of word co-occurrence in the same context. Hence we had to define a text unit that provided such a context for the topic signature extraction. Since the ultimate goal was the extract generation it seemed natural to think of a sentence as such a text unit. In order to account for the cases where the person was referred to by the personal pronoun we used the corpus versions with the co-reference resolution as described in the Subsection 3.1.

However, if we were working on the sentence level only, we would still have missed words from the sentences where the person is pointed to by his/her profession or status rather than name or personal pronoun. To make the analysis sensitive to such cases we extracted topic signatures on the basis of the text unit being defined "document" as well.

3.2.4 Topic signature extraction with the χ^2 and likelihood ratio tests

After the text unit has been defined, we followed the Algorithm 1 in order to create the topic signature for each person[5].

For both techniques we run the procedure on the corpus versions with the document and sentence as a text unit, with the two possible co-referent positions in the latter case. Hence, six sets of the topic signatures per person have been created. Term-weight pairs in the file H together with the person X's name constitute the topic signature for the person X. It is often used to set a threshold weight to 10.83 with confidence level $p = 0.001$ [15, 26]. We chose to set the cutoff weight to 7.88 with confidence level $p = 0.005$ taking into account the small size of our corpus.

[4]The Porter stemming algorithm (or Porter stemmer) is a process for removing the common morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems. More information on the Porter stemmer is available at `http://www.tartarus.org/~martin/PorterStemmer`.

[5]For the detailed explanation of the statistical techniques see [19], pp. 162-163, 169-175, and [6].

Algorithm 1 Create Topic Signatures.

```
for all Person's name $X$ do
  Count how many text units contain $X$ (counter $Res_1$)
  for all Term $t$ do
    Count how many text units contain $t$ (counter $Res_2$)
    Count how many text units contain both $X$ and $t$ (counter $Res_{1,2}$)
    Compute the weight $w$ of $t$ according to $\chi^2$ or likelihood ratio given $Res_1$, $Res_2$, $Res_{1,2}$ and the total number of text units in the corpus, $N$
  end for
  Sort all terms according to their weight in the descending order
  Select the confidence level looking up the $\chi^2$ distribution table; determine the cutoff associated weight
  Output the term-weight pairs $(t, w)$ whose weight is above the set threshold into file $H$
end for
```

3.2.5 Topic signature extraction using frequency counter

The third technique to produce the topic signatures was *tf.idf* weighting scheme[6].

Algorithm 2 describes how we create topic signatures using *tf.idf* term weighting scheme. The procedure was run on the corpus versions with the document and sentence as a text unit, without distinction in the co-rereferent position in the latter case[7]. Consequently, for the *tf.idf* metric we had two sets of the topic signatures per person. The term-weight pairs in the file F together with the person X's name constitute the topic signature for the person X.

Algorithm 2 Topic Signatures from Frequency Counters

```
Classify document sets as relevant or non relevant according to the given person's name.
for all Term $t$ in the relevant set do
  Count how many times $t$ appeared in the relevant set ($tf_{i,j}$).
  Count how many text units in the the whole corpus contained this term ($df_i$).
  Compute the term weight $weight_{i,j}$.
end for
Sort terms according to their weight.
Define a threshold weight.
Output the term-weight pairs whose weight is above the defined cutoff into the file $F$.
```

We empirically set a threshold to 60 words which correspond approximately to the number of words present in the topic signature which is created with the likelihood ratio test with the cutoff associated weight set to 7.88. Note that when a sentence is chosen to be a text unit we still count term frequency in the set of the relevant texts and the text unit definition has no influence on this parameter. When the document frequency is concerned, sentence is treated as a document and we count in how many sentences a word is present. (We can see it as a slightly modified *tf.idf* metric and call it *tf.isf* weighting scheme, where *isf* means *inverse sentence frequency*.)

[6]See [19], pp. 539-544 for the discussion.

[7]Since we did not restrict the choice of the sentences to only those that contain a person name, co-referent position did not influence the performance.

3.3 Summarization

In this section we will describe our approach towards question answering multi-document extracts based on the topic signatures.

In our task, the role of the topic signatures was to point at those portions of information in the source texts that could be used in order to answer a certain question. Earlier in this paper we have mentioned that many aspects have to be taken into account in order to create a quality extract (discourse structure, semantic inter-relations within the text, heuristics on sentence position.) But our primary goal here was to investigate how far one could get in the question-answering (or query-relevant) multi-document extract creation using mainly topic signatures. When summarizing we selected only the sentences that contained words from the topic signatures.

As discussed in the previous subsections, we created four sets of the topic signatures with each of the three techniques: the likelihood ratio test, χ^2 test, and *tf.idf* weighting scheme. To decrease the number of summaries to be produced, we made a preliminary evaluation of the topic signatures. We noticed that the absolute values of the term weights within the topic signatures might change depending on the number of part of speech classes considered. However, there were no significant changes in the term ranking within the topic signatures. Based on this observation we decided to use topic signatures whose terms were nouns.

We summarized ten (out of 50) sets of documents. Given the eight topic signature series per person set (see Subsections 3.2.4, 3.2.5), we have produced 80 summaries.

3.3.1 Sentence scoring

Many sentences in the documents of the set might contain words from the topic signature. At the same time, the length of the extracts we had to produce was very limited. Hence we had to weight and rank all the sentences that contained the topic signature words and include into the abstract only those with the highest rank. We used two formulae to compute the sentence score which indicated the relevance of the sentence to the signature topic:

$$S(s) = \frac{\sum_{t_i \in s} w_{t_i}}{N(s)} \qquad (2)$$

where s means the sentence we try to score, $S(s)$ is the score of s, t_i means a topic signature term, and w_{t_i} means topic signature weight of the term (i.e. weight that was associated to the term when the topic signature was created) which is present in the sentence. The normalization factor $N(s)$ means the number of words in the sentence. We used this formula when the topic signatures were created using *tf.idf* weighting scheme. Similar to the previous one but with logarithmic weights:

$$S(s) = \frac{\sum_{t_i \in s} \log w_{t_i}}{N(s)} \qquad (3)$$

The reason why we suggested to take the logarithm of the term weight was due to the multiplicative nature of the likelihoods (this measure would be equivalent to multiplying the weights w_{t_i}).

3.4 Filter for redundancy

In order to avoid the appearance of sentences that convey essentially the same information, and allow for the more diverse content of the extracts [1, 4], we developed our mechanism that aimed to minimize the duplication of the information within the extracts. The idea was the following: add the top ranked sentence to the extract and remove from the topic signature all the words that appeared in that sentence. This way we insure that the signature words from the first sentence will not influence the choice of the next sentence (though they are allowed to appear in it). Then we re-rank the remaining sentences with respect to the "updated" topic signature. We repeat the procedure of the topic signature update and sentence re-ranking in a recursive way until the extract length expires.

3.5 Extract Generation

The whole process of the extract generation is described by the Algorithm 3. This algorithm is run on a set of sentences about the person X (the algorithm itself is not constrained to work only with a relevant set), ranks those sentences according to the topic signature in a recursive way, and adds the top-ranked sentences into the summary.

Algorithm 3 Extract Generation

```
Procedure Rank-Sentences(Sentences $s$, topic signature $TS$).
for all $s_i$ do
    Compute the score $S(s_i)$ given $TS$ and the formulae.
end for
Sort sentences $s_i$ according to the scores $S(s_i)$.
if summary length $L$ allows to add a sentence then
    Include the sentence with the highest score into the extract $E$.
    Form a new set of sentences $s'$ by removing the sentence with the highest score.
    Form a new topic signature list $TS'$ by removing words covered by the selected sentence.
    Rank-Sentences($s'$, $TS'$).
end if
```

4. RESULTS AND DISCUSSION

In this section we introduce the procedure we followed in order to assess the summaries yielded by our system. Afterwards we present and discuss the results of this evaluation.

4.1 Evaluation Procedure

Summaries can be evaluated from the point of view of *coverage* (the extent to which a system summary bears on the context of the source text) and *quality* (consistency and chronological coherence estimation). In this work we were only interested in the coverage score.

To perform the evaluation we adopted the method proposed by Lin [17] which is used for the manual evaluation of the summaries submitted to the Document Understanding Conference [5]. This method assumes that the system summary assessment is done by comparison to the human made *model* summary of nearly the same length (665 bytes or about 100 words), both decomposed into units. The closer (as measured in percents) the machine summary is to the model one, the better it is. We were given the model summaries divided into the units as prepared by the DUC'2004 staff. Following the DUC'2004 system, we defined the sentence to be the system summary unit. To express the degree of coverage we used 0% 20%, 40%, 60%, 80% and 100% gradation.

Results of the evaluation of the system extracts with respect to the ideal summaries are presented in the Table 1. The data in the table is organized around the type of the topic signatures based on which sentences for the extracts have been picked. There are three main groups that correspond to the techniques we used to create the topic signatures. Each group is divided into the sub-groups which specify the text unit chosen for the topic signature generation. In that order: document, sentence with the co-referent appearing in any position in the sentence, and sentence with the co-referent appearing in the subject position only. The data is available for each of the ten text sets we summarized. The last column presents the best coverage scores obtained by the summarization systems that participated at DUC'2004[8].

4.2 Discussion of the Results

We noticed that the scores obtained by our system are comparable to some of the best systems competing at DUC'2004. However human summaries are still better than those produced by our or other automatic summarization systems. This can be seen from the table 2 which quotes the scores of the best human made summaries as compared to the human made *model* summaries for the respective persons[9]. Non-model human made[10] and the system summaries have been evaluated against the model in the same way. The results that we have obtained can be explained as follows:

- A human summary is not restricted to the source text sentences. The model units can express information in a concise way. This enables the human made summary to reflect many aspects of the original text even when its length is severely restricted. By contrast, our system extracts whole sentence as a single unit. While the sentence length is accounted for in the sentence weighting formulae, the long sentences are not prevented from being selected for the summary. If it is the case, little summary space remains to include other sentences, even if they could have reflected the content of the remaining model units. As a result, the overall coverage of the summary is affected.

- Humans typically melt information conveyed by a number of the source sentences into one unit. Systems that extracts sentences do not have this ability and achieve only partial coverage.

- Summary assessment is subjective. The Table 2 suggests, that the percentage of among-human agreement does not exceed 72.7%. To illustrate this point, we evaluated our best extract about Günter Grass (13.7% coverage as compared to the model summary) against the non-model human made summary. In this alternative evaluation it has got 29.2%.

[8]The data was provided by DUC [5]

[9]The data in the table was provided by the DUC'2004 conference [5].

[10]At DUC'2004 four human summaries for each set have been prepared. One of them was eventually chosen as a model.

Table 1: **Results of our evaluations compared to DUC'2004.**

Person	# sents	# sents	LR			χ^2			*tf.idf*		DUC'04
	(source)	(extract)	doc	coref	coref-subj	doc	coref	coref-subj	doc	coref	best
Albright	98	5-6	0.323	0.353	0.353	0.261	0.261	0.261	0.138	0.246	0.138
Soros	235	5-6	0.276	0.307	0.323	0.338	0.307	0.323	0.215	0.246	0.229
Samaranch	208	7	0.181	0.236	0.236	0.145	0.236	0.236	0.09	0.109	0.250
Kohl	84	5	0.05	0.125	0.125	0.200	0.075	0.150	0.250	0.075	0.450
Grass	220	6-7	0.125	0.137	0.137	0.137	0.137	0.137	0.125	0.150	0.300
Gandhi	268	6-7	0.327	0.254	0.400	0.327	0.327	0.327	0.327	0.327	0.267
Tutu	149	5-6	0.375	0.275	0.325	0.375	0.375	0.350	0.400	0.450	0.340
Chamberlain	371	10-12	0.400	0.300	0.380	0.280	0.240	0.400	0.160	0.360	0.400
Carter	237	6-7	0.345	0.163	0.200	0.290	0.254	0.254	0.163	0.200	0.291
McDougal	201	4-5	0.433	0.433	0.433	0.300	0.316	0.316	0.283	0.500	0.483

Table 2: Maximal agreement between human summaries.

Person	Best coverage by humans
Albright	0.385
Soros	0.371
Samaranch	0.417
Kohl	0.675
Grass	0.512
Gandhi	0.450
Tutu	0.50
Chamberlain	0.60
Carter	0.727
Mcdougal	0.717

- Disparity of the human summaries implies that there exist various opinions on how the answer to the question *Who is the person X?* should look like. According to the model summaries we were given, it had to include person's birth date and place, biographical trivia facts, and chronological report on events she/he was involved to, with the stress on dates, places and people names.

  Dates were often absent in the system extracts which reduced their coverage score. This happens since cardinal numbers were not among the items selected for the topic signature. We performed a quick check adding the cardinals tag (CD) to the items selected for the topic signature. However this did not lead to the inclusion of important dates into the topic signature, those that were included had low weights and eventually did not influence the sentence choice. This is easily explained since the dates (especially the more recent ones) occur too often in the whole corpus to be selected as a representative feature or candidate for a strong collocation. In the Section 5 we will give some ideas on how to improve coverage of the dates. Another observation we can make is that while names of places and people appear often in both - the topic signatures and system extracts, the latter almost never mention person's place of birth or other biographical trivia facts. If a human considers this information important she/he will include it to the summary even if it was marginally mentioned in one of the texts about a person. However statistical methods will fail to assign significance to such single events.

So far we were trying to determine the reasons of the modest coverage achieved by our system.

Now we would like to pass to the comparison of the system performance when using different procedures for the topic signature generation.

- **Technique-wise comparison.** We observe that in most of the cases (17 out of 20) the topic signatures created on the basis of the text unit "sentence" provided slightly better material for the sentence choice than topic signatures created on the basis of the text unit "document". We think that two factors might lead to this result. First is the increase of the corpus size (from 500 documents, with document as a text unit, to 10083, with sentence as a text unit), and statistical methods are known to improve their performance when applied to larger amounts of data. Second factor is the change of the word frequencies with the change of the text unit. Based on the text unit "document", our analysis is blind to the word occurrences within the documents. By contrast, working on the sentence level may help to spot frequent words within a single document. We also would like to emphasize the importance of work with co-referenced sentences, since it extends the base of sentences relevant to a person beyond those sentences where the person's name is explicitly mentioned. Looking at the Table 1 we can also notice that if there were changes in the performance with respect to the co-referent position in the sentence, the better results have been obtained with the coreferent in the subject position. This could be explained by the document narrative form: using co-referent in the subject position might better approximate the majority of cases.
- **Cross-technique comparison.** To extract topic signatures we chose two techniques that emphasize the importance of the term co-occurrence within a given context (i.e. the likelihood ratio and χ^2 tests) and one technique that aims to identify the most important topic words in a given context (i.e. *tf.idf* weighting scheme). The *tf.idf* method was expected to provide a baseline for the evaluation of the system extracts. This is because it does not constrain the terms (in our case person name and other non-stop words in the corpus) to co-occur in the same text unit. Consequently we expected that the terms in the topic signature created with this technique would be salient for the relevant set of texts but less focused on the person in

comparison with the two other techniques. Focus on the person was an important requirement because as we mentioned in the section 3 different texts within the set did not provide equal amounts of information about the person. Consequently we expected the extracts generated on the basis of these topic signatures to be less responsive to the question than the two other sorts of extracts. Results shown in the Table 1 prove our expectations in 7 out of 10 cases.

It has been argued that the likelihood ratio test is better suited for the natural language modeling than the χ^2 test [6, 26, 29]. When choosing both techniques for the topic signature generation we aimed to test this observation. However the small corpus size provided by DUC seems to be insufficient to reach the definite conclusion in this respect.

5. CONCLUSIONS

In this work we studied the topic signature approach to the multidocument *question answering* summarization. We produced summaries that had to answer a question of the form *Who is X*, where *X* was the person. The summaries had also to meet a length requirement and be no longer than 665 bytes which corresponds approximately to 100 words. To create the topic signatures we used three different techniques: likelihood ratio and χ^2 tests and *tf.idf* weighting scheme. We constructed topic signatures based on the document and sentence being defined as a text unit.

Our results showed that the topic signature approach allows to produce question answering multidocument summaries whose coverage is comparable to the coverage obtained by the best systems competing at the Document Understanding Conference [5]. However there remains a large gap between the automatic and human made summaries.

With regard to the topic signatures, we found that using the likelihood ratio or χ^2 tests for the topic signature generation is better than the *tf.idf* weighting scheme in most of the cases. We observed also that in the majority of cases topic signatures generated on the basis of a sentence (with the co-referent resolution) being set as a text unit gave better material for the summary creation.

The main limitation of the topic signatures approach is that it only allows to capture information that was emphasized in the source texts. In contrast, human summaries may contain details that were just briefly mentioned in one or some of the original texts. Humans can also synthesize information from several different sentences/documents, while a sentence extraction system does not have this capability.

5.1 Future Improvements

- Our analysis of the model and non-model human summaries showed that dates and biographical trivia are important components of the answer to the question *Who is the person X?*. However if the source text is not a biography in itself, this data though present, will not be stressed and thus will often be missed by the statistical analysis. It seems that combined approach that integrates topic signatures for the important content identification, with the promotion of terms typical for biographies (such as *born in*, *married to*, *child of* etc.) could be fruitful.

- Another important feature of the life course description is the presence of dates. Currently, dates may appear in our summaries if they occur in the sentences selected for the summary. But the system has no guidelines to prefer sentence that contain both, event and its date, to another sentence that mentions the event only. In our present version the topic signature terms were uni-grams. It seems that extension of the topic signature terms from uni to n-grams is one of the possible ways to facilitate inclusion of dates to the summary. If an event which is important for the person, is systematically mentioned in the source text along with the date, it could be identified as a collocation and extracted for the topic signature term (in that case the term will be a bi-gram). It may also be useful to combine day/month/year items into the high-level concept of "date". This will allow for more focused extraction of dates in the cases when text provides the full date and not just the year of the event.

- Comparison of the scores obtained by our and other summarization systems at DUC suggests that the topic signatures can be used effectively in detecting of prespecified information in the source texts. In this light it might be interesting to evaluate our summaries in a question answering setting, testing how well the question answering system can answer questions using these summaries and different versions thereof.

- The length limit set to 665 bytes allows to use only part of the topic signature terms (approximately 35%). It seems to be important to generate a summary that is not restricted to any specific length, but uses up the whole topic signature up to the cutoff weight. The coverage score obtained by such a summary will indicate an upper bound for the topic signature approach to the summarization. This will also allow to see directly what influence the size of the extract has on the quality of the summaries produced by the system. From this information we could learn which extensions are necessary to improve the overall performance.

- As mentioned in Section 3.4 our current redundancy filtering algorithm removes the already covered topic-signature words from the topic signature in order to rank the remaining sentences. Another option could be to apply a decay factor (for example, dividing the weight of the word by a constant each time the word is covered by the new extract sentence). This method could be of interest in cases of longer summaries.

- As mentioned in Section 3.5 our algorithm of extract generation will work even if run on the full corpus and not only on a relevant set of documents about the person *X*. It might be interesting to check performance of this algorithm in cases when the boundary between relevant and non-relevant documents is not defined.

6. REFERENCES

[1] Roxana Angheluta, Rudradeb Mitra, Xiuli Jing and Marie-Francine Moens. *K.U.Leuven summarization system at DUC 2004*. In DUC Workshop Papers and Agenda, pp. 53-60, Boston, 2004.

[2] Regina Barzilay, Michael Elhadad. *Using lexical chains for text summarization.* In I. Mani and M. Maybury (eds), Advances in Automated Text Summarization, pp. 111-123. Cambridge, Massachusetts: MIT Press, 1997.

[3] Ron Brandow, Karl Mitze and Lisa F. Rau. *Automatic condensation of electronic publications by sentence selection.* Information Processing Management, 31(5), pp. 675-685, 1995.

[4] Jaime Carbonell, Jade Goldstein. *The use of MMR, diversity-based re-ranking for reordering documents and producing summaries.* SIGIR, pp.335-336, 1998.

[5] *Document Understanding Conference,* `http://duc.nist.gov`

[6] Ted Dunning. *Accurate methods for the statistics of surprise and coincidence.* Computational Linguistics, 19, pp.33-64, 1993.

[7] H.P. Edmundson. *New methods in automatic extraction.* Journal of the ACM 16(2), pp.264-285, 1968.

[8] Jade Goldstein. *Automatic text summarization of multiple documents.* Thesis Proposal. Carnegie Mellon University: 1999.

[9] Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, Jaime Carbonell. *Summarizing text documents: sentence selection and evaluation metrics.* SIGIR, pp. 121-128, 1999.

[10] Eduard Hovy, Chin-Yew Lin. *Automated text summarization in SUMMARIST.* In I. Mani and M. Maybury(eds), Advances in Automated Text Summarization, pp. 80-98. Cambridge, Massachusetts: MIT Press, 1997.

[11] Kevin Knight, Daniel Marcu. *Statistics-based summarization - step one: sentence compression.* In Proceedings of the 17th National Conference of the American Association for Artificial Intelligence AAAI'2000, 2001.

[12] Jullian Kupiec, Jan Pedersen, Francine Chen. *A trainable document summarizer.* In Proceedings of the Eighteenth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 68-73, 1995.

[13] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, Chin-Yew Lin. *Question Answering in Webclopedia.* In Proceedings of The 9th Text Retrieval Conference (TREC9), pp. 13-16, 2000.

[14] Chin-Yew Lin. *Robust automated topic identification*: PhD dissertation. University of Southern California, 1997.

[15] Chin-Yew Lin, Eduard Hovy. *The Automated acquisition of topic signatures for text summarization.* In Proceedings of the 18th International Conference on Computational Linguistics, 2000.

[16] Chin-Yew Lin, Eduard Hovy. *Automated multi-document summarization in Neats.* In Proceedings of the Human Language Technology Conference, 2002.

[17] Chin-Yew Lin, Eduard Hovy. *Manual and automatic evaluation of summaries.* In Proceedings of the Workshop on Automatic Summarization post conference workshop of ACL-02, 2002.

[18] H.P. Luhn. *The automatic creation of literature abstracts.* IBM Journal, pp.159-165, 1958.

[19] Christopher D. Manning, Hinrich Schutze. *Foundations of statistical natural language processing.* Cambridge, Massachusetts: MIT Press, 2002.

[20] Daniel Marcu. *From discourse structures to text summarization.* In I. Mani and M. Maybury(eds), Advances in Automated Text Summaries, pp. 82-88. Cambridge, Massachusetts: MIT Press, 1997.

[21] Katleen McKeown, Dragomir R. Radev. *Generating Summaries of Multiple News Articles.* In Proceedings of the Eighteenth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 74-82, 1995.

[22] Marie-Francine Moens, Roxana Angheluta, Jos Dumortier. *Generic technologies for single and multi-document summarization.* Information Processing and Management, 42(3), pp. 569-586, 2005 (to appear).

[23] Jane Morris, Graeme Hirst. *Lexical cohesion computed by thesaural relations as an indicator of the structure of text.* Computational Linguistics, 17(1), pp. 21-48, March:1991.

[24] Joel L. Neto, Alexandre D. Santos. *Document clustering and text summarization.* In PADD: 4th International Conference on Practical applications of Knowledge Discovery and Data Mining. London:2000

[25] Ted Pederson. *Fishing for Exactness.* In Proceedings of the South-Central SAS Users Group Conference (SCSUG-96), 1996.

[26] David A. Smith. *Detecting and browsing events in unstructured text.* In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 73-80, 2002.

[27] Simone Teufel, Marc Moens. *Sentence extraction as a classification task.* In Proceedings of the Workshop on Intelligent Scalable Summarization. ACL/EACL Conference, pp. 58-65, 1997.

[28] Jinxi Xu, W. Bruce Croft. *Query expansion using local and global document analysis.* In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 4-11, 1996.

[29] Yiming Yang, Jan O. Pedersen. *A Comparative Study on Feature Selection in Text Categorization.* In Proceedings of the Fourteenth International Conference on Machine Learning, pp.412-420, 1997.

User Intentions in Information Retrieval

Matthijs Bomhoff
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
m.j.bomhoff@alumnus.utwente.nl

Theo Huibers
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
huibers@cs.utwente.nl

Paul van der Vet
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
vet@cs.utwente.nl

ABSTRACT

We present a study on the possible use of user intentions in information retrieval. First, the feasibility of automatic categorization of user queries with respect to their intention is discussed and tested. The experiments show that it is probably possible to achieve better retrieval if user queries are first classified, after which a dedicated retrieval method performs a search. Further research could be directed at better algorithms, other classifications and other domains.

Keywords

information retrieval, user intention, machine learning

1. INTRODUCTION

To use an information retrieval system, a user is almost always required to enter one or more keywords to search for. One of the main problems with this is, that the system is usually incapable of making sense of the meaning of these keywords and can just use them to somehow search an index of a document collection. In order to overcome this, we investigate to what extent *user intentions* could be used. A *user intention* is here defined as the information need of the user, as opposed to the representation of this need by means of his query.

As representation of the exact need would be very difficult, if not impossible, we restrict ourselves to categorization of intentions according to predefined categories. After selecting a suitable set of intentions, we describe several experiments in which automatic categorization of queries is attempted using features extracted from the queries and their contexts.

We have inspected only queries as they are routinely entered by users of web search engines. The aim has been to automatically capture context information from easily accessible environmental features such as parts of the URLs in the domain we studied, the day and time at which the query was submitted, and part-of-speech information. In this respect, the queries studied in this research differ significantly from the queries studied at the HARD track of TREC 2003 [3] (the reports of TREC 2004 were not yet available at the TREC website at the time of writing this article).

After describing these experiments, we try to formulate several conclusions regarding the results. Some possibilities for further research in this relatively unexplored area of information retrieval are mentioned, followed by an overall conclusion.

This article is a shortened version of the Master Thesis report of Matthijs Bomhoff [4], in which most aspects of this project are treated in more detail.

2. RELATED WORK

Most of the work in this project has been inspired by the work of Kang and Kim [6]. This article will first be briefly discussed in order to create a setting for the work done in this project.

2.1 Intentions

Kang and Kim [6] describe a project that somewhat resembles the goals of this project. The set of user intentions they use comes from a paper by Broder [5]. This work by Broder is first briefly discussed.

Broder divides user information finding tasks (user intentions) into three categories:

topic relevance (informational) The user wants to find information on a specific topic.

homepage finding (navigational) The user is looking for a homepage of a specific person or organization.

service finding (transactional) The user requires a certain service and is searching for a site that offers this service.

Broder has attempted to ascertain the correctness and completeness of this categorization by two means: he took a look at a log from the search engine Altavista, and he analyzed the results of a survey among random users of that same search engine. Both of these techniques for confirmation are doubtful in their respective ways (Broder also recognizes and admits this himself):

The analysis of the survey is subject to questions about its statistical significance. As Broder notes, the survey only has a (self selected) response of about 10%. This can severely influence the results, as certain users are less likely to fill in the survey. (Broder acknowledges this with the example of sex-related queries: only about 1% of the filled-out surveys mentioned sex-related goals, whereas about 10% of the queries were sex-related in that same period.)

The analysis of the query log on the other hand, does not suffer from the problem of statistical significance the way the survey does. Quite another problem arises however, as it is quite hard to determine user intention from just the query without context information for such a broad setting.

2.2 Classification

Kang and Kim [6] drop the service-finding task for their research and focus instead on discriminating between - and improving the handling of - queries for the topic relevance and homepage finding tasks.

First they establish that different searching and ranking strategies are optimal for the two different information needs. This makes it clear that categorizing the queries with respect to their information need is a useful technique, as it enables use of the right strategy on a per-query basis.

After having shown the use of a system that can discriminate between these two information needs, the authors constructed two databases of documents, one with homepages, and one with non-homepages. They constructed language models that describe the differences between these corpora, and attempted to fit queries to both language models. They found that queries with a certain need match better on the language model of pages that satisfy their need, than on the model of the other type of pages. The language model itself is based on four aspects of the documents:

- occurrence of words in either document set
- co-occurrence of words
- usage of words in anchor texts
- POS (Part Of Speech) information

Using this technique for differentiating the two types of queries, the authors go on to assess the performance of the system as a whole regarding information retrieval performance. The overall performance of the system (both precision and recall) is reported to be better when deciding upon and using different search strategies for the two query types compared to using one unified strategy as is usually done in information retrieval systems. This leads the authors to the conclusion that further analysis of user queries is beneficial to the performance of the information retrieval system at hand.

2.3 Conclusion

One of the most interesting things about [6] is the way they choose their possible intentions (informational, navigational and transactional). These categories come straight from [5], but Broder does not explain why these intentions were chosen. This is not necessarily a bad thing, as the categories do seem to be useful for the settings that both [6] and [5] use.

In conclusion, [6] comes close to what our project hopes to achieve. Some possible points of improvement or extension have been touched on already in the discussion above, and some more will follow below.

Overall there is not that much prior work on this subject to be found, so there are a lot of options for research left. The work by Kang & Kim [6] was be used as inspiration, but we took a different approach.

3. USER INTENTIONS

It seems to be quite hard to find good categorizations for *user intentions*, or *tasks*. Only a few interesting articles were found during a search of several online article collections. Several of them contain roughly the categories *informational* and *navigational* (also resp. known as *informative* and *entry page*). As this set of intentions was already discussed in 2.1, it will not be discussed again here.

The choice for the intentions from [5] (informational, navigational, transactional) seems to be the best as there is quite some research available that uses these (or very similar) intentions for a specific task (i.e., improving results for homepage finding in a search engine).

This choice for intentions also has another advantage specific to this project: the number of intentions (three) is more than two, so it is not simply a matter of separability with respect to a single threshold in a single dimension, but it is also not much harder. It is therefore still possible to oversee the categorization by hand.

4. EXPERIMENTATION

Several experiments will be discussed that have been performed to determine if it is possible to actually categorize real-life queries using readily available information. The goal of these experiments was to test to what extent this categorization can be automatically performed using machine learning techniques on easily constructed feature vectors. If it is shown that the categorization can be performed with a reasonable accuracy, it is then possible to consider use of these techniques to improve already existing information retrieval systems.

4.1 Setting and Data

The experiments are performed using several machine learning algorithms from the Weka toolset (see [2]), as this toolset contains overall well-performing versions of most popular machine learning techniques.

In order to carry out experiments with machine learning, a manually tagged set of data is required as training and test set. This sets an important requirement for the experimental setting: the person who performs the manual tagging must to a certain extent be familiar with the domain of the queries in order to be able to guess what the users were trying to find.

In order to meet this requirement, a logfile covering approximately three months with queries collected at the search engine on the website of the *Universiteit Twente* [1] has been selected as the domain. For privacy reasons this file is not publicly available. It is not unreasonable to expect the queries entered on that site to have a strong relation to the university and its campus, a domain quite familiar to the authors.

4.2 Feature Extraction

From a log file containing 11410 lines with valid queries, several features were distilled. These are enumerated below. The name between square brackets is the name of the feature as it appears in the tables with results.

Term Count [term_count] The number of terms in the query.

DNS Parts [dns_parts] The fraction of the terms that occur as part of a hostname in the university domain.

URL Parts [url_parts] The fraction of the terms that occur as part of the directory part of an URL in the university domain.

Day of Week [dayofweek] The day of the week the query was submitted.

Work time [worktime] Does the timestamp of the query fall into "office working hours".

Browser [browser] The browser used for submitting the query.

Language [language] The language the university site was being viewed in (Dutch or English).

Lexical information [celex_*number*] Using the Dutch Celex lexicon, possible word classes for query terms were determined. These features represent the likeliness of terms in the query belonging to each of the Celex word classes:

- **Expression (celex_0)** Words that only occur as part of an expression, e.g. *voorschijn* (this hardly ever occurs in English)
- **Noun (celex_1)** Nouns, e.g. *bank* (*bank*)
- **Adjective (celex_2)** Adjectives, e.g. *groot* (*large*)
- **Numeral (celex_3)** Numerals or other quantifiers, e.g. *meer, zes* (*more, six*)
- **Verb (celex_4)** Verbs, e.g. *verlaat* (*leave*, as in "I leave now")
- **Article (celex_5)** Articles, e.g. *het* (*the*)
- **Pronoun (celex_6)** Pronouns, e.g. *ik* (*I*)
- **Adverb (celex_7)** Adverbs, e.g. *fijntjes* (*nicely*)
- **Preposition (celex_8)** Prepositions, e.g. *over* (*over*)
- **Conjunction (celex_9)** Conjunctions, e.g. *maar* (*but*)
- **Interjection (celex_10)** Interjections, e.g. *asjemenou* (*yeah*)

4.3 Training & Test set

To use supervised machine learning techniques, a manually tagged set of queries is needed. To this end, a set of 169 random queries was tagged by hand with one of the categories (informational, navigational & transactional). Actually, the informational queries were selected randomly, the navigational and transactional queries are partially tagged by hand by adding tags for some frequently used services and homepages on the university network, to increase the number and diversity of entries in these categories. These manual tags are put into a file containing a mapping from query string to category. As some queries occur more than once in the log file, these "doubles" are also tagged, yielding a set of 1736 tagged entries.

The duplicates are used as it is reasonable to assume that identical queries have identical intentions in most cases by far. This use of duplicates has two main advantages. First, it makes tagging a lot of queries in a short time possible. Also, the use of duplicates makes creating a set with a high diversity for certain features easy for the classes that contain only a few distinct query strings with a lot of duplicates.

It is important to note that this way of constructing the training and test set is not focussed on being representative for the data set, but rather on the diversity required for successful machine learning. In practice, the amount of informational queries will probably be a lot higher, considering the trouble we had to find enough navigational and transactional queries to work with. For the experiments that follow, a subset of this set was used with an equal number of entries per category, to get a nice baseline of 33.3% performance for the machine learning algorithms to improve upon.

The fact that it is so hard to find enough *navigational* and *transactional* queries is a strong indication that other sets of intentions will have to be devised for certain settings. Apparently the set of intentions coined by [5] and also frequently encountered in other projects, is not to be the perfect choice for the data set at hand.

4.4 Measuring Performance

For comparison of several experimental setups, a common criterion for optimization has to be defined. Sometimes, in discussions of classifiers, precision and recall are the two all-important values in an information retrieval context. The problem of using precision/recall is however that this provides two scalars per experiment, so it is hard to optimize overall. For that reason, it was decided to use only the fraction of the set that is correctly classified as measure. This measure is called from here on *performance*, as is common in machine learning. (For example: if half of the set is classified correctly, the performance is simply 50%.) The choice for this measure was made as only correctly classified queries are any good for trying to improve information retrieval overall precision or recall in a complete system. Precision and recall values are however often discussed in the text accompanying the experiments, to clarify how the overall performance is reached and what the impact of this on a complete system could be.

4.5 Baseline Performance

feature	*performance*	*feature*	*performance*
term_count (A)	38.8%	celex_2 (J)	35.1%
dns_parts (B)	52.7%	celex_3 (K)	33.2%
url_parts (C)	46.9%	celex_4 (L)	45.3%
dayofweek (D)	33.4%	celex_5 (M)	33.8%
worktime (E)	36.9%	celex_6 (N)	33.2%
browser (F)	34.3%	celex_7 (O)	33.4%
language (G)	33.9%	celex_8 (P)	33.4%
celex_0 (H)	33.5%	celex_9 (Q)	33.2%
celex_1 (I)	56.4%	celex_10 (R)	33.2%

Table 1: Performance based on single features

An important measure for machine learning applications is the performance improvement over the *baseline*. The baseline consists of the amount of categorizations that one would correctly guess, when assigning the most common category to all entries. These values are easily calculated by hand. For the set with equal amounts of queries for each of the three categories (the one that will be used for the experiments), the baseline performance is 33.3%.

4.6 Single Features

The first experiments that give some insight into the "structure" of the data, are simple experiments using the classifier called *OneR* in the Weka tool set. This classifier searches for an optimal classification based on a single feature's value.[1] When the *OneR* classifier is used on the normalized set, the values for the performance in table 1 were obtained for the different features. Some of the numbers are slightly below the 33.3333 baseline that was expected. This is due to the cross-validation: as the model is built on one part of the set and the evaluation takes place over the other part, it is possible that the model has a lower than baseline performance on the part that it was not constructed on. These differences are only marginal though and are therefore acceptable. These results are also plotted in figure 1 for more clarity, the letters in this figure correspond to those in the table.

In conclusion, the results from this section are mixed: some features seem to have quite a good correlation with certain intentions. Some caution is however required, as the *celex_1* and *celex_4* examples show that some features might look significant at first sight, but happen to be based simply on an unforeseen pattern in the data set. In the University of Twente domain there are a number of very popular applications, the names of which happen to coincide with normal Dutch verbs or nouns. After having determined two features that clearly exhibit this problem, we now take a look at the (more interesting) combinations of the other features.

4.7 Two Feature Synergy

Far more interesting than the performance that can be achieved using only one feature, is the result that can be obtained by combining two or more features in the decision making process. But before all the features are thrown into the mix, it is nice to take a look at how well a combination of just two features can get by. All possible combinations of two features have been tried, but as there are a total of 153 possible combinations, not all of them will be discussed here. Only some of the more interesting (those that are either very successful, or show noteworthy patterns in the input) are discussed in what follows. The performance is measured by using the J48 (C45) learning algorithm with pruning and 10-fold cross validation. Furthermore, all combinations containing either *celex_1* or *celex_4* have been ignored, as they have already been dismissed as containing some form of false success. Combinations with either one of these features scored quite high, some even higher than the combinations discussed in more detail here, but a look at their respective decision trees always hinted very strongly at the problem with the acronyms encountered before. They are therefore omitted here, so that they do not distract attention from other, more interesting results. The results of the experiments that are discussed can be found in table 2.

feature 1	*feature 2*	*performance*
dns_parts	url_parts	59.7%
term_count	dns_parts	57.3%
dns_parts	dayofweek	53.8%

Table 2: Performance based on two features (J48 learning)

After running several experiments with two features at a time, two conclusions appear to be justified:

- Some combinations of two features outperform each of these features separately. This was of course to be expected, as the some of the classifiers from the previous chapter favored different classes. And of course, there is simply more information available when more features are used.

- Features that take the context of the document collection into account (such as *dns_terms* and *url_terms*) seem to perform better than features that are only affected by the context of the query (such as *dayofweek* or *worktime*). Combining both of these categories is however good for the performance. For example: *dns_parts* combined with *term_count* seems to work better than *dns_parts* on its own.

After gaining insight into the synergy between two features at a time, the time has come to make full use of the complete feature vectors. The next section investigates the benefits of using more than just two features.

4.8 All features together

After looking at the possible performance of individual features and combinations of two features at a time, it is interesting to see if the use of all features together can still improve the results. To this end, a set with feature vectors consisting of all features, except for *celex_1* and *celex_4*, and a set consisting of all non-lexical features were constructed.

[1]The results of this should be comparable to straight forward statistical analysis, using correlations, but as the WeKa tool set is being used for other experiments, why not use it here, instead of using a statistical package to obtain the same results.

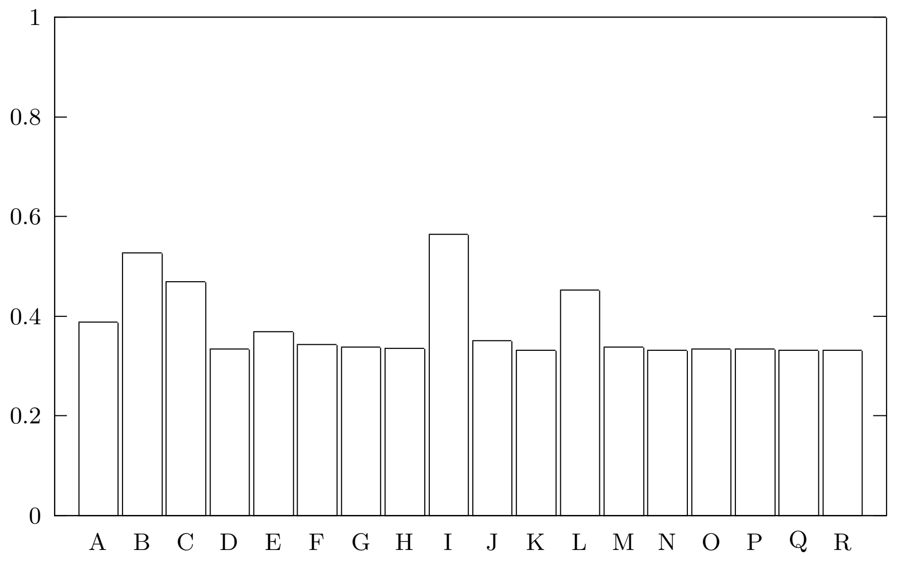


Figure 1: Performance based on single features

features	*performance*	*I prec.*	*I rec.*	*N prec.*	*N rec.*	*T prec.*	*T rec,*
dns_parts & url_parts	60.1%	57.7%	97.9%	95.9%	18.3%	57.6%	64.1%
all but lexical	59.9%	57.7%	97.4%	90.0%	18.8%	57.7%	63.6%
all but celex_{1,4}	60.6%	58.6%	96.9%	89.0%	21.2%	57.5%	63.9%

Table 3: Performance for many features. (Category names abbreviated to I(nformational), N(avigational) and T(ransactional).)

The results of the experiments are shown in table 3. This table contains both the overall performance, and the precision and recall per category. It is obvious that the use of more than the two features that perform best together does not really amount to an improvement. The differences are so small that they cannot be regarded as significant.

The very high precision and low recall for the navigational category is probably caused by the fact that query terms that occur in hostnames (dns) or URLs are a good indicator for navigational queries, but not every navigational query can be classified this way as not all documents have hostnames or URLs that users expect, hence the low recall.

As the comparison of the combination of two and more features shows, adding more features is not a good thing per se. Adding all features other than lexical features leads to a slightly lower performance, but the difference can hardly be called significant. Adding the lexical features (except for *celex_1* and *celex_4*) exposes *celex_2* as yet another lexical feature that seems to be influenced heavily by patterns in the domain, so this performance gain cannot be attributed to using more features either.

In conclusion, it seems that the performance of about 60% is a maximum value for the data set at hand when we try to compensate for spurious information as provided by some of the lexical features.

4.9 Other Machine Learning Techniques

After comparing several different (combinations of) features, it is also interesting to explore other machine learning techniques, to see if the performance achieved by them is significantly different from that achieved using J48 decision trees. It was decided to repeat the experiments described in section 4.8. The choice for these experiments was made as most machine learning techniques ought to perform quite well on just two features. It gets really interesting to see what they do when more features are added. The three experiments from section 4.8 are repeated using three different machine learning techniques:

J48 (J48) The decision tree learning from all the previous sections. Included here for reference.

Multi-layer Perceptron (MLP) A neural network containing one hidden layer.

Naive Bayes (BAY) A simple Bayesian network.

For the *Naive Bayes* classifier, the standard parameters from the Weka tool set were used. For the *Multi-layer Perceptron* the 10 fold cross-validation was dropped in favor of 2 fold cross-validation, to speed up the process a little as training neural networks is by nature very slow. This makes the results a bit less accurate, but that is acceptable as in this section the focus is on relatively large differences in performance.

algo.	*features*	*performance*	*I prec.*	*I rec.*	*N prec.*	*N rec.*	*T prec.*	*T rec,*
J48	dns & url	60.1%	57.7%	97.9%	95.9%	18.3%	57.6%	64.1%
J48	all but lexical	59.9%	57.7%	97.4%	90.0%	18.8%	57.7%	63.6%
J48	all but celex_{1,4}	60.6%	58.6%	96.9%	89.0%	21.2%	57.5%	63.9%
MLP	dns & url	59.9%	57.8%	97.1%	94.6%	18.3%	57.2%	64.4%
MLP	all but lexical	59.4%	58.7%	90.1%	63.9%	26.4%	58.7%	61.8%
MLP	all but celex_{1,4}	57.3%	60.3%	74.6%	55.0%	33.2%	55.4%	64.1%
BAY	dns & url	59.1%	57.3%	94.2%	84.3%	18.3%	56.8%	64.7%
BAY	all but lexical	58.9%	58.0%	91.6%	82.2%	19.4%	55.4%	65.7%
BAY	all but celex_{1,4}	50.6%	63.9%	39.0%	81.5%	19.6%	43.4%	93.2%

Table 4: Performance of different algorithms

The results of the comparison are displayed in table 4. Most of the numbers in this table are not very astounding, there are some interesting details though that will be discussed briefly.

The first thing to note is how the two new algorithms seem to perform a bit worse when more features are added. For the *Multi-layer Perceptron*, the difference is not all that great and can be attributed to other factors, but for the *Naive Bayes* classifier, the performance on all features except for the two notorious celex features is over 8% worse than the performance on just the *dns_parts* and *url_parts*! The reason for this may be the assumption this classifier makes about the distribution of the values for the features. The *Naive Bayes* classifier assumes a normal distribution for all numerical values, whereas this is not the case.

Another interesting number in table 4 is the relatively high recall rate of the *Naive Bayes* classifier for transactional queries in the last column. It appears that the addition of extra features has caused a shift of focus from high recall in the informational category, to high recall in the transactional category. Unfortunately, this shift was not very beneficial to the overall performance. As with the previous point, it is not clear why this happens when more features are added.

The last interesting point is not in obvious differences but on the contrary in the numbers that are remarkably the same: when looking at the overall performance, all algorithms achieve a peak performance of about 60%. If we compare the performance on *dns_parts* and *url_parts* only, all algorithms perform about equally well with the Bayesian network a bit on the low end of the spectrum. It is nice to see that these three different algorithms have almost identical precision and recall rates for the three different categories as well. This could be an indication that the performance for the different categories is mostly determined by the data set and not so much by the algorithm used, which is a good thing.

In the comparison of three different algorithms, it is interesting to note that all algorithms perform about equally well on the combination of the top two features, which is where they also seem to reach peak performance. It is not unthinkable that for all three algorithms some performance can still be gained by tweaking the parameters or by using slightly different versions of the learning algorithms, but the focus of this project is not the top notch performance of one specific algorithm, but rather a meaningful comparison between several aspects of the classification problem. When real peak performance is required for a real-life application of the technology, further research will have to be done. For now it suffices to see that the choice for one of these three commonly used techniques does not have very much of an effect on the best performance that can be obtained using overall best parameters.

5. EVALUATION

5.1 Evaluation of results

After presentation of the results of the experiments and drawing simple conclusions based on direct results, it is time to take a look at the overall results. Taking into account that the best performance achieved during the experiments was about 60%, the performance on *dns_parts* alone is quite high with over 52%. Some extra performance is gained when the *synergy* between multiple features is exploited, but this gain is quite small compared to the gain with respect to the baseline that is achieved using one well-chosen feature.

These characteristic features also seem to be more related to the context of the information retrieval system, than to the context of the query. For example, the URLs and DNS entries that occur in the context of the information retrieval system give a strong hint towards the intention of a query when they occur as a term. The features that are more related to the context of the query, such as *dayofweek* or *language*, are only marginally helpful for determining user intention.

Another observation to make is that certain intentions appear to have a higher precision, while some others are more related to a high recall. This is almost unrelated to the features or algorithm used. (The exception to this rule is constituted by some single features that happen to be good at recognizing a single class.) For example the *informational* class is almost always related to a high value for recall, whereas the other two classes are usually related to relatively high precision values. Therefore, some cases of the other classes are probably easier to recognize. Intentions that are easy to recognize in some cases, are likely to have a high precision, as most entries will be correctly classified by assigning the one class that is hard to recognize to all the entries that are left over after picking the easy ones out.

If we combine these two conclusions (which is justified by looking at the results of the experiments), we come to the conclusion that for the intentions *navigational* and *trans-*

actional, the features *dns_parts* and *url_parts* (two features from the context of the system), are strongly related to recognizing some instances with quite a high precision. One plausible explanation for this is that these features are based on the way the university names the locations of the documents that are related to such queries. This naming seems to be quite consistent with the queries that are used to look for these documents. The opposite also seems to be true: documents related to *informational* queries almost never contain terms that appear in DNS records or in the path component of URLs. The naming that is used for different kinds of documents is therefore also useful for determining query intentions, at least for this domain.

5.2 Representativeness and Portability of Results

After analyzing the results, some comments on their representativeness for this data set and on their portability to other domains are in place. These comments will be discussed below.

5.2.1 The Domain

The first comment that needs to be made regarding the representativeness of the results of this project, is the source of the data that was used. The impact of the use of a data set from the university website regards the applicability of the results of this project to other settings. The *navigational* and *transactional* queries are mostly distinguished using properties of these queries derived from the context of the system, in this case DNS and URLs. In other settings, the location and naming of these documents might not be as clear, severely lowering the applicability of the "star features" from this project.

5.2.2 Duplicate Queries

The hardest aspect of the representativeness of the results to judge is the use of duplicate queries in the training and test set. The question of course is: to what extent are results obtained on this set useful for drawing conclusions about the entire data set? At first sight, the use of duplicates decreases the representativeness of the experiments, as some features are directly influenced by the query string, and the distribution of the values of these queries is no longer representative when doubles are used.

However, the categories that benefit most from duplicates are *navigational* and *transactional*, and we estimate that we have tagged about 50% of the total amount of the queries from these categories.

5.2.3 Equal Distribution

In order to see clearer differences in performance for the different classes, the training and test set is normalized to a set with an equal number of entries per intention. The main advantage of this is, that no category is favored when comparing features and algorithms that are good at recognizing a single category.

This does however severely impact the applicability of the results from these experiments to other settings. In the real world, queries are not equally divided between intentions. In the logs from the university website, *informational* queries are by far the largest category, with *navigational* and *transactional* queries playing only a minor role. [5] states that *informational* queries make up for about half the total amount of queries in a web search engine, with the other half of the queries divided roughly equally over the other two categories. This makes the distribution of the queries over the intentions quite different, so the results for such settings might be quite different.

This research attempts to show which features are good at recognizing what classes for the setting at hand, so even though the conclusions cannot be directly applied to other situations, it is likely that certain features can still be used to discriminate between certain classes. This project can thus provide a basis for further investigations into specific settings by suggesting which features might be useful for the recognition of different intentions.

5.2.4 Conclusion

Overall the representativeness and portability of the results cannot be guaranteed, but taking away the strongest doubts has been attempted. While the exact results can in no way be used to make quantitative claims, qualitative conclusions can be justified and will be drawn in the course of the next section.

6. FURTHER RESEARCH

6.1 Introduction

As this project is only an exploratory study into the possibilities of user intentions in the field of information retrieval, it leaves a lot of questions unanswered. Other projects could (and hopefully will) take up the challenge of further investigating these possibilities and coming up with better ways of using this extra source of information. Several possible directions for further research are lined out below.

6.2 Other Categorizations

One of the most fundamental decisions made during this project is the choice for the intentions that are used. The choice for *informational*, *navigational*, and *transactional* was a good one for the situation at hand, although the informational category occurs quite often compared to the other two. For most situations and applications, though, different categorizations are probably better.

Further research is needed to determine possible sets of intentions that perform well in specific settings, or perform well overall. For most applications, a specific set of intentions will usually perform best, but determining this set can be too expensive, so it would be nice to have some reference sets of intentions that tend to perform well in most situations. These sets need to be determined by further research.

6.3 More or Other Features

Another area that certainly needs more research is that of the set of features that can (and should) be used to perform categorization. In this project, the choice of features was largely dictated by the available information and the ease of use of that information. There are however many more possible features that could be considered, for example:

- External contextual information. For example the weather conditions or stock market developments.
- Client connection endpoint. For this project, it seemed too hard to actually do something with this, but for subsequent projects, it might be possible to try and draw conclusions based on certain aspects of the client address, such as: country code in hostname or connection type.
- Chronological context. For this project, every query was treated as a single event without context in time. I.e., previous queries by the same user (determined through client address for example), were not considered relevant. In real life however, a *profile* of a user's previous queries may help determine the intention for a query more accurately.

Further research projects should be undertaken to find out to what extent such features, as well as others that remained unmentioned here, can contribute to the categorization process.

6.4 Other & Tweaked Algorithms

The algorithms used for this study are a couple of quite popular machine learning techniques using parameters found to perform well overall. The reason for this is, that this project is merely an explorational study, rather than an exhaustive quest for the best tweaked algorithm for this problem. For practical applications however, it may be worth the extra effort to optimize the choice and parameters of the algorithm to use.

6.5 Cross Domain Classification

For this study, the semantics of query terms are not taken into account. A direct result of this is that no separate classification per semantic domain is possible. All queries undergo the same categorization process, after which their semantics can be used to find proper results through, for example, a specific retrieval strategy. Another possibility would be to split the document set into several domains and first assign a query to one (or more) of these domains using terms from the query. Subsequently, a different intention categorization system can be used per domain, possibly improving the categorization performance. For example: a company offers both physical products and services. If the terms in the query can be used to distinguish what part of the company it is about, it might be better possible to determine an intention for that query as it might be easier to distinguish informational queries about the services and transactional queries about the physical products that way.

7. CONCLUSION

Based on the experiments, the conclusion that automatic categorization into different user intentions is possible, is not justified, based on this setting, data set and choice of possible user intentions alone. Based on these results alone, however, it is possible to say that this will quite likely be possible under certain circumstances. The catch is, that (almost) all successful classifications have to be attributed to the two features that are related to the context of the system. Classification based on the other features is hardly worth the trouble, because of the bad results that are obtained with them. For correct classification, the context of the system and document set appears to be needed after all, simply using the query's own context or linguistic features seems to be insufficient in this setting.

The question of the possible performance improvement is not as easily answered. Alas, the experiments needed for research on this question require use of a complete search engine over the same document set, as well as some tagged result sets (to calculate precision/recall values) for these queries. Both of these were not available for this project and could not be constructed due to time and scope constraints. This will have to be further investigated in subsequent projects.

8. REFERENCES

[1] Universiteit Twente, de ondernemende research universiteit. Available at *http://www.utwente.nl* ; last verified Aug. 24, 2004.

[2] Weka software. Available at *http://www.cs.waikato.ac.nz/ml/weka/* ; last verified Aug. 24, 2004.

[3] J. Allan. Hard track overview in trec 2003: High accuracy retrieval from documents. In *TREC 2003*, pages 24–37, 2003.

[4] M. J. Bomhoff. *The Best of Intentions*. University of Twente, Enschede, the Netherlands, 2004. Available online at http://www.bomhoff.nl/pub/doc/best_of_intentions.pdf.

[5] Broder. A taxonomy of web search. *IRFORUM: SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, 36, 2002.

[6] I.-H. Kang and G. Kim. Query type classification for web document retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71, 2003.

Hierarchical topic detection in large digital news archives

Exploring a sample based approach

Dolf Trieschnigg
University of Twente
Enschede, The Netherlands
trieschn@cs.utwente.nl

Wessel Kraaij
TNO
P.O. Box 155, 2600 AD Delft, The Netherlands
kraaij@tpd.tno.nl

ABSTRACT
Hierarchical topic detection is a new task in the TDT 2004 evaluation program, which aims to organize a collection of unstructured news data in a directed acyclic graph (DAG) structure, reflecting the topics discussed in the collection, ranging from rather coarse category like nodes to fine singular events. The HTD task poses interesting challenges since its evaluation metric is composed of a travel cost component reflecting the time to find the node of interest starting from the top node and a quality cost component, determined by the quality of the selected node. We present a scalable architecture for HTD and compare several alternative choices for agglomerative clustering and DAG optimization in order to minimize the HTD cost metric. The alternatives are evaluated on the TDT3 and TDT5 test collections.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; I.5.3 [**Pattern recognition**]: Clustering—*Algorithms,Similarity measures*

Keywords
Information Retrieval, Hierarchical Topic Detection, TDT

1. INTRODUCTION
The Topic Detection and Tracking (*TDT*) project is an annually held evaluation study in the field of TDT organized by the National Institute of Standards and Technology(NIST).

TDT has included a Topic Detection task since its inception in 1996. In this task systems are required to organize news stories in clusters, corresponding to the topics discussed. The result can be regarded as a partition of the corpus, in which each news item is assigned to one and only one partition representing a topic.

The systems are scored by comparing the system result to a manually composed *ground truth*. The cost of a (cluster) structure defines the 'distance' to the ground truth; a better structure has a lower cost. The ground truth is composed by annotators of the Linguistic Data Consortium and consists of manually labelled clusters containing news stories discussing a particular topic. A topic is defined as an event or activity, along with all directly related events and activities. The topics are selected from a random sample of documents from the corpus. The annotation is search guided, i.e. the related stories are found using a search engine. Important to mention is that the annotation for the most recently published TDT 5 corpus is incomplete, that is, there will be no guarantee that every story on each topic will have been located [5]: The search for stories related to one particular topic is ceased after 3 hours, in contrast to previous annotations where the annotators decided when all on-topic stories were found.

The Task Definition and Evaluation Plan of TDT 2004 [6] describes two reasons for introducing a new *Hierarchical Topic Detection* task. The first shortcoming is that a flat partitioned structure does not allow a single news item to belong to multiple topics. Furthermore a flat structure does not allow multiple levels of granularity, i.e. topics cannot be introduced at various levels of detail.

The new HTD task enables stories to be assigned to multiple clusters. Furthermore clusters may be a subset of, or overlap with other clusters. The resulting structure must be characterizable as a DAG with a single root node. The root node represents the complete document collection whereas child clusters further down the DAG represent more specific subsets comprising finer detailed topics. For this initial trial evaluation, the task simplifies treatment of time: the task is treated as retrospective search, i.e. the documents may be processed in any order, in contrast to the old task in which the items should be processed in the order they were published [6].

The metric used for the old Topic Detection task is not suitable for this new task. Allan et al [1] discuss various methods for evaluating hierarchical cluster structures. The TDT 2004 HTD task is evaluated by using the minimal cost method described in Allan et al's paper.

The minimal cost metric finds for each annotated topic the system's optimal cluster, having the lowest cost. This cost consists of a *detection cost* representing the 'goodness' of the cluster and a *travel cost* representing the complicatedness to find the cluster. The detection cost is the same as for previous topic detection tasks and consists of a penalty for false alarms and misses, misses have more impact than false alarms however. The travel cost has been introduced to penalize 'powerset' cluster structures, i.e. structures having clusters containing all possible combinations of document sets. The travel cost of a cluster is, independent of its content, related to the shortest path to this cluster from the structure's root cluster. The number of encountered branches and the length of the path are the major components in the travel cost calculation, representing the number of choices a user has to make and the number of cluster titles a user has to read to find the best matching cluster. The score function is parametrized, i.e. the impact of the various cost components is set by using parameters. A more detailed explanation of the metric can be found in Allan et al's paper [1] and the TDT evaluation plan [6].

1.1 Overview of this paper

TNO has participated in the HTD task of TDT 2004. This paper discusses TNO's approach, the experiments on the TDT 3 corpus with this system and the final TDT 2004 results.

We would like to answer the following questions:

- Are conventional agglomerative clustering techniques appropriate for the new HTD task? If not, what additional actions do make these techniques suitable?
- Is the resulting structure intuitive and how does this relate to the minimal cost metric?

Paragraph 2 introduces our approach followed by paragraph 3 outlining related work. Paragraph 4 describes the experiments carried out using this approach. Paragraph 5 discusses the most important results from the experiments and participation in TDT 2004. Conclusions and future work will be outlined in paragraph 6.

2. OUR APPROACH

The corpus for TDT 2004, the TDT 5 test collection, contains news corpora from a number of sources and languages. The total corpus consists of around 400,000 stories (see table 1). The stories are multilingual but all all non-English stories are also available in machine translated English. The system only works with the (translated) English stories. The size of the corpus makes it difficult to use a conventional agglomerative hierarchical clustering approach, like single, complete or average clustering, which uses a distance matrix for building a binary cluster tree. The complexity of such methods usually is $O(n^2 \log(n))$ in time and $O(n^2)$ in space [3]. To illustrate this, a set of 400,000 documents would typically[1] require 80 gigabytes of memory (preferably working memory). After that 80 billion document pair comparisons should be made just to fill the matrix.

The goal of this research is to explore possibilities to make agglomerative clustering scalable for large document datasets.

[1]Using a symmetric distance matrix, $O(\frac{1}{2}n^2)$, optimistically using only 1 byte per comparison

Table 1: TDT 5 corpus statistics

	TDT3	*TDT5*
Arabic stories	0	72,910
English stories	34,600	278,109
Mandarin stories	n.a.	56,486
Total stories	n.a.	407,505
Annotated topics	160	250

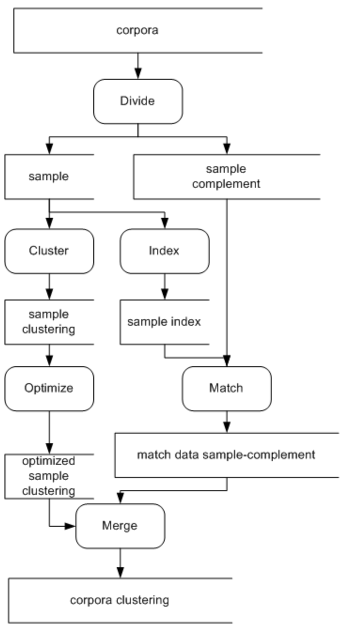


Figure 1: Data Flow Diagram

The following approach has been used:

1. Take a sample from the corpus;
2. Build a hierarchical cluster structure of this sample;
3. Optimize the resulting binary tree for the minimal cost metric;
4. Assign the remaining documents from the corpus to clusters in the structure obtained from the sample.

Figure 1 shows this approach graphically. The steps are explained in the following paragraphs.

2.1 Sampling

The first step is to take a random sample from the corpus. The size of this sample is 20,000 documents, its corresponding distance matrix requires an acceptable 800 megabytes of working memory[2].

2.2 Clustering

The second step is to build a hierarchical cluster structure. Starting point for the clustering method is the cross-entropy reduction scoring function [4]. Suppose we have two documents D_1 and D_2. Both documents are represented by simple unigram language models M_{D_1} and M_{D_2}, a reference

[2]4 bytes per comparison in a symmetric matrix

unigram model for general English M_C is estimated on the complete document collection. Now the cross-entropy reduction (CER) of M_{D_1} and M_{D_2} compared to M_C is defined as:

$$\begin{aligned} CER(D_1; C, D_2) &= H(D_1, C) - H(D_1, D_2) \qquad (1) \\ &= \sum_{i=1}^{n} P(\tau_i|M_{D_1}) \log \frac{P(\tau_i|M_{D_2})}{P(\tau_i|M_C)} \end{aligned}$$

where τ_i is an index term and n is the number of unique index terms in C

The generative document model M_{D_2} is smoothed by linear interpolation with the background model M_C [9]. Normalization of scores (by subtracting $H(D_1, C)$) is essential for adequate performance.

The symmetrical version of this scoring function is defined as

$$sim(D_1, D_2) = \frac{CER(D_1; C, D_2) + CER(D_2; C, D_1)}{2} \qquad (2)$$

A distance matrix is filled using this scoring function. For the actual clustering 3 basic hierarchic agglomerative clustering methods are used: single, complete and average pairwise linkage.

2.3 Optimizing

The result of this clustering process is a, usually unbalanced, binary tree. An uneven cluster, i.e. a cluster which has childclusters containing an uneven number of documents, adds extra travel cost to all of the clusters below this cluster, especially if this cluster is near the root of the tree. Relating to the real world, the 'user' should consider more branches and titles to find the desired cluster. A more balanced tree will reduce the expected travel cost, but how can the structure be rebalanced without losing clustering information? The metric shows whether the changes to the tree have thrown away clustering information: if after rebalancing the detection cost for any 'optimal' cluster grows, the rebalancing has thrown away valuable information from the original structure. The detection cost should remain the same (or decrease) and the travel cost is decreased.

The method used for rebalancing the tree, without large changes to the optimal clusters, is quite simple. First the clusters are removed which have no documents directly[3] attached and have a dissimilarity higher or equal to a certain threshold. A group of unconnected clusters now remains. These clusters are used to form a better balanced tree with a branching factor of three, suiting the HTD evaluation metric preferring tertiary or quadruple trees[6]. This is done by recursively taking the smallest three (or a different number of) clusters to form a new cluster, until only one root cluster remains.

Figure 2 and 3 show the impact of this rebranching on an average pairwise clustering of 100 documents. The black squares in the bottom of the visualization represent documents, the rectangles represent clusters grouping documents and clusters in new clusters at a higher level. The marked clusters in the first figure will be removed: their dissimilarity is higher than or equal to the chosen threshold and they don't contain documents directly below them. After removing these clusters and corresponding edges, a group of small cluster branches remains. The second figure shows the result of building a more balanced tree with these small branches. The marked clusters now represent newly added clusters.

[3]a directly attached document only appears in this cluster and not in child clusters

2.4 Merging

An index is built from the sample document set. The documents from the corpus which are not in the sample are used as queries on this index returning the best document-likelihood matches. For each document in this complement dataset the best 10 matches are used for merging. The document from the complement dataset is added to all of the matching documents' clusters.

The new documents which don't have any matching documents are collected in one cluster. This ensures new documents at least are assigned to one cluster.

The result of the merging process is a so called fuzzy cluster structure: News items can belong to multiple topics.

3. RELATED WORK

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). It's applied for pattern recognition, image processing, information retrieval and others. The major steps in clustering patterns are: (1) choosing a pattern representation, (2) defining a proximity measure appropriate to the dataset domain, (3) the actual clustering process, (4) data abstraction, e.g. labelling the clusters and optionally (5) assessment of the output. Jain et al give a very good introduction to the concepts of data clustering [3].

In general there are two types of clustering techniques, hierarchical and partitional, which determine the resulting structure. The hierarchical approach produces a nested series of partitions whereas the partitional approach yields a flat structure, in which the relationship between clusters is not as clear as in the first approach.

Clustering techniques can either be agglomerative or divisive. The first is a bottom-up approach which starts with the patterns treated as distinct (singleton) clusters and successively merges clusters together until a stopping criterion is satisfied. Divisive clustering works top-down: the complete dataset is treated as one cluster and splits the clusters until a stopping criterion is met.

A clustering technique can either be hard or fuzzy. Hard clusterings assign each pattern to only one cluster, whereas fuzzy clusterings may assign patterns to multiple clusters based on the degree of membership.

Most hierarchical agglomerative approaches are variants of single link, complete link and minimum-variance (e.g. Ward's method) algorithms. The main difference is the way distance between existing and new clusters is calculated. A very popular partitional method is k-means. By choosing k patterns as initial centroids and assigning the remaining patterns to

Figure 2: Before rebranching, marked clusters will be removed

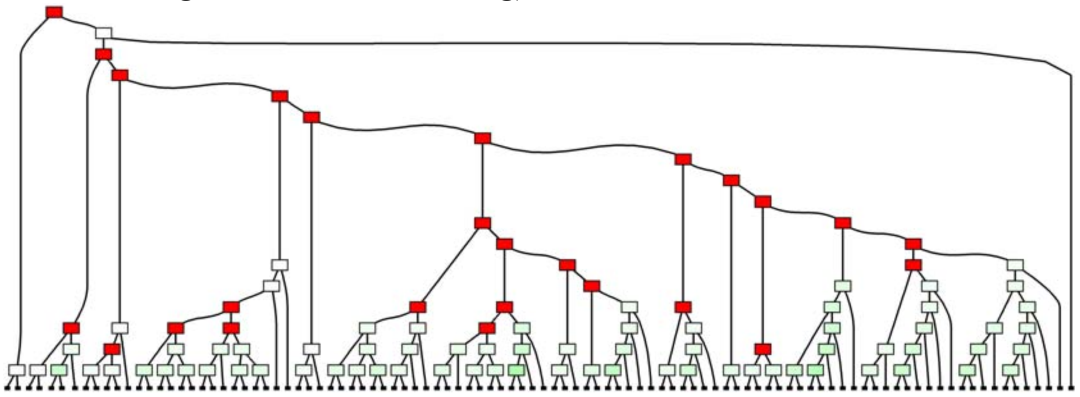
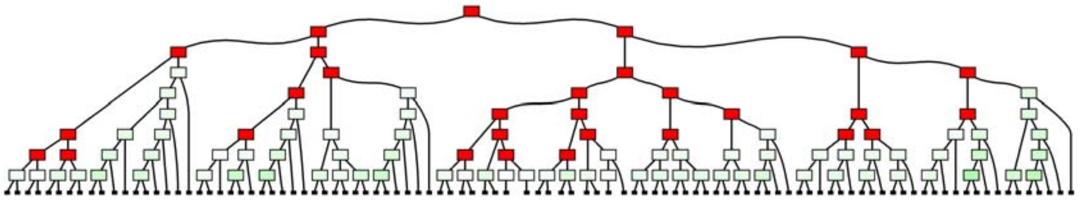


Figure 3: After rebranching, marked clusters are new



one one of these centroids a clustering is obtained. The clusters

Topic detection is a specialization of cluster analysis to facilitate the task of information analysis. Van Rijsbergen [10] formulated the cluster hypothesis for document clustering: "Closely associated documents tend to be relevant to the same requests". Document clustering has been extensively investigated as a methodology for improving document search and retrieval [2].

The high dimensionality of text data and usually large size of datasets do not allow simple application of hierarchical clustering methods because of its high time and space complexity. Much research has been done on finding scalable methods for clustering. This has resulted in different hybrid clustering systems, combining hierarchical and partitional clustering techniques [11, 12].

Cutting et al [2] introduced the Buckshot algorithm, which combines average link clustering with k-means clustering. The average linking is used to find relatively good initial centroids used for further k-means clustering.

Smeaton et al [8] developed a method using a much smaller distance matrix for hierarchical clustering. New documents were added to the clustering by using document-likelihood.

Pantel et al [7] introduced document clustering with committees, which also is a variation on k-means clustering. The centroids are the average feature vectors of carefully chosen committees of patterns representing a cluster.

4. EXPERIMENTS

Experiments were carried out using the English sources from the TDT 3 dataset as a preparation for participation in the trial HTD task of TDT 2004. The size of this dataset is around 35,000 documents, roughly one tenth of the TDT 5 dataset. As a sample we took 10,000 documents from the TDT 3 dataset.

For this sample a symmetric distance matrix was created, filled with the dissimilarity between each document pair. Using this matrix a cluster structure was built using single, complete and average link methods. The sample structure was scored using the minimal cost metric and TDT 3 ground truth containing 160 topics. Based on the bad results for single linkage was decided to exclude this method from further experiments.

Experiments were carried out with rebranching, varying the cut threshold (0, 0.90, 0.95, 0.96, 0.97 and 0.98) and varying the number of branches (3 and 4) to use when glueing the pieces together. Furthermore experiments were carried out applying rebranching before and after the merging process. Other tree simplifying operations were studied, also changing the structure in the lower parts of tree, but these resulted in similar or worse results and are not further discussed in this paper.

The documents in the complement dataset were used as queries for the built sample index. The 20 best matching documents from the sample were searched, using document likelihood. The new documents were assigned to the clusters to which the best matching documents from the sample belong. Two methods were used:

- Adding the new document to the first n (1, 10, 20) matching clusters.

Table 2: Comparison of clustering methods

Method	*Minimum cost*	*Norm. detection cost*	*Norm. travel cost*	*Depth*
Average link	0.2747	0.3722	0.0855	11.68
Complete link	0.6120	0.8778	0.0962	13.14
Single link	0.6970	1.0003	0.1084	24

- Adding the new document to the first matching cluster and to to all matching clusters having a document likelihood higher than a certain threshold (0.5, 1, 2)

The minimal cost was calculated over all of the generated cluster structures, including structures only containing sample documents.

The configuration with optimal result for the TDT 3 test collection was used for the TDT 2004 participation. This was constructing a sample cluster structure using average pairwise link for 20,000 documents, applying a rebranch with branching factor 3 and cut threshold 0.96 and finally merging the best 10 matching documents. Creating the cluster structure of the TDT 5 corpus took around one complete day of processing time on a 900 Mhz machine having 2 Gb of working memory. The sample based clustering system from TNO scored best in the HTD evaluation task of TDT 2004!

5. DISCUSSION

In this paragraph the most interesting results from the experiments and participation in TDT 2004 are discussed.

5.1 Linkage method

First of all the choice of linkage method. The sample TDT 3 dataset was clustered using complete, average and single linkage methods. Average pairwise linking gave the best results by far. For each of the topics in the ground truth, the best cluster, i.e. the cluster having the minimum cost, was calculated. The rows in table 2 show the average characteristics of these best clusters. Without rebranching average pairwise linking gave the best results by far. The metric indicated the cluster structures obtained by using single linkage and complete linkage were much worse.

Further investigation showed that single linkage, as expected[3, 8], performed bad because of its chaining behaviour. A smaller sample of 100 documents was taken, clustered using single linkage and visualized in a tree (figure 4). The figure shows how, especially in the upper part of the tree, new clusters are created by merging an existing cluster and a single document. As a result, the travel cost to reach a more meaningful cluster, i.e. a cluster more closely resembling topics from the ground truth residing at the bottom of the structure, is very high. The travel cost overshadows the detection cost in such a way that the cluster having the lowest overall cost (consisting of travel cost and detection cost) is in the upper part of the structure, although the recall is very poor.

The visualization of a structure with 100 documents obtained by using complete linkage (figure 5) does not clearly show any chaining behaviour. However, details of the experiment outcome showed that a few clusters were chosen

Figure 4: Single link clustering suffers from chaining

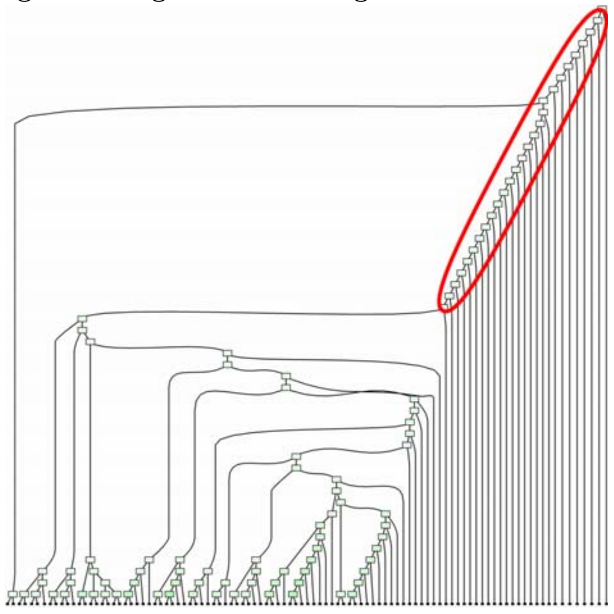
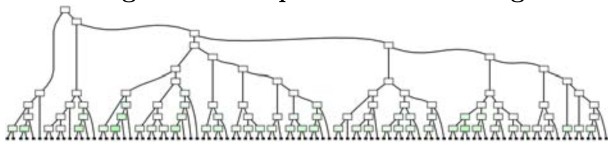


Figure 5: Complete link clustering

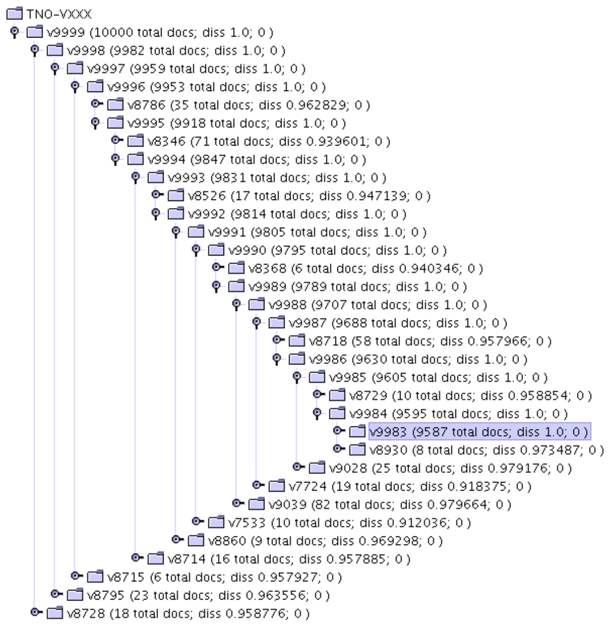


frequently as best matching cluster for a topic, just like the single link cluster structure. A screen shot of a cluster structure browser (Figure 6) shows the complete linkage structure also suffers from some kind of 'chaining' behaviour. At the root the document set is divided in two clusters: one tight, relatively small cluster with a dissimilarity little less than 1, and one heavy cluster with a dissimilarity equal to 1. The heavy cluster subsequently is divided again in one small tight cluster and one very heavy cluster. This continues downwards the tree. The visualization of the structure of 100 documents did not show this behaviour, simply because the dataset is too small. The result, just like the single linkage structure, does not allow the best clusters to be found deep down the clustering tree because of the high travel cost to get there. Some of the best matching clusters found (with a smaller travel cost less influencing the complete cost) were promising however. Table 3 gives a sample of the best clusters found for particular topics and its score. The clusters found at depth 2 can be considered as chosen under influence of travel cost - most probably a cluster with a lower detection cost can be found further down the tree. The other clusters however do seem to cover the topics quite well; the recall is quite high, but the precision can be further improved.

The structure obtained by using average linkage seems to be more balanced, naturally enabling more clusters to be considered, not being limited by travel cost. This is one of the major reasons average linkage performs much better when evaluating with the minimal cost metric.

Table 3: Sample of best matching clusters using complete linkage

System cluster	*Minimum cost*	*Norm detect. cost*	*Norm travel cost*	*#Ref*	*#Sys*	*#Union*	*Depth*
v7102	0.6656	1.001	0.0146	5	2	0	2
v7102	0.6656	1.001	0.0146	33	2	0	2
v7102	0.6656	1.001	0.0146	60	2	0	2
v8514	0.2333	0.1686	0.3588	30	29	25	49
v7102	0.6656	1.001	0.0146	1	2	0	2
v7102	0.6656	1.001	0.0146	4	2	0	2
v7102	0.6656	1.001	0.0146	9	2	0	2
v7102	0.6656	1.001	0.0146	1	2	0	2
v8933	0.5553	0.0152	1.6036	10	41	10	219
v7102	0.6656	1.001	0.0146	3	2	0	2
v8500	0.1387	0.0064	0.3954	8	21	8	54
v5701	0.1454	0.0015	0.4247	1	4	1	58
v7102	0.6656	1.001	0.0146	41	2	0	2
v7102	0.6656	1.001	0.0146	5	2	0	2
v7102	0.6656	1.001	0.0146	9	2	0	2
v7102	0.6656	1.001	0.0146	17	2	0	2
v8013	0.2758	0.1765	0.4686	12	30	10	64



```
TNO-VXXX
  v9999 (10000 total docs; diss 1.0; 0 )
    v9998 (9982 total docs; diss 1.0; 0 )
      v9997 (9959 total docs; diss 1.0; 0 )
        v9996 (9953 total docs; diss 1.0; 0 )
          v8786 (35 total docs; diss 0.962829; 0 )
          v9995 (9918 total docs; diss 1.0; 0 )
            v8346 (71 total docs; diss 0.939601; 0 )
            v9994 (9847 total docs; diss 1.0; 0 )
              v9993 (9831 total docs; diss 1.0; 0 )
                v8526 (17 total docs; diss 0.947139; 0 )
                v9992 (9814 total docs; diss 1.0; 0 )
                  v9991 (9805 total docs; diss 1.0; 0 )
                    v9990 (9795 total docs; diss 1.0; 0 )
                      v8368 (6 total docs; diss 0.940346; 0 )
                      v9989 (9789 total docs; diss 1.0; 0 )
                        v9988 (9707 total docs; diss 1.0; 0 )
                          v9987 (9688 total docs; diss 1.0; 0 )
                            v8718 (58 total docs; diss 0.957966; 0 )
                            v9986 (9630 total docs; diss 1.0; 0 )
                              v9985 (9605 total docs; diss 1.0; 0 )
                                v8729 (10 total docs; diss 0.958854; 0 )
                                v9984 (9595 total docs; diss 1.0; 0 )
                                  v9983 (9587 total docs; diss 1.0; 0 )
                                  v8930 (8 total docs; diss 0.973487; 0 )
                              v9028 (25 total docs; diss 0.979176; 0 )
                          v7724 (19 total docs; diss 0.918375; 0 )
                        v9039 (82 total docs; diss 0.979664; 0 )
                    v7533 (10 total docs; diss 0.912036; 0 )
                  v8860 (9 total docs; diss 0.969298; 0 )
              v8714 (16 total docs; diss 0.957885; 0 )
        v8715 (6 total docs; diss 0.957927; 0 )
      v8795 (23 total docs; diss 0.963556; 0 )
    v8728 (18 total docs; diss 0.958776; 0 )
```

Figure 6: 'Chaining' behaviour of complete linkage clustering

5.2 Influence of rebranching

The preference of the minimal cost metric for clusters closer to the root of the tree in combination with an unbalanced tree resulted in bad results for complete and single linking. The tree should be balanced without modifying important cluster information. This is done using the rebranching method described before. Table 4 shows the minimum cost of the rebranched structures. The dissimilarity thresholds used are adapted to the various methods. The complete linkage causes the dissimilarity to reach 1 quickly for clusters higher in the tree. The threshold is set to 1 correspondingly. Average linkage will not quickly reach a dissimilarity close to 1, so a threshold value smaller than 1 is chosen. Single linkage suffered so badly under the chaining effect, no threshold value could be chosen to build a better tree. Therefore it was decided to not continue further experiments using the single link clustering technique. The rebranching action did not have a big (-6%) impact on the minimum cost for the structure built with average linking. The normalized travel cost however decreased significantly (-65%). As expected the minimum cost for the cluster structure built using complete linking decreased (-50%)as a result of rebranching. A more balanced tree will be searched more thoroughly, i.e. more clusters down the tree are considered, enabling all of the compact clusters to be picked as optimal clusters. As a result the travel cost *and* the detection cost decreased after rebranching the structures built using complete linkage.

5.3 Influence of matching

It was expected that the complete cluster structures, i.e. the structures obtained by adding the rest of the document dataset to the sample structure, would increase the average minimum cost of the optimal clusters. The contrary was true: adding the new documents to multiple clusters, resulting in a fuzzy cluster structure, improved the results! Table 5 shows the cost before and after the matching process. It's particularly interesting that the average detection costs for the TDT 3 and TDT 5 dataset are so different. For the TDT 5 dataset the normalized detection cost and normalized travel cost are in the same order, whereas for the TDT 3 the detection cost is much higher. This might be caused by differences in the dataset, or the way the ground truth was composed.

Furthermore it is noteworthy that the detection cost for the TDT 5 after matching has decreased. The recall has improved (lower P_{miss} rate) but the precision has gone down (higher P_{fa}). The fuzzy matching is causing this. By simply adding new documents to multiple clusters, recall is bound to go up. The cost of a false alarm is very low because the dataset is quite large and topics are quite small[4]. Simply guessing related documents for a particular topic using this

[4]the cost of a false alarm is normalized by the chance a random document does not belong to a topic, which is quite low if the dataset is large and the topics small

matching method pays off: the chance an on-target document is guessed is quite large, resulting in a high chance to increase recall, while the cost of a false alarm is low.

The results give the impression the approach is quite scalable, further investigation has to show this indeed is true and how to explain this performance.

5.4 Intuitiveness

The results do raise questions about the intuitiveness of the metric. For example consider the cluster named 'v18100' in table 6. It represents a topic supposedly to have 81 new items, but itself contains 2826 items, of which 80 actually overlap with the truth cluster. The penalty for missing 1 of the 81 documents is calculated as 0.0123, whereas the false alarm of 2746 items only adds 0.0099 of cost. Just imagine a user trying to find its way through a 'topic' polluted with so many unrelated items. The averages in table 6 show the clusters do have a good recall, but it's precision is terrible. This phenomena also is apparent in the results of table 5: after the fuzzy matching of new documents the the misses decrease but the false alarm rate goes up. The metric allows a large increase of the recall by adding documents to multiple clusters - the loss in precision is not penalized.

Although the idea behind the introduction of the travel cost is intuitive, it does not really penalize 'powerset' structures as it was intended. The travel cost penalizes structures not having the desired branching factor or which are not balanced very well, although the ground truth does not provide any information about this. Another cost component should be introduced to penalize scattering documents over relatively unrelated clusters as is the case when constructing powerset cluster structures.

6. CONCLUSION & FUTURE WORK

In this paper the results of a prototype HTD system were presented. The usage of conventional agglomerative clustering techniques combined with dissimilarity measurement using language modelling looks promising. Cluster structures built with complete linkage using this distance measurement do need restructuring to be effective however. The system has been used for participation in the newly introduced HTD evaluation task of TDT 2004 and achieved best results. The intuitive quality of the clusters is questionable however. At this time the results have too little precision to be really useful. The results give thought about the metric used for HTD evaluation.

Future work should point out what steps in this clustering approach are of most importance and how the precision of the structures can be increased. Furthermore it seems interesting to study the scalability of the system in more depth.

7. REFERENCES

[1] J. Allan, A. Feng, and A. Bolivar. Flexible intrinsic evaluation of hierarchical clustering for TDT. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 263–270. ACM Press, 2003.

[2] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM Press, 1992.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[4] W. Kraaij. *Variations on language modeling for information retrieval.* PhD thesis, University of Twente, May 2004.

[5] Linguistic Data Consortium. TDT 5: Project resources for 2004 evaluation. `http://www.ldc.upenn.edu/Projects/TDT2004`.

[6] NIST. The 2004 Topic Detection and Tracking (TDT2004) task definition and evaluation plan. `http://www.nist.gov/speech/tests/tdt/index.htm`.

[7] P. Pantel and D. Lin. Document clustering with committees. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 199–206. ACM Press, 2002.

[8] A. F. Smeaton, M. Burnett, F. Crimmins, and G. Quinn. An architecture for efficient document clustering and retrieval on a dynamic collection of newspaper texts. In *Proceedings of the 20th BCS-IRSG Annual Colloquium*, 1998.

[9] M. Spitters and W. Kraaij. Unsupervised event clustering in multilingual news streams. *Proceedings of the LREC2002 Workshop on Event Modeling for Multilingual Document Linking*, pages 42–46, 2002.

[10] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

[11] P. Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing Management*, 24(5):577–597, 1988.

[12] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM Press, 2002.

Table 4: Influence of rebranching

Cluster method (size)	*Minimum cost*	*Norm. detection cost*	*Norm. travel cost*	*Depth*
Average linkage	0.2747	0.3722	0.0855	11.68
... after rebranching (threshold 0.97)	0.2579	0.3620	0.0559	6.11
Complete linkage	0.612	0.8778	0.0962	13.14
... after rebranching threshold 1.0)	0.3497	0.5006	0.0567	5.89

Table 5: Influence of matching on average costs

Cluster method (size)	*Minimum cost*	*Norm. detection cost*	*Norm. travel cost*	*P(miss)*	*P(fa)*
TDT3 sample (10,000)	0.2579	0.3620	0.0559	0.3069	0.0112
... after matching (35,000)	0.2430	0.3581	0.0195	0.2681	0.0184
TDT5 sample (20,000)	0.0565	0.0629	0.0441	0.0493	0.0028
... after matching (278,000)	0.0282	0.0406	0.0041	0.0224	0.0037

Table 6: Sample results from one complete TDT 5 cluster structure

System cluster	*Minimum cost*	*Norm detect. cost*	*Norm travel cost*	*#Ref*	*#Sys*	*#Union*	*P(miss)*	*P(fa)*
v13965	0.0039	0.0045	0.0028	5	261	5	0	0.0009
v15445	0.0023	0.0023	0.0022	1	133	1	0	0.0005
v14140	0.0024	0.0019	0.0035	27	133	27	0	0.0004
v16401	0.0095	0.0131	0.0025	13	759	13	0	0.0027
v18100	0.0411	0.0607	0.0031	81	2826	80	0.0123	0.0099
v3969	0.0013	0.0004	0.0031	1	24	1	0	0.0001
v5859	0.0019	0.0012	0.0032	2	71	2	0	0.0002
v1076	0.0029	0.0018	0.0051	1	104	1	0	0.0004
v3440	0.0019	0.0013	0.0031	2	76	2	0	0.0003
v9072	0.0094	0.0117	0.0050	21	683	21	0	0.0024
v2590	0.0017	0.0005	0.0042	1	28	1	0	0.0001
v8772	0.0448	0.0664	0.0030	63	223	59	0.0635	0.0006
v17828	0.0016	0.0009	0.0030	1	50	1	0	0.0002
v15435	0.0065	0.0073	0.0051	2	417	2	0	0.0015
v17092	0.0037	0.0042	0.0028	5	241	5	0	0.0008
...	...	...	...	...	...	...	...	...
average	0.0282	0.0406	0.0041	43.15	1073.1	42.05	0.0224	0.0037

A Ground Truth For Half A Million Musical Incipits

Rainer Typke, Marc den Hoed, Justin de Nooijer, Frans Wiering, Remco C. Veltkamp
Utrecht University, ICS
Padualaan 14
3584 CH Utrecht,
The Netherlands
{rainer.typke|mhoed|jnooijer|frans.wiering|remco.veltkamp}@cs.uu.nl

ABSTRACT

Musical incipits are short extracts of scores, taken from the beginning. The RISM A/II collection contains about half a million of them. This large collection size makes a ground truth very interesting for the development of music retrieval methods, but at the same time makes it very difficult to establish one. Human experts cannot be expected to sift through half a million melodies to find the best matches for a given query. For 11 queries, we filtered the collection so that about 50 candidates per query were left, which we then presented to 35 human experts for a final ranking. We present our filtering methods, the experiment design, and the resulting ground truth.

To obtain ground truths, we ordered the incipits by the median ranks assigned to them by the human experts. For every incipit, we used the Wilcoxon rank sum test to compare the list of ranks assigned to it with the lists of ranks assigned to its predecessors. As a result, we know which rank differences are statistically significant, which gives us groups of incipits whose correct ranking we know. This ground truth can be used for evaluating music information retrieval systems. A good retrieval system should order the incipits in a way that the order of the groups we identified is not violated, and it should include all high-ranking melodies that we found. It might, however, find additional good matches since our filtering process is not guaranteed to be perfect.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Symbolic music representation*; J.5 [**Arts and Humanities**]: Performing arts (music)

General Terms

Algorithms, Measurement, Performance

Keywords

music information retrieval, RISM, ground truth, filtering

1. INTRODUCTION

For evaluating the performance of a music retrieval system, one needs a ground truth for its data collection and some given queries. The music retrieval systems we have in mind serve the information need for music that is melodically similar to a given query. Such search engines can be useful not only for retrieving sheet music or recordings in libraries or stores, but also for investigating how composers have influenced one another, or for raising or resolving copyright disputes.

The RISM A/II collection [8] contains 476,600 incipits, short excerpts of notated music from the beginnings of manuscripts in libraries, archives, cloisters, schools, and private collections worldwide. This collection is useful for content-based music retrieval because of its size and the fact that it contains real music written by human composers. A music retrieval system that works well with this collection should also perform well for real-world applications in general. Our ground truth can serve as a benchmark for deciding how well a music retrieval system works with the RISM A/II collection.

In TREC [12], relevance assessments are mostly binary ("relevant" or "not relevant"). Only in more recent TREC web tracks such as at TREC-9 [3], this was extended to ternary ("irrelevant"/"relevant"/"highly relevant"). Studies such as Selfridge-Field [9] show that melodic similarity is continuous. Local melodic changes such as lengthening a note or moving it up or down a step are usually not perceived as changing the identity of a melody, and by applying more and more changes, the perceived relationship to the original becomes only gradually weaker. Also, melodies are generally quite resistant to the insertion of all sorts of ornamentation. Because of the continuity of melodic similarity, there are no sensible criteria for assigning one out of a few distinct degrees of relevance to a melody, so any relevance assessment with a given scale length seems inappropriate. Instead, we asked human experts to rank all incipits where they saw any similarity to the query. Our ground truth therefore does not consist of sets of highly relevant, relevant and irrelevant documents, but of ranking lists of documents.

A valid way of establishing such a ground truth would be to ask a number of human experts to look at all possible matches for a given query (carefully making sure that they stay concentrated long enough) and order them by similarity. Since we cannot expect our human experts to sift through half a million melodies, we needed to filter out incipits of which we can be reasonably sure that they do not resemble the query.

This paper describes how we filtered the collection for a list of 11 queries, how we employed 35 human experts for ranking the remaining incipits by their similarity to the queries, and how we established a ground truth as a result. For each of the 11 queries, we built a list of groups of incipits that is ordered by similarity to the query. The Wilcoxon rank sum test [13] gives us a measure for the statistical significance of the differences between the groups of incipits.

Related Work. We are not aware of any previous efforts to establish a ground truth for the RISM A/II collection or a similarly large collection of musical incipits, themes, or scores. However, there is high interest in establishing a systematic, TREC-like paradigm for the music information retrieval research community [2], so that having a ground truth could be very helpful.

2. FILTERING MELODIES

To be able to exclude incipits that are very different from our selected queries, we calculated some features for every incipit in the database. Filtering could then easily be done by issuing SQL statements with selections based on those features.

- **Pitch range:** the interval between the highest and lowest note in the incipit.
- **Duration ratio:** the duration of the shortest note (not rest), divided by the duration of the longest note (not rest). The result is a number in the interval (0,1], where 1 means that all notes have the same duration, while a very small number means a very high contrast in durations.
- **Maximum interval:** the largest interval between subsequent notes. Rests are ignored.
- **Editing distance between gross contours:** the editing distance between two character strings is the sum of the costs of the cheapest possible combination of character insertion, deletion, and replacement operations that transform one string into the other. We determined the gross contour as a string of characters from the alphabet U ("up"), D ("down"), and R("repeat") and calculated the distance to every query for each incipit in the database, using the editing distance described by Prechelt and Typke in [7]. They had optimized the costs for the insertion, deletion, and replacement operations for gross contour strings such that the resulting similarity measure corresponds well with human perception.
- **Editing distance between rhythm strings:** we also represented the incipits as rhythm strings with one character from a three-character alphabet for each pair of subsequent notes: longer, shorter, and same duration.
- **Interval histogram:** the number of occurrences for each interval between subsequent notes, normalized with the total number of intervals. With this feature, we can base selections on things like "incipits with many thirds".
- **Interval strings:** one string of diatonic intervals and one string of chromatic intervals for every incipit. This makes it possible to select incipits that contain a certain sequence of intervals.
- **Motive repetitions:** in order to be able to select things like "all incipits with at least three repeated notes in two different places", we collected sequences of intervals that were repeated at least once, along with their number of occurrences, for every incipit.

We used different filtering steps and features for every query since every query has its own characteristic features. Every filtering step had the aim of reducing the number of candidates for matches for a given query by excluding incipits with features that make them very different from the query. As long as this holds for every filtering step, different people should arrive at similar candidate lists even if they apply different filtering steps. However, they need to have similar notions of melodic dissimilarity.

For example, we used the following filtering steps for the "White Cockade" incipit whose ground truth is shown in Table 3:

- Exclude incipits whose pitch range is less than an octave or greater than a minor tenth. This excluded 78 % of the incipits in the database.
- Exclude incipits whose maximum interval between subsequent notes is less than a minor sixth or greater than a diminished seventh. This excluded 79 % of the remaining incipits.
- Exclude incipits with a duration ratio greater than 0.51, i. e. incipits where all notes have quite similar durations. This excluded a further 4 % of incipits.
- Exclude incipits that do not contain at least one of the two interval sequences "fifth up, third down, unison, sixth up" or "third up, unison, unison, sixth up". This left us with 88 incipits.

Because of the dangers of filtering too strictly and thereby accidentally excluding incipits that are similar to the query, we stopped the filtering process once the number of remaining incipits had fallen below 300. To arrive at the desired number of about 50 candidates, we manually excluded remaining incipits that were very different from the query.

As an additional measure to limit the error introduced by accidentally filtering out similar incipits, we used our prototype of a search engine based on transportation distances (described in [10] and [11]) as well as two algorithms from [5] for finding incipits that are similar to the query. The latter two algorithms, called P2 and P3 by their authors, find incipits containing transpositions of the query where many onset time/pitch combinations match, and incipits containing transpositions of the query with maximum common duration with matching pitch. From these search results, we included candidates that we considered similar although they had been filtered out. Also, we used the metadata in the RISM A/II collection. For example, for "Roslin Castle" (see Table 1), we made sure that every incipit whose title contains the word "Roslin" was included. With these methods, we found between 0 and about 8 additional candidates for each query, with an average of about 4.

Once we had filtered out all incipits that are not similar to the query, we also removed incipits that were either identical to other incipits or parts of other incipits. Including identical incipits multiple times in the candidate list would have amounted to asking our experts the same question multiple times, and we wanted to put their time to a more productive use. As a result, only 6 versions of "Roslin Castle" occur in our ground truth in Table 1 although we list 16 known occurrences of this melody in our paper about using trans-

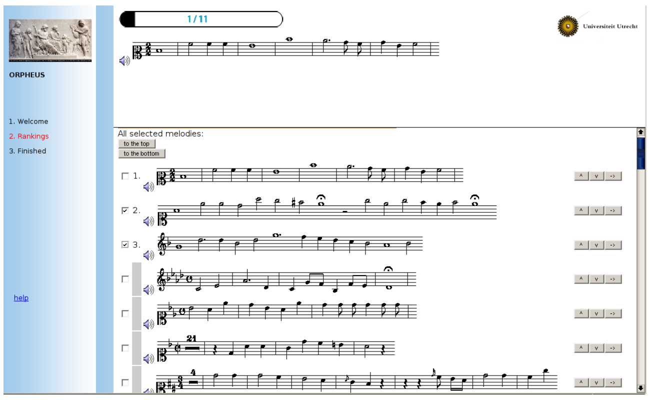


Figure 1: The user interface for the experiment. MIDI files are provided for listening to incipits. In the bottom half of the screen, the subjects can change the order of the candidate incipits, while the query always remains visible at the top.

portation distances for measuring melodic similarity [10], for which we used the same 2002 version of the RISM database.

3. EXPERIMENT DESIGN

3.1 Notated music, MIDI files

Our goal was to establish a ground truth for the incipits that are contained in the RISM A/II collection. These incipits can be exported from the database in the "Plaine & Easie" format [4] and then rendered in common music notation. In order to prevent differences in the rendition of the notated music from having an impact on the ground truth, we used the software that is included with the RISM A/II database [8] for rendering the music notation bitmaps and took screen shots of the results. Only in cases where the RISM software fails to show the whole incipit because it is too long for fitting on the screen, we rendered the notated music ourselves by converting the Plaine & Easie data into the Lilypond[1] format. In addition to the notated music, we also provided MIDI files generated from the Plaine & Easie data as an illustration of the incipits. However, we told the experiment subjects that the definitive source for similarity judgements is the notated music, and that the MIDI files only serve as an illustration.

The metadata from the RISM A/II collection (composer, work title, title of the movement, instrumentation etc.) was not shown to the human experts. They only saw the notated music of the incipits and could listen to a MIDI rendition, as can be seen in Figure 1.

3.2 Experts

Müllensiefen et al. point out [6] that music experts tend to have stable similarity judgements, in other words, do not change their mind on what is melodically similar when asked to perform the same judgements a few weeks apart. Subjects with stable similarity judgements, in turn, seem to have the same notion of melodic similarity. In order to establish a meaningful ground truth, we therefore tried to recruit music

[1]Lilypond (see `http://lilypond.org`) is an open source music typesetter.

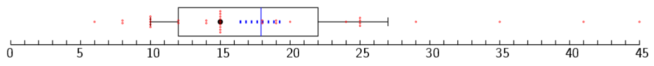


Figure 2: The experience of our experts, in years. The box extends from the first to the third quartile. The whiskers mark the bottom and top 10 percent. Every data point is shown as a little dot. The median is marked with a fat dot, the mean is shown as a vertical line. The dashed horizontal line around the mean marks one standard deviation below and above the mean.

experts as our experimental subjects. We asked people who either have completed a degree in a music-related field such as musicology or performance, who were still studying music theory, or who attended the International Conference on Music Information Retrieval Graduate School in Barcelona 2004 to participate in our experiment.

All of our experts play at least one instrument or sing, most play several instruments. See Figure 2 for a box-and-whisker plot showing their musical experience in years.

3.3 Instructions, tasks

We asked the subjects to rank all candidates that resemble the query by their melodic similarity to the query. Candidates that seemed completely different from the query could be left unranked. The ranking was to be done by reordering the given candidates such that the candidate most similar to the query was at the top, followed by less and less similar candidates, and finally a number of candidates without any assigned ranks that did not resemble the query at all. By asking people to reorder a list instead of picking a rank from a scale, we avoided suggesting how long the ranked list should be, and we also made it easy for the experts to judge whether they ranked all candidates correctly by looking at a local ordering only. It was sufficient to ensure that for each pair of consecutive candidates in the ranked part of their reordered list, the incipit that was ranked higher was more similar to the query than the other incipit of the pair.

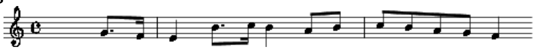
We asked the experts to regard transpositions of a melody as identical, as well as melodies that are notated slightly differently, but in a way that does not affect the way they sound. For example, melodies that are notated with different clefs, but are otherwise the same, should not be viewed as different. In cases where two incipits were taken from similar pieces, but covered different amounts of musical material, we asked the subjects to only consider the common parts of the two incipits for the comparison.

We asked every subject for about 2 hours of his time. If somebody did not manage to work on all queries within that time, we ended the experiment anyway. Therefore, not all queries were judged by all experts. For example, only 25 out of all 35 experts ranked the top candidate shown in Table 3. We asked the experts to work carefully, even if that meant that they could not finish all 11 queries within two hours.

3.4 Threats to the validity of results

- **Filtering errors.** It is possible that we filtered out some incipits although they are similar to the query. Our ground truth, therefore, could be incomplete. However, this does not threaten the validity of the ranking of those candidates that we did include.

Table 1: **Ground truth for "Roslin Castle". Table contents: median rank, incipit with title and RISM A/II signature, box-and-whisker plot showing the ranks assigned by our subjects, and a bar composed of squares visualizing the Wilcoxon rank sum test results for every preceding incipit. For details see Section 4.2.**



Query: Anonymus: Roslin Castle, RISM A/II signature: 800.000.193

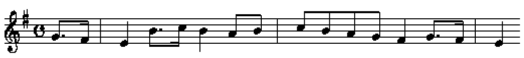
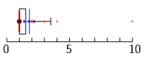
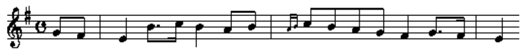
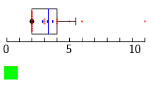
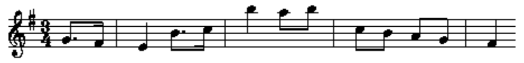
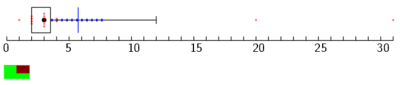
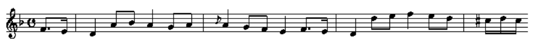
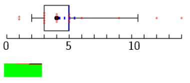
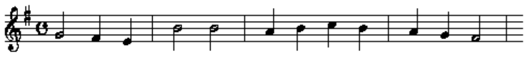
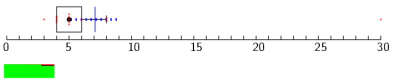
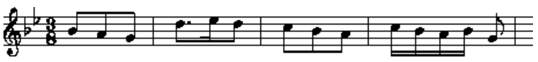
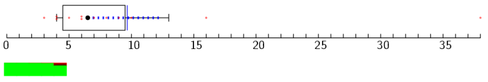
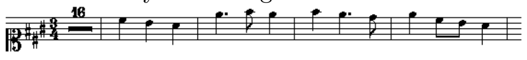
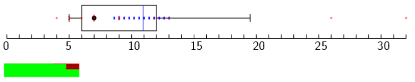
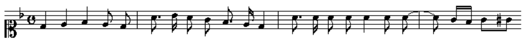
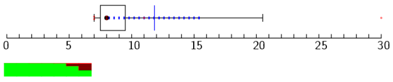
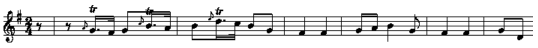
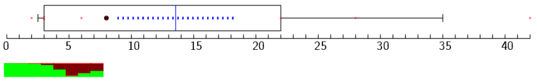
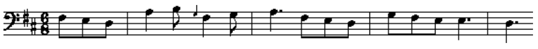
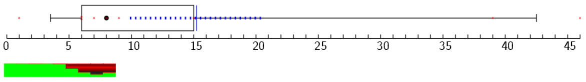
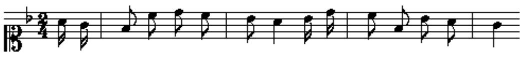
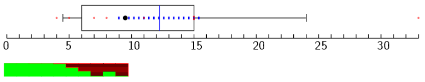
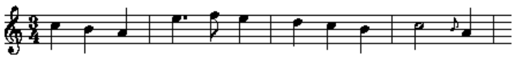
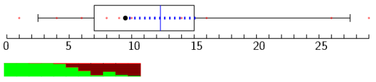
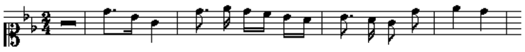
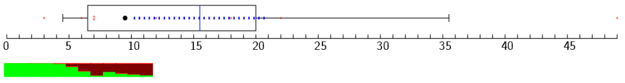
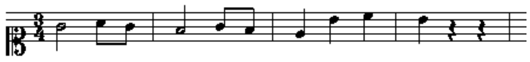
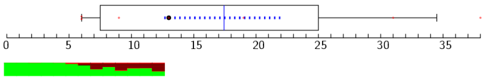
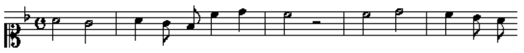
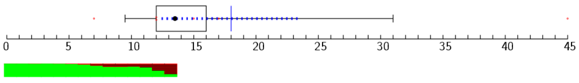
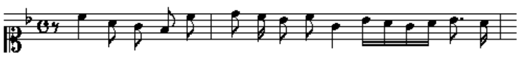
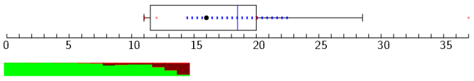
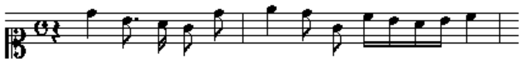
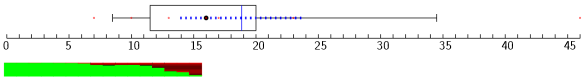
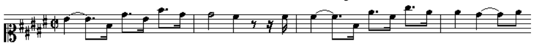
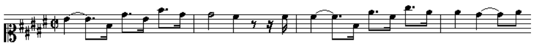
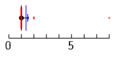
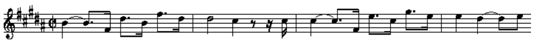
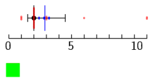
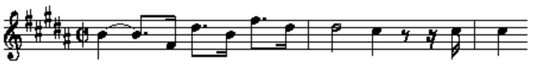
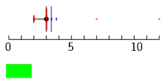
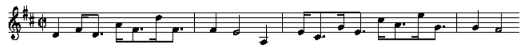
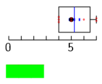
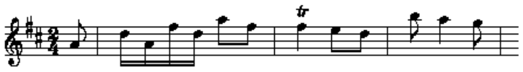
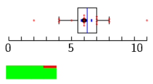
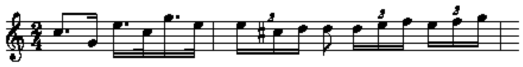
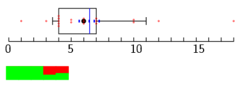
Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature	Ranks, Wilcoxon Test Results (p-values: dark upper area)
1	 Anonymus: Roslin Castle. 000.109.446	
2	 Anonymus: Roslin Castle. 000.111.779	
3	 Anonymus: Roslin Castle. 000.112.692	
4	 Anonymus: Roslin Castle. 000.132.330	
5	 Anonymus: Roslin Castle. 000.112.625	
6.5	 Anonymus: Allegro. 704.000.704	
7	 Anonymus: Care Jesu. 400.196.546	
8	 Robert Führer: Vesperae. 220.000.909	
8	 Florian Leopold Gaßmann: Trios. 702.001.807	
8	 Anonymus: De France et de Navarre. 700.008.178	
9.5	 Anonymus: Vernünftige Lust. 000.114.257	
9.5	 Anon.: Pour me venger de l'ingrate. 000.109.156	
9.5	 G. Molinari: Mecum enim habeo. 850.503.217	
13	 T. W. Fischer: Masses. 550.161.144	
13.5	 G. J. Werner: Puer natus. 530.004.292	
16	 Anonymus: Constitues eos principes. 850.028.763	
16	 Girolamo Chiti: Per singulos dies. 850.503.825	

Table 2: Ground truth for an Aria by Johann Adolf Hasse as query. For details see Section 4.3.



Query: J. A. Hasse: Artemisia, Aria no. 16, Andantino/Allegretto, RISM A/II signature: 270.000.749

Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature	Ranks, Wilcoxon Test Results (p-values: dark upper area)
1	 J. A. Hasse: Artemisia. 270.000.749	
2	 J. A. Hasse: Artemisia. 270.000.746	
3	 J. A. Hasse: Artemisia. 270.000.748	
5	 J. A. Hasse: Tito Vespasiano. 270.000.530	
6	 A. C. Rezel: Ihr die ihr mit vergnügtem Blick. 240.003.707	
6	 J. Touchemoulin: Sonatas. 706.000.461	

- **Sequence effects.** The initial order of candidates as well as the order in which queries are presented to the experts could have an impact on the results. Experts could be tempted to leave the order similar to the initial order, and they get more tired and at the same time more skilled at using our interface over the course of the experiment. We addressed these problems by randomizing the order of queries for every participant, and we also put the candidates in a new random order whenever a new query appeared on the screen.
- **Carelessness of experts.** For some queries, such as the "White Cockade" shown in Table 3, we included the query itself among the candidates. Careful experts should put it at the very top of the ranked list. Not everybody did, but enough of the experts were careful. This query was recognized as most similar to itself with high statistical significance: the Wilcoxon rank sum test, which we used as described in Section 4.1, shows that for every candidate that was not identical to the query, the probability of the null hypothesis is ≤ 0.0001123.

4. RESULTS

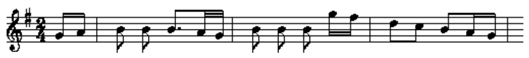
4.1 Evaluation methodology

For every query, the subjects were asked to choose and rank as many of the candidates for matches as they thought had some similarity to the query. Those candidates without any similarity could be left unranked. This gives us a list of ranks for every candidate. These lists tend to be longer for the candidates that are more similar to the query. For example, 20 subjects ranked the first candidate in Table 1. Not all of them assigned rank 1 to this incipit, but the median is still 1. Only 6 people, however, ranked the incipit whose median rank is 13.5.

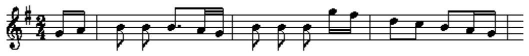
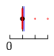
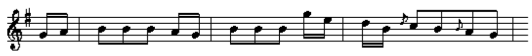
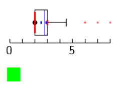
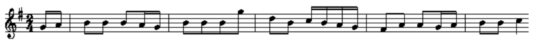
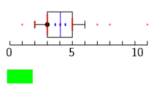
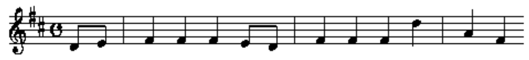
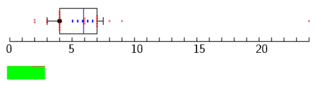
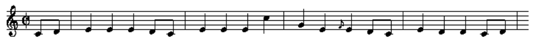
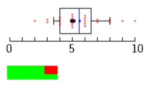
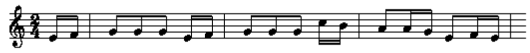
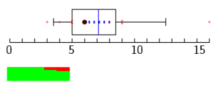
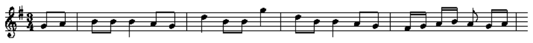
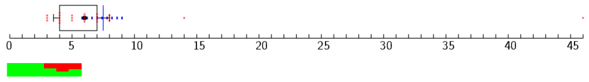
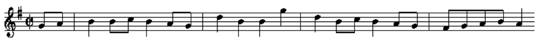
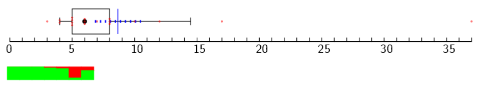
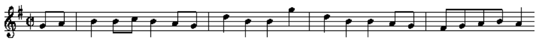
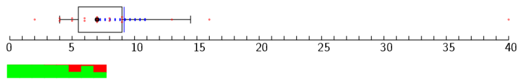
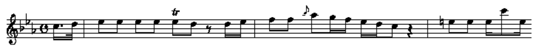
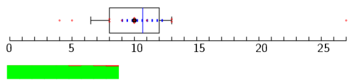
We did not exclude any expert's opinion from our final ground truth. Four data sets were left out of our evaluation because they had resulted from aborted experiments which were restarted from scratch by the same experts after encountering technical problems with browsers. If we had included those four data sets, we would have partially counted four experts' opinions twice.

To obtain a ground truth, we ordered the candidates by their median rank and then by their mean rank. In addition, for every ranked candidate, we applied the Wilcoxon rank sum test to the ranked candidate and every incipit that was ranked higher. The Wilcoxon rank sum test, given two samples, determines the probability of the null hypothesis (p-value), that is, the hypothesis that the median values are the same for the whole two populations from which the samples were taken. We used it to find out how likely it is that the differences in ranking observed by us are only caused by our choice of 35 people out of the whole population of music experts. A low p-value resulting from the Wilcoxon test means that the difference in medians is probably not a coincidence. A large p-value does not necessarily mean that the medians are the same, but just that we do not have compelling evidence for them being different.

Table 3: Ground truth for "The White Cockade" by J. F. Latour as query. Only one out of the top nine pieces, "Cotillons", is not the same piece as the query. As one should expect, the Wilcoxon rank sum test results warrant a separator between the first nine incipits and the tenth, which is from a different piece and at the same time clearly different from the preceding incipits. For details see Section 4.3.



Query: J. F. Latour: The White Cockade, RISM A/II signature: 000.111.706

Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature	Ranks, Wilcoxon Test Results (p-values: dark upper area)
1	 J. F. Latour: The White Cockade. 000.111.706	
2	 Anonymus: White cockade. 000.113.506	
3	 J. F. Latour: The White Cockade. 000.116.073	
4	 E. Hille: Der verurteilte Hochlandsmann. 451.503.814	
5	 J. F. Latour: The White Cockade. 000.113.932	
6	 Anonymus: Cotillons. 190.018.612	
6	 Anonymus: White Cockade. 000.135.676	
6	 Anonymus: White Cockade. 000.127.493	
7	 Anonymus: White Cockade. 000.132.448	
10	 Friedrich II. (der Große): Sonatas. 200.022.611	

4.2 The resulting ground truth tables

We visualize the ranks assigned to each candidate with a box-and-whisker plot. The box extends from the first to the third quartile. The whiskers mark the bottom and top 10 percent. Every data point is shown as a little dot. The median is marked with a fat dot, the mean is shown as a vertical line. The dashed horizontal line around the mean marks one standard deviation below and above the mean. The numbers on the scales reflect ranks.

Below every box-and-whisker plot except for the first one, we visualize the Wilcoxon rank sum test results with a horizontal bar that is composed of one square for every incipit which is ranked higher than the current one. Each of these squares has a dark upper area with a fine pattern of horizontal stripes and a lower area with a lighter, solid colour. The size of the dark upper area reflects the p-value (see Section 4.1 for an explanation of what the p-value means).

For incipits where every square in the Wilcoxon visualization is almost entirely light-coloured, we can be reasonably sure that all preceding incipits should indeed be ranked higher. Wherever this is the case, we draw a horizontal line immediately above the incipit. For Table 1, we set the threshold for the maximum probability for the null hypothesis at 0.25. In other words, we draw a horizontal line above every incipit where the p-value is less than 0.25 for every single incipit that appears higher in the list. Most actual probabilities are much lower than that, as the visualization of the Wilcoxon tests in Table 1 shows.

For "Roslin Castle" (Table 1), we find five clearly distinguishable groups that way. The incipit with median rank 1 is generally considered the most similar incipit to the query. For the incipit with median rank 2, the Wilcoxon test shows that the probability for the null hypothesis is p=0.00006722. Therefore, we consider the difference in median values statistically significant and separate the second incipit from the first with a horizontal line. For the incipit with median rank 3, the difference in medians is statistically significant for the comparison with the first incipit (p=0.0002765), but not for the comparison with the second incipit (p=0.6363). This is reflected in the Wilcoxon visualization bar, which consists of one almost entirely light-coloured square on the left for the comparison of the third incipit with the first one, and one mostly dark square on the right for the comparison of the third incipit with the second one. Since there is no statistically significant difference between the second and third incipit, we group them together and do not separate them with a horizontal line. The third group consists of the incipit with median rank 4. The highest of its three p-values resulting from the Wilcoxon tests for its three predecessors is 0.07633. The fourth group again consists of one single incipit, while for all other incipits, there are no statistically significant differences in median ranks. Either we did not have enough subjects who ranked these incipits, or people simply do not consider the dissimilarities between the remaining incipits and the query significantly different.

The tables shown in this paper are not complete. We cut them off a bit after the last detected border between clearly distinguishable groups because the ranking becomes less reliable and therefore less interesting towards the bottom of the tables. The complete data are available online at http://give-lab.cs.uu.nl/orpheus.

4.3 Musical properties of the identified groups

In Table 1 ("Roslin Castle"), the candidate with the highest rank looks as if it would begin with the query and therefore should, according to our instructions, be regarded as identical to the query since only the common part should be considered. If one looks more closely, however, one notices that the key signatures are different. The resulting differences in two notes, however, are not big enough for our experts to consider it very different from the query. The incipits with median ranks 2 and 3 constitute the second group. Both begin differently from the query - the incipit with median rank 2 has slight differences in rhythm at the beginning and two grace notes added in the second measure, while the incipit with median rank 3 has its second measure transposed by an octave. Otherwise their beginnings are the same as the query. Our experts agree that these incipits are both less similar than the incipit with median rank 1, but they disagree on whether the transposition of a measure by an octave or the modified rhythm and added grace notes should be seen as a greater dissimilarity. Because of this, these two incipits are combined into one group. The experts agree that the incipit with median rank 4 is significantly different from those preceding it. This is justified by a minor difference in rhythm in measure 1 and a major one in measure two – the first note is just a grace note, so there is no group of four descending eighth notes in that measure as in all preceding incipits. The incipit with median rank 5 is again significantly different. The rhythm is changed in several ways, leading to a very noticeable difference in measure 3. The third note in this measure corresponds to the first note in measure 2 of all preceding incipits. Because here this note is not at the beginning of the measure, it is much less emphasized, which changes the character of the melody. The last statistically significant border between groups is that between the incipits with median ranks 5 and 6.5. The latter is the first incipit of a different piece, and it also has a different time signature, so we would expect a border between groups here. Another border could be expected between the second and third incipit with median rank 9.5 because the interval sequence at the beginning changes noticeably here. However, at this point in the ranked list, we do not have enough votes per incipit for finding a statistically significant difference.

The top three candidates for J. A. Hasse's "Artemisia" (see Table 2) are very similar. The incipit with median rank 1 is identical to the query, that with median rank 2 is written with a different clef, but otherwise identical to the query, and the incipit with median rank 3 is identical to the first half of the query. Although they were instructed to disregard such differences, our experts still agreed that simply notating the query differently changes it less than omitting the second half, leading to statistically significant differences in the rankings. The incipit with median rank 5 is a somewhat similar melody by the same composer, but from a different work ("Tito Vespasiano"). It is similar to the query because it also begins with a motive built from notes from a triad (the tonic) and with a dotted rhythm, followed by a variation of the same motive that is based on another triad, the dominant. However, the rhythm is inverted in "Tito Vespasiano". All other candidates are ranked lower, but without further statistically significant differences in rank. The next candidates also begin with a triad that is split up in a similar way, sometimes also with a dotted rhythm, but not followed by a similar motive based on the dominant.

Table 3 shows that our experts correctly recognized that the incipit that is most similar to the query is the query itself. The incipit with median rank 2 has some minor differences in rhythm, some added grace notes, and two different eighth notes instead of one quarter note in the last measure. Surprisingly enough, the incipit with median rank 3, about which we could say pretty much the same as about that with median rank 2, is ranked lower, and this difference is statistically significant. The remaining incipits of "The White Cockade" or a German version of the same song, "Der verurteilte Hochlandsmann", are all ranked lower, but without statistically significant differences in their median ranks. In that group, there is one incipit from a different piece (Anonymus: "Cotillons") that is melodically very similar. There is again a noticeable border between the incipit with median rank 7 and that with median rank 10. The latter is a sonata by Friedrich II, where only the first five notes are similar.

5. CONCLUSIONS

Our ground truth for 11 incipits from the RISM A/II collection can serve as a basis for a benchmark for evaluating music information retrieval systems. The complete ground truth, along with the sets of queries, candidates, and experimental results, can be found at http://give-lab.cs.uu.nl/orpheus. We encourage music retrieval researchers to apply their favourite methods to the RISM A/II collection and compare their results to our ground truth.

In order to use standard evaluation procedures using measures such as precision and recall, one could simply define a threshold and call every match in our ground truth that is ranked higher than the threshold "relevant" and the rest of the matches "irrelevant". Any report on the basis of this ground truth must then mention this threshold value. However, to take advantage of the ranking we established, one could also look at the precise order in which matches are returned. A good music information retrieval system should not only retrieve the matches that are ranked highly in our ground truth, but also return them in an order that does not violate the order of the groups of incipits we found by using the Wilcoxon rank sum test. The correct order of our incipits within groups, however, should be regarded as unknown. For example, for Table 1, we do not know whether the incipit with median rank 2 or that with median rank 3 should really be ranked higher, but we do know that both of them should be ranked lower than the incipit with median rank 1 and higher than any other incipit in our list of candidates. It is also important to take into consideration that we excluded candidates that are identical to other candidates or transpositions of other candidates, or candidates that are identical to the beginning of other candidates. Any additional such candidates should be regarded as belonging to the same group as their identical counterparts, transpositions, or the incipits with whose beginnings they are identical. Finally, for two reasons, there is the possibility that a good music retrieval system can find additional good matches: our filtering method is not guaranteed to be perfect, and later editions of the RISM A/II collection will always contain additional incipits. If such additional matches occur, one should check whether those matches were included in our lists of candidates.

Our ground truth is already applicable as it is, and the Wilcoxon test results give an indication of how much one can rely on the borders between the distinguishable groups we found. However, additional work could be beneficial, both for increasing the number of queries and for making more borders between groups emerge and for making the existing ones more reliable. Buckley's and Voorhees's work [1] indicates that having about 25 queries would be desirable for being able to apply statistical tests for comparing different retrieval systems. Also, more experiments could help if our candidate lists need to be extended either because a mistake in the filtering process gets noticed or because new incipits that are similar to a query were added to the RISM A/II collection after 2002.

Our ground truth can not only be useful as a benchmark for music retrieval systems, but also for finding out which musical features are relevant and how much they influence melodic similarity. For doing that, one needs to find incipits in the ground truth where very few features are different, but lead to significant differences in ranking. An example can be seen in Table 2, where the first three incipits show that notating music in a different clef leads to a smaller difference than removing about two measures from the end. A somewhat less trivial observation that can be made in the same table is that repeated harmonic patterns matter, and that a dotted rhythm seems to be perceived as more similar to an inverted dotted rhythm than to a sequence of notes with the same durations.

6. ACKNOWLEDGEMENTS

We thank all participants of our experiments, both the ISMIR 2004 Graduate School attendees and the participants from Utrecht.

References

[1] C. Buckley and E. M. Voorhees (2000). Evaluating evaluation measure stability. *Proceedings of the 23rd ACM Conference on Research and Development in Information Retrieval (SIGIR),* pp. 33–40.

[2] J. S. Downie (2003). Toward the scientific evaluation of music information retrieval systems. *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 25–32, Johns Hopkins University, Baltimore (MD), USA.

[3] D Hawking (2001). Overview of the TREC-9 Web Track. *Proceedings of the 9th Text Retrieval Conference TREC-9*, Gaithersburg, MD, 2001. http://trec.nist.gov/pubs/trec9/papers/web9.pdf

[4] J. Howard (1997). Plaine and Easie Code: A Code for Music Bibliography. In: *Beyond MIDI: The Handbook of Musical Codes,* edited by E. Selfridge-Field. pp. 362–72

[5] K. Lemström, V. Mäkinen, A. Pienimäki, M. Turkia, and E. Ukkonen (2003). The C-BRAHMS Project. *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 237–238, Johns Hopkins University, Baltimore (MD), USA.

[6] D. Müllensiefen and K. Frieler: Measuring melodic similarity: Human vs. algorithmic judgments. *R. Parncutt, A. Kessler & F. Zimmer (Eds.) Proceedings of the Conference on Interdisciplinary Musicology (CIM04) Graz/Austria, 15-18 April, 2004.*

[7] L. Prechelt and R. Typke (2001). An Interface for Melody Input. *ACM Transactions on Computer-Human Interaction* 8(2), pp. 133–149.

[8] *Répertoire International des Sources Musicales (RISM). Serie A/II, manuscrits musicaux après 1600.* (2002) K. G. Saur Verlag, München, Germany. http://rism.stub.uni-frankfurt.de

[9] E. Selfridge-Field (1998). Conceptual and Representational Issues in Melodic Comparison. *Computing in Musicology (11),* pp. 3–64.

[10] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering and R. van Oostrum (2003). Using Transportation Distances for Measuring Melodic Similarity. *ISMIR 2003: Proceedings of the Fourth International Conference on Music Information Retrieval*, pp. 107–114.

[11] R. Typke, R. C. Veltkamp, and F. Wiering (2004). Searching notated polyphonic music using transportation distances. *Proceedings of the ACM Multimedia Conference, New York, October 2004*, pp. 128–135.

[12] E. M. Voorhees and D. K. Harman (2000). Overview of the eighth Text REtrieval Conference (TREC-8). *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pp. 1–24. NIST Special Publication 500-246. Electronic version available at http://trec.nist.gov/pubs.html

[13] F. Wilcoxon (1945). Individual comparisons by ranking methods. *Biometrics,* 1(6):80–83.

An In-depth Study of the Automatic Detection and Correction of Spelling Mistakes

Sau Kwan Chan
University of Glasgow
17 Lilybank Gardens
Glasgow
chansk@dcs.gla.ac.uk

Ben He
University of Glasgow
17 Lilybank Gardens
Glasgow
ben@dcs.gla.ac.uk

Iadh Ounis
University of Glasgow
17 Lilybank Gardens
Glasgow
ounis@dcs.gla.ac.uk

ABSTRACT
This paper discusses the issues involved in an information retrieval system when spelling errors are encountered in a query. We look at two classical algorithms that may be used to correct these errors and their consequent effect on the system's performance. The algorithms are the Levenshtein Distance and the Longest Common Subsequence. We experiment on a variety of test data and explore the impact of certain errors on an information retrieval system. We produce an in-depth study of the use of the two algorithms in an information retrieval environment. We evaluate the algorithms using two measures. Using two different data collections, the WT2G and .GOV collections, of differing size, we observe the effects of different types of spelling errors and their successful error correction rate. In particular, we find that the precision/recall results for query terms with spelling errors, regardless of the collection and its size, are reduced by up to half when compared to correctly spelled queries, showing that there is a justifiable need for error correction.

1. INTRODUCTION
Information retrieval (IR) systems are ultimately designed and developed to be used by end users. The task of formulating a query to use with the system is not necessarily a simple task and, among other things, users are prone to making errors with the spelling of a query for various reasons. A user may be in a rush and can not afford the time to check the spelling of a query, the user may not be searching with their native language or the user simply does not know the exact spelling of the term. We define a spelling error to be a query term that is not spelt in the same way as the terms in the lexicon of the system. For example, 'retreival' is a spelling mistake of the term 'retrieval'.

It is well known that queries containing spelling errors are common place. The work done by Kukich supports this fact [11]. Kukich showed that there exists approximately 0.2% - 3% spelling errors in English texts. This is regardless of the size of the corpora and will usually cause the IR system's recall rate to decrease.

While it is logical to correct spelling errors, there does not exist a comprehensive study into the effects of using an error correction algorithm in an IR environment. It can be questioned as to whether using string analysis algorithms to correct spelling errors is justifiable in terms of successful results and the costs involved with respect to an IR system. This paper aims to provide an in-depth study on the use of two classical spelling correction algorithms and to ascertain if the benefits do or do not outweigh the costs, regardless of the collection size or content.

The vast differences in collection size brings forward the design issue of whether to correct the errors in the documents of a collection, to correct the given query, or to correct both the documents and the queries. Similar to cross-language retrieval, due to the ease with which a collection can grow in size, the most logical method is to detect and correct spelling errors made in the queries rather than in the collection. We take advantage of the fact that queries tend to consist of a few words, so our algorithm needs only to check these terms making the querying process faster.

While other error correction techniques exist [1, 11], we concentrate on two classical and well established string matching algorithms to detect misspelled terms, namely the Edit Distance and the Longest Common Subsequence algorithms. The results show that both algorithms can correct misspelled terms but the degree of success varied. Our two methods of evaluation provide evidence supporting this. The Edit Distance was found to be more effective than the Longest Common Subsequence with most of the tested type of errors. Our first method of evaluation details the limits of each algorithm in regards to the number of spelling mistakes that can be made before a severe degradation in performance is observed. This is the rate of successful error corrections. We use precision/recall measures as the second evaluation measure. Our analysis shows that using either of these algorithms prevented precision/recall measures to fall to the values of those queries without error correction supporting the need for these algorithms to be used.

In the next two sections, we discuss various types of spelling errors and two classical algorithms that can be used to correct these errors. We look at how they differ in their approach to correcting erroneous terms. In Section 4, we set out the environment in which our experiments are performed and explain the two evaluation methods used. In addition, we consider how we can keep processor performance high and memory usage low. We evaluate our algorithms in Section 5 and show the results from our study, and discuss the reasons why we got the results we did. Finally, we draw the conclusions of the experiments, discussing to what extent our hypotheses were correct and give possible directions of future developments in Section 6.

2. SPELLING ERROR DETECTION AND CORRECTION

There are four different basic types of spelling errors according to Durham, Lam and Saxe [6]. These are:

1. There are extra characters in the term. For example, *'retrievall'* has an extra 'l'.
2. There are letters missing from the term. For example, *'retreval'* where the missing letter is 'i'.
3. Letters in the term are wrong. For example, *'retreeval'* has an 'e' where it should be 'i'.
4. The transposition of letters. For example, *'retrEIval'* instead of *'retrIEval'*.

We use these four types as a basis for our study. It is clear that there is no limit to the number of errors made in one term. A term could contain as many or even more errors than correct characters. Hence, a correction algorithm should not be limited by the number of errors made nor by the length of the term.

In addition, a query can have multiple different types of errors within it. For example, a query with the term *'inforatoin'* may contain a letter missing (Type 2) as well as a transposition of letters (Type 4) when it should have been *'information'*. Therefore, an IR system that has a spelling error detection and correction functionality should take this into account.

We aim to investigate the effects of having this functionality in regards to querying performance across different corpora and test which correction algorithm is better suited for which type of spelling error.

3. LEVENSHTEIN DISTANCE AND LONGEST COMMON SUBSEQUENCE ALGORITHMS

We use the lexicon of the system as our 'dictionary' of correctly spelled words. This is a logical approach since the system will not propose correction suggestions that are not in the index and so any corrected terms are guaranteed to retrieve some documents.

An error is detected when a query term does not exist in the lexicon. If no error is found then the system runs as normal but if an error is detected then we use one of two classical algorithms to search for possible corrections: the Levenshtein Distance algorithm or the Longest Common Subsequence algorithm.

The Levenshtein Distance was a concept introduced by V. Levenshtein in 1965 to measure the distance between two strings [12]. This algorithm is more commonly known as the Edit Distance algorithm. This is because the algorithm measures the difference between two strings by finding the number of edits it takes to change one term to another. An edit is taken as a deletion of a character, substituting a character with another, or an addition of a character. So to change 'bread' to 'bean' would require two edits, a deletion of the character 'r' and a substitution of the character 'd' with the character 'n'. Therefore, the distance between these two strings is two.

Mathematician G. de B. Robinson first described an algorithm for finding the longest common subsequence (LCS) of two strings in 1938 [13]. The Longest Common Subsequence (LCS) algorithm finds the string sequence of largest length that exists in both of the two input strings. Say the two input strings were 'the sand over there' and 'cat sat on the mat', then the longest common subsequence would be 't sa o the' with length value ten, since the algorithm uses sequences regardless of the sequence's content. Whitespace is included in the computation.

Both these algorithms find possible corrections for misspelled query terms by looking at different properties of the input string. The Edit Distance measures the difference between two strings while the LCS algorithm measures what the two strings have in common. For the best possible correction to a misspelled term we desire the Edit Distance between the error term and the possible correction from the lexicon to be as small as possible. Whereas the larger the LCS between the terms in the lexicon and the misspelled term, the better the chance of the error being corrected with the intended term.

4. ENVIRONMENTAL SETTING

We use an existing search engine and choose what weighting model we use (see Section 4.1). In addition, we detail the two data collections (WT2G and .GOV), on which we performed our experiments (see Section 4.2). Finally, we discuss our evaluation methodology and the steps taken to ensure optimal processor performance and memory usage, an important consideration in developing IR systems (see Section 4.3).

4.1 PL2 Weighting Model

For the experiments we used the Terrier IR system[1]. Terrier uses various models for its matching functionalities. We used the PL2 matching function in the experiments which is a member of the Divergence From Randomness models [2]. These models were developed by Amati and van Rijsbergen based on their work on a probabilistic framework for IR. Using PL2, a document's relevance score d for a query Q is computed by the following formula:

[1]Terrier is developed by the University of Glasgow Information Retrieval Group. Further information can be found in http://ir.dcs.gla.ac.uk/terrier

$$\begin{aligned} score(d,Q) &= \sum_{t \in Q} w(t,d) \qquad (1) \\ &= \sum_{t \in Q} qtf \cdot \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{tfn}{\lambda} + (\lambda + \\ &\quad \frac{1}{12 \cdot tfn} - tfn) \cdot \log_2 e + 0.5 \cdot \log_2(2\pi \cdot tfn)\big) \end{aligned}$$

where λ is the mean and variance of a Poisson distribution, $w(t,d)$ is the weight of document d for query term t and qtf is the number of occurrences of term t in the query.

The normalised term frequency tfn in equation (1) is computed using *normalisation 2*[2]:

$$tfn = tf \cdot \log_2(1 + c \cdot \frac{avg\_l}{l}) \qquad (c > 0) \qquad (2)$$

where tf is the within document frequency of a given term, l is the document length and avg_l is the average document length in the whole collection. The value c is a free parameter and can be different for different collections. By measuring the normalisation effect, c is automatically estimated [9]. We use short queries and so the value of c is 10.57 and 1.25 for WT2G and .GOV, respectively.

4.2 TREC Data Collections

In order to have an idea on how the algorithms perform with any given corpora regardless of size, we experiment using two collections. One relatively small collection (WT2G) and one that is much larger (.GOV).

The WT2G collection is 2 gigabytes in size with 50 queries, i.e. TREC[2] topics 401 - 450 [7]. This collection contains documents of a miscellaneous nature. The larger collection is called .GOV and is 18 gigabytes [4, 5]. The .GOV collection is a sample of the '.gov' domain. We used the collection along with TREC11-12 which has 99 queries in total. All TREC queries were checked beforehand and found to be correctly spelled.

We ran our algorithms on both collections so that we could observe the effects of spelling error detection and correction on both small and large collections, as well as on a range of different data.

4.3 Experimental Methodology

Our purpose in these experiments is to explore and survey the effects on querying performance with respect to two classical algorithms over different corpora. We evaluate the two algorithms by measuring their successful error correction rate and by using precision/recall measures.

As mentioned in Section 2, there are four basic types of spelling errors. As the TREC queries are correctly spelled, we can simulate a spelling error by randomly changing one of the existing terms of each TREC query. Then we incrementally increase the number of errors to quantify the amount of errors each algorithm can handle before a visible degradation in performance. To illustrate, we change the TREC queries to have one extra letter (Type 1), then change the queries to have two extra letters and so on. This is repeated for each error type. This allows us to measure the effect each different error type has and whether one error type is more difficult to correct than another, as well as each algorithm's cut off point in performance degradation regarding the number of errors.

We also measure the effects on querying performance using the traditional precision/recall values for each type of error. We use the original provided TREC queries and record the precision/recall values. This is our baseline, which we hope to match. On the other hand, we also record the precision/recall values for when there is no error correction for the modified query terms. By comparing these values with the precision/recall values obtained when the error correction algorithms are used, we can decide on the impact of using error correction algorithms on querying performance.

As with any system, we wish the running time of the system to be as fast as possible due to the large size of the corpus. It has been found that spelling errors have a frequency of at least a factor of 100 lower than correctly spelled terms [14]. Hence, we do not need to process terms in the lexicon with a lower term frequency than the misspelled query term since any term that has a lower frequency than a misspelled term is not likely to be correctly spelled itself. This cuts down the number of terms in the lexicon that need to be processed which in turn saves processing time.

To further reduce processing time, it is safe to assume that a user will enter correctly the first initial of a query term. This is the same assumption that the popular search engine Google uses. With this assumption, the system needs only to look at terms in the lexicon that have the same initial character. We created an index of the places in the lexicon where each initial character starts. This requires thirty six restricted binary searches but is run once on loading of the lexicon.

The amount of memory used by the system must also be kept to a minimum for practical purposes. We can not store all the terms in memory. Using the idea of dynamic programming by Hirschberg, we can use just two 1-dimensional arrays to store previous and current calculations instead of a $m \times n$ matrix for the character comparisons [10]. This reduces the memory needed for the Edit Distance and LCS algorithms from $O(mn)$ to $O(n)$. This is possible for both algorithms as we only need the last value of the matrix, which contains the distance or LCS length, for the purposes of this experiment.

5. EVALUATION

We evaluate the two algorithms by graphing each algorithm's performance on the various types of errors (Types 1-4), to see the rate of successful error corrections and behaviour of the algorithms as the number of errors increase. In addition, by calculating the precision/recall values for each error type for both algorithms, we can compare and contrast the

[2]More information about the Text REtrieval Conference can be found on http://trec.nist.gov

two algorithms and promote one over the other in terms of querying performance improvement. We evaluate using precision/recall measures for the two used web collections to find out whether the effects of error correction are independent of the size or content of a collection.

5.1 Aims

In our experiments we measure how well each error correction algorithm performs. We compare them against each other and as stand alone algorithms. Based on how these algorithms work, we aim to explore and provide support for the following hypotheses:

1. Precision/Recall measures are higher using error detection and correction algorithms compared to systems without this functionality.
2. The two classical algorithms look at different properties of the query term and will not necessarily give the same results/performance. One algorithm may be better for specific types of errors.
3. Overall, the Edit Distance algorithm should perform better than the LCS algorithm, since the Edit Distance algorithm utilises 3 out of the 4 errors tested in our experiments.
4. The less errors there are in the query term, the better the chances of an accurate correction.
5. If an algorithm performs well on certain types of errors then the algorithm should perform well on a combination of these error types.
6. Error correction is independent of the collection's size and content.

5.2 WT2G Collection Results

Using the WT2G collection, we analyse the experimental results by measuring the rate of successful error corrections and using traditional precision/recall measures. We graph the percentage of accurate corrections while incrementally increasing the number of certain types of errors. We also use the precision/recall measures. With this range of evaluation methods, we can survey the effectiveness of using the Edit Distance and Longest Common Subsequence algorithms in errors correction in IR.

5.2.1 Errors Correction Graphs

For each error type, we graph the percentage of accurate corrections for each increment in the number of errors in the query term (Figure 1).

There is an overall downward trend to the percentage of accurate corrections as the number of mistakes increase, which marginally supports Hypothesis 4. In addition, it is easy to see from the graphs that the two classical algorithms do not give the same corrections all the time since they concentrate on different aspects of a term (Hypothesis 2) - if they did then surely the graphs would be the same for both Edit Distance and Longest Common Subsequence.

While overall, our graphs support Hypothesis 4, error correction rates for a term containing one error are at times less than those for two errors. This is because our dictionary with which we check our query terms is the lexicon of the system. Since the lexicon has not been spellchecked beforehand, we may not detect the same misspelled query terms with one error (which is more common) if they already exist in the lexicon. So while a query term with more errors is harder to correct accurately, it also has a better chance of being detected as an error.

The Edit Distance performs well with error types 1, 2 and 3 (Figures 1(a), 1(b) and 1(c) respectively), with an error correction rate of over 60% for terms with one spelling error. Only the case when characters are swapped around (Type 4 with Figures 1(c)) is the correction rate below 50% for one error. This is expected since as mentioned in Section 3, it is exactly these three edits that are included in the algorithm's calculations. Indeed it performed better than the Longest Common Subsequence algorithm for queries with a single error, supporting Hypothesis 3.

However, in the case of deletion errors (Figure 1(b)) as the number of errors increased, the LCS algorithm managed to outperform the Edit Distance algorithm. Although the percentage did decrease for both algorithms, it was not as sharp a decrease as for the Edit Distance algorithm. In fact, in all four graphs the LCS algorithm appears to have a more stable behaviour even as the number of errors increases.

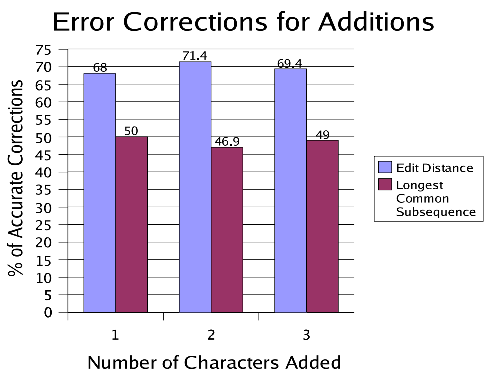
5.2.2 Precision/Recall Results

We calculate the precision/recall measures for the TREC queries which are correctly spelled. We compare the precision/recall values obtained using each spelling algorithm with the precision/recall values obtained from the original unaltered TREC queries and the precision/recall values when there is no error correction. Tables 1, 2, 3 and 4 detail these precision/recall values for each error type.

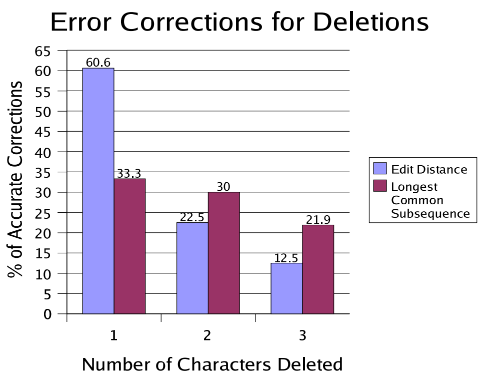
The average precision for queries that had used a spelling correction algorithm never fell below that of those queries that contained errors but were not corrected. Therefore, precision will on average increase with a spelling correction function. The two algorithms are able to cope with all the four common types of errors mentioned for this collection. This supports the first of our hypotheses made in this paper; that precision/recall measures will improve with error correction.

The Edit Distance algorithm performed particularly well for all error types apart from deletion errors (Type 2), outperforming the LCS algorithm. But it should be taken into consideration that while the algorithm performed poorly for increasing amounts of deletion errors, query terms are often short and so as more characters are missing, there are less remaining characters to work with.

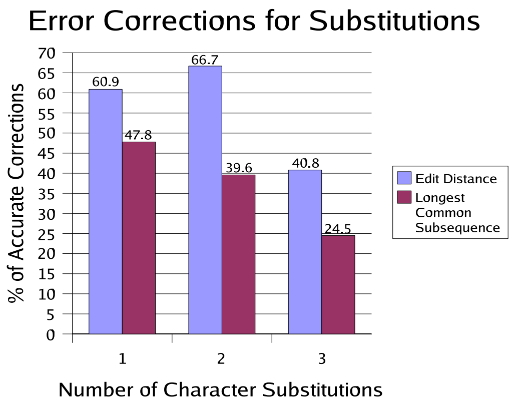
In the case for queries with errors of three extra characters (Table 1) using Edit Distance correction, the number of relevant documents retrieved is greater than those for the original queries with no errors by two documents. However there was a 0.0192 decrease in average precision compared to the baseline average precision. As the query terms now have three extra miscellaneous terms and since the query terms are short the mistake is more likely to be detected.



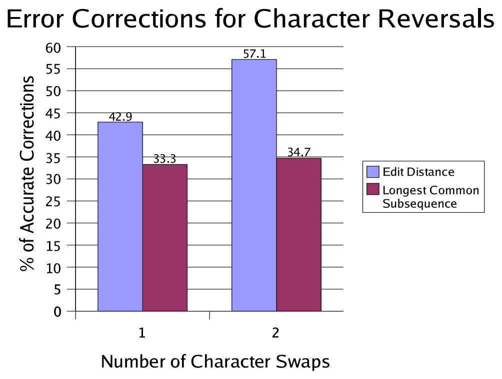
(a) Character additions (Type 1).



(b) Character deletions (Type 2).



(c) Character substitutions (Type 3).



(d) Character reversals (Type 4).

Figure 1: Percentage of accurate corrections for various degrees of four error types (WT2G).

Therefore, in this case it could simply be that more of the query terms are being picked up as errors.

The Edit Distance algorithm did consistently well considering that the average precision for correctly spelled queries was 0.2971. Using error correction average precision was in the region of 0.2 while without correction the average precision was no higher than 0.1548.

Like the Edit Distance algorithm, the LCS algorithm still managed to improve precision/recall values regardless of error type. The number of relevant documents also increased.

While the LCS algorithm did not perform as well as the Edit Distance, supporting Hypothesis 3, the LCS managed to outperform the latter with deletion errors as the number of deletion errors increased as mentioned earlier. Having two character deletions causes a sharper drop in average precision with the Edit Distance when compared with just one deletion, whereas the LCS algorithm has a more stable decrease (Table 2). Deletion errors do not change the sequence of correct characters as reversal errors, (Type 4), do and there are no noisy incorrect characters to take into account like those that addition and transposition errors create (Types 1 and 3 respectively).

5.3 .GOV Collection Results

We continue our experiments with a different collection, the .GOV collection, in order to achieve an overview of how these classical algorithms perform with a different and larger data set.

5.3.1 Errors Correction Graphs

Similar to Section 5.2.1, we graph the percentage of accurate corrections. For the .GOV collection, the LCS algorithm slightly outperformed the Edit Distance algorithm for deletion errors Type (2), as can be seen in Figure 2(b). This could be due to the fact that deletion errors still have their sequence of characters intact in a given query term, which is key to the LCS algorithm.

Figure 2 shows again that, overall, as the number of errors increase, it becomes increasingly difficult for the system to decide what the correct term should be from the possible corrections (Hypothesis 4). It is quite possible that the system will choose a term that has the same property (same Edit Distance or LCS value) as another term yet the former is not the desired term.

Similar to the WT2G collection in Section 5.2.1, the Edit Distance algorithm outperformed the LCS algorithm for most

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**48181**	**2279**	**1682**	**0.2971**	**0.3263**	**0.4880**
Type 1 - 1 error (no correction)	43499	2279	1153	0.1452	0.1697	0.2320
Edit Distance Correction	48781	2279	1665	0.2609	0.2890	0.4140
LCS Correction	48616	2279	1334	0.2040	0.2335	0.3360
Type 1 - 2 errors (no correction)	46440	2279	1154	0.1330	0.1556	0.2340
Edit Distance Correction	48781	2279	1519	0.2210	0.2490	0.3760
LCS Correction	47902	2279	1330	0.2061	0.2318	0.3260
Type 1 - 3 errors (no correction)	41585	2279	1013	0.1548	0.1823	0.2220
Edit Distance Correction	48781	2279	1684	0.2779	0.3107	0.4540
LCS Correction	48181	2279	1255	0.1937	0.2280	0.3200

Table 1: The precision/recall value for correction of character addition errors (WT2G).

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**48181**	**2279**	**1682**	**0.2971**	**0.3263**	**0.4880**
Type 2 - 1 error (no correction)	46378	2279	1072	0.1244	0.1390	0.2220
Edit Distance Correction	47658	2279	1334	0.1938	0.2194	0.3180
LCS Correction	48107	2279	1088	0.1676	0.1937	0.2680
Type 2 - 2 errors (no correction)	46390	2279	1072	0.1229	0.1352	0.2240
Edit Distance Correction	47433	2279	1234	0.1405	0.1483	0.2360
LCS Correction	48569	2279	1137	0.1525	0.1723	0.2560
Type 2 - 3 errors (no correction)	46495	2279	1079	0.1222	0.1373	0.2200
Edit Distance Correction	47237	2279	1655	0.1263	0.1392	0.2300
LCS Correction	48674	2279	1096	0.1293	0.1417	0.2220

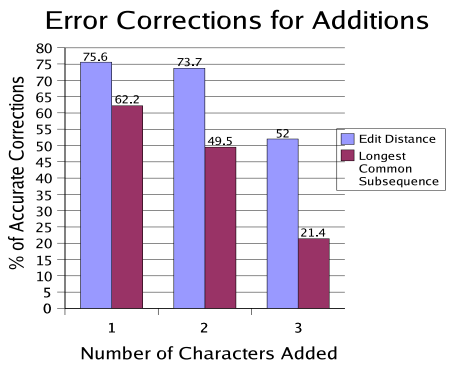
Table 2: The precision/recall value for correction of character deletion errors (WT2G).

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**48181**	**2279**	**1682**	**0.2971**	**0.3263**	**0.4880**
Type 3 - 1 error (no correction)	43751	2279	1107	0.1391	0.1616	0.2420
Edit Distance Correction	48336	2279	1294	0.2224	0.2371	0.3680
LCS Correction	47487	2279	1213	0.1958	0.2210	0.3200
Type 3 - 2 errors (no correction)	43749	2279	1106	0.1390	0.1616	0.2420
Edit Distance Correction	48449	2279	1465	0.2369	0.2513	0.3740
LCS Correction	49443	2279	1163	0.1799	0.2027	0.3080
Type 3 - 3 errors (no correction)	43749	2279	1106	0.1390	0.1616	0.2420
Edit Distance Correction	46137	2279	1243	0.1739	0.1919	0.3120
LCS Correction	48289	2279	1024	0.1632	0.1816	0.2720

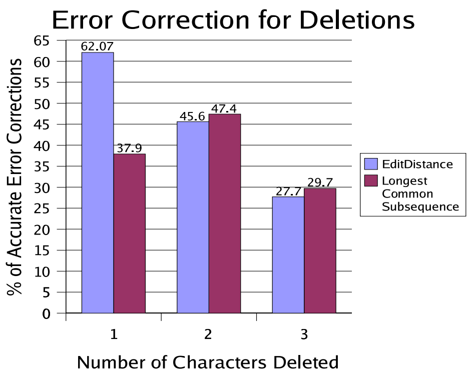
Table 3: The precision/recall value for correction of character substitution errors (WT2G).

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**48181**	**2279**	**1682**	**0.2971**	**0.3263**	**0.4880**
Type 4 - 1 error (no correction)	46367	2279	1174	0.1489	0.1706	0.2600
Edit Distance Correction	48546	2279	1284	0.1993	0.2115	0.3160
LCS Correction	48053	2279	1225	0.1785	0.1952	0.3160
Type 4 - 2 errors (no correction)	45719	2279	1095	0.1356	0.1569	0.2380
Edit Distance Correction	48781	2279	1389	0.2248	0.2518	0.3680
LCS Correction	49787	2279	1108	0.1748	0.1944	0.3020

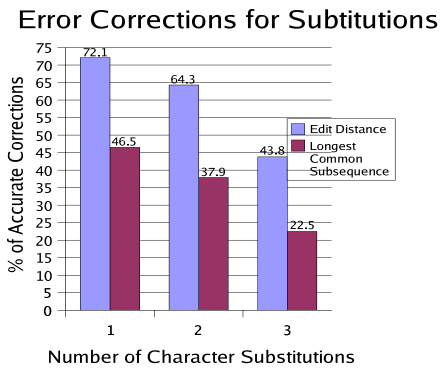
Table 4: The precision/recall value for correction of character reversal errors (WT2G).



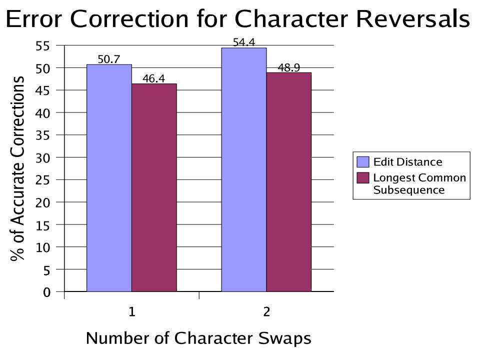
(a) Character additions (Type 1).



(b) Character deletions (Type 2).



(c) Character substitutions (Type 3).



(d) Character reversals (Type 4).

Figure 2: Percentage of accurate corrections for various degrees of four error types (.GOV).

experiments. In particular, there is a 25.6% difference between the two algorithms for the correction of a one character substitution (Type 3 with Figure 2(c)). A possible reason for this behaviour is that the Edit Distance actually takes into account three out of the four types of error tested, as was mentioned in Hypothesis 3.

5.3.2 *Precision/Recall Results*

As we did for the WT2G collection, we calculate the precision/recall measures for the .GOV collection for each error type (Tables 5, 6, 7 and 8).

In general, average precision is always better *with* error correction than without correction. In almost all cases, the number of relevant documents retrieved also increases when using error correction. There was a slight reduction in relevant documents retrieved when there were increasingly more deletion of characters (Table 6).

It can be seen that the average precision is consistent with the results for percentage of successful error corrections found in Subsection 5.3.1. When the percentage of accurate corrections is high, average precision is high and vice versa.

Much like the precision/recall values found in Section 5.2.2, the Edit Distance performs well on all types of error supporting Hypothesis 1.

Not only did precision/recall improve with the algorithms but the number of relevant documents retrieved also increased. In particular, for terms with three extra characters, the Edit Distance managed to increase the number of relevant documents retrieved from 952 to 1371 (Table 5). This is an extra 419 relevant documents. This value is just 11 documents less than when the original queries without spelling mistakes are used (our baseline).

The results for the Longest Common Subsequence algorithm, while not as high as the Edit Distance algorithm, still managed to improve precision/recall values. Hence, Hypotheses 1 and 3 still hold.

As pointed out in the previous subsection, the Edit Distance did not outperform the LCS in all the experiments. The LCS algorithm did manage to perform better in two of the experiments. Without any error correction, average precision dropped by 0.0854 when two characters were transposed (Table 8). Using Edit Distance correction, average precision was improved slightly but more significantly, LCS correction had an average precision more than double that when no error correction was used.

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**98477**	**2090**	**1382**	**0.1334**	**0.1596**	**0.1545**
Type 1 - 1 error (no correction)	97115	2090	949	0.0624	0.0725	0.0838
Edit Distance Correction	98780	2090	1338	0.1187	0.1374	0.1394
LCS Correction	98780	2090	1208	0.0968	0.1087	0.1202
Type 1 - 2 errors (no correction)	97111	2090	952	0.0626	0.0725	0.0848
Edit Distance Correction	98780	2090	1349	0.1279	0.1502	0.1424
LCS Correction	99068	2090	1208	0.1001	0.1198	0.1202
Type 1 - 3 errors (no correction)	97111	2090	952	0.0626	0.725	0.0848
Edit Distance Correction	99068	2090	1371	0.1150	0.1343	0.1434
LCS Correction	99428	2090	1160	0.0853	0.1012	0.1162

Table 5: The precision/recall value for correction of character addition errors (.GOV).

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**98477**	**2090**	**1382**	**0.1334**	**0.1596**	**0.1545**
Type 2 - 1 error (no correction)	97948	2090	893	0.0554	0.0634	0.0737
Edit Distance Correction	95049	2090	971	0.0571	0.0704	0.0838
LCS Correction	95049	2090	900	0.0506	0.0569	0.0707
Type 2 - 2 errors (no correction)	97403	2090	875	0.0573	0.0668	0.0778
Edit Distance Correction	97602	2090	928	0.0671	0.0755	0.0919
LCS Correction	99585	2090	871	0.0542	0.0626	0.0798
Type 2 - 3 errors (no correction)	96950	2090	870	0.0532	0.598	0.0727
Edit Distance Correction	96496	2090	862	0.0492	0.0518	0.0697
LCS Correction	98458	2090	817	0.0515	0.0512	0.0768

Table 6: The precision/recall value for correction of character deletion errors (.GOV).

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**98477**	**2090**	**1382**	**0.1334**	**0.1596**	**0.1545**
Type 3 - 1 error (no correction)	97196	2090	856	0.0524	0.0623	0.0687
Edit Distance Correction	99463	2090	1266	0.0913	0.1153	0.1071
LCS Correction	98374	2090	960	0.0647	0.0793	0.0808
Type 3 - 2 errors (no correction)	97188	2090	861	0.0529	0.0633	0.0687
Edit Distance Correction	97524	2090	1221	0.0820	0.0975	0.0980
LCS Correction	98883	2090	969	0.0736	0.0916	0.0909
Type 3 - 3 errors (no correction)	97188	2090	834	0.0514	0.0606	0.0677
Edit Distance Correction	96538	2090	980	0.0716	0.0741	0.0879
LCS Correction	97334	2090	863	0.0514	0.0565	0.0657

Table 7: The precision/recall value for correction of character substitution errors (.GOV).

Error Type	Retrieved	Relevant	Relevant Retrieved	Average Precision	R Precision	Precision at 10
Original Queries	**98477**	**2090**	**1382**	**0.1334**	**0.1596**	**0.1545**
Type 4 - 1 error (no correction)	97932	2090	876	0.0480	0.0516	0.0636
Edit Distance Correction	99003	2090	1111	0.0675	0.0751	0.0879
LCS Correction	98780	2090	1208	0.0968	0.1087	0.1202
Type 4 - 2 errors (no correction)	97932	2090	879	0.0483	0.0528	0.0636
Edit Distance Correction	100000	2090	1209	0.0813	0.0958	0.1091
LCS Correction	98071	2090	1019	0.0688	0.0775	0.0909

Table 8: The precision/recall value for correction of character reversal errors (.GOV).

5.3.3 Combinations of errors

We now look at what happens when a term contains a combination of two of the types of errors tested. We tested for each possible combination of the four error types mentioned in Section 2. For example, we tested the case when a term had an addition error (Type 1), and character substitution error (Type 3), like misspelling the word '*housing*' with '*huosinng*'. Again, we compare the results of the Edit Distance algorithm with the Longest Common Subsequence algorithm, found in Table 9.

We saw that the Edit Distance algorithm performs well particularly for Types 1 and 3, (Figures 2(a) and 2(c) respectively). The algorithm achieved over 60% accurate corrections for up to two errors for both error types. Hence it is reasonable to expect that a combination of both an addition and a substitution will perform just as well (Hypothesis 5). When the results from the experiments were analysed, this was indeed the case. The Edit Distance managed to correct 79.7% of the errors that had both a character addition error and a character substitution error supporting our hypothesis.

Furthermore, the Edit Distance algorithm does not dominate the results as much as when there was only one type of error. The LCS algorithm outperformed the Edit Distance in two cases: a combination of an addition and a character reversal (Types 1 and 4), and a combination of a character deletion and a character reversal (Types 2 and 4). It is clear that the latter combination is the hardest to correct since both algorithms failed to correct even half of the errors, suggesting that the type and combination of errors do affect a spelling error correction algorithm.

5.4 Discussion

We have analysed our results with respect to two collections. We now look at all the results as a whole and draw our conclusions to give a more general overview.

We have seen that, overall, the Edit Distance performed better than the Longest Common Subsequence algorithm. In both collections the Edit Distance algorithm performs well. Addition, deletion and substitution errors have been corrected to a much higher success rate than reversal errors as the algorithm takes this into account in its calculations (Hypothesis 3).

The size of the corpora and content does not seem to be an important factor (Hypothesis 6). Indeed, our experimental results from the WT2G and the .GOV collections are consistent (Figures 5.2.1 and 5.3.1, and Tables 1 - 8).

While we have said that the Edit Distance generally performs better than the LCS (Hypothesis 3), the latter algorithm is not without merits. The LCS is better for the case when there are three deletion errors. Three deletion errors appear to be the breakdown point in performance for the Edit Distance. While the LCS's degradation in accurate corrections is still apparent, the degradation is less severe as the number of errors in a query term increase (Figures 1(b) and 2(b)).

There is a decrease in performance for query terms with three errors regardless of the error type. However, the Edit Distance is still able to accurately correct around half of the mistakes made (apart from deletion errors). Hence, this may or may not be the cut-off point in which the algorithm's costs outweigh its benefits. Our experimental evidence suggests that the LCS's limit is at two errors. Its performance, while not as good as the Edit Distance, manages to cope with one error well enough but after that only character reversal and addition errors are corrected at a rate of around 50%.

Transposition errors appear to be the most difficult to correct. The two algorithms can only manage to accurately correct about half of errors for just one transposition error. In this case, the percentage of accurate corrections for the two algorithms is similar to each other. The Edit Distance algorithm uses two edits rather than one to correct this type of error and so is bound not to perform as well as on the other error types. This is still in line with the LCS's performance rate.

For the case of query terms consisting of a combination of error types, the performance of the two algorithms was more suggestive in terms of which algorithm performed better. For terms that have both an addition error and a character deletion, the Edit Distance error is best suited bettering the LCS algorithm by 12.5%. Yet for terms containing a character deletion as well as a character reversal the Edit Distance only managed to correct under a third of the errors accurately while the LCS algorithm corrected nearly a half of the same errors. Based on our experiments on one collection (.GOV), the results highlight again the differences between the two algorithms and that the type of spelling error is a factor in how well an algorithm performs (Hypothesis 2).

Clearly, the best way to correct spelling errors is to first be able to detect what type of spelling it is. From there an algorithm could be picked which would guarantee the best chance of correcting the error accurately. To illustrate, if an error was detected as several deletion errors, then the system should use LCS as it performed better than the Edit Distance (Figure 2(b)). However, if no classification of error is made and a general error correcting algorithm was needed, then the Edit Distance algorithm is recommended as three of the different types of errors (Types 1, 2 and 3), are taken into consideration when calculating its distance. Finally, according to the obtained results, the rate of successful corrections correlates with the average precision found using precision/recall measures.

6. CONCLUSION AND FUTURE WORK

This paper has looked at two classical algorithms that can be used in IR systems to detect and correct spelling errors. These algorithms are the Levenshtein Distance algorithm (more commonly known as the Edit Distance algorithm) and the Longest Common Subsequence algorithm. The Edit Distance measures the difference between two strings while the LCS measures how much two strings have in common with one another.

Experiments were performed to evaluate how well the algorithms performed for four types of errors. We also experimented with these errors on two different sized data collections of varying content. This enabled us to observe how

Error Combination	Edit Distance	Longest Common Subsequence
Character Addition & Deletion	63.0%	50.5%
Character Addition & Reversal	63.0%	68.0%
Character Addition & Substitution	79.7%	66.6%
Character Deletion & Reversal	30.9%	48.4%
Character Deletion & Substitution	57.1%	45.9%
Character Substitution & Reversal	54.0%	51.0%

Table 9: Percentage of accurate corrections of errors containing a combination of two error types.

the used string analysis algorithms perform on a range of data with a variety of spelling errors. In our evaluation, we identified and analysed the properties that an error correction algorithm might have difficulty with and those that are relatively simple to correct accurately. It was found that the Edit Distance performed generally well and bettered the LCS three out of four times for both the WT2G and .GOV collection. This seems to suggest that error correction is independent of the collection used.

It has also been pointed out that the algorithms can be used as an approach to tackle combinations of these errors and that their performance is related to how the algorithm performed for each error singularly. If the types of error could somehow be detected beforehand then we could choose one error correction algorithm over another based on the results of our experiments and its success in correcting that type of error. This is certainly an area of the work done in this paper that could be further developed in the future and reinforces the notion that spelling error correction impacts querying performance on varying degrees depending on the algorithm used.

We have only looked at two string matching algorithms but there are others such as those used in correcting optical character recognition (OCR) errors [3, 11]. It would be worthwhile to explore whether these techniques could be applied to spelling errors in queries and if they have better success at correcting errors than the two classical algorithms. We have also briefly mentioned that the size of the query term can affect the performance of the algorithm as seen when the number of deletion errors was significant. Further work can be done on the effects of average word length which is dependant on what language is in use.

The experiments have also shown that without an error correction functionality for query terms, average precision can fall by more than half of its original value. With just a small increase in time processing ($<$ 850ms per query) error detection and correction can prevent such a severe degradation in performance.

7. ACKNOWLEDGMENTS

This work is funded by a UK Engineering and Physical Sciences Research Council (EPSRC) project grant, number GR/R90543/01. The project funds the development of the Terrier IR framework (http://ir.dcs.gla.ac.uk/terrier).

8. REFERENCES

[1] R. C. Angell, G. W. Freund, and P. Willett. Automatic spelling correction using a trigram similarity measure. *Information Processing & Management*, 19(4): 255–261, 1983.

[2] G. Amati and C. J. van Rijsbergen. Probabilistic Models of Information Retrieval based on Measuring the Divergence from Randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389, October 2002.

[3] S. M. Beitzel, E. C. Jenson, and D. A. Grossman. A Survey of Retrieval Strategies for OCR Text Collections. *Invited to the 2003 Symposium on Document Image Understanding Technologies*, Greenbelt, Maryland, April 2003.

[4] N. Craswell and D. Hawking. Overview of the TREC-2002 Web Track. In *Proceedings of the Eleventh Text Retrieval Conference (TREC-2002)*, Gaithersburg, MD, 2002.

[5] N. Craswell and D. Hawking. Overview of the TREC-2003 Web Track. In *Proceedings of the Twelvth Text Retrieval Conference (TREC-2003)*, Gaithersburg, MD, 2003.

[6] I. Durham, D. A. Lamb, and J. B. Saxe. Spelling correction in user interfaces. *Comm. ACM*, 26(10):764–773, 1983.

[7] D. Hawking, E. Voorhees, N. Craswell and P. Bailey. Overview of the TREC-8 Web Track. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, 131–150, 1999.

[8] D. Harman. How effective is suffixing? In *Journal of the American Society for Information Science,* 42(1):7-15, 1991.

[9] B. He and I. Ounis. A Study of Parameter Tuning for Term Frequency Normalization. In *Proceedings of the Twelveth ACM CIKM International Conference on Information and Knowledge Management*, New Orleans, LA, 10–16, November 2003.

[10] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Comm. ACM*, 18(6):341–343, 1975.

[11] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, December 1992.

[12] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.

[13] G. de B. Robinson. On the representations of the symmetric group. *American Journal of Math*, 60:745–760, 1938.

[14] U. Quasthoff. Tools for automatic lexicon maintenance: Acquisition, error correction, and the generation of missing values. In *Proceedings of the first International Conference on Language Resources & Evaluation*, ELRA 1998, S. 853-856.

An experimental methodology to study collections size impact on retrieval effectiveness

Amélie IMAFOUO
Ecole Supérieure des Mines de Saint-Etienne
158 Cours Fauriel
42023 Saint-Etienne Cedex,France
imafouo@emse.fr

Michel BEIGBEDER
Ecole Supérieure des Mines de Saint-Etienne
158 Cours Fauriel
42023 Saint-Etienne, Cedex, France
beigbeder@emse.fr

ABSTRACT

Information grows continuously; for professional or personal reasons the need of easy access to it comes under the Information Retrieval (IR) field. Few works tackled the questions of Information Retrieval Systems (IRS) *effectiveness* and *efficiency* in the context of scalability in corpus size.

We propose a general experimental methodology which makes it possible to study the scalability influence on IR models. This methodology is based on the construction of a collection on which a given characteristic C is the same whatever be the portion of collection selected. This new collection called uniform (according to the characteristic C) can be split into sub-collections of growing size on which some given properties will be studied.

We apply our methodology to WT10G (TREC9 collection) and consider the characteristic C to be the distribution of relevant documents on a collection. We build a uniform WT10G, sample it into sub-collections of increasing size and use these sub-collections to study the impact of corpus volume increase on standards IRS evaluation measures (recall/precision, high precision).

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Scalability, Relevance, Retrieval Evaluation, Experiments

Keywords

Scalability, sampling collections, relevance, evaluation measures

1. FORMER IR WORKS ON SCALABILITY

The information retrieval process consists in providing, in response to a user request, documents which as well as possible meet his information need (relevant documents). This process is divided in several steps: collection construction, indexing, querying and evaluation. In the following sections, after some brief recalls on these phases, we will review works which were interested in scalability.

1.1 Collection construction

IR experiments are usually based on static collections which are composed of a set of documents, a set of information needs or topics, and a set of relevance judgments on those documents using the topics. The techniques currently used (pooling) described in the case of TREC [21] come up against many limits with the growth of information volume [23]. Some pooling improvements were proposed without resolving all the limits [23], [8], [18]. But [23] showed that using the pooling techniques to identify documents to be assessed introduces a skew that does not have a significant impact. He also showed that the relevance judgments obtained by the pooling technique provide a credible base for the evaluation of IRS that did not take part in TREC campaign.[19] showed the impact of judges disagreements on the reliability of relevance judgments can be neglected.

In the case of IRS using several distributed sources, [9] propose to scale in corpus size by reducing the duplication of identical documents coming from different sources. [10] are interested in replicating the Web hyperlinks structure features on a collection of reduced size. This relates to collection sampling: determine some properties of a large collection, build a collection of reduced size which has the same properties and make studies on the reduced collection (what should be easier than on the whole collection) and defer the results obtained on this reduced collection on the whole collection.

1.2 Indexing and querying

The average time to index collections increases in a very significant way according to their size [6]. Solutions suggested for this scalability limit are based on the physical information compression but their impact on the retrieval performance remains not significant [22]. A track that will allow the reduction of the representation space is the increase in the information granularity (using aggregate concepts to represent information rather than low level information units like the terms or the n-grams).

The average time for processing requests also scale badly. Techniques that aim to reduce this time need better knowledge of the user information need, which makes it possible to identify sub-collections (i.e. to reduce the size) which would be enough to carry out search and meet the user's need. The question is then how to segment the collection (collection segmentation based on questions answered by users, segmentation based on a users profile [16]). Other works rely on additional metadata related to the user information need and/or to the documents allowing a better classification of documents [6].

1.3 Evaluation

Two measures generally help to carry out IR evaluation: precision is the proportion of retrieved documents that are relevant; recall is the proportion of relevant documents that are retrieved. Precision and recall put forward two different aspects of an IRS performance. Hitherto, few work tackled the collection size impact on effectiveness. The relevance evaluation of retrieval results depend neither on collection size nor on corpus diversity, and this can generate biases when comparing different IRS performance. This is one of the main goal of the TeraByte task introduced at TREC in 2004. Heterogeneity is more important in large collections. Thus [5] showed that for large collections, the terms discrimination is amplified : the number of frequent terms does not increase in relation with the collection size and the proportion of discriminating terms decreases. The use of these collections for IRS evaluation will require new measures that put forward the precision on the first retrieved documents. Our work focuses on precision in first retrieved documents; they determine the users satisfaction particularly in the Web context.

1.4 Former research work on the scaling impact

Following the 5 assumptions put forth by the TREC-6 VLC task participants (they noticed a significant increase of high precision when the collection size increases) [13]. [11] studied the impact of the collection (size) on attribution of a score to a document and used various forms of relevant documents distribution and irrelevant documents distribution. He carried out experiments by building three types of samples of the collection:

- *Uniform samples*: One creates N primary disjoint samples of equal size and compose them to have several sub-collections of a certain size (e.g 3/7). The tests are done on each of these sub-collections and the result deferred for the sample of this size (e.g. 3/7) corresponds to an average of the results on all these sub-collections and takes into account all the data.
- *Replicated samples*: One takes the primary samples of size 1/10 of the whole collection and replicate them a desired number of times.
  Example: $(0), (0, 0), \cdots, (0, 0, 0, 0, 0, 0, 0, 0, 0, 0), (1), \cdots, (1, 1, 1, 1, 1, 1, 1, 1, 1, 1), (9), \cdots, (9, 9, 9, 9, 9, 9, 9, 9, 9, 9)$.
- *Biased samples*: the TREC6 data is subdivided in 5 disjoined sub-collections according to their origin. Each of the 5 sub-collections is considered as a sample, but these sub-collections have neither the same number of documents nor the same size and their documents do not have the same probabilities of relevance.

Both our methodology and Hawking's work aim to sample a given collection to allow experiments on the scalability influence in a reproducible way.

2. OUR METHODOLOGY

2.1 Assumptions and general methodology

We want to study the influence of collection growth on IR models. To achieve this goal, we have to carry out experiments on collections of growing size and analyze the behavior of different IR models when the collection size increases. The question is how to build collections of growing size that will be used to carry out experiments and obtain reliable results on scalability influence? Let C be a characteristic of a collection and $\{P_i\}$ be some properties. The methodology goal is to obtain a collection on which the characteristic C is the same whatever be the portion selected in the collection. If we have such a collection, we can split it into portions of different size, study properties $\{P_i\}$ on different portions and analyze the influence of portion size on these properties. The way the initial collection is split must not be a constraint. This mean we want to allow any splitting of the collection, the only constraint is that the characteristic C must be the same whatever be the portion of the collection selected. Our methodology is divided in 4 steps:

1. We suppose in this step that we have an collection. We study the characteristic C on this collection. If the constraints we want on C are satisfied, we move to step 3;else we continue to step 2. In the same way, if we do not have an initial collection, move to step 2.
2. (Re)build the collection to have a new one on which the characteristic C remain unchanged whatever be the portion of the collection selected
3. Split (sample) the new collection into portions of growing size
4. Study the properties $\{P_i\}$ on different portions and analyze the influence of portion size on these properties.

We apply this general methodology on retrieval evaluation. Given the fact that IR evaluation is based on relevant documents retrieving, we assume that for a study on scalability influence, the way relevant documents are distributed within collections of growing size is important. For two collections of different size, the relevant documents distribution must not change. We have an initial large collection and want to split it into sub-collections of growing size. We want to allow any splitting of the collection, the only constraint is that the distribution of relevant documents must be the same whatever be the portion of the collection selected. So, we apply the general methodology explained above with characteristic C chosen to be the distribution of relevant documents on the collection and the properties $\{P_i\}$ are evaluation measures.

2.2 Step 1: Study the characteristic C

The first step of our methodology is to study the characteristic C (in this case, it is the distribution of relevant documents among the collection) to notice if this distribution has a particular shape. If the distribution is unspecified, we move to next step and our goal is to have a uniform collection.

2.3 step 2: build a uniform collection

This step aims to obtain a collection that can be split in different ways, by respecting our assumptions. In this new collection (called uniform), whatever be the portion selected, the number of relevant documents for a given topic is proportional to the portion size and the total number of relevant documents (for all topics) is proportional to the portion size. We can obtain such a collection if relevant documents for a topic are distributed uniformly on the collection. This means that relevant documents for a given topic t must be separated by the same number of documents. To carry out this distribution, we first of all compute $E(t)$which is the wished distance between two documents relevant to the topic t. Let $R(t)$ be the set of the relevant documents for topic t, T the set of all topics and D the set of all the documents of the collection. We have

$$E(t) = \frac{\mid D \mid - \sum_{k \in T} \mid R(k) \mid}{\mid R(t) \mid}$$

Thus within the new collection, for each topic t we want its relevant documents to be separated by $E(t)$ irrelevant documents (i.e. documents which are not considered to be relevant for any topic) and possibly by documents considered to be relevant for some topics different from t. For documents considered to be relevant for several topics, they are inserted only once in the new collection, with the position defined by the first of the concerned topics that is processed. This can possibly change the real distance between two relevant documents of a topic t. So for a topic t, the real distance between two relevant documents is $E_r(t) \leq E(t) + \sum_{k \in T, k \neq t} \mid R(k) \mid$

This introduces a skew on the uniform collection. We assume that our collection is large and that

$$\sum_{k \in T} \mid P(k) \mid <<< \mid D \mid$$

: the total number of relevant documents is much smaller than the collection size. The skew on the real distance between two relevant documents for a topic t is then not significant; it does not affect the uniformity we wanted on the collection.

Let us look at this on an example. Let us suppose that we have a collection made up of 30 documents ordered as follows:

$D = \{d_1, \cdots, d_{30}\}$ and $T = \{t_1, t_2\}$. Let us suppose $R(t_1) = \{d_1, d_7, d_{18}\}$ and $R(t_2) = \{d_7, d_{21}\}$. Document d_7 is relevant for both t_1 and t_2.

We compute $E(t_1) = \frac{|D| - \sum_{k \in T} |R(k)|}{|R(t_1)|} = \frac{(30-4)}{3} \approx 8$ and $E(t_2) = \frac{(30-4)}{2} \approx 13$

In the uniform collection, the documents are ordered as follows:

$$\{d_2, d_3, d_4, d_5, d_6, d_8, d_9, d_{10}, \underbrace{d_1}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, \overbrace{\underbrace{d_7}},$$
$$d_{16}, d_{17}, d_{19}, d_{20}, d_{22}, d_{23}, d_{24}, d_{25}, \underbrace{d_{18}}, d_{26}, d_{27}, d_{28},$$
$$d_{29}, d_{30}, \overbrace{d_{21}}\}$$

We envisaged that the relevant documents for topic t_1 will be separated by $E(t_1)$ irrelevant documents. In the worst case for our example, two relevant documents for topic t_1 will be separated by $E(t_1) + \mid R(t_2) \mid = E(t_1) + 2$.

Notice that the document d_7 is inserted once. Because this document is relevant for more than one topic, the real distance between 2 relevant documents of topic t_1 is different from our envisaged distance. This skew is not significant if we assume that the collection is large.

2.4 Step 3 : Sample the collection

With the methodology explained above, we obtain a uniform and reusable collection for various types of experiments. This collection can be split out in portions of different sizes using various ways since the number of relevant documents on a portion is proportional to the portion's size. The choice of the way the collection will be split can vary according to experiments needs. One could split out the whole collection into N sub-collections of equal size and then put those N sub-collections together in various ways to obtain collections of various sizes since it is known that the number of relevant documents on a portion is proportional to the portion's size.

For the example we showed previously, to obtain sub-collections of growing size, one can split the collection in $N = 3$ portions of size $\frac{|D|}{N} = 10$

- A first portion is
$$D_1 = \{d_2, d_3, d_4, d_5, d_6, d_8, d_9, d_{10}, d_1, d_{11}\}$$
- A second portion is
$$D_2 = \{d_{12}, d_{13}, d_{14}, d_{15}, d_7, d_{16}, d_{17}, d_{19}, d_{20}, d_{22}\}$$
- A third portion is
$$D_3 = \{d_{23}, d_{24}, d_{25}, d_{18}, d_{26}, d_{27}, d_{28}, d_{29}, d_{30}, d_{21}\}$$

Then one can build sets of sub-collections

$$\{S_1 = D_1, S_2 = D_1 \bigcup D_2, S_3 = D_1 \bigcup D_2 \bigcup D_3\}$$

or

$$\{S_1 = D_2, S_2 = D_1 \bigcup D_3, S_3 = D_1 \bigcup D_2 \bigcup D_3\}$$

. In the two cases, we obtain three sub-collections of growing size with the same distribution of relevant documents.

2.5 Step 4: Study the influence of collection size on IR models properties

In this step, we have sub-collections of growing size with the same distribution of relevant documents and want to study the scalability impact on a given property of an IR model (for example evaluation measure like recall, precision, high precision). We study the property for each-sub collection and analyze the property's behavior as the collection size grows.

3. USING OUR METHODOLOGY WITH WT10G

3.1 Relevant document distribution on WT10G

3.1.1 Data.

We worked on the TREC test collection WT10G [4]. Information nee ds for our tests are topics 451-500. This test collection contains 1,692,096 documents including 2,371 documents considered to be relevant for topics 451-500 and distributed among those topics as the Fig. 1 shows.

Figure 1: Number of relevant documents per topic in WT10G

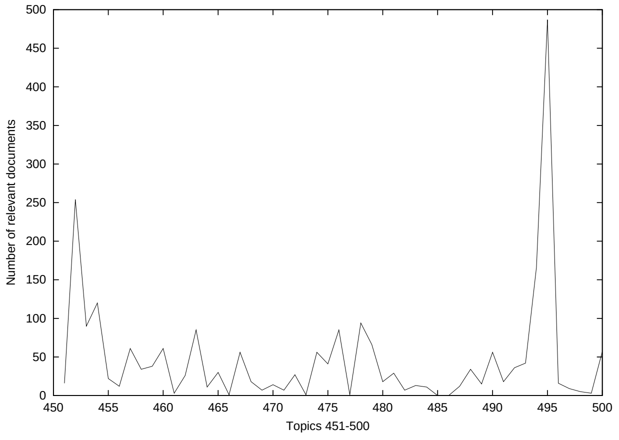


Table 1 gives some statistics on relevant documents in the WT10G collection.

The characteristic C is the relevant document distribution. $T = \{451, \cdots, 500\}$, D is WT10G and $\mid D \mid = 1,692,096$, $\sum_{t \in T} \mid R(t) \mid = 2,371$

3.1.2 Query building.

From the TREC9 topics, we manually built a request set for querying the collection. A request is a list of key-words based on the topic title, the description and the narrative of the relevant documents awaited for this topic as provided by TREC. For the topic 460 for example:

<top>

<num> Number: 460

<title> Who was Moses?

<desc> Description: Find documents that discuss the biblical figure of Moses.

<narr> Narrative: A relevant document includes any information concerning Moses and his deeds regarding the Israelites.

</top>

Table 2: Some statistics on our queries: number of words per topic

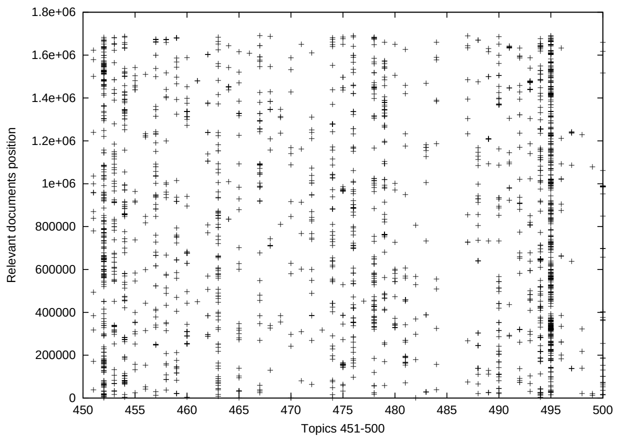
Min	Max	Average
2	8	4.76

We built the request "*Moses Israelites bible biblical*" (title (Moses), description (biblical, bible) and narrative (Israelites)). Table 2 shows the statistics on the number of key-words for the queries we built.

3.1.3 Relevant document distribution on WT10G.

We studied the per topic relevant document distribution within the WT10G collection. This distribution is plotted on Fig. 2. The relevant documents distribution over

Figure 2: Relevant documents per topic distribution (initial collection)



WT10G is unspecified and it varies according to topics. Relevant documents For a given topic are not uniformly distributed. The number of relevant documents is not a linear function of the collection size. Given that we will subdivide our test collection in sub-collections of growing size, it is important to take into account the documents distribution on each sub-collection so that the properties we want to study (precision and recall for example) remain meaningful as the sub-collections size grows. We thus redistributed the relevant documents within the collection.

3.2 The uniform WT10G

We redistributed the relevant documents within WT10G collection to obtain a uniform collection. In this new collection, the number of relevant documents for a given topic is a linear function of the collection size and the total number of relevant documents (for all the topics) is also a linear function of the collection size.

For example, the topic number $t = 495$ has 487 documents considered to be relevant, then $E(t)$ is equal to $\frac{(1,692,096-2,371)}{487} = 3,469$

For the WT10G collection, $E_r(t) \leq E(t)+2,371$. This introduces a skew on the uniformity of our distribution but given the total number of documents in the collection (1,692,096

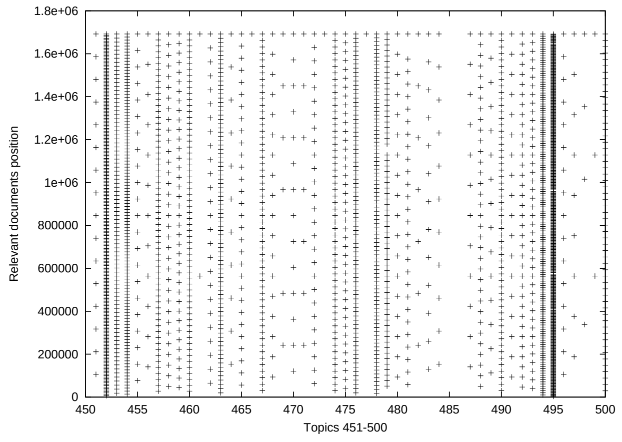
Table 1: Collection WT10G: statistics on the number of relevant documents per topic

Min	Max	Median	Average(Avr)	Topics t/ $\mid R(t) \mid < Avr$	Topics t/ $\mid R(t) \mid > Avr$
0(topics 485 and 486)	487	25	47.42	35	15

documents), this skew is not significant. This skew has no influence on the uniformity of the distribution of relevant documents which we want to draw up on the test collection. When the whole collection is split into sub-collections, the size of these sub-collections remains big enough to allow us to continue neglecting the skew. Moreover, this skew is a compromise to obtain a collection within which the relevant documents are at the same time distributed uniformly for each topic and are distributed uniformly (with little near) if the whole set of the topics is considered.

Figure 3 shows the distribution of relevant documents on the uniform collection. In the uniform collection, the relevant documents for a given topic are distributed in a uniform way on the collection and the relevant documents (all topics fused) are distributed in a uniform way on the collection. Thus, whatever the collection portion selected, the number of relevant documents (either all topics fused or topic by topic) is proportional to the portion size.

Figure 3: Relevant documents per topic distribution (uniform collection)



3.3 Sampling uniform WT10G

Within the framework of our experiments, we built portions of increasing size with a variation of 200,000 documents by taking the documents in the order of appearance in the collection. We obtained 8 portions and we worked on 7 of them (The portions we worked on are size 200,000 documents to size 1,400,000 documents). However, the choice of the variation can be modified according to experiments needs. One could split out the whole collection into N sub-collections of equal size and then mix those N sub-collections in various ways to obtain collections of various sizes since it is known that the number of relevant documents on a portion is proportional to its size.

4. USING UNIFORM COLLECTION TO STUDY SCALABILITY IMPACT

4.1 IR models used

We used five IR models for our experiments:

- The LUCY tool [1] we used implements the Okapi model, which is an extension of the probabilistic model.
- We used the tool MG [2] which is based on the vector model.

The 3 other models are based on the proximity between the request term; the implementations we used are described with more details in [15].

- The Cover Density Ranking method [7] rank the relevant documents according to "the density of cover" of the request keywords in the documents. We call this method *Clarke model.*
- For the method proposed in [12], a request is a set of tuples (U, A) composed of a set of terms U and an importance coefficient A. Each element of $I(U, A)$ (set of documents intervals which contains all terms of U) takes part in the relevance score of the document. We call this method *Hawking model.*
- The method in [17] allots to each document, for a given request, a score computed by adding the Okapi model score and a proximity score. We call this method *Rasolofo model.*

4.2 Precision /recall graphs

4.2.1 Assumptions.

When the relevant documents are not distributed in a uniform way on the collection, it is difficult to evaluate the impact of increasing the collection size on precision and recall. If the relevant documents are distributed in a uniform way on the collection, then the precision for a given level of recall is the same never mind that one works on the whole collection or on one of its samples.

Recall/precision curves for the 5 IR models we used are plotted on Fig. 4 (Clarke model), Fig. 5 (Hawking model), Fig. 6(Okapi model as implemented in the Lucy tool), Fig. 8 (vector model as implemented in MG tool) and Fig. 7 (Rasolofo model).

Hawking model and Clarke model : for the first recall level, recall/precision curves for big collections are very close. For high recall level, the curves remain close. This means that those models have a certain *stability* regarding the precision/recall measure when the collection size increase. For the two models, the retrieval status value (RSV) for a document does not depend on the collection; it depends only on the document content and on the query.

For the OKAPI model (Fig. 6), the precision/recall ratio is much better for big collections than for small ones for

Table 3: Uniform WT10G: statistics on documents size for every sub-collection

Sub-collection size	Min size	Max size	Empty doc	Average size(Avr)	doc size <=Avr	doc size>Avr
200,000	3	2,326,790	3	3,875.22	155,090	44,910
400,000	3	2,326,790	14	4,103.59	314,199	85,801
600,000	3	2,326,790	21	3,902.08	468,430	131,570
800,000	3	2,344,747	29	3,876.66	628,158	171,842
1,000,000	3	2,344,747	33	3,857.18	785,360	214,640
1,200,000	3	2,344,747	34	3,766.91	943,802	256,198
1,400,000	3	2,344,747	36	3,773.19	1,101,592	298,408
1,600,000	3	2,344,747	38	3,790.24	1,260,289	339,711

Figure 4: Recall/Precision curves on the uniform WT10G for our 7 collection sizes - Clarke IR model

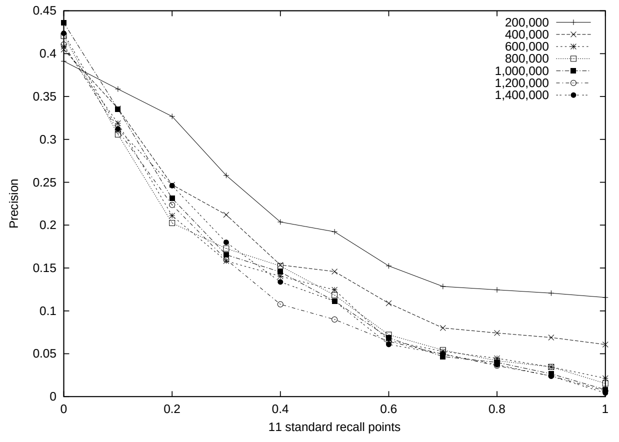
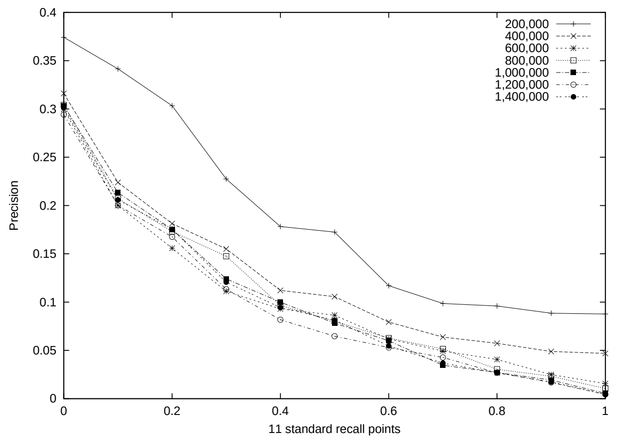


Figure 5: Recall/Precision curves on the uniform WT10G for our 7 collection sizes - Hawking IR model



the first recall level. When recall level increases (up 10%), this changes and the curves become closer. For the OKAPI model, the RSV for a document depends on others documents in the collection. So the RSV is not free from collection size. The *stability* is less than for the two preceding models.

For the Rasolofo model (Fig. 7), big collections are below the small collections for very small recall level. For recall level from 10% to 30%, the curves are ordered from the

Figure 6: Recall/Precision curves on the uniform WT10G for our 7 collection sizes - Okapi IR model from Lucy Tool

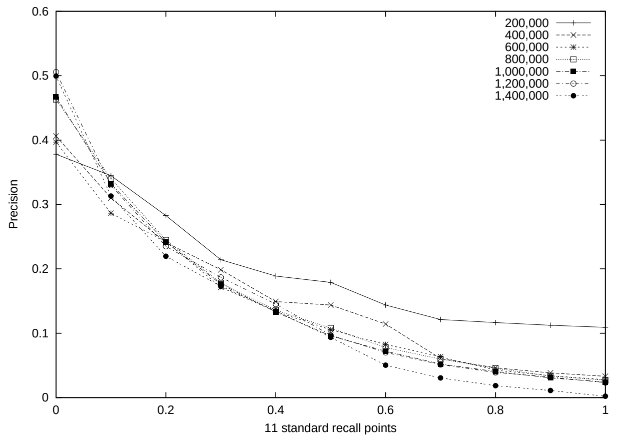
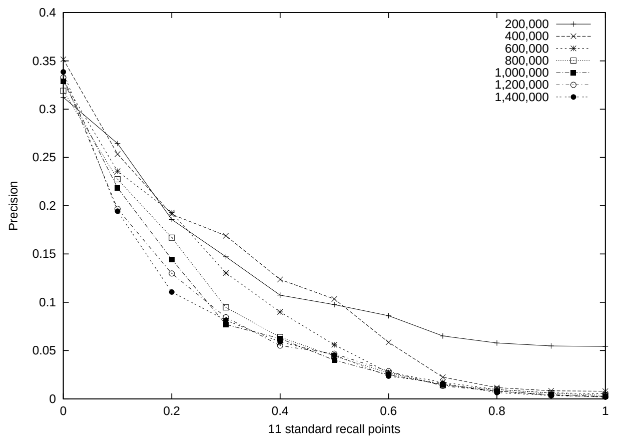
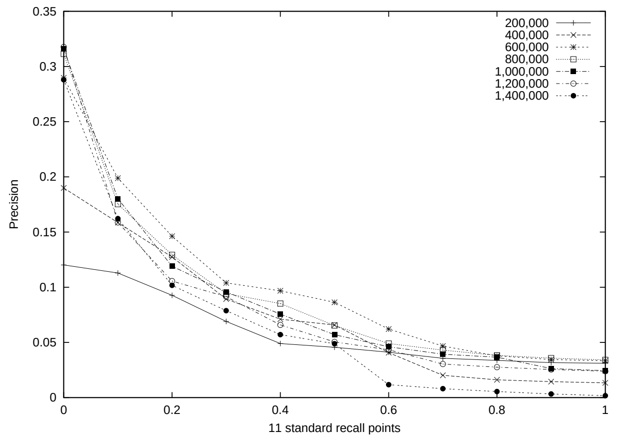


Figure 7: Recall/Precision curves on the uniform WT10G for our 7 collection sizes - Rasolofo IR model



smallest collection to the biggest collection and curves are not close. For big collections, curves start to be close from 30% recall. This model combines the proximity between query's terms in a document and the Okapi model to give an RSV to a document. So, it combines a method for which a document's RSV depend on the collection and a method for which a document's RSV depend only on the document and the query. Its *stability* starts at high recall level.

Figure 8: Recall/Precision curves on the uniform WT10G for our 7 collection sizes - Vectorial IR model from MG tool



For the vector model (Fig. 8), big collections curves are close for small recall level (until 10%). The order between curves according to the collection size changes from a recall level to another. This implies a certain *instability* regarding the precision/recall measure when the collection size increase. The RSV given to a document by this model depends on query's term frequency in the document (*tf*) and on inverse document frequency (*idf*). The *idf* depends on the collection.

The role the collection has in the attribution of document scores affect the way the model scale (regarding the precision/recall measure). IR models for which a document score depends only on the query terms and on the document seam to scale better than those for which documents score depends on collection.

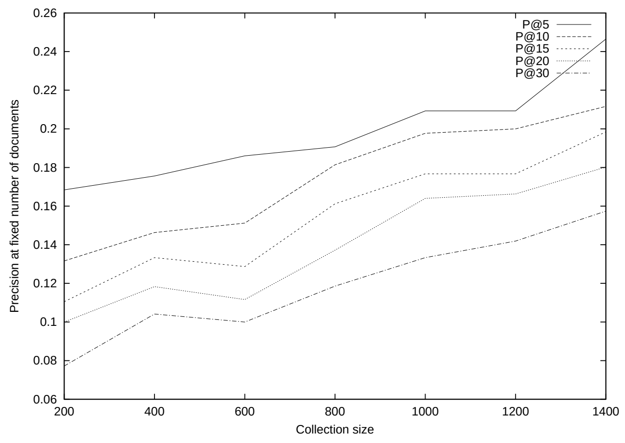
4.3 High Precision

Precision after a fixed number of retrieved documents closely correlates with user satisfaction in tasks such as Web searching and is easy to interpret. The experimental results of [11] are based on a unique IR model, the Okapi BM25 model implemented through the Padre system. We generalize this results to 4 $(5, 10, 15, 30)$other constant cutoff level and to 5 IR models. Figure 9 (Clarke model)show the evolution of these precisions when the collection size grows. For the four others model we used, the curves shape are similar to those of Clarke model Fig.9. These results show that the precision on the first retrieved documents increases with the sample size.[11] conclude that this increase is due to two main factors: the number of relevant documents in a portion of the collection (This number increases with the portion size) and to the ability of the couple (E, Q) (where E is the search engine and Q the query)to rank relevant documents ahead of irrelevant ones.

4.4 First relevant document retrieved rank

Table 4 gives some statistics on the rank of the first relevant document retrieved when collection size increases. Table 4 shows that the rank of the first relevant document retrieved reduces when collection size increases (this means that the first document retrieved goes upper in the top list of retrieved documents when collection size increases). We also

Figure 9: Precision on first documents retrieved on the uniform WT10G for our 7 collection sizes (Collection size/1000) - Clarke IR model



studied the percentage of first relevant document retrieved that are between the 10 first documents retrieved and notices that this percentage increases with collection size.

5. DISCUSSION AND CONCLUSIONS

The methodology proposed in this work build a collection on which a given characteristic C is the same whatever be the portion of collection selected. This new collection is called uniform (according to the characteristic C) and can be split in sub-collection of growing size on which some given properties will be studied. This methodology is general and reproducible.

We apply our methodology and consider the characteristic C to be the distribution of relevant documents on a collection. We uniformize WT10G, sample it into sub-collections of increasing size and study the impact of corpus volume increase on some standards IRS evaluation measures. We noticed that IR models for which a document score depends only on query terms and on the document seam to scale better than those for which documents score depends on collection. We are interested to know more about how the collection role on the attribution of document RSV influences the IR models scalability. Our results for precision at a constant cutoff level$(5, 10, 15, 30)$ generalize those of [11] and we extend them to various IR models.

We are now using the uniform WT10G collection to study the scalability impact on others IR models properties, like the position of the first relevant document retrieved, other IR measures (R-precision, the Mean Average Precision) or new measure like bpref [20] that seam to be robust to incomplete relevance judgments.

While information grows continuously, it becomes important to know what impact the scalability has on retrieval models to improve these models and ensure they scale correctly. The methodology we propose is a general one. We believe it can be used to build others "uniform" collections according to some others characteristics (proportion of query terms when evaluating relevance feedback algorithms for example) and to use these new collections to study the scalability impact

Table 4: Rank of the first relevant document retrieved

Collection	Average rank and median rank of the first relevant document retrieved														
size	percentage of the first relevant document between the 10 first														
	Hawking			Rasolofo			Lucy			Vectoriel			Clarke		
200.000	33,76	3,5	66%	88,35	18	48%	39,5	6	63%	146,13	94	26%	54,5	3	71%
400.000	38,46	4,5	63%	53,22	3	61%	44,31	2	76%	83,60	15,5	44%	33,31	3	41%
600.000	39,43	5,5	63%	33,58	3	77%	24,5	3	78%	74,10	8	52%	32,62	3	90%
800.000	20,26	5,5	63%	24,35	4	77%	14,97	2	84%	64,73	6,5	57%	28,40	3	87%
1.000.000	26,36	7	63%	20,19	3	77%	17,97	2,5	86%	77,84	6,5	55%	5,29	3	90%
1.200.000	29,7	5	63%	22,03	3	77%	20,07	2	86%	90	6,5	57%	5,70	2	87%
1.400.000	22,33	5,5	63%	17,64	3	83%	14,97	2,5	81%	63,64	7	56%	12,15	2,5	81%

on IR models.

6. REFERENCES

[1] http://www.cs.mu.oz.au/mg/.

[2] http://www.seg.rmit.edu.au/lucy/.

[3] http://trec.nist.gov/.

[4] P. Bailey, N. Craswell, and D. Hawking. Engineering a multipurpose test collection for web retrieval experiments draft. In *Proceedings of the 24th annual international ACM SIGIR conference*, 2001.

[5] M. Beigbeder and A. Mercier. Etude des distributions de tf et de idf sur une collection de 5 millions de pages html. In *Atelier de recherche d'informations sur le passage à l'échelle Congrès INFORSID 2003*, Nancy, France, 2003.

[6] J. P. Chevallet, J. Martinez, M. Boughanem, L. Lechani-Tamine, and S. Calabretto. Rapport final de l'AS-91 du RTP-9 'passage à l'échelle dans la taille des corpus', 2004.

[7] C. L. A. Clarke, G. Cormack, and E. Tudhope. Relevance ranking for one to three term queries. *Information Processing and Management*, 26(2):291–311, 2000.

[8] G. Cormack, C. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–289, 1998.

[9] O. Frieder and D. Grossman. On scalable information retrieval systems. In *Invited Paper, 2nd IEEE International Symposium on Network Computing and Applications.*, Massachusett, Cambridge, 2003.

[10] C. Gurrin and A. Smeaton. Replicating web structure in small-scale test collections. *Information retrieval*, 7:239–263, 2004.

[11] D. Hawking and S. Robertson. On collection size and retrieval effectiveness. *Information retrieval*, 6(1):99–105, 2003.

[12] D. Hawking and P. Thistlewaite. Proximity operators - so near and yet so far. In *Proceedings of the Fourth Text Retrieval Conference TREC-4*, pages 131–143, 1995.

[13] D. Hawking and P. Thistlewaite. Scaling up the trec collection. *Information retrieval*, 1(1):115–137, 1999.

[14] P. Lyman, H. R. Varian, K. Swearingen, P. Charles, N. Good, L. L. Jordan, and J. Pal. How much informations 2003. http://www.sims.berkeley.edu/research/projects/how-much-info-2003/, October 2003.

[15] A. Mercier. Etude comparative de trois approches utilisant la proximité entre les termes de la requête pour le calcul des scores des documents. In *INFORSID 2004 - 22ème congrès informatique des organisations et des systèmes d'information et de décision*, 2004.

[16] G. B. Newby. The science of large scale information retrieval. Internet archives, 2000.

[17] Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *Proceedings of European Conference on Information Retrieval Research*, pages 207–218, 2003.

[18] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proceedings of the 24th annual international ACM Conference on research and Development in Information Retrieval*, pages 66–73, 2001.

[19] E. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information and Processing Management*, (36):697–716, 2000.

[20] E. Voorhees and C. Buckley. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 25–32, 2004.

[21] E. Voorhees and D. Harman. Overview of the sixth text retrieval conference (trec-6). In *NIST Special Plublication 500-420-The Sixth text retrieval Conference*, 1997.

[22] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes - Compressing and indexing documents and images*, pages 451–468. Morgan Kaufman Publishers, second edition, 1999.

[23] J. Zobel. How reliable are the results of large scale information retrieval experiments. In *Proceedings of the 21th ACM SIGIR Conference on research and development in information retrieval*, pages 307–314, 1998.

Using a Reference Corpus as a User Model for Focused Information Retrieval

Gilad Mishne Maarten de Rijke Valentin Jijkoun

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
gilad,mdr,jijkoun@science.uva.nl

ABSTRACT

We propose a method for ranking short information nuggets extracted from a text corpus, using another, reliable reference corpus as a user model. We argue that the availability and usage of such additional corpora is common in a number of IR tasks, and apply the method to answering a form of definition questions. The proposed ranking method makes a substantial improvement in the performance of our system.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*linguistic processing*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering, search process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*question-answering (fact retrieval) systems*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Information Retrieval

Keywords

Question Answering, Information Retrieval

1. INTRODUCTION

The area of Question Answering (QA) is at the focus of a lot of research interest lately, both in the Information Retrieval (IR) community and among Computational Linguists. It is seen as one of the few applications to successfully combine techniques from Natural Language Processing and IR. The QA track at the annual Text REtrieval Conferences (TREC, [20]) has become an important factor in shaping and giving direction to QA research. Introduced in 1999, this track attracts a significant number of participants each year, and provides a focal point for much modern QA research.

When the QA track at TREC was introduced, it focused on so-called "factoid" questions (typically having a short named entity as an answer) such as *How many people live in Tokyo?* or *When is the Tulip Festival in Michigan?*. As the track evolved, it was argued that this type of questions does not accurately model the needs of real users of QA technology. In addition to named entities as answers, users often search for definitions of concepts, or for summaries of important information about them. As a result, in 2003 TREC introduced *definition questions*—questions for which the answer is not a single named entity, but a list of *information nuggets* [19].

In the TREC 2004 QA track this was taken a step further. The questions were now clustered in small groups, organized around the same topic. For example, the topic *Concorde* included questions such as *How many seats are in the cabin of a Concorde?* and *What airlines have Concordes in their fleets?*. Finally, for every topic, the track guidelines required participants to supply "additional important information found in the corpus about the target, that was not explicitly asked." This last requirement has been dubbed *"other"* questions [20]. In our view, the task presented at the TREC 2004 QA track, and the introduction of the "other" questions makes a big step towards more realistic user scenarios. According to our own analysis of web query logs, users tend to ask much more "knowledge gathering" questions than factoid questions about specific facts.[1]

This new type of "other" questions puts more emphasis on the *user* aspect in the QA process—an issue that has mostly been neglected in the QA community. The TREC criteria for what is a *good answer* to a given question has so far been rather vague, but QA systems dealt with this vagueness fairly effectively for factoid questions. With the "other" questions, where systems are required to return only *important* information, there is an implicitly assumed user model that can discriminate between important and unimportant facts about a topic. For example, for the topic *Clinton*, his birthday might be considered important, while the day of the week when he left Mexico probably is not. In order to give reasonable responses to "other" questions, a QA system needs to model such preferences.

[1]The analysis of this data is preliminary and will be published elsewhere as soon as it is completed.

DIR 2005, 5th Dutch-Belgian Information Retrieval Workshop.

We present an approach for answering "other" questions using an explicit user model. We describe a method for gathering important facts about an entity from a collection of documents and for ranking the facts with respect to their importance for the user. We show that our ranking improves over plain retrieval of facts from the corpus. The core idea of our method is to estimate the importance of facts found in the target collection by using external "reference" corpora, high-quality sources of information that model a user's ability to distinguish between important and unimportant facts. The proposed method is our first step towards user-oriented QA, and further refinements of the underlying techniques are needed. We identify additional areas where this method is or may be helpful, and discuss its strengths, weaknesses and directions for further research.

The rest of the paper is organized as follows. In Section 2 we survey related work regarding answering definition questions, and about using high-quality external sources. Next, in Section 3 we describe the details of the re-ranking method. Our experiments and results follow in Section 4, and we wrap up with conclusions in Section 5.

2. RELATED WORK

Our approach to ranking snippets extracted from a given document collection is based on "double" ranking: a regular IR ordering by decreasing relevance to the query, followed by re-ranking based on a comparison of those snippets with information mined from a "reference" corpus. Such "double" ranking and re-ranking schemes have been used widely in IR. E.g., in their Maximal Marginal Relevance criterion Carbonell and Goldstein [4] strive to reduce redundancy while maintaining query relevance in selecting appropriate passages for text summarization. Similar ideas have been widely used in work carried out at TREC's novelty track [17], where two things have to be done: relevant sentences should be extracted from a list of relevant documents, and from the resulting list of sentences only the ones contributing novel information should be retained. Kamps [10] and Van Hage et al. [18] use two-step ranking procedures in which the list of documents output by a retrieval engine is re-ranked based on hierarchical relations of relevant metadata concepts in a thesaurus or ontology, respectively.

We are not the first to be using electronic encyclopedias in open domain QA. Kupiec's Murax [11] was probably the first modern open domain QA system, combining IR techniques with shallow natural language processing to answer factoid questions against an electronic encyclopedia. More generally, many teams participating in the TREC QA track use resources other than the corpus against which the questions need to be answered. E.g., at TREC 2002, the University of Waterloo's QA system consulted the web as well as locally developed corpora and knowledge bases with answers to questions of frequently occurring types [6]; IBM's usage of the CYC knowledge base provides another example [5].

Our re-ranking mechanism is related to BBN's use of so-called *question profiles* for re-ranking candidate answers to definitional questions at TREC 2003 [21]. Question profiles are vectors of word frequencies generated from existing definitions of the question target in electronic dictionaries, as well as from biography collections and search engine results.

The work in this paper is also related to the so-called *answer projection* task that data-intensive QA system often face: given an answer to a question (possibly obtained from an outside source), find supporting documents in a given collection for it [15]. Phrased this way, the task resembles a known-item search task. Accordingly, answer projection has been addressed using the kind of high precision retrieval models that have typically been employed for known item search, such as specific Okapi settings [3], passage retrieval, and combinations of heuristics [14].

The use of external, high-quality data sources in IR is not limited to the QA setting. Some examples of IR tasks in which a reference corpus of some kind has been or is being used are *filtering*, *spoken document retrieval (SDR)*, and *web retrieval*. Filtering, which was evaluated at TREC for a large number of years, relies on the availability of standing information needs, possibly with a reference corpus of documents known to be relevant to the information need. In SDR, the corpus is usually noisy (literally), containing incomplete documents; parallel "clean" corpora are often used in this case to expand the noisy documents or the query and improve retrieval significantly [9].

In the popular area of *web retrieval*, some search engines, such as Yahoo! and Google (to some extent), maintain a human-generated catalog of internet sites, in addition to the index of crawled data from the internet. We are not aware of published research on using this catalog to improve retrieval, but this seems a viable option. Even if we take as an example "reference" corpus the relatively small English edition of Wikipedia (`http://en.wikipedia.org`), an open-content encyclopedia, rather than the large human-generated indexes, there are many web queries that can benefit from using it. In query logs released by AltaVista [2], 13% of about 7M queries have an entry in Wikipedia (this was checked without removing stopwords and with no morphological normalization, which will most likely increase the percentage further).

3. RANKING WITH A USER MODEL

In this section we provide the details of our method for extracting, ranking, and re-ranking information nuggets from a corpus. In a nutshell, after identifying a suitable "reference" corpus for our domain (our user model), we first use IR and NLP methods to identify information nuggets—short excerpts of text—related to the topic, both from the given document collection and from the "reference" corpus. Then, we use sentence-similarity metrics to rank the nuggets from the collection: the facts similar to those found in the "reference corpus" are considered more important and ranked higher.

3.1 Target Corpus and Reference Corpus

In the TREC QA task, answers to questions (including "other" questions) must be found in a given text corpus. In recent years, this corpus has been a part of the AQUAINT corpus, containing more than 1 million newswire documents, and a total of 3.1GB of text. In our experiments this corpus is used as the *target* corpus, where important information nuggets have to be located. The corpus is unstructured: we do not know beforehand which articles or passages contain "important" information about a topic.

The "reference" corpus to be used should be a relatively small, high-quality collection of documents, which is catalogued in a way that facilitates selecting documents which contain important information for a given topic. Typical corpora that can be used for such reference purposes are encyclopedias (e.g., biography pages from `http://biography.com`) and various knowledge bases (e.g., the Internet Movie Database `http://www.imdb.com`). Since TREC QA is an open domain task, we used the English edition of Wikipedia (`http://en.wikipedia.org`), an open domain encyclopedia. The version we used contained 768,000 entries (including placeholders and disambiguation entries), for a total of 900 MB of text.

3.2 Mining Facts from the Target Corpus

When answering an "other" question for a given topic, we use IR to locate documents containing information about the topic, and then split the sentences from the retrieved documents into more easily "digestable" shorter nuggets.

Retrieval

First, from the target collection we retrieve the top 20 documents containing the topic as a phrase, using a traditional vector space model for the retrieval. Our collection is composed of news articles with headlines. Since an occurrence of a topic in a headline can be very indicative of the document's importance for the topic, we indexed the headlines and the article bodies separately, and calculate the retrieval score as a combination of the different representations; this is a common technique for semi-structured IR [16].

Extraction

Since a response to an "other" question is a list of short nuggets, we have to split the retrieved documents into separate facts. This raises several problems. First, we observed a notorious use of referential NPs: even in highly focused documents the topic is introduced initially, and then referred to with pronouns or definite NPs (e.g., "*PRESIDENT CLINTON arrived today at the . . . HE will leave to Mexico on Monday*"). We therefore resolve pronouns in the documents using a simple anaphora resolution module described in [1]. Then, we extract all sentences which contain the topic (either originally or after the resolution); this is a natural way to restrict our attention to document sections which potentially include facts about the entity.

Still, the sentences are often too long to be presented as nuggets. Moreover, as the next step of our method involves comparison of nuggets, we need to keep them atomic, i.e., as short as possible. We observed that most facts in the extracted sentences could be described with simple predicates (e.g. "[President Clinton] *will leave* to Mexico"). We therefore parse the sentences with Minipar—a wide-coverage dependency parser [12]—and consider as a *fact nugget* every predicate (usually, a verb) with all its arguments and modifiers. Table 1(a) shows an example for the topic *Cassini space probe*.

Finally, every extracted fact is given a *prior importance estimation*: the retrieval score of the document from which the fact was extracted.

3.3 Mining Facts from the Reference Corpus

In order to obtain a list of "good" facts for a given topic, we now repeat the fact extraction stage, with slight modifications, for the reference corpus. First, we extract a high-quality document (i.e., an encyclopedia entry) for the topic. We then apply the anaphora resolution and sentence splitting methods described in the previous section. Next, we assign *importance* to each fact, based on layout cues in the document, such as proximity to the beginning of the entry. These heuristics are based on the fact that in encyclopedia entries, important information is typically given first, data in tables is usually significant, and so on. An example of facts extracted from an encyclopedia entry is given in Table 1(b).

3.4 Estimating Importance of Facts

At this stage, we have two lists of nuggets: facts from the target corpus, with prior importance estimation, and reliable facts from the reference corpus, each with its importance value. To refine the importance estimation for the target facts, we calculate sentence-level similarity between the target and reference nugget lists: we exhaustively compare each target fact to each fact from the reference corpus. We experimented with two types of sentence-level similarity measures: lexical and semantic.

We measure lexical similarity by determining the word overlap between the sentences, using metrics such as Jaccard [8] to normalize over the sentence lengths. Prior to the comparison we use standard stemming and stopword removal on both sentences to increase the morphological uniformity. As to semantic similarity between sentences, we use linguistically motivated techniques to find similarities also between sentences which do not match on the surface level. We use two types of metrics; the first is the total WordNet distance of words appearing in the sentences, based on methods described in [7]. Alternatively, we use similarity scores between pairs of words derived from proximities and co-occurrence in large corpora, described in [13], and sum the total proximity measure for the words in the two segments.

In the experiments described below we used the lexical similarity with Jaccard metric. Later we found that co-occurrence-based measures seem to give better estimates of sentence similarity. A careful evaluation of different measures for this task is in our future plans.

Let $\{t_i\}$ denote the list of facts extracted from the target corpus and $\{r_j\}$ the reliable facts from the reference corpus. We denote the similarity between two facts as $\mathrm{sim}(t_i, r_j)$, the prior importance estimation of a target fact as $I_{pr}(t_i)$ and the importance of the reliable fact as $I(r_j)$. Then the updated, posterior importance estimation of a target fact is calculated as follows:

$$I_{post}(t_i) = I_{pr}(t_i) \cdot \max_j \left(I(r_j) \cdot \mathrm{sim}(t_i, r_j) \right).$$

We sort the target facts by decreasing posterior importance and present the top N as the key facts about the topic.

3.5 Removing Redundant Facts

At the TREC 2004 QA track, each "other" question was asked after a sequence of factoid questions, all about a given topic. Therefore, an additional requirement was set on the response to the "other" question: the retrieved facts should not duplicate the information conveyed by (answers to) the factoid questions.

To avoid such duplication, we performed another filtering step: from the ranked list, we omit nuggets that are *similar* to other nuggets higher in the ranking, or to one of the factoid questions together with its answer (as found by our factoid-QA system). We use the same sentence-level similarity measure $\mathrm{sim}(\cdot, \cdot)$ as for the posterior importance estimation.

Document text	The Cassini space probe , due to be launched from Cape Canaveral in Florida of the United States at dawn , is carrying 33 kg of plutonium needed to power A rocket's seven-year journey to Venus and Saturn . Local mass media quoted opponents of Cassini as saying at the weekend that the mission will cross Panama , the Caribbean , Southern Africa and Madagascar be fore hurtling into space . Foreign affairs spokesman Pieter Swanepoel said neither had anything's department received any request or contacted the American authorities to find out what was happening with Cassini . . . .
Extracted facts	• The Cassini space probe : due to be launched from Cape Canaveral in Florida of the United States at dawn • Local mass media quoted opponents of Cassini as saying at the weekend the mission will cross Panama • Foreign affairs spokesman Pieter Swanepoel said neither had anything's • department to find out what was happening with Cassini • Pieter Swanepoel : Foreign affairs spokesman • . . .

(a) Extracting facts from the target corpus.

Encyclopedia entry	Cassini-Huygens is a joint NASA/ESA unmanned space mission intended to study Saturn and its moons. The spacecraft consists of two main elements: the Cassini orbiter and the Huygens probe. The spacecraft was launched on October 15 , 1997 and entered Saturn's orbit on July 1 , 2004. October 15, is the first spacecraft to orbit Saturn and just the fourth spacecraft to visit Saturn.
Extracted facts	1. Cassini - Huygens a joint NASA/ESA unmanned space mission intended to study Saturn and its moons 2. The spacecraft consists of two main elements 3. the Cassini orbiter the Huygens probe 4. The spacecraft entered Saturn's orbit on July 1, 2004 5. October 15, is the first spacecraft to orbit Saturn just the fourth spacecraft to visit Saturn 6. October 15, to visit Saturn

(b) Extracting facts from the reference corpus.

Re-ranked facts	• Cassini will be carrying 12 separate packages of scientific instruments a probe [3] • Saturn's largest moon [1] • department to find out what was happening with Cassini [3] • the instruments on Cassini to provide pictures of Saturn Nearly seven meters' rings moons radar to pierce the orange [1] • The Cassini space probe : due to be launched from Cape Canaveral in Florida of the United States at dawn [3] • . . .

(c) Re-ranked facts from the target corpus (with the id of the most similar reference fact in brackets).

Table 1: Fact extraction and re-ranking in action.

4. EVALUATION

4.1 Experimental Setting

We applied the described method to find answers to the "other" questions and evaluated it within the TREC Question Answering track at TREC 2004 [19]. For the QA task, a list of questions was given, divided into 65 groups, each organized around a certain topic; examples include *James Dean*, *cataract* and *Teapot Dome scandal*. For each topic, a number of factoid questions were given, and an additional "other" question which requires as a response a list of important information nuggets regarding the topic. The important nuggets were set in advance by the assessors, and were divided into "essential" facts and less important "acceptable" ones. Each participant in the track returned a list of information nuggets for each topic.

The response was scored using the F-measure of precision and recall with recall three times more important. The recall measures the fraction of the essential nuggets returned, and the precision penalizes nuggets not considered essential or acceptable, and very long nuggets. More precisely, let E be the number of essential nuggets identified by the assessors; ER and AR are the number of nuggets returned by the system and judged as essential and acceptable, respectively;

length denotes the total length (the number of non-white-space characters) in the returned nuggets. Then

$$\begin{aligned} R &= ER/E \\ allowance &= 100 \cdot (ER + AR) \\ P &= \begin{cases} 1, \text{ if } length < allowance \\ 1 - [(length - allowance)/length], \text{otherwise} \end{cases} \\ F &= (10 \cdot P \cdot R)/(9 \cdot P + R) \end{aligned}$$

In essence, this F-measure gives a higher importance to recall than precision, and rewards responses with lengths which are less than a per-topic threshold.

4.2 Results

In order to evaluate the effect of ranking nuggets using an external reference corpus, we included two versions of answers to “other” questions in the our official TREC QA runs, the baseline and a re-ranked version:

- In the baseline version we extracted nuggets from the target collection, as described above, and used prior estimates for the importance of the facts (I_{pr}) to rank the nuggets. We submitted 20 or less nuggets per topic (20 was an arbitrary threshold).

- For the re-ranked version, we used posterior estimates of the nugget importance (I_{post}) instead; also 20 or less facts were submitted per topic.

The results of the runs are given in Table 2. For comparison, the best system at TREC 2004 achieved an F-measure of 0.46, while the median F-measure over all 63 submitted runs is 0.184.

Measure	**Baseline**	**Re-ranked**
Precision	0.176	0.220 (+25%)
Recall	0.208	0.237 (+14%)
F-measure	0.184	0.210 (+14%)

Table 2: Evaluation results for “other” questions.

Note that the version of the F-measure used at TREC 2004 is biased towards recall. As is clear from Table 2, our re-ranking method substantially improves both recall and precision, but more so for precision.

A further per question breakdown of the change of performance in terms of F-measure is given in Figure 1 (top), indicating that while the gain in F-measure averaged over all questions is positive, there are questions whose score is affected negatively by our re-ranking mechanism. The results in Table 2 indicate that our re-ranking mechanism affects precision and recall differently; this is reflected in Figure 1 (middle) and (bottom), where we provide per-question breakdowns for precision and recall.

For 26 questions the F-measure of the original sentences (without re-ranking) is 0, preventing any improvement from our re-ranking method; in Figure 2 we present the breakdown for all but the 26 zero scoring questions.

4.3 A Closer Look

An analysis of the assessed runs revealed that often good nuggets were in the collection, but not in the top 20 documents we used for the extraction. Indeed, the threshold of

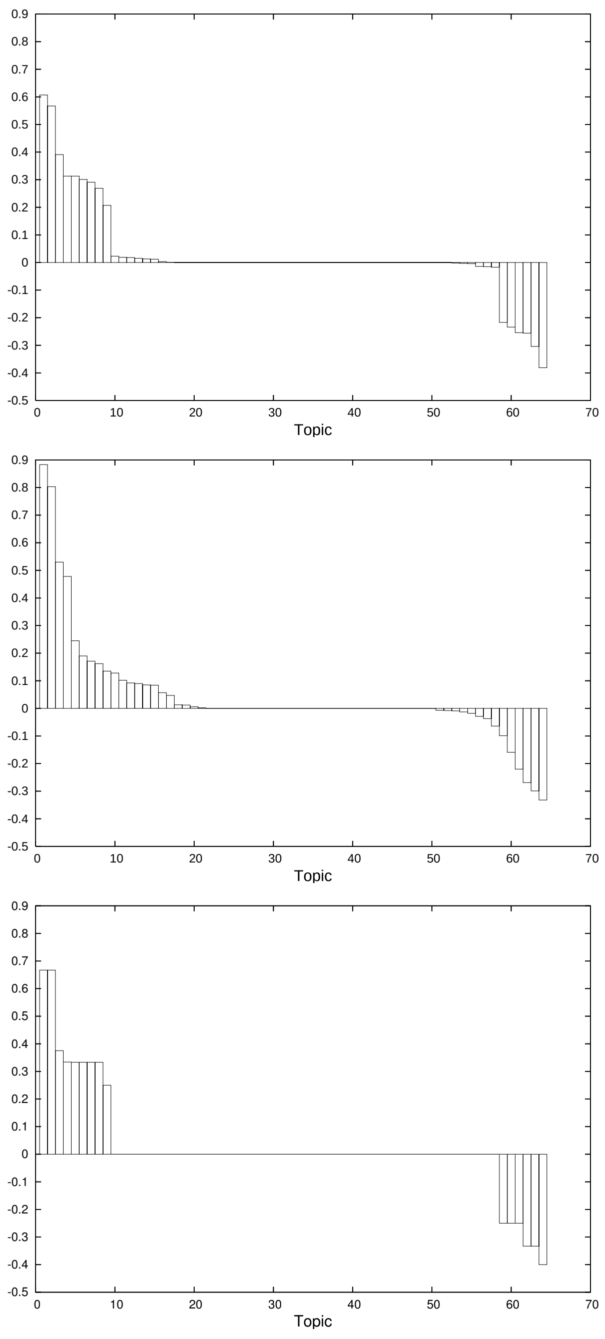


Figure 1: Per-question breakdown of effect of re-ranking, all questions: F-measure (top), precision (middle), and recall (bottom).

20 was set mainly for computational reasons, and further experiments with higher thresholds has shown clear improvements. Since the evaluation of new runs has to be done manually, we have no numerical support for this claim.

Another major source of errors was the similarity measure (normalized word overlap) used in our submitted runs. Because of its sparsity, often the decision about a match was based on a single common word, as, e.g., for the third nugget in Table 1(c). Again, experimenting with more suit-

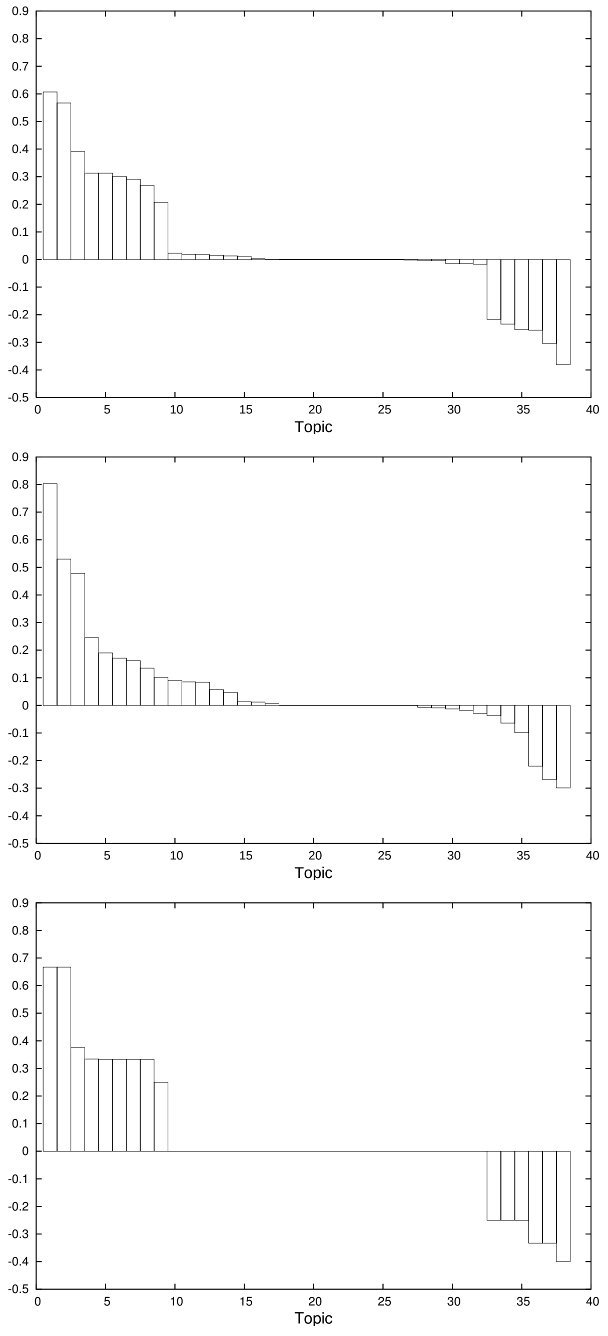


Figure 2: Per-question breakdown of effect of re-ranking, for questions with non-zero F-measure before re-ranking: F-measure (top), precision (middle), and recall (bottom).

able measures is hindered by the lack of automatic evaluation methodology: unlike the factoid questions at TREC, for "other" questions it is difficult to create patterns of correct answers. Developing such effective automatic evaluation methods is essential for improving the systems.

Re-ranking errors were also caused by our sentence splitting and anaphora resolution methods. For example, for the topic "*Carlos the Jackal*" one of the important nuggets "*the man known as Carlos the Jackal, once considered the world's most wanted terrorist, is serving a life sentence there*" was discarded after re-ranking, although the reference corpus provided the nugget "*on December 23 he was found guilty and sentenced to life imprisonment.*" Although both contain the key words *sentence* and *imprisonment*, the nugget from the target collection was too long for the similarity to be detected by our method. A better sentence splitter (capable of ignoring reduced relative clauses) could make nuggets shorter and the similarity more obvious. Another reason for discarding the snippet was the incorrectly resolved referential "*there.*" Had it been resolved to its true antecedent "*La Sante,*" the snippet could have matched the reference nugget "*he was sent to La Santé de Paris prison to await trial.*"

4.4 Discussion

In our method, we assume availability of a reference corpus, a high-quality, clean and well-structured text collection, reflecting *the user's perspective* on which information is relevant or important for which topics. When faced with new information (in our case, coming from a different, unstructured, less reliable and less focused corpus), we use the reference collection to identify and rank new facts. Sentence similarity is used as a device to check how well a new bit of information matches the needs of users.

Our implementation of this model, as described in this paper, is far from complete. First of all, currently our system is capable of identifying facts from the target corpus which are very similar to those in the reference collection. But, users would probably also be interested in finding new facts, different from those the reference corpus can provide. To address this quite natural need, we can generalize the notion of fact similarity. Abstracting from concrete entities in the reference corpus, we can observe, for example, that if the user model considered the fact "*X was founded...*" important for the topic X, then the fact "*Y was founded...*" is likely to be important for the topic Y. Modifying the similarity metric to use information about, e.g., named entities and their types, we can use our reference corpus on a more abstract level and provide the user with both new and important information.

Second, the method we use to split long sentences (typical for newspaper text) into manageble facts is not very robust. It is based on full syntactic parsing and suffers from parsing errors, often producing ungrammatical and hardly interpretable nuggets. While parsing (identification of predicate-argument structure) can lead to a more informed estimation of fact similarity, more robust chunking methods should probably be used to present results to the user.

There are many other parts of the system that need attention: disambiguating entries in the reference corpus using previous questions of the user, improving the anaphora resolution module (see, e.g, nugget 5 in Table 1(b), where the pronoun *it* was incorrectly resolved to *October 15*) and extending it to handle definite NP anaphora.

5. CONCLUSIONS

We described a way to use high-quality semi-structured resources to model preferences of users for retrieval of short facts. By comparing facts extracted from a target collection to the information from a reference resource, we identify those facts that are potentially *important* for the user. Even with a simple word overlap-based similarity measure, this method shows reasonable performance: applying it to answer "other" questions in the TREC 2004 QA track, we show substantial improvements over the baseline.

Our preliminary analysis of the TREC 2004 results suggest experimenting with more sophisticated sentence-level similarity measures and improving sentence splitting for extraction of atomic facts.

Acknowledgments. We thank David Ahn for his work on the anaphora resolver used in our experiments. This research was supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. Maarten de Rijke was also supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 612.-000.106, 612.000.207, 612.066.302, and 264-70-050.

6. REFERENCES

[1] D. Ahn, V. Jijkoun, J. Kamps, G. Mishne, K. Müller, M. de Rijke, and S. Schlobach. The University of Amsterdam at TREC 2004. In *TREC 2004 Conference Notebook*, Gaithersburg, Maryland USA, 2004.

[2] AltaVista Search Logs, URL: `ftp://ftp.archive.org/pub/AVLogs/`.

[3] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In *Proc. 10th Text REtrieval Conference*, 2001.

[4] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM Press, 1998.

[5] J. Chu-Caroll, J. Prager, C. Welty, K. Czuba, and D. Ferrucci. A multi-strategy and multi-source approach to question answering. In *Proceedings TREC 2002*, 2003.

[6] C. A. Clarke, G. Cormack, G. Kemkes, M. Laszlo, T. Lynam, E. Terra, and P. Tilker. Statistical selection of exact answers (multitext experiments for trec 2002). In *Proceedings TREC 2002*, pages 823–831, 2003.

[7] M. De Boni and S. Manandhar. The use of Sentence Similarity as a Semantic Relevance Metric for Question Answering. In *Proceedings of the AAAI Symposium on New Directions in Question Answering*, 2003.

[8] P. Jaccard. The distribution of the flora of the alpine zone. *New Phytologist*, 11:37–50, 1912.

[9] P. Jourlin, S. Johnson, K. Spärck Jones, and P. Woodland. Improving retrieval on imperfect speech transcriptions. In *Proc. SIGIR '99*, pages 283–284, Berkeley, CA, 1999.

[10] J. Kamps. Improving retrieval effectiveness by reranking documents based on controlled vocabulary. In S. McDonald and J. Tait, editors, *Advances in Information Retrieval: 26th European Conference on IR Research (ECIR 2004)*, LNCS 2997, pages 283–295. Springer-Verlag, 2004.

[11] J. Kupiec. MURAX: a robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 181–190. ACM Press, 1993.

[12] D. Lin. PRINCIPAR – an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, 1994.

[13] D. Lin and P. Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[14] J. Lin, A. Fernandes, B. Katz, G. Marton, and S. Tellex. Extracting Answers from the Web Using Knowledge Annotation and Knowledge Mining Techniques. In *Proc. 11th Text REtrieval Conference*, 2002.

[15] G. Mishne and M. de Rijke. Query formulation for answer projection. In *Proceedings ECIR 2005*, 2005.

[16] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 143–150. ACM Press, 2003.

[17] I. Soboroff and D. Harman. Overview of the TREC 2003 novelty track. In *Proceedings TREC 2003*, pages 38–53, 2004.

[18] W. van Hage, M. de Rijke, and M. Marx. Information retrieval support for ontology construction and use. In S. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Proceedings 3rd International Semantic Web Conference (ISWC 2004)*, LNCS 3298, pages 518–533, 2004.

[19] E. Voorhees. Overview of the TREC 2003 question answering track. In *Proceedings Twelfth Text Retrieval Conference (TREC 2003)*, pages 54–68, 2003.

[20] E. Voorhees. Overview of the TREC-2004 question answering track. In *Proceedings 13th Text REtrieval Conference*, Gaithersburg, Maryland USA, To appear.

[21] J. Xu, A. Licuanan, and R. Weischedel. TREC 2003 QA at BBN: answering definitional questions. In *Proceedings TREC 2003*, pages 98–106, 2004.

DIR'5

Utrecht University
Center for Content and Knowledge Engineering
P.O. Box 80.089
3508 TB Utrecht
the Netherlands

ISBN: 90-393-0031-3