Final Year Project

B.Sc. CS&IT

Development of a System for Summarization of Multi-Party Dialogue; An Exploration of Unsupervised Automatic Keyphrase Extraction Techniques

Author:
Michael Neely

Supervisor:
Dr. Colm O'Riordan

April 17th, 2018

Abstract

Summarization and keyword extraction of text documents is a common Natural Language Processing task with increasingly relevant applications to computing, academic, and social domains. The majority of previous work is focused on structured textual sources, such as articles, books, and scientific papers. This project introduces the motivation for a focus on the more complex effort of summarizing unstructured multi-party dialogue. Three standard algorithms, along with a proposed novel technique, are presented and evaluated on a custom dataset of conversation transcripts. Results, future work and potential applications in this domain are discussed in detail.

Keywords

Keyphrase Extraction, Automatic Keyphrase Extraction (AKE), Unsupervised Automatic Keyphrase Extraction, Natural Language Processing (NLP), Information Retrieval (IR), Information Storage, Summarization, Conversations

Acknowledgements

First and foremost I would like to thank my supervisor, Dr. Colm O'Riordan, for his tireless support, patience, and understanding during what can only be described as a rather chaotic year. I've yet to meet a more passionate and genuine scholar. It was an honor to work with him, and I wish him all the best.

Secondly, I would like to thank everyone who contributed to the development and evaluation of the test collection: Maire Hogan, Matthew Ferson, Ken and Dorothy McGowan, and Barbara, Gene, and Kelly Neely. These individuals sacrificed their time to craft summaries for each of the conversations in the corpus and subjectively evaluate the performance of the various keyphrase extraction methods.

Third, to my family. For their continued and unconditional love, support, and guidance.

Additional thanks go to the folks in the Natural Language Processing Department at Stanford University, for making their incredible CoreNLP Server available to everyone via the GNU General Public License[1].

Last, I would like to thank everyone who made my time as undergraduate in Ireland the most incredible experience of my life. I have repeatedly been awed by the generosity, kindness, patience, openness, and welcoming nature of the Irish people. I'm honored to have met so many wonderful individuals. You've made it very hard to leave.

This project is dedicated to Máire Hogan, Matthew Ferson, Ross Daly, Oisin Curran, Chris Cornwall, Joey Maloney, Hasan Syed, Aisling Fitzsimmons, Shaun Ward, Ciaran Martin, Fergal Larkin, Eamon, Bernadette, and Eoin Hogan, James, Angela, and Aisling Neylon, Gary McMahon, Brian Casey, Mark McConn, Conor Proctor, Diarmaid Ryan, and David Daly.

[1]`https://www.gnu.org/licenses/gpl.html`

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivations

The dawn of 'Big Data' and the cross-industry fervor generated by Gartner analyst Doug Laney's highly influential 3V's framework [1] has focused attention on the process of transforming large, unstructured datasets into concise, meaningful information. As computing power and storage improve, it is frequently possible to collect and explore previously untapped sources of raw data such as sensors, personal devices, and server logs. These data stores often contain valuable insights that are difficult to extract. As respected Computer Scientist and Machine Learning pioneer Ian Witten accurately states: "There is a growing gap between the generation of data and our understanding of it" [2]. Thus, the difficulty in reducing this gap lies not only in the processing and analysis necessary to turn abstract data into concrete knowledge but also in the efficiency of the techniques employed.

Textual data is arguably the most tedious and lengthy form of data to sift through. There is certainly no shortage of digital text available for consumption on the Internet. Readers are spoiled for choice in a world where over 80 million WordPress blog entries[2], 20 million peer-reviewed academic articles[3], and a collective 11,000 news stories from the Wall Street Journal and New York Times [4] are published online each month. With a veritable sea of content to sift through, the modern reader has a rapidly reducing amount of time available to spend surmising the relevancy of documents. This trend has led to the emergence of authors ocassionally including "too long; didn't read" (TLDR) sections in their works, an idea that naturally corresponds to the abstracts that preface academic papers. When these useful summaries are not available for a large majority of documents, the question emerges: is it possible to develop intelligent systems capable of automating this task?

Research primarily focuses on formulating models that infer or extract the key points necessary to build sufficiently descriptive summaries of structured textual sources. Books, academic papers, newspaper articles, and even broadcast recordings tend to have patterns that can be exploited to reduce analytical complexity. These sources have standard layouts with, clear boundaries, and typically deal with a small number of topics or ideas. Thus, models in

[2]https://wordpress.com/activity/posting/

[3]https://www.stm-assoc.org/2015_02_20_STM_Report_2015.pdf

[4]https://www.theatlantic.com/technology/archive/2016/05/how-many-stories-do-newspapers-publish-per-day/483845/

these areas have become extremely proficient. Recently, attempts have been made to summarize semi-structured sources like blog posts and emails, which deviate from formal written language, yet still contain exploitable patterns. However, the domain of unstructured text sources, like social media posts and comments, forum discussions, podcast transcripts, text messages, and conversations, remains relatively unexplored. The sparsity of knowledge in this area warrants addressing.

Summarization of spontaneous dialog has various academic, social, and personal applications. Humans engage in a multitude of conversations with different parties throughout the day, discussing information such as names, places, appointments, tasks, dates, and times. The bulk of this information tends to be forgotten soon after the meeting is ended. Concise summaries of discussions impart clarity by capturing essential ideas and objectives and removing unnecessary conversational clutter.

1.2 Research Questions

This project aims to at least partially answer the following questions:

- What are the domain characteristics of typical conversations? What makes their summarization a difficult task?
- Will any existing algorithms or strategies prove useful in the domain?
- Are there any existing corpus's on which to build and test the model?
- What metrics can be used to evaluate the generated summaries?
- What are the possible applications of these models?

1.3 Hypothesis

An algorithm that combines existing Natural Language Processing and Information Retrieval techniques, along with domain-specific heuristics, will produce a reasonably effective model for summarizing single or multi-party dialog.

1.4 Goals and Aims

This project seeks to:

- Explore the unique domain characteristics of conversations, as well as the challenges associated with summarizing them.
- Develop a unique corpus of multi-topic conversations.
- Procure 'gold standard' summarizations of various lengths from human annotators.
- Implement algorithms that use existing Information Retrieval techniques.
- Develop an interactive application for exploring the knowledge base.
- Develop a summarization algorithm that combines existing techniques and domain-specific heuristics.
- Evaluate the performance of the algorithms against the test collection with different objective and subjective measures.
- Prove that algorithms that consider noun phrase chunks instead of individuals words exhibit better performance.
- Offer the beginnings of a platform for real-world use.

1.5 Contributions

This project suggests a novel algorithm for the automatic ranked keyphrase extraction of conversations, as well as the beginning of an interactive web application that offers personalized keyphrase extraction from submitted text.

1.6 Layout of Report

Chapter 1 introduced the reader to the domain and the author's motivations for contributing original research. Chapter 2 discusses important concepts as well as previous academic work necessary for the understanding of the all aspects of the project. Chapter 3 explains the high-level design of the algorithms and system developed and Chapter 4 discusses the technical details and implementations. In Chapter 5, the models are objectively and subjectively evaluated over several experiments, the results of which will are analyzed and discussed in detail. Finally, in Chapter 6, the report concludes with a discussion of future work and potential applications of the models and the author's personal reflections.

2 Background

2.1 Document Summarization Approaches

Document summarization approaches can be divided into two categories:

- **Extraction-Based** techniques, which build summaries using only words, sentences, or paragraphs contained in the document.
- **Abstraction-Based** techniques, which uses Natural Language Processing to generate a summary that emulates human tendencies.

Consider the following excerpt from Wikipedia[5]:

> The earliest credible evidence of coffee-drinking or knowledge of the coffee tree appears in the middle of the 15th century in the accounts of Ahmed al-Ghaffar in Yemen. It was here in Arabia that coffee seeds were first roasted and brewed, in a similar way to how it is now prepared. Coffee was used by Sufi circles to stay awake for their religious rituals. Accounts differ on the origin of coffee seeds prior to its appearance in Yemen.

Figure 1: A sample document on "Coffee" from Wikipedia

Extraction-Based techniques might select 'coffee', 'seeds', and 'Yemen' as important terms, whereas a human might choose a more abstract phrase such as 'The history of coffee'. This would be extraordinarily difficult for an Abstract-Based technique to capture because it requires a deep understanding of the text. Due to this difficulty, Extraction-Based techniques have received considerably more attention from the research community. Most methods focus on extracting keyphrases as summaries from documents instead of longer segments, because of their additional utility in categorization, indexing, clustering, and search.

The task of extracting keyphrases is known as Automatic Keyphrase Extraction (AKE). AKE techniques are classified as supervised or unsupervised [3]. Supervised AKE is presented as a traditional Machine Learning (ML) classification task, which can be beneficial, as it does not require an in-depth understanding of the NLP domain. However, as with most classification tasks, it requires the development of a manually annotated dataset for training. Unsupervised AKE offers the attraction of eliminating this need entirely, by attempting to develop rules or heuristics that capture how humans assign importance to certain terms in documents.

[5] `https://en.wikipedia.org/wiki/Coffee`

2.2 On Keyphrases vs Key Terms

It is worth noting that AKE is defined as the process of extracting **keyphrases** as opposed to keywords. Most human-assigned keyphrases are classified as, or contained within, noun phrases [4] [5] [6]. Thus it is appropriate to consider n-grams (a sequence of n terms) as keyphrases instead of individual words.

Alrehamy and Walker define n-gram keyphrases as 'candidate terms', if they match the following patterns[7]:

Table 1: Candidate term POS extraction patterns with examples, as defined by Alrehamy and Walker

Pattern	Example
N = (NN|NNS)	coffee/NN, computer/NN, printer/NN
C = (JJ)∗(NN|NNS)+	new/JJ computer/NN, printer/NN cartridge/NN
E = (NNP|NNPS) ∗ (S)∗ (NNP|NNPS)+	Galway/NNP, Sumatran/NNP Coffee/NN, University/NNP of/IN Galway/NNP

Here, ***N*** denotes a single *Noun*, ***C*** denotes a *Compound Noun*, a sequence of nouns optionally starting with an adjective, and ***E*** denotes an *Entity*, a sequence of proper nouns with up to one stop word (e.g. *of*, *and*, *the*, etc. . .) in the middle.

Considering candidate terms instead of individual words should yield superior performance. If conversation participants frequently discuss attending the 'University of Limerick' for example, it is desirable for the algorithm to consider the entity as a single keyphrase and avoid populating the list of terms to consider with both 'University' and 'Limerick'.

2.3 Existing Unsupervised AKE Techniques

Unsupervised AKE techniques normally employ either statistical, graph-based, or clustering approaches. Graph-based techniques generally model documents as a collection of keyphrase nodes connected by weighted edges that capture the relationships between them. Some ranking function then determines the top-scoring candidate terms. Previously proposed graph-based models include Litvak's application [8] of Kleinberg's [9] Hubs and Authorities (HITS) algorithm, various extensions of Page et al.'s PageRank algorithm [10] such as Mihalcea and Tarau's TextRank which weights edged based on their 'semantic relatedness' [11], and Tsatsaronis, Varlamis, and Nørvag's high performance Semantic Rank [12].Clustering techniques group document terms into distinct clusters based on their "lexical, semantic, and statistical information" [7]. Bracewell et al. [13] presents an algorithm which groups similar noun phrases into clusters. Liu et al. [14] propose KeyCluster, a model which extracts and clusters single nouns using similarity measures and external knowledge sources like Wikipedia and then selects keyphrases that contain centroid nouns. Alrehamy and Walker note that KeyCluster "outperforms many prominent AKE methods but suffers several practical drawbacks", namely that "clustering-based methods, in general, cannot guarantee that all generated clusters are important in representing the document theme, and selecting the centroid of an unimportant cluster as a heuristic to identify and extract keyphrases leads to inappropriate keyphrases" [7]. They introduce their own algorithm, SemCluster, which uses Wordnet [15] as the core ontology supplemented by external knowledge sources like DB-Pedia [6] to capture the candidate term synsets for similarity computation and clustering. Statistical approaches include those introduced by Steir and Belew [16] and Barker and Cornacchia [17] which select bigrams and noun phrases as keyphrases, respectively. Statistical attempts to model the topics contained within a document include TextTiling, [18], Latent Dirichlet Allocation (LDA) [19], and Pachinko Allocation (PAM), an improvement on LDA that additionally captures correlations between topics [20]. This report focuses primarily on leveraging the well-tested $tf - idf$ frequency statistic technique explained in the section below, an approach used in 83% of digital library text-recommender systems [21].

[6]`http://www.dbpedia.org`

2.4 Towards a Robust Term Weighting Scheme

Interest in term indexing and retrieval has a rich history spanning more than half a century. The question of how to best model the problem was first answered by Luhn in an influential 1957 paper [22], where he suggested to represent documents as vectors of individual terms.

Thus an Information Retrieval system, S, consists of a collection of documents, D, where each document d is represented as a vector of the terms:

$$S = \sum_{i=1}^{n} d_i =< t_1, t_2, \ldots, t_j > \tag{1}$$

Information requests, or queries, from users are also represented in this vector form:

$$Q =< t_1, t_2, \ldots . t_j > \tag{2}$$

Distinction among terms is achieved by introducing a weighting scheme, where each term is assigned a weight indicative of its importance to the information need of the user interacting with the system. The simple boolean weighting scheme suggested by Luhn assigns binary, discrete weights to terms. Terms are simply absent (0), or present (1), based on the query.

The query-document similarity equation selects relevant documents from the collection via a pairwise comparison between the query vector and each document vector.

$$sim(d_j, q) = \sum_{i=1}^{n} w_{q_i} \cdot w_{d_i} \tag{3}$$

This model additionally supports boolean expression queries, which more naturally correspond with how users tend to view information requests. Documents are deemed relevant if they satisfy any components obtained by mapping the expression to disjunctive normal form.

$$q = t_1 \wedge (t_2 \vee (\neg t_3)) = 100 \vee 110 \vee 111$$

The so-called ‘Boolean Model’ is formal, correct, and easy to implement. However, it suffers from a serious shortcoming. Since documents are either relevant or irrelevant, there is no ranking of results. Users need to formulate very specific queries to more accurately filter the collection, thus making interactions with the system laborious. Imagine if the Google search engine used the Boolean Model. A simple query for ‘cat AND NOT dog’ could return billions of unranked results, encompassing every webpage on the internet that contains the term ‘cat’ without the term ‘dog’.

Salton et. al., introduced the Vector Space Model in 1975 as an improvement, by removing the limitation on binary weights in both the document and query representations. [23] In this model, documents can be ranked based on their similarity to the query by obtaining the cosine of the angle between the two vectors.

$$sim(d_j, q) = cos(d_j, q) = \frac{d_j \bullet q}{|d_j||q|} = \frac{\sum_{i=1}^{n}(w_{i,j} \times w_{i,q})}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,q}^2}} \quad (4)$$

This model, along with all information retrieval systems, depends on an accurate weighting system. In their 1988 review of term-weighting approaches, Salton and Buckley define the the development of such a system as the processing of answering the question: “is the determination of the term weights capable of distinguishing the important terms from those less crucial for content identification?” [24].

Previous works posited the existence of two separate factors that contribute to a term’s weight. Luhn suggested term-frequency (tf) as a natural metric, arguing “There is also the probability that the more frequently a notion and combination of notions occur, the more importance the author attaches to them as reflecting the essence of his overall idea” [22]. Yet, frequency of a term does not necessarily correspond to importance. The English language contains a high number of semantically meaningless words, such as “the”, “of”, “and”, etc. . . , that occur in documents with high frequency. Luhn is credited with coining the term ‘stop word’ to refer to these terms after his 1959 presentation [25], even though he did not use the phrase directly. Thus, any weighting scheme solely based on term frequency is subject to noise from terms that do not relate to the meaning of the document. To solve this problem, Spärck Jones argued a statistical interpretation of a term’s specificity should also be considered, claiming “we should treat matches on non-frequent terms as more valuable than ones on frequent terms, without disregarding the latter altogether” [26]. She defined the specificity of a term as “an inverse function of the number of documents in which it occurs” [26]. This concept is called inverse-document-frequency (*idf*), and measures the rarity of a term is across a document collection. Spärck Jones justifies *idf* by connecting it to Zipf’s law [27], though the argument is merely probabilistic. Researchers such as Robertson have attempted to connect the measure to information theory [28], but it is still considered to only be a heuristic.

$$idf(t_d) = \log\left(\frac{N}{N_t}\right) \tag{5}$$

Combining these two metrics into term-frequency-inverse-document-frequency ($tf - idf$) captures both how important a term is to any particular document as well as how important the term is across the collection. Most measures of $tf - idf$ perform scale term frequency in a sub-linear manor via logarithmic normalization. This is only natural: the nth + 1 occur of a term is less important than the nth occurrence.

Thus we arrive on a final formula to rank the importance of individual terms in a document:

$$w_{t,d} = (1 + \log f_{t,d}) \cdot \log\left(\frac{N}{N_t}\right) \tag{6}$$

2.5 On Document Pre-processing

A certain degree of document preprocessing is required to prevent noisy terms from damaging performance. Information Retrieval systems commonly employ stemming algorithms, such as those introduced by Lovins in 1968 [29] and Porter in 1980 [30]. Stemming is the process of reducing inflected words to their root form, and is primarily used to allow matches between documents and query terms that share the same root. As Manning, Raghavan, and Schütze discuss, "For grammatical reasons, documents are going to use different forms of a word...In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set." Yet they are quick to argue that stemming is a "rather crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time" [31]. Stemming increases recall at the risk of damaging precision. Manning, Raghavan, and Schütze highlight the problematic nature of stemming with a simple example, noting "the Porter stemmer stems all of the following words: *operate operating operates operation operative operatives operational* to *oper*. However, since *operate* in its various forms is a common verb, we would expect to lose considerable precision on queries such as the following with Porter stemming"[31]:

- operational and research
- operating and system
- operative and dentistry

Natural Language Processing offers a more advanced process known as *lemmatization* which reduces words to their canonical (dictionary) form by analyzing their context from information such as intended parts of speech, and position within the enclosing sentence, neighboring sentences, and the entire document.

Sample Text: The traditional method of planting coffee is to place 20 seeds in each hole at the beginning of the rainy season. This method loses about 50% of the seeds' potential, as about half fail to sprout.

Porter Stemmer: The tradit method of plant coffe is to place 20 seed in each hole at the begin of the raini season. Thi method lose about 50% of the seed potenti, as about half fail to sprout.

Lemmatization: The traditional method of plant coffee be to place to place 20 seed in each hole at the beginning of the rainy season. This method lose about about 50% of the seed potential, as about half fail to sprout.

Figure 2: A comparison of stemming and lemmatization on sample text from Wikipedia

2.6 On Document Structure

Some models represent documents as an n-gram 'bag of words' [32], where keyphrase frequency is captured, but order and positioning are disregarded. In structured documents, this information can be leveraged to improve AKE performance. In 2010, the 5th International Workshop on Semantic Evaluation held a competition where researchers developed algorithms to extract keyphrases from a corpus of articles from the Association for Computer Machinery's Digital Library (ACM DL) [33]. The HUMB system developed by Lopez and Romary won by exploiting the structure of the articles, preferentially weighting terms in the important sections like the abstract and conclusion [34]. In contrast to scientific articles, conversations tend to be unstructured: there is no guarantee participants will begin or end the dialog with the most important terms. Thus, the algorithms used in this report will not consider positional information for term weighting.

2.7 Evaluation Metrics

An AKE algorithms performance must be evaluated to determine its effectiveness. Evaluation measures for document summarization can be classified as being **intrinsic** (related to the correctness of the summary) or **extrinsic** (related to the application of the summary). Steinberger and Ježek categorize intrinsic metrics as evaluating either the content or the text quality of the generated summary [35]. Because AKE merely selects phrases instead of building plain-English summaries, only content evaluation metrics are appropriate. Popular content evaluation metrics include content-based metrics such as ROUGE [36], cosine similarity, unit overlap, longest common subsequence (LCS), and Landauer, Foltz and Laham's Latent Semantic Analysis (LSA) [37], as well as co-selection metrics like precision, recall, and F-measure.

2.8 Challenges

Spoken conversation bares little resemblance to the mostly standardized conventions of written text. Topics shift regularly and typically build on previous discussions. Grammatical rules tend to be loosely observed, with liberal use of slang and new words and phrases that are often unrecognized by lexical databases. The problematic characteristics of the domain are defined as:

- Unclear Boundaries - It is often unclear when a sentence or paragraph starts or ends.
- Fragmentation - Discussions on one topic are often interspersed with discussions on another.
- Temporality - the meaning of the discussion is dependent on the time/place.
- Topicality - words or concepts may be new or not previously encountered.

Any accurate AKE algorithm must incorporate strategies and heuristics that factor in these issues. As discussed in the subsection on Document Structure, a bag of words model will need to be adopted as it is not possible to rely on document or even sentence structure in a conversation transcript. There are also issues with generating and comparing summaries. Document summarization is a subjective and personal task. Ask multiple individuals to draft one or two sentences that capture the essence of some text, and it is highly likely that the summaries will differ immensely. The number of challenges surrounding the domain of conversation AKE indicates that achieving high-performance levels and accurate evaluations will be difficult.

2.9 Stanford CoreNLP

This project leverages the performance of Stanford's CoreNLP software [38] to perform common NLP tasks [7]. The algorithms presented will explain the individual annotators available where appropriate.

2.10 Summary

This section introduced the reader to the task of Automatic Keyphrase Extraction, detailing the rich academic history leading to present algorithms, as well as listing metrics for evaluating generated summaries and highlighting several important challenges that make Automatic Keyphrase Extraction of conversations a difficult task.

3 Experimental Design

Four different algorithms were designed to evaluate the initial hypothesis, inspired by previous AKE approaches presented in Section 2.

3.1 Workflow

The basic workflow for an AKE algorithm is as follows:
Input: A conversation transcript

1. Tokenization and Sentence Splitting [8] - split the transcript into individual sentences with individual word tokens.
2. Pre-processing and Stopword Removal - perform additional pre-processing as required, and remove words present in a curated list of stop words if needed.
3. Term or Phrase Ranking - rank phrases or terms according to a weighting.
4. Additional Processing - perform additional processing as required.
5. Storage - store the document future calculations.

Output: Ranked list of keyphrases that capture the meaning of the conversation

[7] `https://stanfordnlp.github.io/CoreNLP/`

[8] The tokenization and sentence splitting tasks are performed with CoreNLP's TokenizerAnnotator and WordsToSentenceAnnotator, respectively.

3.2 Algorithm 1 - Lemma TF-IDF

The first algorithm serves primarily as a benchmark and uses the classic logarithmic normalization $tf - idf$ weighting scheme described in Section 2.4 to rank individual words in the conversation. Lemmatization (via CoreNLP's MorphaAnnotator [9] and stopword removal are performed during the pre-processing step to reduce noise and increase performance.

3.3 Algorithm 2 - Candidate Term TF-IDF

The second algorithm uses the same $tf-idf$ weighting scheme, but introduces a part-of-speech tagging pre-processing step to identify 'candidate terms' (noun-phrases) as defined by Alrehamy and Walker [7]. POS tagging is done using CoreNLP's POSTaggerAnnotator [10] which is derived from Toutanova and Manning's work on log-linear part-of-speech taggers, presented in [39] and [40]. $Tf - idf$ calculations are performed on candidate terms instead of individual words to generate more accurate keyphrases.

3.4 Algorithm 3 - Candidate Term Latent Dirichlet Allocation

An improvement on the initial LDA algorithm proposed by Blei et al. [19] was introduced by Griffiths and Steyvers, who implemented an alternate inference of the LDA approach using collapsed Gibbs Sampling [41]. Qiu et. al, argue that "though the collapsed Gibbs sampling approach is in principle more accurate, it has high computational complexity, which makes it inefficient on large datasets." [42]. Accuracy is more important than efficiency for this project, so this variant is considered. It has been observed by the author that without a sufficiently large corpus, the performance of the collapsed Gibbs Sampling variant is significantly higher when only considering the document intended for summarization and disregarding additional corpus document vectors. This LDA technique is included as an algorithm due to its potential to select keyphrases that may not rank highly in $tf - idf$ based approaches, but are still indicative of conversation topics. As with the second algorithm, the vocabulary is defined by candidate terms instead of individual words.

[9] `https://stanfordnlp.github.io/CoreNLP/lemma.html`
[10] `https://stanfordnlp.github.io/CoreNLP/pos.html`

3.5 Algorithm 4 - *Candidate Term and Named Entity Weighted Semantic Evaluation*

The final algorithm leverages domain knowledge and Natural Language Processing tasks to generate a ranked list of keyphrases.

3.5.1 Named Entity Recognition

Certain textual strings — such as people, location, organizations, dates, and times — represent abstract or physical objects with special semantic importance. These are referred to as *Named Entities*, as introduced by [43]. *Named Entity Recognition* (NER) is the NLP task concerned with the location and classification of these entities within text.

The noun-phrase candidate term extraction method can inherently capture some named entities that follow one of the three phrase patterns, but this is typically a small subset of the total entities contained in the text.

(1) The earliest credible evidence of coffee-drinking or knowledge of the coffee tree appears in the middle of the **15th century (DATE)** in the accounts of **Ahmed al-Ghaffar (NAME)** in **Yemen (COUNTRY)**. It was here in **Arabia (COUNTRY)** that coffee seeds were **first (ORDINAL)** roasted and brewed, in a similar way to how it is now prepared. Coffee was used by **Sufi (RELIGION)** circles to stay awake for their religious rituals. Accounts differ on the origin of coffee seeds prior to its appearance in **Yemen (COUNTRY)**.

(2) The earliest **credible evidence(C)** of **coffee-drinking(N)** or **knowledge(N)** of the **coffee tree(N)** appears in the **middle(N)** of the **15th century(C)** in the **accounts(N)** of **Ahmed al-Ghaffar (E)** in **Yemen (E)**. It was here in **Arabia(E)** that **coffee seeds(N)** were first roasted and brewed, in a **similar way(C)** to how it is now prepared. **Coffee(N)** was used by **Sufi circles(C)** to stay awake for their **religious rituals(C)**. **Accounts(E)** differ on the **origin(N)** of **coffee seeds(N)** prior to its **appearance(N)** in **Yemen(E)**.

Figure 3: A comparison between named entities and candidate terms extracted from a passage of text.

Named entities are clearly important to the meaning of a passage of text or conversation, so their accurate extraction is desired. It is observed that over

20% of the human-defined keyphrases from the test collection are named entities.

This algorithm uses the CoreNLP NERClassifierCombiner annotator [11], which adds distribution similarity features to a Viterbi [44] and local Conditional Random Field (CRF) model, as presented in [45]. The following table shows all entity types recognized by the the software, with the subset considered by this algorithm highlighted.

Table 2: CoreNLP named entity recognition labels.

PERSON	LOCATION	ORGANIZATION	MISC
COUNTRY	NATIONALITY	STATE_OR_PROV	TITLE
IDEALOGY	RELIGION	CRIMINAL_CHARGE	CAUSE_OF_DEATH
MONEY	NUMBER	ORDINAL	PERCENT
DATE	TIME	DURATION	SET

Temporal and numerical entities are useful in question-and-answering type IR systems, where users can query for specific information, such as: “What do I need to have done by this Friday?”, or “Do I have any appointments at 2pm?”. However, they typically do not occur in human-provided summaries of text. Only 2% of manually assigned keyphrases were classified as numerical or temporal entities, 80% of which came from a single human annotator. For this reason, only the subset of named entity tags that correspond to names, places, or events are used.

3.5.2 Algorithm Overview

The algorithm only considers a set of terms obtained by the union of the extracted candidate terms and named entities. To increase keyphrase topic coverage, near-duplicate phrases are removed by calculating the Sørensen–Dice coefficient of each tuple of the considered terms. This metric is defined as the “ratio of the number of shared character bigrams to the total number of bigrams in both words” [46]. When a pair of strings exceeds the defined similarity threshold, the longer of the two is removed.

[11] https://stanfordnlp.github.io/CoreNLP/ner.html

As explained in the Challenges section, conversations tend to contain a high frequency of terms that do not contribute to an accurate summary. Examples include slang and informational shortening of words, such as “lotta” (meaning “lot of”). Many of these terms end up polluting the set of candidate terms. Consider the following POS tagged sentence from one of the test transcripts:

“Uh//**VB** a //**DT** lotta //**NN** people //**NNS** not//**RB** thrilled //**VBN** to //**TO** death //**NN** with //**IN** this //**DT** study //**NN** uh //**VB** a //**DT** lotta //**NN** people //**NNS** also //**RB** supporting //**VBG** it /**PRP**”

The phrase “lotta people” is considered as a compound noun (C), and appears frequently throughout the rest of the conversation; giving it a high rank under a pure $tf - idf$ weighting scheme. A custom ***term specificity*** metric is used to reduce the rank of these types of phrases based on the hypothesis: *“A specific term or phrase will contain a limited number of relevant articles in Wikipedia”*. Consider the term *‘London’*, which seems pretty specific at first glance. Yet, Wikipedia contains over 50 articles that match the term [12], covering a diverse range of topics like musical albums, airports, films, and other global cities bearing the same name.

By querying the external knowledge source DBpedia[13] for each phrase in the set of filtered keyphrases we can define a specificity score s for a keyphrase kp based on the ratio of relevant articles returned from the entire collection, A:

$$s_{kp} = \frac{\mid a_{kp} \mid - l}{l} \tag{7}$$

Where $a_{kp} \subset A$, and some $l < \mid A \mid$ is a limit on the total number of articles that can be returned.
Solely considering this metric would lead to poor results. If a conversation was about one of the disambiguations of the term London, for example, the term would have such a low specificity score that it would not place highly in the ranking used to determine keyphrases. This score can be combined with the classic logarithm-normalized $tf - idf$ weighting scheme used in previous algorithms to reach a final formula for keyphrase weight:

$$w_{kp,d} = \gamma \cdot ((1 + \log f_{kp,d}) \cdot \log(\frac{N}{N_{kp}})) + (1 - \gamma) \cdot (\frac{\mid a_{kp} \mid - l}{l}) \tag{8}$$

[12]`https://en.wikipedia.org/wiki/London_(disambiguation)`
[13]`http://www.dbpedia.org`

Where $0 < \gamma < 1$ is some weighting factor to favor either metric if desired.

The previous hypothesis can be modified to the following:

A keyphrase is highly descriptive of a document if it is semantically specific and occurs with a high frequency in the document and low frequency across the collection.

Figure 4: The final hypothesis for the proposed *Candidate Term and Named Entity Weighted Semantic Evaluation* algorithm

4 Technical Implementation

4.1 Overview

The system was implemented as a 3-tier architecture Node.js [14] web application. The code was written in TypeScript [15], a typed superset of JavaScript, using new ECMAScript 6[16] features.

Node.js uses an event-driven and non-blocking (asynchronous) Input/Output (I/O) model. Blocking operations — such as file system I/O, database I/O, and network access — are isolated and assigned to available worker threads through callback registration. The main loop is free to continue processing other requests while the operation is processed by the worker thread. The callback is executed when the process is complete, notifying the loop that the result is available for the next processing step. Coordination of a complex sequence of tasks is achieved through the new ***await*** command. This unique architecture means that a server running Node.js can handle hundreds, if not thousands more requests simultaneously on machines with little memory and processing resources, compared to other web-servers such as Apache[17]

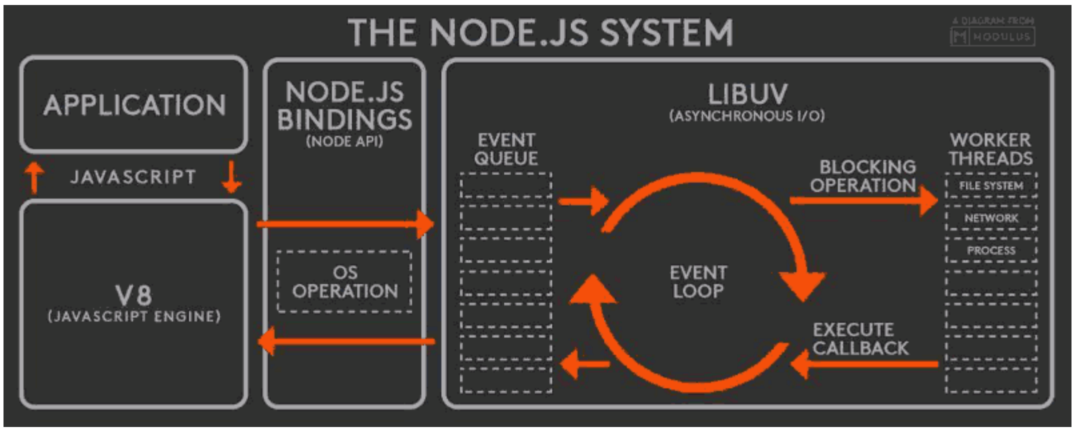


Figure 5: The Node.js system architecture

Source: User-created diagram at `https://stackoverflow.com/questions/21596172/`

[14]`https://nodejs.org/en/`
[15]`https://www.typescriptlang.org/`
[16]`http://es6-features.org/`
[17]`https://httpd.apache.org/`

4.2 Three-Tier Architecture

An n-tier architecture is commonly used for web applications. Components are organized into horizontal layers in this architectural pattern, with each layer performing a specific, (often isolated) role.

4.2.1 Presentation Tier

The highest-level layer in an n-tier architecture. The presentation tier is responsible for displaying the graphical user interface (GUI) and communicating with the application logic tier to update the display as appropriate. In this application, HTML templates are rendered by the server using the Pug templating engine [18] and then served to the client. Pages are relatively simple, using Bootstrap4[19] and jQuery 3.3.1 [20] for presentation and interactivity. Data is sent and received through HTTP GET and POST requests to the various exposed APIs.

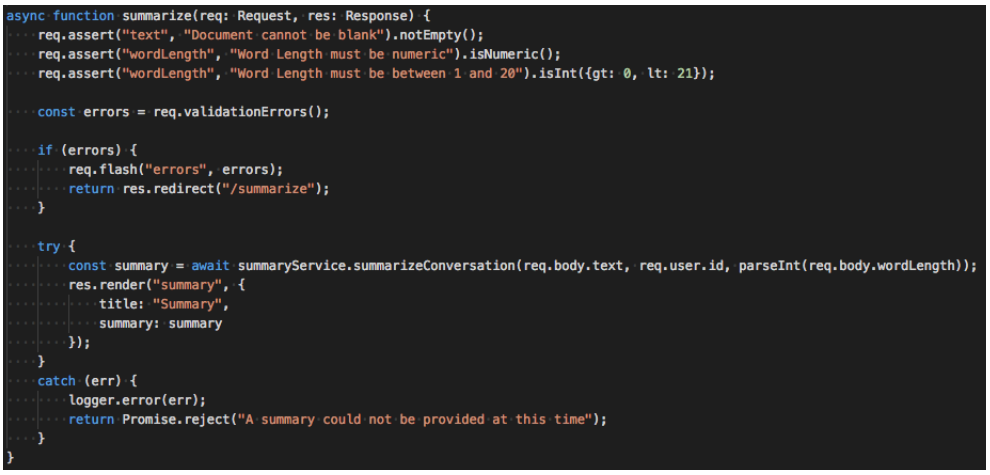
4.2.2 Application Logic Tier

This layer is responsible for coordinating completion of application-level tasks. The application follows a Model-View-Controller (MVC) framework[21] for sufficient separation of concerns and ease of development. Isolating application logic into multiple domains promotes robust, testable, and reusable code. It also has the additional benefit of enabling independent parallel development in multi-developer projects.

- Controller - The controller is responsible for accepting user input and invoking the necessary business-logic functions to return an appropriate response. This application uses the Express.js [22] web application framework to expose APIs that respond to user requests over HTTP. Various controller modules define how the application handles specific routes. For example, a GET method request to "/summarize" returns the web form used to submit user-provided conversation transcripts for AKE, while a POST method to "/summarize" submits the transcript to be summarized, as well as the number of keyphrases desired, to the application in the body of the request. Note that business-logic functions are invoked in the controller, but defined in the model.

[18]https://pugjs.org/
[19]https://getbootstrap.com/docs/4.0/
[20]https://jquery.com/
[21]https://en.wikipedia.org/wiki/Model-view-controller
[22]https://expressjs.com/

Consider the following figure which demonstrates the isolation of an individual function in the controller:



```
async function summarize(req: Request, res: Response) {
    req.assert("text", "Document cannot be blank").notEmpty();
    req.assert("wordLength", "Word Length must be numeric").isNumeric();
    req.assert("wordLength", "Word Length must be between 1 and 20").isInt({gt: 0, lt: 21});

    const errors = req.validationErrors();

    if (errors) {
        req.flash("errors", errors);
        return res.redirect("/summarize");
    }

    try {
        const summary = await summaryService.summarizeConversation(req.body.text, req.user.id, parseInt(req.body.wordLength));
        res.render("summary", {
            title: "Summary",
            summary: summary
        });
    }
    catch (err) {
        logger.error(err);
        return Promise.reject("A summary could not be provided at this time");
    }
}
```

Figure 6: An example of a controller function

The controller function merely validates the user input and then passes the submitted text, number of keyphrases, and user ID number to the model for further processing. It then returns the result (asynchronously) when the appropriate model function responds. Besides route definition and management, the controller in this application is also responsible for user authentication, session management, access control, caching, API rate limiting (explained later), and security measures to prevent common attacks such as cross-site scripting (XSS) [23].

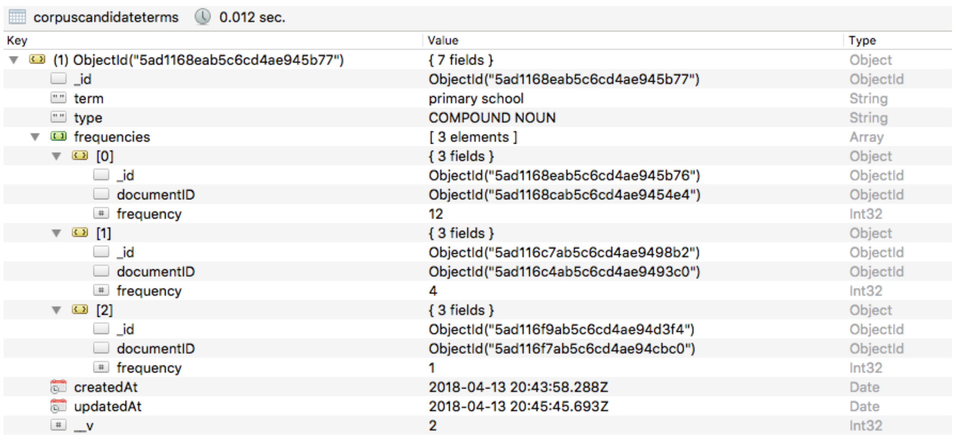
[23] https://en.wikipedia.org/wiki/Cross-site_scripting

- Model - The model is responsible for abstracting business logic away from the controller. Individual services correspond to the controller sub-modules, and handle the following tasks:
    - **Communication with the CoreNLP server** - The Stanford CoreNLP server is run in a Docker[24] container exposed over port 9000. Communication between the Java-based NLP server and the Node.js application is handled by the open-source Node CoreNLP adapter[25], and a custom-written type definition file for compatibility with TypeScript [26]. The custom Docker image is included in the appendix.
    - **Communication with the Database** - Documents are stored in a NoSQL MongoDB database[27], which is further in the 'Data Storage Tier' section below.
    - **Querying DBpedia** - DBpedia is queried to determine keyphrase specificity scores via the DBpedia Lookup web service[28] and the GOT HTTP request module[29].
    - **Metric Calculations** - The model handles the calculations of all metrics used for keyphrase ranking ($tf - idf$, specificity, etc...) and evaluation (precision, recall, etc...)
    - **Summary Algorithms** - All four AKE algorithms are defined in the model.
- View - The view is responsible for representing information. Static Pug templates are used for common pages, while JavaScript-Object-Notation (JSON) data from the controller is used to render the appropriate templates for display after the user submits a request.

[24]`https://docs.docker.com/`
[25]`https://github.com/gerardobort/node-corenlp`
[26]This type definition file could be submitted as an open-source contribution in the future.
[27]`https://docs.mongodb.com/manual/`
[28]`https://github.com/dbpedia/lookup`
[29]`https://github.com/sindresorhus/got`

4.2.3 Data Storage Tier

The data storage tier is responsible for the modeling, storage, and retrieval of information. TypeGoose[30] schemas to model the data in an Object-Oriented structure. Corpus conversations are stored in their raw and CoreNLP annotated forms. Lemmas, candidate terms, and named entities are stored in separate record-level inverted indices[31]: each one is stored with its term (e.g., "primary school"), type (e.g., "COMPOUND NOUN"), and an array of [frequency, pointer] tuples indicating which documents the term occurs in, as well as its frequency. This information is used for *idf* calculations.


corpuscandidateterms 0.012 sec.

Key	Value	Type
(1) ObjectId("5ad1168eab5c6cd4ae945b77")	{ 7 fields }	Object
_id	ObjectId("5ad1168eab5c6cd4ae945b77")	ObjectId
term	primary school	String
type	COMPOUND NOUN	String
frequencies	[3 elements]	Array
[0]	{ 3 fields }	Object
_id	ObjectId("5ad1168eab5c6cd4ae945b76")	ObjectId
documentID	ObjectId("5ad1168cab5c6cd4ae9454e4")	ObjectId
frequency	12	Int32
[1]	{ 3 fields }	Object
_id	ObjectId("5ad116c7ab5c6cd4ae9498b2")	ObjectId
documentID	ObjectId("5ad116c4ab5c6cd4ae9493c0")	ObjectId
frequency	4	Int32
[2]	{ 3 fields }	Object
_id	ObjectId("5ad116f9ab5c6cd4ae94d3f4")	ObjectId
documentID	ObjectId("5ad116f7ab5c6cd4ae94cbc0")	ObjectId
frequency	1	Int32
createdAt	2018-04-13 20:43:58.288Z	Date
updatedAt	2018-04-13 20:45:45.693Z	Date
__v	2	Int32

Figure 7: An example of an entry in the candidate term inverted index

There are two separate databases for corpus documents (those in the test collection) and user documents. Users have personal record-level inverted indices for lemmas, candidate terms, and named entities. The reason for this is explained in the Future Work section.

4.2.4 An Example of the 3-Tier Architecture in Action

The user enters the text into the form and selects the number of keyphrases desired. When the form is submitted, the input is validated by the controller and passed to the summary service, along with the user's id. The text is then passed to the CoreNLP server for annotation and then summarized using one of the algorithms. As explained above, each lemma, candidate term, and named entity is saved/updated in the user's personal inverted index. Once the final ranked list of keyphrases has been generated by the

[30]https://github.com/szokodiakos/typegoose
[31]https://en.wikipedia.org/wiki/Inverted_index

summary algorithm, the service selects and returns n of those keyphrases to the controller (as specified by the user). The controller then renders and returns an HTML template.

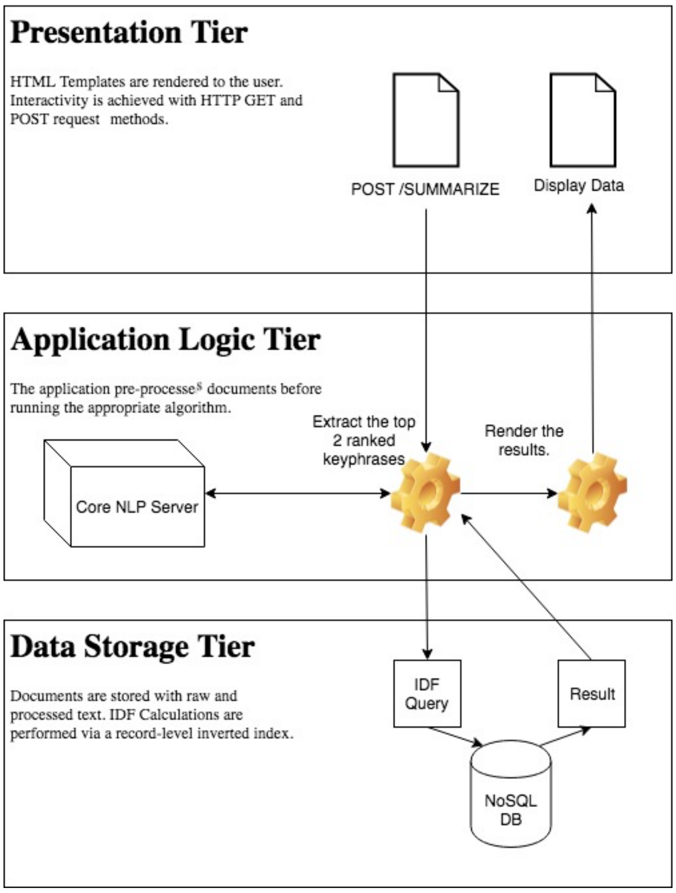

Presentation Tier
HTML Templates are rendered to the user. Interactivity is achieved with HTTP GET and POST request methods.
POST /SUMMARIZE
Display Data
Application Logic Tier
The application pre-processes documents before running the appropriate algorithm.
Extract the top 2 ranked keyphrases
Render the results.
Core NLP Server
Data Storage Tier
Documents are stored with raw and processed text. IDF Calculations are performed via a record-level inverted index.
IDF Query
Result
NoSQL DB

Figure 8: An example of the 3-tier architecture in the context of the application

4.3 A Note on Good Programming Practices

Application code is extensively commented, documented, and tested. Code formatting[32], coverage[33], continuous integration[34], and dependency watcher[35] tools were used to ensure code consistency, quality, and readability. These are all common practices expected of modern programmers.

[32]https://palantir.github.io/tslint/
[33]https://coveralls.io/
[34]travis-ci.org/
[35]david-dm.org/

5 Evaluation and Results

5.1 Developing a Test Collection

An existing collection of spoken dialog along with human selected keyphrases could not be found; so a new 25-document test collection of multi-party conversation transcripts was constructed by randomly selecting articles from the Australian Radio Talkback Corpus (ART)[36], SACODEYL European Youth Language Corpus [37], and BACKBONE Pedagogic Corpora for Content & Language Integrated Learning [38]

Five different volunteers were assigned five conversations and asked to produce summaries of short, medium, and long length, as well as ten descriptive keyphrases for each transcript.

5.2 Selecting Evaluation Measures

Algorithms were used to produce summaries of 5 (short), 10 (medium), and 15 (long) key terms or phrases for each document in the test collection. Their performance was evaluated objectively using the following metrics:

Participant defined summary precision: Each keyphrase in a generated summary is given one point if it is also present in the human defined summary of the same size. The total score is then divided by the length of the generated summary. Scores are denoted as "Precision@k", with k representing a fixed number of keyphrases. Precision at 5 keyphrases is measured against the given short summary, precision at 10 keyphrases is measured against the medium summary, and precision at 15 keyphrases is measured against the long summary.

Participant defined keyphrase recall: Percentage of the total participant defined keyphrases successfully recalled by a 15 keyphrase generated summary.

Their Performance was also evaluated subjectively with a **perceived keyphrase quality score**. Document summarization is a highly subjective task; each participant produced summaries of varying length and specificity. Many used terms and phrases in their various summaries and lists of keywords that

[36]`https://www.ausnc.org.au/corpora/art`
[37]`http://www.um.es/sacodeyl/`
[38]http://projects.ael.uni-tuebingen.de/backbone/moodle/

were not present in the original transcript. As a more indicative effort of keyphrase quality, the volunteers were asked to rate the quality of the generated keyphrases at each length (short, medium, and long) on a scale of 1 to 5. 1 being terrible, nothing to do with the conversation, and 5 being amazing, perfectly captures the essence of the conversation.

5.3 Selecting Algorithm Parameters

Algorithms 3 and 4 required certain parameters to be selected. Latent Dirichlet Allocation was performed with the same parameters suggested as Griffith and Steyvers [41]: $\alpha = 0.01$, $\beta = 0.1$, and Collapsed Gibbs sampling over 1000 iterations with a burn interval and sample delay of 100. To generate k keyphrases, the algorithm was run on k topics, with the highest probability candidate term for each topic selected. Algorithm 4 was run with a $l = 50$ and $\gamma = 0.75$, as it was observed that preferentially weighting $tf - idf$ over the specificity score yielded higher results. Increasing γ any further reduced performance.

5.4 Evaluation and Results

Table 3: Comparison of average algorithm performance

Method	Summary Prec.@5	Summary Prec.@10	Summary Prec.@15	Keyphrase Recall@15
Lemma TF-IDF	0.264	0.241	0.315	.280
Candidate Term TF-IDF	0.328	0.348	0.361	.284
Candidate Term LDA	0.056	0.148	0.156	.120
Candidate Term and Named Entity Weighted Semantic Evaluation	0.376	0.268	0.3204	.328

The Lemma TF-IDF approach served primarily as a benchmark for the other algorithms. Its precision score was artificially high due to the nature of the technique. Consider one of the test collection documents that captures an interview with the owner of a company that provides tours in an amphibious vehicle. The lemma algorithm earned two points for returning the terms ‘amphibious’ and ‘vehicle’, while the other three algorithms earned only one point for correctly identifying and returning the compound noun ‘amphibious vehicle’. Perhaps a more appropriate score would consider both the ‘correctness’ of the returned keyphrase, as well as the length of n-gram.

The Candidate Term LDA algorithm displayed a remarkably low performance, often returning phrases that contributed very little to the overall meaning of the documents. Independent test readers who were asked if they could reconstruct the gist of a summarized conversation from its keyphrases made fair attempts of the ones provided by the other algorithms but often fell far short when faced with the ones selected by LDA. This is attributed to the probabilistic nature of the model. It is worth noting, however, that the LDA algorithm frequently selected important keyphrases that were ranked quite low in the $tf - idf$ based approaches. Whether this was caused by the initial random seed or was a systematic feature could not be determined with such a small test collection.

The Candidate Term-TF-IDF and the *Candidate Term and Named Entity Weighted Semantic Evaluation* algorithms shared a similar distribution of precision@k values, though the former yielded a superior performance at higher values of the k, and the latter at lower ones. A determination of whether this result was statistically significant would require a larger test collection. The *Candidate Term and Named Entity Weighted Semantic Evaluation* approach exhibited a higher keyphrase recall value, which was easily explained considering the previously presented statistic that nearly 20% of participant defined keyphrases were named entities (as an example, note that this algorithm correctly selected ‘sam mcguire’[39] as a keyphrase from a conversation on Gaelic football [40], while the Candidate Term TF-IDF algorithm selected the term ‘cup’ instead).

[39]`https://en.wikipedia.org/wiki/Sam_Maguire_Cup`
[40]`https://en.wikipedia.org/wiki/Gaelic_football`

Table 4: Comparison of average perceived keyphrase quality

Method	Quality@5	Quality@10	Quality@15	Total
Lema TF-IDF	3.16	3.52	3.72	3.47
Candidate Term TF-IDF	3.72	4.08	4.28	4.027
Candidate Term LDA	2.08	2.6	2.6	2.43
Candidate Term and Named Entity Weighted Semantic Evaluation	3.32	3.68	3.52	3.51

The Candidate Term TF-IDF algorithm was the clear winner on the question of perceived keyphrase quality. Participants were asked to select the "best" method after assigning all of their scores to the different generated summaries. The summaries were presented to the volunteers unlabelled, yet this method was picked unanimously. Both the *Candidate Term and Named Entity Weighted Semantic Evaluation* and Lemma TF-IDF methods presented respectable scores and received nearly identical evaluations, while again the LDA method proved unsuitable.

5.5 Discussion

With the exception of the LDA approach, all of the algorithms presented performed surprisingly well. The failure of this technique is attributed to the lack of a proper training phrase. Performance may have been higher when only the conversation to be summarized was considered, but it is likely that this is a local minimum. Higher performance would have been achieved with a sufficiently large corpus.

The Lemma TF-IDF algorithm served as an excellent benchmark, and proved more than adequate as the basis of any AKE technique on spoken dialog. The unigram 'bag-of-words' model worked well in an environment prone to irregularity in punctuation and grammar. Participants noted that nearly all of the terms produced by this method were 'technically correct', even though they frequently did not capture the true essence of the conversation. This technique often ranked spoken interjections (e.g. 'hmm', 'ahh') among the top lemmas because their frequent occurrence produced high tf values, and their varied spelling (e.g. 'hm' was spelled with anywhere from one to seven m's) produced high *idf* values. A specially curated list of spoken stop words, or a pre-processing algorithm to remove them, is necessary for any future performance gains.

The Candidate Term TF-IDF method corresponded naturally to the way human participants selected keyphrases. Perhaps this similarity elicited high quality scores from the volunteers. Any method using $tf - idf$ based keyphrase ranking is dependent on a large and diverse corpus to prevent inappropriately high IDF scores on keyphrases that may be unique, or at least semi-unique, to a particular document or speaker's style. For instance, one of the conversations in the test collection contained twelve occurrences of the term 'cow', a word which was only present in one other conversation. As a result, it was the highest ranked keyphrase after running the Lemma and Candidate Term TF-IDF algorithms on the conversation. This term was absent in both the keyphrases and all three summaries generated by the participants assigned to this document. Perhaps the best way to overcome this issue is to modify the approaches to consider term positioning in addition to term frequency and inter-collection rarity. The word 'cow' may have occurred a dozen times within the document, but all twelve appearances were within a few sentences of each other. It was previously noted that AKE techniques cannot depend on reliable sentence and paragraph separation in conversation transcripts, but it is still possible to calculate the distance between terms. The possibility that the most descriptive terms in documents are those with both high frequency and large distance between subsequent appearances is one left open for future consideration.

Final consideration is directed towards the relative failure of the *Candidate Term and Named Entity Weighted Semantic Evaluation* technique. Inspection of some of the highly ranked keyphrases highlighted the author-defined specificity score as the main issue. The top five keyphrases returned by this algorithm on a conversation concerning the risks and benefits of hormone replacement therapy for the relief of menopause systems, were: *"menopause", "oestrogen", "last month", "alternative therapies",* and "time reading". Three of these terms make sense, but the inclusion of *"last month"*, and *"time reading"* was puzzling. Despite their low tf scores (2 occurrences each), the algorithm inappropriately ranked these phrases among the most important. These terms were assigned perfect and near-prefect specificity scores (1 and 0.98) because Dbpedia contained one entry for *"last month"* and 2 entries for *"time reading"* on *The Last Month of the Year, Time Reading Program, and All-time Reading United A.C. roster*, respectively. While these might be important articles for other purposes, they clearly are irrelevant to a document on hormone replacement theory. The proposed solution to this issue is to determine the particular meaning of the term by analysing its context and then filtering out entries that do not match this context.

The final question proposed in this discussion explores the limited applicability of present text evaluation metrics to AKE methods that only consider n-grams in the analysed text. It was noted that the five participants wrote reference summaries of highly variable length and abstraction. Some participants built summaries by selecting sentences from the conversation and others chose to write their own sentences that captured the essence of the document, but not necessarily the contents of the conversation. For example, consider the following summaries:

(1) Sam is promoting an amphibious tour operation in Plymouth, England. Tourists board an amphibious vehicle which travels on land and water. The vehicle is regulated by government standards.

(2) A girl called Sarah from Reading has an interview about her life.

Figure 9: A comparison of concrete and abstract summaries generated by participants

Summary (1) is concrete: it contains a large overlap with the n-grams present in the conversation. In contrast, (2) describes the overall purpose of the conversation, but only uses 3 terms present in the transcript. An Extraction-based AKE algorithm might generate a summary similar to (2), but struggle to accurately capture the abstract idea of (1).
There exists a need for an objective metric that can capture the notion of 'relative correctness' in keyphrase selection, where a keyphrase that is 'technically correct' still receives a full, or almost-full score. This question is left open for future debate.

6 Conclusion

6.1 Summary

This project covered the motivations behind Automatic Keyphrase Extraction as well as the history of efforts in the Information Retrieval domain leading up to modern techniques. It highlighted the lack of methods aimed at summarizing unstructured textual sources, particularly conversation transcripts, and explained why an effective algorithm would be beneficial. Four algorithms were presented and evaluated on a newly constructed corpus, the results of which were explained in detail. Additionally, a brief overview of the technical implementation was provide. The project concludes with a discussion of future work in and useful applications in this domain.

6.2 Future Work

6.2.1 Enhancing the Model

The algorithms presented and evaluated in previous sections are all open to future improvement. Some possible methods were described in the Discussion section.

6.2.2 Towards Personalized Summarizations

This project frequently highlighted the challenge of developing summaries consistent with common user behavior. It would be highly beneficial to tailor the AKE process to individuals by learning the way they process, abstract, and summarize information. The application already stores personalized record-level indices for candidate terms, lemmas, and named entities extracted from conversations submitted by users. Thus, a personal $tf - idf$ score could introduced to differentiate it from $tf - idf$ scores obtained from the corpus. The final $tf - idf$ value for a keyphrase could be modified by introducing some weighting factor δ to combine the two scores. Denoting the N as the collection of corpus documents, and U as the collection of user documents, the final $tf - idf$ value of a keyphrase could be defined as:

$$w_{kp,d} = \delta \cdot ((1 + log f_{kp,d}) \cdot \log(\frac{U}{U_{kp}})) + (1 - \delta) \cdot ((1 + \log f_{kp,d}) \cdot \log(\frac{N}{N_{kp}})) \quad (9)$$

6.2.3 Developing Query Systems

An interactive query system could be developed to allow users to ask questions about their past conversations. Potential examples of queries and their corresponding answers include:

- **Topic Retrieval**
  - Question: "What did I talk about at 2 pm today?"
  - Answer: "At 2 pm today you discussed the Seattle Seahawks last game against the Texans."
- **Conversational Partner Identification**
  - Question: "Who did I talk to yesterday?"
  - Answer: "Yesterday you talked to John, Mary, and Barbara."
- **Appointment Reminders**
  - Question: "Am I supposed to meet someone today?"
  - Answer: "You told James Smith you would meet him for coffee at 3 pm in Starbucks."
- **Task Reminders**
  - Question: "What do I need to have done by Friday?"
  - Answer: "You agreed to draft the minutes for last week's meeting by Friday."

6.3 Applications

6.3.1 Web Bots

Automatic summaries of newspaper articles and text submissions on the popular website Reddit[41] are already performed by the AutoTLDR[42] bot. These summaries assist users in determining whether an article or post is relevant to their information needs or interests. Bots such as AutoTLDR could be extended with advanced AKE algorithms to summarize message threads, which is a more complicated task. Threads commonly contain thousands of messages, the majority of which are irrelevant to the overall discussion amongst users. An effective AKE technique would need to employ domain-specific heuristics to reach a high level of performance.

[41]`https://www.reddit.com`

[42]`https://www.reddit.com/r/autotldr/`

6.3.2 Integration with Artificial Intelligence Agents

Artificial Intelligence (AI) agents like Siri[43], Alexa[44], or Cortana[45] could transcribe real-time audio obtained via a microphone on the dedicated device (in the case of Alexa), or mobile phone. These agents could leverage an advanced conversation AKE algorithm to perform useful functions based on the keyphrases (particularly named entities) derived from the transcripts: scheduling appointments, creating reminders, providing a valuable method of querying the record storage via conversation, etc... Consider the benefits of such a system in the following domains:

The Workforce - Modern society's demanding pace rewards those who can keep track of rapidly shifting appointments, deadlines, and responsibilities. A typical employee walking through the office on the way to the coffee machine may be approached by several co-workers demanding various deliverables or tasks. Doctors may see dozens of patients during their rotations or clinic hours, each with a unique illness or request. Lawyers might discuss multiple cases in rapid succession, debating possible angles and strategies with clients or partners. The universal need for clear and concise summaries of each conversation is readily apparent. Individuals could quickly scan their records to remind themselves of what they need to follow up on or prepare for. Artificial intelligence agents could access the summaries and automatically schedule future meetings, notify users of what they must accomplish, and remind them of the previous conversational topic before they rendezvous with an acquaintance.

Academia - University professors and students commonly discuss several academic concepts or domains throughout the day. A project supervisor, for example, may deal with students working on diverse topics such as genetic algorithms, network security, and information retrieval. The challenge lies in remembering what was discussed and with whom. Conversation summaries will help students and lecturers readjust their headspace before meetings and recall previously theorized concepts or ideas.

Elderly Assistance - As the brain ages it becomes more difficult to remember past social interactions. Typically the elderly require a small 'jog' to their memory to recall the rest of the conversation. The common frustration amongst surveyed family members and friends was that they "remember they

[43]`https://www.apple.com/ios/siri/`

[44]`https://developer.amazon.com/alexa`

[45]`https://www.microsoft.com/en-us/windows/cortana`

talked with someone in the afternoon about something important," but they remember neither the individual nor the subject. A small reminder from an AI agent, such as: "you discussed your wife's birthday present with John", would be sufficient for complete recollection. Conversation summaries will reduce missed commitments and appointments, minimize personal irritation and assist those with the early stages of Alzheimer's.

Information Management - Digital media archives are difficult to index automatically, and typically require manual labeling. As the sizes of media collections increase, so too will the need for efficient methods of sorting through the data. Automatic unsupervised summaries of stored audio can serve as indices to assist query systems in rapidly locating relevant files and segments. This approach would potentially work for any media that includes sound, such as meeting recordings, movies, newscasts, and podcasts.

Note that all of the algorithms all depend on the textual quality of the transcript. Misspelled words, incorrect punctuation and unnecessarily included interjections have serious impacts on performance. Additionally, storing transcripts for every conversation analyzed may prove too taxing on cloud storage if the an AI agent was used. It is worth looking into the possibility of performing AKE on raw streamed audio and storing only the extracted keyphrases.

6.4 Broader Societal Impact

As with all technology that improves performance by consuming personal information, a privacy-minded approach is advised. The use of AI technology and AI assistants opens multiple avenues to theft or misuse of user data. Researchers, developers, and students are urged to read *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems (A/IS)*[46], a work by the IEEE Standards Association to "ensure every stakeholder involved in the design and development of autonomous and intelligent systems is educated, trained, and empowered to prioritize ethical considerations so that these technologies are advanced for the benefit of humanity"[47]. It is their recommendation that policy is created that:

- Allows every global citizen/individual access to tools allowing them control over a minimum common denominator of attributes that define his/her identity.

- Allows the possibility for citizens/individuals to access, manage, and control how their data is shared.

- Provides easily understandable ways for citizens/individuals to choose how or whether to share their data with other individuals, businesses, or for the common good as they choose.

- Provides for future educational programs training all citizens/individuals regarding the management of their personal data and identity, just as many countries provide training in personal finances and basic legal understanding.

Figure 10: Personal Data and Individual Access Control policy recommendation, as defined by the IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems

6.5 Closing Remarks

The future of AKE techniques in the domain of conversation summarization is clearly promising. The author has enjoyed his exploration of the topic, and hopes that the findings and reflections presented will stimulate future work.

[46]`http://standards.ieee.org/develop/indconn/ec/ead_v1.pdf`

[47]`https://standards.ieee.org/develop/indconn/ec/autonomous_systems.html`

7 References

[1] Doug Laney. 3d data management: Controlling data volume, velocity, and variety, 2012. `https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf`, Last accessed on 2018-04-14.

[2] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[3] Kazi Saidul Hasan and Vincent Ng. Automatic keyphrase extraction: A survey of the state of the art. In *In Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL*, 2014.

[4] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 257–266, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[5] Ken Barker and Nadia Cornacchia. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '00, pages 40–52, London, UK, UK, 2000. Springer-Verlag.

[6] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 216–223, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[7] Hassan H. Alrehamy and Coral Walker. *SemCluster: Unsupervised Automatic Keyphrase Extraction Using Affinity Propagation*, pages 222–235. Springer International Publishing, Cham, 2018.

[8] Marina Litvak and Mark Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics, 2008.

[9] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[10] Larry Page, Sergey Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1998.

[11] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics, 2004.

[12] George Tsatsaronis, Iraklis Varlamis, and Kjetil Nørvåg. Semanticrank: ranking keywords and sentences using semantic graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1074–1082. Association for Computational Linguistics, 2010.

[13] D. B. Bracewell, F. Ren, and S. Kuriowa. Multilingual single document keyword extraction for information retrieval. In *2005 International Conference on Natural Language Processing and Knowledge Engineering*, pages 517–522, Oct 2005.

[14] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 257–266, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[15] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[16] Amy M. Steier and Richard K. Belew. Exporting phrases: A statistical analysis of topical language. In *Second Symposium on Document Analysis and Information Retrieval*, pages 179–190, 1993.

[17] Ken Barker and Nadia Cornacchia. Using noun phrase heads to extract document keyphrases. In Howard J. Hamilton, editor, *Advances in Artificial Intelligence*, pages 40–52, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[18] Marti A Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16. Association for Computational Linguistics, 1994.

[19] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[20] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM, 2006.

[21] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, Nov 2016.

[22] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, 1(4):309–317, October 1957.

[23] Gerard Salton, A. Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975.

[24] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513 – 523, 1988.

[25] H. P. Luhn. Key word-in-context index for technical literature (kwic index). *American Documentation*, 11(4):288–295.

[26] Karen Sparck Jones. Document retrieval systems. chapter A Statistical Interpretation of Term Specificity and Its Application in Retrieval, pages 132–142. Taylor Graham Publishing, London, UK, UK, 1988.

[27] Zipf, george k. human behavior and the principle of least effort. cambridge, (mass.): Addison-wesley, 1949, pp. 573. *Journal of Clinical Psychology*, 6(3):306–306.

[28] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 60(5):503–520, 2004.

[29] Julie B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[30] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[31] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[32] Zellig S. Harris. Distributional structure. 10:146–162, 08 1954.

[33] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 21–26, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[34] Patrice Lopez and Laurent Romary. Humb: Automatic key term extraction from scientific articles in grobid. 07 2010.

[35] Josef Steinberger and Karel Ježek. Evaluation measures for text summarization. *Computing and Informatics*, 28(2):251–275, 2012.

[36] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

[37] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis, 1998.

[38] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[39] Kristina Toutanova, annd Manning Christopher D. Klein, Dan, and Yoram. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *In Proceedings of HLT-NAACL 2003*, pages 252–259, 2003.

[40] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 63–70, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[41] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

[42] Zhuolin Qiu, Bin Wu, Bai Wang, Chuan Shi, and Le Yu. Collapsed gibbs sampling for latent dirichlet allocation on spark. In *Proceedings of the 3rd International Conference on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications - Volume 36*, BIGMINE'14, pages 17–28. JMLR.org, 2014.

[43] Ralph Grishman and Beth Sundheim. Design of the muc-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 1–11, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[44] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

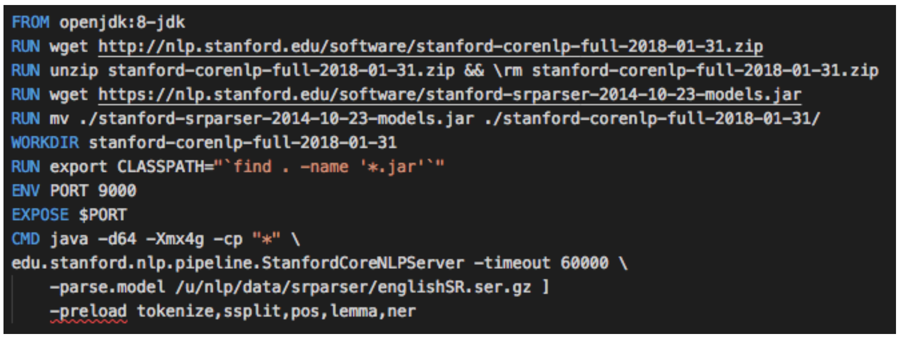
[45] Jenny Rose Finkel, Trond Grenager, and Chistopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370, 2005.

[46] Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. Cognates can improve statistical translation models. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*, NAACL-Short '03, pages 46–48, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Appendices

A Stanford CoreNLP Custom Docker Image

The StanfordCoreNLP server can was run as a detached process in a Docker container exposed over port 9000 with the open-jdk-8 image shown below. This custom image downloads and unzips the core distribution of the server, as well as the higher performing (yet more complex) Shift Reduce Parser[48], and also preloads all of the annotators. Note the server requires at least 4GB of allocated heap memory to successfully run.



```
FROM openjdk:8-jdk
RUN wget http://nlp.stanford.edu/software/stanford-corenlp-full-2018-01-31.zip
RUN unzip stanford-corenlp-full-2018-01-31.zip && \rm stanford-corenlp-full-2018-01-31.zip
RUN wget https://nlp.stanford.edu/software/stanford-srparser-2014-10-23-models.jar
RUN mv ./stanford-srparser-2014-10-23-models.jar ./stanford-corenlp-full-2018-01-31/
WORKDIR stanford-corenlp-full-2018-01-31
RUN export CLASSPATH="`find . -name '*.jar'`"
ENV PORT 9000
EXPOSE $PORT
CMD java -d64 -Xmx4g -cp "*" \
edu.stanford.nlp.pipeline.StanfordCoreNLPServer -timeout 60000 \
    -parse.model /u/nlp/data/srparser/englishSR.ser.gz ]
    -preload tokenize,ssplit,pos,lemma,ner
```

Figure 11: Custom Stanford CoreNLP docker image file

B Source Code Repository

The source code for the developed application is available at `https://github.com/michaeljneelysd/Final-Year-Project`

[48]`https://nlp.stanford.edu/software/srparser.html`