IIIT Hyderabad at TAC 2009

by

Vasudeva Varma, Vijay Bharath Reddy, Sudheer K, Praveen Bysani, GSK Santosh, kiran kumar, kranthi Reddy, karuna Kumar, nithin M

in

Text Analysis Conference

National Institute of Standards and Technology Gaithersburg, Maryland USA

Report No: IIIT/TR/2009/222

Centre for Search and Information Extraction Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2009

IIIT Hyderabad at TAC 2009

Vasudeva Varma
Vijay Bharat
Sudheer Kovelamudi

Praveen Bysani
Santosh GSK
Kiran Kumar N

Kranthi Reddy
Karuna Kumar
Nitin Maganti

Search and Information Extraction Lab
Language Technologies Research Center
IIIT Hyderabad, India.

Abstract

In this paper, we report our participation in Update Summarization, Knowledge Base Population and Recognizing Textual Entailment at TAC 2009. This year, we enhanced our basic summaization system with support vector regression to better estimate the combined affect of different features in ranking. A Novelty measure is devised to effectively capture relevance and novelty of a term. For Knowledge Base Population, we analyzed IR approaches and Naive Bayes Classification with Phrase and Token searches. Finally for RTE, we built templates using WordNet and predict entailments.

Part I
Update Summarization Track

1 Introduction

Update Summarization is a new stride in summarization community. Ever since its introduction in DUC 2007 [1], there has been a consistently growing focus in this direction. The task is to summarize a cluster of documents under the assumption that user had some prior knowledge on topic. The major challenge in update summarization is to detect information that is not only relevant to users need but also novel given the user's prior knowledge. Update summarization is relevant for newswire, since a topic in news stories evolves over time and user/reader would only be more interested about new information about that topic.

NIST first introduced update summarization as a pilot task at DUC 2007, later as a main task at TAC 2008 and continued it in TAC 2009. While the problem definition remained the same, the quality of data have improved through out the years. In DUC 2007, the update task data was just a subset of Multi-Document Summarization data. In 2008 sufficient care is taken such that there are distinctive events between clusters, but a huge time gap is observed between clusters. For TAC 2009, a lot of time and efforts has been put into choosing news topics and creating appropriate document clusters.

Update summarization shares similarity with Novelty track introduced at TREC 2002[2]. The Novelty track was designed to investigate systems' ability to locate relevant, novel information within the ranked set of documents retrieved in answer to a topic. Update Summarization is in a way an extension to Novelty track as it needs to summarize the content along with detecting relevant and novel information. Researchers have approached the problem of "Update Summarization" at varying levels of complexity during past couple of years at TAC. Ruifang He et. al [6] proposed an iterative feedback based evolutionary manifold ranking of sentences for update summarization. Ziheng Lin et. al [20] followed time stamped graph approach incorporating information about temporal ordering of events in articles to focus on the update summary. There are also simple content filtering approaches [19] which identify dynamic content and generate summaries.

Recent advances in machine learning like perceptrons [4], markov models [13], CRF's [17] and bayesian classifiers [9] have been adapted to summarization throughout the years. We use a machine learning method, Support vector regression (SVR) for sentence ranking. Sujian Li, You Ouyang [10] were the first to use regression in the context of text summarization to predict sentence scores. FastSum [16] also utilizes support vectors to score sentences using multiple features.

In this work, we introduce a new feature Novelty Factor (NF), that is devised to capture relevance and novelty of a sentence within a topic. Computationally NF is a very simple feature, yet very effective. Official TAC results support our argument on NF. We

[1] http://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html#pilot

[2] http://trec.nist.gov/data/novelty.html

secured 1st position in avg-modified pyramid scores for cluster B, and 3rd in ROUGE-2 and ROUGE-SU4 recall scores for both clusters A, B. Our Post TAC experiments have shown significant imporovement over current results for cluster A.

2 System Description

We built a sentence extractive summarizer, that extracts, scores and ranks a sentence before finally generating a summary. During ranking, instead of manually weighting each sentence scoring feature 3, we utilize a machine learning algorithm, Support Vector Regression (SVR) to predict *sentence rank*. In following sections we briefly explain SVR, estimation of sentence importance and our algorithm to generate summaries.

2.1 Support Vector Regression

Regression analysis refers to techniques for modeling values of a dependent variable from one or more independent variables. Support Vector Machines, a popular mechanism for classification purposes could also be used for regression purposes (Vapnike ,Gunn 1988) [5].

Consider the problem of approximating the set of training data

$$T = \{(F_1, i_1), (F_2, i_2)...(F_s, i_s)\} \subset F \times R.$$

where F is space of feature vectors.

A tuple (F_s, i_s) represents feature vector F_s and importance score i_s of sentence s. Each sample satisfies a linear function $q(f) = \langle w, f \rangle + b$, with $w \in F, b \in R$.

The optimal regression function is given by minimum of functional,

$$\Phi(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i{}^- + \xi_i{}^+$$

where C is a pre-specified value, and $\xi_i{}^-, \xi_i{}^+$ are slack variables representing upper and lower constraints on the outputs of the system.

We use Radial bias kernel function for our experiments.

2.1.1 Sentence Importance Estimation

Importance score (i_s) is not pre-defined for sentences in training data, we estimate the value of importance using human written summaries(also known as models) on that topic. ROUGE-2 and ROUGE-su4 scores highly correlate with human evaluation [11]. Hence we make a safe assumption that importance of a sentence is directly proportional to its overlap with model summaries.

Sentence importance is estimated as the ROUGE-2 score of that sentence. The importance of a sentence s, denoted by i_s is computed as follows

$$i_s = \frac{\sum_{m \in models} |Bigram_m \bigcap Bigram_s|}{|s|} \quad (1)$$

$|Bigram_m \bigcap Bigram_s|$ is number of bigrams shared by both model m and sentence s. This count is normalized using sentence length $|s|$.

2.2 Algorithm

Our system follows a 3 stage algorithm to generate summaries,

1. **Pre-Processing**
   In pre-processing stage, documents are cleaned from news heads and HTML tags. Stop words are removed and porter stemmer is used to derive root words eliminating suffixes are prefixes. Sentences are extracted from each document.

2. **Feature Combination**
   Features used for sentence scoring are combined to rank sentences. Normally features are manually weighted to compute sentence rank. This process is automated with use of SVR in 3 steps,

   - *Sentence tuple generation*: Feature values of every sentence are extracted and its importance(i_s) is estimated as described in Section 2.1.1. Each sentence s in training data is converted into a tuple of form (F_s, i_s). F_s is vector of feature values of sentence $F_s = \{f_1, f_2, f_3\}$
   - *Model building*: A training model is built using SVR, from generated sentence tuples.
   - *sentence scoring*: Importance of a sentence in testing dataset is predicted based on the the trained model. The estimated importance value is considered as rank of sentence.

     $$i_s = q(F_s)$$

3. **Summary Generation**
   During summary generation, a subset of ranked sentences are selected to generate summary. A redundancy check is done between a sentence and summary generated so far, before selecting it into summary. This step helps to prevent duplicate sentences in summary. Sentences are adjusted based on their order of occurrence in documents to improve readability. Reported speech is removed from summary to alleviate its conciseness.

3 Features

For our previous participation at TAC 2008 and DUC 2007, complex language modeling techniques like Probabilistic hyperspace analogue to language (PHAL) [7], KullbackLeibler divergence (KL) [1] are used as features in our system. This year we have used Sentence Position (SL1 and SL2) and Novelty Factor (NF) that is specifically devised for update task as sentence scoring features.

3.1 Sentence position

Sentence position is a very old and popular feature used in summarization [3]. It is well studied and still used as a feature in most state of art summarization systems [8]. We use the location information of a sentence in two seperate ways to score a sentence.

Sentence Location 1 (SL1):

First three sentences of a document generally contain the most informative content of that document which is proved by our analysis on the oracle summaries (in Section 5.1). Nearly 40% of all the sentences of the oracle summaries come from among the first three sentences of each document.

Score of a sentence s at position n in document d is given by,

$$\begin{aligned} Score(s_{nd}) &= 1 - \frac{n}{1000} \qquad if \;\; n <= 3 \\ &= \frac{n}{1000} \qquad else \end{aligned}$$

(Assuming that number of sentences in a document will be less than 1000)

Such that,

$$Score(s_{1d}) > Score(s_{2d}) ... >> Score(s_{nd})$$

Sentence Location 2 (SL2):

Positional index of a sentence in the document is assigned as the value of feature. Training model will learn the optimum sentence position for the dataset based on its genre. Hence this feature is not inclined towards top or bottom few sentences in a document like SL1.

$$Score(s_{nd}) = n$$

where s_n is nth sentence in document d.

3.2 Novelty Factor (NF)

We propose a new feature ***Novelty Factor* (*NF*)** that primarily focuses on update summarization problem.Consider a stream of articles published on a topic over time period T. All the articles published from time 0 to time t is assumed to have been read previously (*previous clusters*). Articles published in the interval t to T are unread articles that might contain new information (*new cluster*). Let the publishing date of a document d is represented by t_d. *NF* of a word is calculated by

$$nf(w) = \frac{|nd_t|}{|pd_t| + |D|}$$

$nd_t = \{d : w \in d \wedge t_d > t\}$
$pd_t = \{d : w \in d \wedge t_d <= t\}$
$D \;\; = \{d : t_d > t\}$

Numerator $|nd_t|$ is the number of documents in the new cluster that contain word w. It is directly proportional to relevancy of the term, since all the documents in the cluster are relevant to the topic. The term $|pd_t|$ in denominator will penalize any word that occurs frequently in previous clusters, in other words it elevates novelty of a term. $|D|$ is total number of documents in current cluster, this is useful for smoothing values when w do not occur in previous clusters.

Update task data in TAC 2009 consists of two clusters, cluster A is the only *previous cluster* and B is the *new cluster*. Hece the NF for a word in

cluster A
$nf_{clusA}(w) = \frac{d_A}{D_A}$

cluster B
$nf_{clusB}(w) = \frac{d_B}{D_A + D_B}$

Score of a sentence s is the average nf value of its content words (w). We exclude query words while computing the score since they are equally important in both the clusters.

$$Score(s) = \frac{\sum_{i \in s} nf(w_i)}{|s|}$$

NF score of a sentence is a measure of its relevance and novelty to the topic.

4 Evaluation and Results

TAC 2008 update task documents and corresponding models are used to generate data for training SVR. It provides 48 topics, each topic contains 20 documents divided in chronological order between cluster A and cluster B. Summary for cluster A is normal multi document summary of length 100 words, where

as summary for cluster B is an update summary of 100 words.

TAC 2009 Update summarization data has 45 topics with documents distributed in each topic the same way as TAC 2008. NIST evaluates all peer summaries manually for overall responsiveness, readability and linguistic quality. All summaries were also automatically evaluated using ROUGE and BE. Evaluations are also conducted using pyramids [15], which are built using Semantic Content Units(SCU) from corresponding model summaries of a topic.

We submitted two runs, Run1 and Run2 for TAC 2009 update summarization track,

Run 1(System id:35) uses two features NF and SL1 for sentence ranking. The training model for SVR is built upon TAC 2008 update task data.

Run 2(System id:51) generates summary using two features NF and SL2. DUC 2007 update task data is used to build training model.

4.1 Results

Official TAC evaluation results of Run1 and Run2 on various intrinsic and extrinsic evaluation criterion for clusA and clusB are presented in Table 1 and Table 2 respectively.

Run1 has proved to work exceptionally well in *update scenario*. Run1 is ranked 1st in Average modified pyramid scores(APS), 4th in Overall Responsiveness(OS) and 3rd in ROUGE Scores(R2 and Rsu4) for cluster B which is essentially the update cluster. It worked equally well for cluster A, given that it secured 3rd in ROUGE, 4th in Overall Responsiveness. Average modified pyramid scores of cluster A is slightly below than best systems.

Run2 is an experimental run to find the effect of training on overall performance of the system. The main difference between Run1 and Run2 is the quality of training data. Even though Run2 performs comparatively well than most of the systems that participated, it still does not make to the top unlike Run1. Hence it is inferred that as the quality of training improves so do the performance of system.

5 Post TAC experiments

5.1 Oracle Summaries

We generated *sentence-extractive Oracle Summaries* using test document set and their model summaries. Each oracle summary is the best sentence extractive summary that can be generated by any sentence extractive summarization system for that topic. Sentences are ranked using Equation 1 to produce these summaries. Motivation behind generating these summaries is to find the scope of improvement in sentence extractive summarization.

After TAC submission, some new and simple features are devised to improve the summary quality especially in update scenario.

5.2 Query Focus

All the features (NF,SL1, and SL2) in current summarizer are query independent and makes no use of any information provided by query. Even without query focus these features are able to perform very well. Hence, a new feature *Qterms* is introduced to incorporate query focus into current summarization system.

Qterms In query focused summarization systems, a sentence is considered relevant only if it contains a query term. We use the same intuition to score query focused sentences. A sentence (s) is scored by the amount of query terms it contains

$$Score(s) = \frac{\sum_{w \in s} F_Q(w)}{|s|}$$

where

$$\begin{aligned} F_Q(w) &= n \quad if \ w \in Q \\ &= 0 \quad else \end{aligned}$$

Q is list of all the words in query
n is frequency of w in Q

5.3 Novel Word Count (Nwords)

Novelty Track at TREC is in many ways similar to the update summarization. The task is to mark novel sentences within a set of relevant sentences for a topic. As state of art summarizers are sentence extractive and update summarization requires to extract novel sentences to build an update summary, novelty track approaches can compliment update summarization in identifying sentences with new information.

A simple approach to detect novel sentences is to compute the amount of *new* words in a sentence. Words that never occurred before in document cluster are considered *new*. In this case, we consider any word that occured in cluster A other than query words as old and all the remaining words as new.

A sentence (s) is scored by the amount of Novel (New) words it contains,

$$Score(s) = \frac{\sum_{w \in s} F_{clusA}(w)}{|s|}$$

$$\begin{aligned} F_{clusA}(w) &= 0 \quad if \ w \in clusA \\ &= n/N \quad else \end{aligned}$$

	R-2	R-su4	Average modified Pyramid score	Overall Responsiveness
Run1	0.10840	0.14475	0.299	4.864
Run2	0.10491	0.14167	0.295	4.727

Table 1: *TAC Official results for cluster A*

	R-2	R-su4	Average modified Pyramid score	Overall Responsiveness
Run1	0.10100	0.13833	0.307	4.614
Run2	0.09572	0.13644	0.299	4.568

Table 2: *TAC Official results for cluster B*

$clusA$ is set of words in cluster A
n is number of times w occured in cluster B
N is total term frequency of cluster B

We present in Table 3, Table 4, the effect of these new features along with oracle summaries to depict the scope of improvement in sentence extractive summarization.

	R-2	R-su4
Run1+Qterms	**0.11350**	**0.14969**
Oracle	0.15620	0.18093

Table 3: *ROUGE-2 and ROUGE-SU4 scores of Post-TAC experiments for cluster A*

	R-2	R-su4
Run1+Qterms	0.09106	0.13132
Run1+Nwords	0.09807	**0.14058**
Run1+Nwords+Qterms	0.08923	0.13047
Oracle	0.14978	0.17767

Table 4: *ROUGE-2 and ROUGE-SU4 scores of Post-TAC experiments for cluster B*

Query focus (Qterms) has helped in producing better summaries for cluster A. Around 4% improvement in ROUGE scores is observed. It is interesting to see that, the same feature failed to improve quality of cluster B summaries.

Inclusion of New Novelty feature (Nwords) does not show any significant growth over Run1. Adding Qterms to this combination only resulted further dip in scores.

6 Discussion

Novelty Factor(NF) is able to perform well in summarization because all the documents given under a topic are relevant to it. Hence, the importance of a term is directly proportional to the number of documents in which it occurs. In context of update summarization, there is a need to penalize the terms that are authoritative in cluster A. So, the importance of a term in cluster B is inversely proportional to the number of documents in which it occured in cluster A. Both sentence positional algorithms (SL1 and SL2) have performed decently. SL1 is inclined towards top sentences in a document, and SL2 is unbiased towards positional index of a sentence. SL1 works because of the intuition that top sentences would always have informative content. SL2 is a feature that helps boosting informative sentences based on genre of the corpus. SL2 learns significant sentence positions in corpus based on training data. As both training and testing belong to same genre (Newswire) SL2 performs well. In our experiments both SL1 and SL2 performs in a similar way as important sentences in training data are present among top 3 sentences of a document.

The major difference between Run1 and Run2 is the amount and quality of training data. DUC 2007 update task data is just a subset of Multi Document summarization data and TAC 2008 data is specifically handcrafted for update task. It is observed that Run1 produces better summaries than Run2 both in automatic and manual evaluations.

There is still a huge gap between ROUGE scores of oracle summary and machine generated summaries. It reveals the fact that there is a lot of scope for improvement in sentence extractive summarization.

In future we plan to involve NF within a formal language modeling framework. We are currently working on predicting word level importance using SVR. Experiments are being carried out to predict the role of word position in a sentence to decide its importance. Further we plan to explore the use of entailment in summarization scenario.

Part II
Knowledge Base Population (KBP)

1 Introduction

TAC 2009 Knowledge Base Population (KBP) track focuses on automatic updation of structured Knowledge Bases (KB). The task has been broken down into two sub tasks.

1) Entity Linking: The aim of this task is to determine for each query, which knowledge base entry is being referred to, or if the entry is not present in the knowledge base. The query consists of a name-string and a document id from the document collection. Each query string will occur in the associated document in the test collection. The purpose of the associated document is to provide context for that might be useful for disambiguating the name-string. Each query must be processed independently of one another and for each query a knowledge base node id should be returned if present else NIL.

2) Slot Filling: The Slot Filling task involves learning a pre-defined set of relationships and attributes for target entities based on the documents in the test collection. A query in the Slot Filling task will contain a name-string, docid, entity-type, node-id, and a list of slots to ignore. The node id that is provided will refer to a node representing the entity in the KB. For targets for which no node exists in the KB, the node-id will begin with NIL. As in the entity linking task the provided docid is intended to give context for the entity.

Systems must process the target entities (i.e., each query) independently from one another. For each slot value returned, systems must also return a single docid from the test collection that supports the value returned for the given entity and slot. Slots can be single valued slots or multi valued slots.

2 Approach For Entity Linking

We have broken down the Entity Linking task into three separate modules.

1. **Preprocessing:** Our aim during the preprocessing step is to build a Knowledge Repository of entities that contains vast amount of world knowledge of entities like name variations, acronyms, confusable names, spelling variations, nick names etc. We use Wikipedia[3] to build our knowledge repository.

The advantages of using wikipedia are

- It has better coverage of named entities[18].
- Redirect pages provide us with synonyms[18].
- Disambiguation pages can be used for homonym resolution[14].

We use the following link structure from Wikipedia to extract the name variations of an entity.

(a) **Redirect Links:** A redirect page in wikipedia is an aid to navigation. When a page in wikipedia is redirected it means that those set of pages are referring to the same entity. They often indicate synonym terms, but can also be abbreviations, more scientific or more common terms, frequent misspellings or alternative spellings etc.

(b) **Disambiguation Pages:** Disambiguation pages are specifically created for ambiguous entities, and consist of links to articles defining the different meanings of the entity. This is more useful in extracting the abbreviations of entities, other possible names for an entity etc.

(c) **Bold Text from First Paragraph:** In wikipedia the first paragraph usually contains a summary of the article or most important information about the article, thus containing the most relevant words for that article. We extract phrases from the first paragraph of wikipedia article that are written in bold font. This bold text generally refers to nick names, abbreviations, full names etc.

(d) **Metaphones:** In order to identify spelling variations for a given entity we use the metaphone algorithm [2] .We generate the metaphonic codes for each token generated using the above 3 features.

(e) **Lucene:** Lucene [4] is used as an underlying retrieval system by us to retrieve entity mapping from the knowledge repository created using the above features. Metaphone codes for each token of knowledge repository is also stored in this index. We also index the knowledge base and wikipedia to facilitate fast retrieval of data.

2. **Candidate List Generation:** Since the entity linking task involves the determination of

[3] Wikipedia is a huge collection of articles. Each article is identified by a unique title. These articles define and describe about events and entities.

[4] Lucene is a high-performance, full-featured text search engine. http://lucene.apache.org/java/docs/.

whether a knowledge base node exists or not for a given query, we try to generate the possible candidate maps for a given query from the knowledge base as well as wikipedia. The motive behind adding articles from wikipedia also to the candidate items list is that our knowledge base is a subset of articles from wikipedia. The addition of wikipedia articles to the candidate items list helps us in identification of nill valued queries, that is if for a given query we have nodes from both knowledge base and wikipedia in our candidate items list and our algorithm maps to the wikipedia article we can conclude the non-presence of the node in knowledge base describing about this entity.

Candidate items list is generated using only the title information of the articles and the given query terms, also we follow different approaches for queries that are Acronyms and those that are not.

In order to identify if a given query is an acronym or not we use a simple heuristic based approach. If all the letters present in the query are capitals we consider it as an acronym else it is not. The algorithm for handling these two cases is as follows.

(a) **Not an Acronym:** If the given query is not an acronym we search for the query terms directly in the title field of the nodes of knowledge base. If an hit is found we add that node to the candidate items list. If no hit is found it means that the query could be a variation of the entity or variation of the entity is present in the knowledge base. We then search on the knowledge repository index created by us in order to get the possible variations for the entity. If we find any possible variations for an entity in our knowledge repository index, we search in the knowledge base index on title field using these variations. If any hits are found we add those nodes to the candidate items list. We also search in the wikipedia index using the retrieved variations. If any hits are found in the wikipedia index we add them to the candidate items list.

If no variations are found in the knowledge repository index as well, we assume that the entity might have been written in a different form than it is usually written. we generate the metaphone code for each token present in the query. We search in the metaphone field of the knowledge base index. If any hits are found we add those nodes to the candidate items list. The algorithm is depicted in Fig.1 .

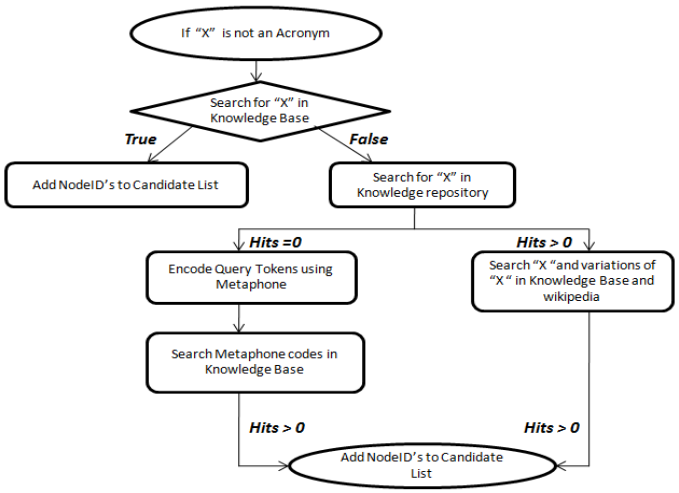

Fig. 1: *Flow chart when query is not an Acronym*

(b) **Acronym:** If the given query is an acronym we try to get the expanded form from the document content which has been given as disambiguation text. We remove stop words from the disambiguation text and use an N-Gram based approach to find the expanded form. If the length of the acronym is "N" characters, we check if "N" continuous tokens in the disambiguation text have the same initials as the characters of the acronym. If we are successful in finding the expanded form from the disambiguation text, we search in the knowledge base index using these tokens. If any hits are found we add those nodes to the candidate items list. If we don't find the expanded form from the disambiguation text we search in the knowledge repository index for the acronym. If we find the expanded form in the knowledge repository index, we search in the knowledge base index using this expanded form. If any hits are found in the knowledge base index we add those nodes to the candidate items list. We also search in the wikipedia index using the expanded form. If we find any hits in the wikipedia index we add those articles to the candidate items list. We also search using the acronym in the the title field of the knowledge base and wikipedia index. If any hits are found we add those nodes to the candidate items list. The algorithm is depicted in Fig.2 .

Refining the Candidate items list: We delete articles that belong to wikipedia from our candidate items list if it is also present in the knowledge base because if a node is present in the Knowledge base describing about an entity, it

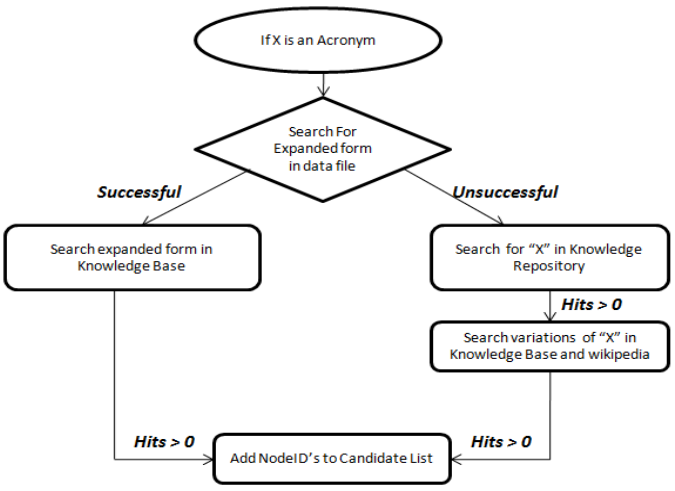

Fig. 2: *Flow chart when query is not an Acronym*

could be the possible link for our query.

While searching in the article title of the knowledge base or wikipedia index for generating the candidate item list we can apply 2 different methods.

(a) **Phrase Search:** In this method we see if the exact phrase of the query or the expanded form of the acronym is present in the title field or not. Only on finding the exact phrase we add those nodes to the candidate items list. A variation of this is searching for phrase with a noise of one word.

EG: If the given query is "UT" and if we find the expanded form from our knowledge repository as "University of Texas". In simple phrase search we would be retrieving nodes that have exact phrase "University of Texas" present in the title. But in Phrase search with noise we would be retrieving nodes that have the titles "University of Texas at austin, University of Texas at Dallas" also.

(b) **Token search:** In this method we check if all the tokens of the query or the expanded form of the acronym are present in the title field or not. Variation of token search is searching with a noise of one word. The difference between Phrase search and Token search is that in phrase search the word order is constrained where as in token search just the presence of each token is vital and not the order in which they occur.

EG: If the given query is "CCP" and if we find the expanded form from our knowledge repository as "Chinese Communist Party". In token search we would be retrieving nodes that have titles either "Chinese Communist party" or "Communist Party of China". The node having title "Communist Party of China" is not found as a candidate item in the phrase search.

3. **Calculating Similarity Score:** If there are no nodes present in the candidate list, it means that no node is present in the knowledge base describing that entity. We return NILL in this case. If there is only one node belonging to knowledge base present as a candidate item , it means that we have found an exact match for the query and there are no other entities in our knowledge base node describing about this entity. But if we find that single node belongs to wikipedia, it means that the likely link for our query is not present in Knowledge base and hence we return NIL.

If there are more than one nodes in our candidate items list and all of those belong to wikipedia, we return NILL as there are no nodes present in the knowledge base describing about our entity. But if the candidate items list contains our knowledge base nodes as well, we follow a different approach to solve this problem. In this case we have to find which node is the mostly likely link for the given query. This can be solved in two different methods.

Classification Approach:

(a) **Rainbow Text Classifier** [5] **:** It has several built methods for classification like Naive Bayes[12], SVM, KL-divergence, TFIDF, K-Nearest Neighbor. We have conducted our experiments using Naive Bayes and K-Nearest Neighbors.

If we consider all the possible candidate items as different classes, we need to find which class is the best map for our query. In this approach we use bag of words as a feature and build models for classification using Rainbow Classifier. For building these models we view each candidate item as a separate class and train the model. We then give the disambiguation text provided along with the query as test document. This test document is classified into one of the classes and the score obtained is the likelihood of the test document belonging to that class.

(b) **Information Retrieval Model:** We index the candidate items text using lucene. Each

[5] http://www.cs.cmu.edu/ mccallum/bow/rainbow/

candidate item is treated as a separate document. Query formulation is an important part in the success of this approach. We try and reduce as much noise as possible while generating the query from the disambiguation text along side trying to boost the terms that seem to the most likely terms that describe about our entity. Since the disambiguation text provided has been tagged neatly into different paragraphs, we consider only those paragraphs where the query terms are present. Once we have extracted all the paragraphs that contain the query terms, we remove the stop words. We form a boolean "OR" query of all the tokens generated from the paragraph text.

We give this query to the candidate items index and the relevance score for each document is calculated.

Result generation: Once we are able to find the closest matching node for a given query using the disambiguation text we check if the node belongs to the knowledge base or the wikipedia. If the node belongs to the knowledge base we give the knowledge base node id as the mapping link else if the node belongs to wikipedia we give the link as NILL saying that we don't have a link for the query entity in our knowledge base.

3 Description of runs

We have submitted 3 runs for the entity linking task.

1)Phrase search with noise for candidate list generation and Naive Bayes for classification.
2)Token search with noise for candidate list generation and Naive Bayes for classification.
3)Phrase search with noise for candidate list generation and Information Retrieval approach.

Table 5 shows the results of our various runs. It also includes post TAC experiments. Infact an IR approach with token search performs better than the runs we have submitted for TAC 2009.

Micro-Average Score we obtained through our system outperforms all 35 runs submitted at TAC 2009. The average-median score over all the 35 runs is 71.08% and the base line score is 57.10% when NILL is returned for all the queries. Our system score outperforms median score by as much as 11% and the base line score by 25%.

4 Approach for Slot Filling

In the slot filling task we have to populate the slot value pairs of an entity in the knowledge base. The query for slot filling task will be of the form (Name-String, Doc-id, Entity-type, Node-id, Ignore-slots).

Name-string refers to the entity name whose slot values pairs have to be modified and Entity-type refers to the whether the given entity is either a person, location or organization. Node-id will refer to the knowledge base entry which has to be populated. If the node-id is NIL, in that case we have to populate the whole template of the given entity type.

1. **Preprocessing:** For slot filling task, during the preprocessing stage we index all the documents of the document collection using lucene. This facilitates searching and fast retrieval of the documents.

2. **Approach:** Given a query, we search for the name-string on the index built during the preprocessing stage. We boost the documents that have name-string appearing in the headline of the document. From the retrieved results we consider the top 50 documents for further processing. Our assumption is that these top 50 documents might contain information related to the slots.

   Once we have these documents we need to extract the sentences that might contain the slot values. For this we index the sentences of the top 50 documents we retrieved as separate documents.

3. **Query Formulation:** Since we have been given a generalized mapping from different Wikipedia slot names to a single slot name of the knowledge base. EG: "Nick names", "Also known as" are all mapped to "Alias name". So we have used this information during our query formulation. We form a boolean "OR" query of all the tokens of the mappings provided.

   Now based on the slot value to be filled, we query the index built using the sentence extracted from the top 50 documents and consider the top 10 retrieved sentences for further processing. We assume that these top 10 sentences might contain our slot values.

   We have categorized the slot values as

   (a) Person
   (b) Organisation
   (c) Place
   (d) Date
   (e) Integer
   (f) String

Algorithm	Noise	Phrase/Token search	Micro-Average Score	Nill-valued precision	Non-nill valued precision	Macro-Average Score
IR	1	Token Search	82.25	86.32	76.84	75.70
IR	**1**	**Phrase Search**	**82.17**	**86.41**	**76.54**	**75.39**
IR	0	Phrase Search	81.81	86.90	75.04	75.46
IR	0	Token Search	81.76	86.45	75.52	75.54
NB	**1**	**Token Search**	**81.43**	**85.42**	**76.12**	**75.38**
NB	**1**	**Phrase Search**	**81.25**	**85.37**	**75.76**	**75.10**
NB	0	Phrase Search	81.12	85.91	74.75	75.45
NB	0	Token Search	80.87	85.51	74.69	74.96

Table 5: *Results of Various Experiments.*
IR = Information Retrieval, NB = Naive Bayes

Type	Our Score	Median Score	Best Score
Single Valued Slots	0.761	0.514	0.816
List Valued Slots	0.604	0.439	0.742
SF-Value score, All Slots	0.682	0.461	0.779

Table 6: *Results of Various Experiments.*

Now we need to extract the slot value from the 10 sentences we have. For this we have used Stanford Named Entity Recognizer(NER). We tag the 10 sentences using this NER and based on the slot value type we extract the value from the sentences. Different sentences might return different values, we consider the value with highest frequency as the correct slot value and return it as the output.

For list valued slots we return the top 3 frequency results.

4. **Results:** Table 6 shows the results of our single run submitted at TAC 2009 for slot filling task.

5 Conclusion and Future Work

We have presented a general overview of building our system for TAC 2009 which analyzes a query and it's disambiguation text in order to find a link in a knowledge base. We showed that an indexing based approach with word search noise we able to outperform all the othe approaches. Our approach and experiments indicate that the 2 level methodology helps in achieving high accuracies. We believe that this approach is particularly promising because, Wikipedia is constantly growing and being updated. With it continuous growth we are gauranteed of the latest up to date information.

Part III
Recognizing Textual Entailment (RTE)

1 Introduction

Textual entailment recognition is the task of deciding, when given two text fragments, whether the meaning of one text (Hypothesis) is entailed from the other text (Text). The definition of entailment is based on (and assumes) common human understanding of language as well as common background knowledge.
We define textual entailment as a directional relationship between a pair of text fragments, which we call the Text (T) and the Hypothesis: (H). We say that:

T entails H, denoted by T -> H, if a human reading T would infer that H is most likely true.

For example, given assumed common background knowledge of the business news domain and the following text:

T1: Internet Media Company Yahoo Inc. announced Monday it is buying Overture Services Inc. in a 1.63-billion dollar (U.S.) cash-and-stock deal that will bolster its on-line search capabilities.

The following hypotheses are entailed:

- H1.1 Yahoo bought Overture
- H1.2 Overture was acquired by Yahoo
- H1.3 Overture was bought
- H1.4 Yahoo is an internet company

If H is not entailed by T, there are two possibilities:

1) H contradicts T

2) The information in H cannot be judged as TRUE on the basis of the information contained in T.

For example, the following hypotheses are contradicted by T1 above:

- H1.5 Overture bought Yahoo
- H1.6 Yahoo sold Overture

While the following ones cannot be judged on the basis of T1 above:

- H1.7 Yahoo manufactures cars
- H1.8 Overture shareholders will receive 4.75 dollar cash and 0.6108 Yahoo stock for each of their shares.

However, entailment must be judged considering the content of T AND COMMON KNOWLEDGE TOGETHER, and NEVER ON THE BASIS OF COMMON KNOWLEDGE ALONE.

2 Approach

This time in RTE-5 the average length of the text is made higher than is RTE-4. Texts will come from a variety of sources and will not be edited from their source documents. Thus, systems will be asked to handle real text that may include typographical errors and ungrammatical sentences. The sentence structure of text and hypothesis would rarely be similar. Hence, there is a need to reduce dramatically the complexity of required input sentences.

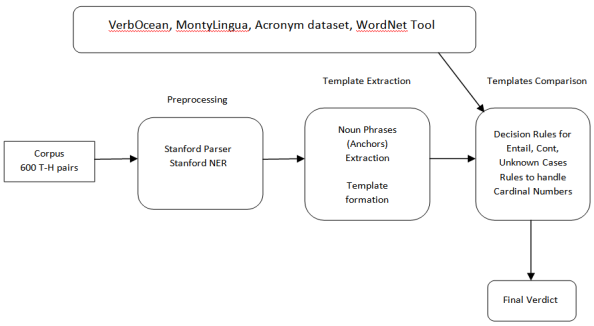

Fig. 3: *System Architecture*

The major idea of our approach is to find linguistic structures, here termed templates that share the same anchors. Anchors are lexical elements describing the context of a sentence. Templates that are extracted from different sentences (text and hypothesis) and connect the same anchors in these sentences are assumed to entail each other. For example the sentences 'Yahoo bought Overture' and 'Yahoo acquired Overture' share the anchors {X= Yahoo, Y = Overture}, suggesting that the templates X buy Y and X acquire Y entail each other.

In later subsections, a) the problem of finding matching anchors and b) identifying the template structure are addressed.

Apart from this major idea, initially the dataset is preprocessed to extract the dependency trees using Stanford parser, and similarity values between common-nouns using WordNet similarity tool are also preprocessed as the WordNet would take few seconds to load for every T-H pair and that would be costly if we want to make a quick run over the test set. Additionally, we added few rules and used semantic resources such as Stanford NER, VerbOcean (for comparing verbs alone), MontyLingua (to get the base form of a verb) and Acronyms dataset. Fig:3 best describes our system

3 Pre-processing

The corpus is first treated with the Stanford parser to generate dependency trees. We used the collapsed version of the dependency trees. To recognize the

Named Entities, the Stanford Named Entity Recognition tool has been used.

4 Template Extraction

A template is a dependency parse-tree fragment, with variable slots at some tree nodes e.g.,

X < –subj– prevent –obj–> Y

An entailment relation between two templates T1 and T2 holds if the meaning of T2 can be inferred from the meaning of T1. For example

T1: 'Aspirin reduces heart attack' can be inferred from

T2: 'Aspirin prevents a first heart attack' using the above template structure.

The anchor set here is {Aspirin, heart attack}

4.1 Anchor Set Extraction

In a sentence, the verb action is related to entities of subject and object. If we know the subject and object entities (or anchor sets), then we can say the event a sentence could probably be describing about. The goal of this phase is to find substantial number of promising anchor sets for each sentence. Usually these are the Noun Phrases of a sentence. A noun phrase can be a combination of common- nouns and proper nouns. If the proper nouns are once identified then the event or fact of the sentence can be judged.

Initially on the output of the Stanford parser, the Anchor Set Extraction algorithm is applied. Our algorithm would capture all Noun Phrases of the sentence as the anchor sets, with more relevance given to the named-entities together as a different anchor set. The named-entities can be identified by applying the Stanford NER tool upon the dataset.

4.2 Template Formation

The Template Extraction algorithm accepts as its input a list of anchor sets extracted from the ASE. Then, TE generates a set of syntactic templates from the dependency tree structure.

This is applied only on Hypothesis Data. And the templates extracted are searched in the text for availability.

For example:

T: Madonna has three children.

Output of ASE: {Madonna, three children}

Now, using the dependency tree, the connecting element for these anchor sets is extracted, in order to get a relation between each of the anchor sets.

After applying the TE algorithm

Output Template(s):

has=VBZ subject-*Madonna*=NNP object-*three*=CD *children*=NNS

The pos-tags are attached at the end, because it would help in the further modules to distinguish proper-nouns, common-nouns, cardinal numbers and verbs.

Similarly if there are more anchor sets, there would be more templates best describing the dependency between the anchor sets.

5 Template Comparison

After the templates from the Hypothesis are extracted, these templates are searched in the text information. For the text sentences, the ASE algorithm is applied and all the anchor sets are extracted.

Now, for the hypothesis anchor sets, Anchor Set Match algorithm is applied, where each of the anchor sets is made to match with the anchor sets of the text sentences. Here, immense care is taken while matching the anchor sets. For Named-Entities (like {China, Chinese}, {Malay, Malaysia}) special rules are designed to cover majority of such cases. For nouns WordNet similarity tool is used; as the values are calculated in the preprocessing stage, there wouldnt be much time lapse while comparison. The threshold value was initially 0.75. For cardinal numbers rules are mentioned below.

The output of this ASM algorithm would be the best matching anchor set in the text for every anchor set in the hypothesis. If any of the anchor set in the hypothesis doesnt found a matching anchor set, then the decision is said to be UNKNOWN.

With the anchor set matching, the text and hypothesis are believed to be describing roughly the same event or fact. Now their verbs or modifier heads have to match to decide whether they describe the exact same event or not.

The best matching anchor set from the text is retrieved. This text anchor set modifies the parent nodes in its dependency tree. All such nodes are extracted from the dependency tree. The Comparison Algorithm is applied to these nodes to compare with the modifier verb or noun of the corresponding hypothesis anchor set.

The comparison algorithm would first look for a direct match of verb or noun in the list of nodes. If not, it uses the *'WordNet'* similarity values for noun comparisons. Because WordNet doesnt have enough database for the verbs, the *'VerbOcean'* is used which has a huge collections of verbs and different combinational values. But the Verbocean would only have the base form of the verbs.

So, in order to convert the verbs to base form we used *'MontyLingua'* tool which would do the work. This algorithm would return if the event of the anchor sets matches or not.

6 Rules for Entailment Prediction

- When the event of an anchor set is identified, it is verified with the events of the other anchor sets. If all other anchor sets satisfy the same event, then the verdict is ENTAILMENT.
- The acronyms are expanded using the acronym database, so the acronyms are also matched with the expanded acronyms, and entailment is predicted accordingly. Ex: UK United Kingdom
- Rules for Cardinal Numbers Matching (Numeric Named Entities)
  - The basic rule would first resolve the Numeric Named Entities into numeric values.

    Ex: 24 = 24; twenty-six = 26; 1L = 100000; 50,000 = 50000; 45 thousand =45000
  - Next basic rule is to directly compare the numeric values of hypothesis and text.

    Then there would be quantification modifiers before or after the numeric entities.

    Ex: 'more than 100' in Hypothesis should match 'at least 110' in text.
  - The modification words are first considered and then the difference of the numeric values, to decide whether they match or not.

7 Rules for Contradiction Prediction

- The basic rule would be considering the negation words such as – 'not', 'n't', 'never' etc.
- In the VerbOcean database, the list of antonyms for each verb is hugely available. So, the antonyms can be detected using the VerbOcean database and the contradiction can be judged.
- In cases when the events of the anchor sets match, but the numerical values, the numbers may mismatch. Such cases are predicted as Contradiction.

  Consider the following example

  killed nsubj=*John* obj=*Lucy*

  killed nsubj=*Lucy* obj=*John*
- Here, the anchor sets have matched, and the event also are the same. But the roles are reversed. Such cases are declared as CONTRADICTION.

8 Rules for Unknown Prediction

- Whenever an anchor set from the hypothesis do not match any anchor set in the text, then it judge as UNKNOWN.
- When the numbers along with their associated noun phrases from the hypothesis cannot be mapped to any associated noun phrase in the text, and then it is judged as UNKNOWN.
- When a T-H pair doesn't satisfy any of the above rules of entailment or contradiction, then the system assumes it to be an UNKNOWN case.

9 Results and Conclusion

In TAC-2009, RTE-5 task has to predict the relation between 600 pairs of Text and Hypothesis. We submitted three runs for RTE 3-way task. In the first run, considering the comparison of nouns using the WordNet tool, the upper limit value is set to 0.75, which maintains a balance between generality and specificity. In case of VerbOcean there are two considerations, one are the verb synonyms with tag 'similar', and other are verb antonyms with tag 'opposite-of'. For verb synonyms the threshold value is set to 10.5 (max value found was 25 approx). For verb antonyms the threshold value is set to 12.

For the other two runs, experiments are carried out by varying the above mentioned threshold values.

The results we obtained for the 3 runs submitted for RTE 3-way task are presented in Table 9

	Run1	Run2	Run3
Accuracy	46.8	46.8	46.83

Table 7: *TAC 2009 results for 3 runs*

We obtained an accuracy of 60.66 for the RTE 2-way task (evaluation was done upon the gold standard set) with the threshold values set as mentioned in the above paragraph.

The confusion matrix for the 3-way task of run 3 with accuracy 46.83 is displayed in Table 8

	Entailment	Unknown	Contradiction	Total
Entailment	148	111	41	300
Unknown	48	122	40	210
Contradiction	36	43	11	90

Table 8: *Confusion matrix*

Confusion matrix for 3-way task run-3

We got an accuracy of 46.83 as our best result. This year, our approach was more concentrated upon extracting templates and comparing the anchor sets, which we think has covered more cases than the approach we followed previously.

10 Results and Analysis of Ablation tests

In the system we built, when two words are required to be analyzed for comparison, we have considered the following cases for the categorization of those words

1)Both the words are nouns
2)At least one of them is a verb
3)At least one of them is an acronym.

We used different tools for each of the cases mentioned above.
In the first case when the similarity of nouns has to be calculated, we used the similarity tool WordNet.
While in second case for the comparison of a verb with any other word, we used VerbOcean along with MontyLingua tool.
And for the third case we used the acronym data set available at the acronym-guide web page, which was suggested in the RTE knowledge resources.

10.1 Experimental Setting

For all the ablation tests the experimental settings was the same. And they were run upon the experimental settings of the run-3 of 3-way task. The threshold value for the similarity of nouns using WordNet tool is set as 0.75. And the threshold value was 10.5 for comparing the synonyms verbs and the antonym verbs using Verbocean.

10.2 Ablation Tests

10.2.1 Test-1

Tool Ablated: WordNet
In this ablation test, the similarity tool WordNet is ablated. If this tool is removed, then the two nouns are compared using only the string comparison techniques. Given two words, if they are similar to a certain string length (starting from the first letter), proportionate to their lengths, then they are said to be similar.
Results obtained for this test

2way accuracy = 0.6033; Change in 2way Accuracy = 0.01; 3 way accuracy = 0.47; Change in 3way accuracy = 0.0017

10.2.2 Test-2

Tool Ablated: MontyLingua
When a verb needs to be compared with another word, then we used VerbOcean. But, the VerbOcean has all the verbs in their base form. So, there is a need to convert the verb into its base form. For that purpose we used MontyLingua tool. When this is removed, the verb is checked in the VerbOcean in its original form. So, in this ablation test, we removed the tool and tested with the same experimental setting. We obtained zero change in the accuracies of 2-way and 3-way task. Accuracy for 2-way task was 0.6067 and the accuracy for 3-way was 0.4683.

10.2.3 Test-3

Tool ablated: VerbOcean
Here the verb comparison tool VerbOcean is removed. When this is done, those words are compared based on string comparison techniques mentioned in the case of wordnet ablation test. Here too, we obtained zero change in the accuracies of 2-way and 3-way task. Their accuracies remained the same.

10.2.4 Test-4

Tool ablated: Acronyms set
The acronyms data set downloaded from the acronym-guide is removed in this ablation test. Whenever two words are compared, it is checked if any of those words is an acronym. If so, its expanded form is retrieved from the acronym set if available, then they are compared. If a word is embedded in the expanded set of words, it is said to be similar. So, if this is removed, the acronyms are compared as such, after removing the punctuations. There was no change in the accuracies of 2-way or 3-way task in this ablation test.

References

[1] L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval.

[2] S. Deorowicz and M. Ciura. Correcting spelling errors by modelling their causes. *International journal of applied mathematics and computer science*, 15(2):275, 2005.

[3] H. Edmundson. New methods in automatic extracting. pages 268–285. In Journal of ACM, Volume 16, 1969.

[4] S. Fisher and B. Roark. Query-focused summarization by supervised sentence ranking and skewed word distributions. In *In proceedings of DUC 2006*. DUC, 2006.

[5] S. R. Gunn. Support vector machines for classification and regression. May 1998.

[6] R. He, Y. Liu, B. Qin, T. Liu, and S. Li. Hitirs update summary at tac2008:extractive content selection for language independence. In *TAC 2008 Proceedings*. Text analysis conference, December 2008.

[7] J. Jagarlamudi, P. Pingali, and V. Varma. Query independent sentence scoring approach to duc 2006. In *In proceedings of DUC 2006*. DUC, 2006.

[8] R. Katragadda, P. Pingali, and V. Varma. Sentence position revisited: A robust light-weight update summarization baseline algorithm. In *Proceedings of the Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies (CLIAWS3)*, pages 46–52, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[9] J. kupeic, J. pedersen, and F. chen. A trainable document summarizer. In *In proceedings of ACM SIGIR 95*, pages 68–73. ACM, 1995.

[10] S. Li, Y. Ouyang, W. Wang, and B. Sun. Multi-document summarization using support vector regression. In *DUC 2007 notebook, 2007*. Document Understanding Conference, November 2007.

[11] Lin and Chin-Yew. Looking for a few good metrics: Automatic summarization evaluation - how many samples are enough? In Proceedings of the NTCIR Workshop 4, June 2004.

[12] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996.

[13] J. M.Conro, J. D. schlesinger, J. Goldstein, and D. P.O'leary. Left-brain/right-brain multi-document summarization. In *In proceedings of DUC 2004*, 2004.

[14] R. Mihalcea. Using wikipedia for automatic word sense disambiguation. In *Proceedings of NAACL HLT*, volume 2007, 2007.

[15] A. Nenkova, R. Passonneau, and K. McKeown. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2):4, 2007.

[16] F. schilder and R. Kondadandi. Fastsum: fast and accurate query-based multi-document summarization. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*. Human Language Technology Conference, 2008.

[17] D. shen, J.-T. sun, H. Li, Q. yang, and Zheng. Document summarization using conditional random fields. In *In proceedings of IJCAI'07*, pages 2862–2867. IJCAI, 2007.

[18] T. Zesch, I. Gurevych, and M. Muhlha user. Analyzing and accessing Wikipedia as a lexical semantic resource. *Data Structures for Linguistic Resources and Applications*, pages 197–205, 2007.

[19] J. Zhang, X. Cheng, H. Xu, X. Wang, and Y. Zeng. Summarizing dynamic information with signature terms based content filtering. In *TAC 2008 Proceedings*. Text analysis conference, December 2008.

[20] ziheng Lin, H. H. Hoang, L. Qiu, S. Ye, and M.-Y. Ka. Nus at tac 2008:augmenting timestamped graphs with event information and selectively expanding opinion contents. In *TAC 2008 Proceedings*. Text analysis conference, December 2008.