The University Of Sheffield.

Triple Scoring: Scoring and ranking the truth of factual triples

Callum William Booth

Supervisor: Dr. Andreas Vlachos

COM3610 Dissertation Project

02/05/2018

Word Count: 13,764

This report is submitted in partial fulfilment of the requirement for the degree of Computer Science (MComp) by Callum William Booth

The University of Sheffield

Faculty of Engineering

Department of Computer Science

DECLARATION

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Callum William Booth
Signature:

Date: 02/05/2018

ABSTRACT

In a lot of cases, what we consider a fact essentially boils down to “x is y”. Facts such as “the sky is blue”, “space is cold”, “the universe is huge”, all essentially follow this format: subject-relation-object.

This is advantageous, as it allows us to represent facts in a very mathematical way, one which can be parsed by a computer. This project aims to create a model that can assign a numeric truth score to a type-like relation triple. These triples take the form of (*subject*, *relation*, *object*), and can be used to represent a fact.

In this project, a triple scoring model is designed for the WSDM Cup 2017 Triple Scoring task. The model utilises large corpora, and performs natural language processing and information retrieval techniques to corroborate and rank facts, culminating in a model that achieves an 8th place result out of a pool of 21 solutions.

ACKNOWLEDGEMENTS

My deepest thanks to my supervisor Dr. Andreas Vlachos for his help and guidance throughout this project, and to my family for having to put up with me pacing around the living room at midnight while I mumbled about thesaurus generation being a pain in the *donkey*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

1 INTRODUCTION

In this chapter, the problem to be solved and the various constraints it must work within are defined, and some potential use cases for a solution are briefly considered. The various challenges and obstacles that this project entails are also explored.

1.1 BACKGROUND

Verifying the relevance and truth of a fact was a topic that gained considerable traction throughout 2016 and 2017. Events such as the UK's EU referendum and the 2016 US presidential election saw rise to masses of information of questionable veracity, the volume of which cannot be verified by human effort alone. Since then, it has become increasingly clear that automated solutions are required to sort fact from fiction.

A system which can verify facts is useful for many reasons. The most significant use case is to verify information spread during campaigns. A prominent example was the 2016 US presidential election, which saw the rise of the term "fake news", defined as "news articles that are intentionally and verifiably false, and could mislead readers" (Allcott & Gentzkow, 2017). Fact checking systems in general can be an effective way to quash these malicious attempts at misdirection, by rating a fact or statement within a range of false and true values.

Systems that can verify and score facts also have applications such as search engines, capable of answering simple questions or show summarised results from a user query. For example, a search engine might return a list of professions a person named in a query has. To rank results like this, a scoring system is needed to allow comparisons, and to produce an ordering over the set of results.

For large knowledgebases, the most interesting results are wanted first, and these will be the ones that score highest. Bast, et al. (2015) find that in a Freebase query of entities with profession "Actor" and nationality "American", a simple popularity ranking lists the top 5 American actors as "*George Bush, Hillary Clinton, Tim Burton, Lady Gaga, Johnny Depp"*. They conclude that this is tenuously correct in the sense that they have appeared in movies, however only one of these returned entities can really be classed as an actor, making it clear that more sophisticated techniques of deriving facts about entities are needed.

1.2 PROJECT

This project focuses on verifying (*subject*, *relation*, *object*) triples that represent a fact to verify against a corpus or knowledgebase. For example, the triple ("Guillermo del Toro", "profession", "Director") represents the statement "Guillermo del Toro's profession is Director", a fact not only verifiable by common knowledge, but also by information freely available on the internet that can corroborate the fact, known as a "witness" to the fact. The more of these witnesses we

can find, the truer the fact is. (Bast, et al., 2015). To find these witnesses, the produced triple scoring system needs to be able to extract pertinent information from a corpus and assess whether it substantiates the claim; to do this, information extraction methods on a corpus of sentences extracted from Wikipedia will be used. Each of these witnesses will then be weighted in some way, the methods of which will be explored in later chapters, and a model will be used to transform these weightings into a score.

Since this project has been created as part of the Web Search and Data Mining Cup 2017 (WSDM Cup 2017), certain parameters for the project have been pre-defined: the triple scorer must produce a score in the range [0, 7], where 0 represents complete falsity and 7 represents complete truth (Wsdm-cup-2017.org, 2017), and two main relations are to be explored: *profession*, and *nationality*. WSDM Cup 2017 also provides a suite of evaluation functions that will be used to evaluate the performance of the project against a set of ground truth values judged by a human panel (which has also been provided). A full listing of the files provided by WSDM Cup 2017 can be found in Appendix A.

In this project, two main models for solving the task will be explored: a bag-of-words model, which carries out rudimentary full-text search; and a semantic relation model with synonym search, their performance relative to the winning solutions described by Bast, et al. (2017) will be evaluated.

1.3 Constraints and Challenges

A project like this is fraught with various challenges, the most prominent being the time implications of reading corpora. Processing large corpora is a task that can be bound by many hardware factors, such as the speed and amount of RAM on the host machine, and I/O speed of the storage medium the corpus is stored on. As this project will be run on consumer grade hardware, special care will have to be taken to ensure the corpus can be read quickly and efficiently.

In addition to I/O constraints, there is an issue surrounding the complexity of the task itself. Triple scoring as a concept is a difficult task due to the dynamic nature of language, for example the task of dealing with synonymic language, which Bast et al. (2015) also refer to as "semantically related" language. Sentences in a corpus aren't always guaranteed to contain the exact name of a profession, but instead have language related to a profession such as "cast as" or "starred in" in context of "actor", so additional datasets may have to be used to allow the system to find a broader range of witnesses to the triple.

This, however, broaches the issue of how wide a semantic search space to explore. The more synonyms are explored, the more diluted the underlying concept being searched becomes. For example, if the profession word "footballer" is to have its synonyms explored, there runs a risk that words such as "sport" would eventually count as a witness to a triple containing it, leading to overzealous witness capture. However, without any sort of synonym search, the solution is

restricted purely to exact vocabulary matches, akin to a very rudimentary full-text search engine. This more conservative method preserves underlying meaning but can lead to potential witnesses being ignored.

Finally, as this is a theoretical project focused on experimenting with models and techniques, it will be limited by time constraints. Exploring and evaluating many iterations of different models and extraction techniques is time consuming, particularly when hardware constraints are considered. Executing multiple runs of a system for testing purposes or training a learning model on corpus data will take a lot of time and computing power every time the code is changed. Therefore, for brevity's sake, this project will aim to produce a system that works reasonably well having explored a variety of different methods. Further discussion around improving the final solution will be carried out.

2 LITERATURE REVIEW

In this chapter, various relevant topics in the field of natural language processing that will be useful throughout this project will be explored, and the research behind them will be discussed. Current research into the problem of triple scoring will be analysed, by looking at solutions submitted as part of the WSDM Cup 2017 competition.

In particular: the work of Ding, et al. (2017), who submitted the winning solution: the BOKCHOY triple scorer; and Yamada, et al. (2017) who submitted the second-place solution: the Lettuce triple scorer; will be discussed.

2.1 CURRENT RESEARCH

Current research into triple scoring is mostly centred around the work of Bast, et al. (2015), who propose the notion of a "type-like relation" and suggest methods and techniques for extracting information from corpora, and generating triple scores.

Bast, et al. (2015) discuss the idea of counting *profession* words, which mimics the behaviour of humans in solving this problem (restricted to the *profession relation* domain). A person doesn't intrinsically know that x is y without seeing evidence of this, but this evidence won't always be explicitly defined. For instance, say "x is an actor" is a fact that can be verified; evidence in a corpus that bears witness to this fact may be of the form "x acted in…", "x starred as…", etc. These profession words can consist of synonyms of the profession itself, words relating to the profession, or words that are generally evocative of the profession.

Bast, et al. (2017) describe the methodology behind the winning solution submitted as part of the WSDM Cup competition: the BOKCHOY Triple Scorer submitted by Ding, et al. (2017).

2.1.1 The BOKCHOY Triple Scorer

Ding, et al. (2017) describe their approach as an ensemble learning model based on four initial base scorers:

- A word classification scorer, that utilises a binary classifier trained on 100 uniformly sampled persons for each profession, where the training feature set consists of words from the corpus valued by TF.IDF values, and uses the classifier's confidence value as the base score.
- A word counting scorer, that for each profession, counts instances of indicative vocabulary, and uses the sum of products of word occurrence count and TF.IDF value as the relevance score calculation.
- A generative model to backwards generate text associated with a person based on *profession* vocabulary. Each word in the corpus of person-related text has an associated probability correlated to the relevance of the *profession* to the person, found via maximum likelihood estimation, and is used to measure the relevance between the *profession* and the person.

- A Freebase path ranking strategy, to build paths between entities in Freebase data based on the relation, using each path as a feature in a binary classifier.

Ding, et al. (2017) describe their method of combining these scores into a single [0,7] ranged integer, by employing an ensemble learning model whereby each score is combined as an average weighted by the score given for the triple's entry in the **.train* files.

This solution is interesting as it employs multiple different classification methods to produce a score, allowing their solution to adapt to corpus data differences between people. For example, if the *wiki-sentences* corpus didn't contain enough information on a person, the Freebase path ranking step would be able to compensate.

This solution was the first-place entry, achieving a 0.868 accuracy (Bast, et al. 2017).

2.1.2 The "Lettuce" Triple Scorer

Yamada, et al. (2017) describe the approach behind their "Lettuce" triple scorer as a two-step classification and scoring model, based on neural network and machine learning based approaches. This solution was the second-place entry in terms of accuracy, at 0.823 (Bast, et al. 2017).

Yamada, et al. (2017) describe the classification step, which carries out an extraction of terms related to entities by tokenising Wikipedia articles related to the entity, and any sentence in any article containing a link to the entity. This creates two item vectors: one containing terms, and one containing any other entities also linked in the corpus. From these vectors, a multi-layer perceptron classifier is trained on feature vectors computed from these item vectors (Yamada, et al., 2017).

Yamada, et al. (2017)'s score mapping step consists of a gradient boosted regression tree based classifier model trained on probabilities of a type applying to an entity from the output vectors of their classification step.

2.2 PRE-PROCESSING METHODS

Before text can be effectively learnt from, several pre-processing steps are normally performed to "normalise" the corpus. These steps can consist of tokenisation, stop-word removal, and stemming and/or lemmatisation.

2.2.1 Tokenisation

Tokenisation is the process that forms the basis of most parsing processes in natural language processing. It involves splitting a sentence into a list of words, in a process described by Grefenstette & Tapanainen (1994) as "the isolation of word-like units from a text". For most basic uses, the delimiter which marks the split between words will be a whitespace character, or punctuation. For example, the sentence "I love natural language processing" would be tokenised as ['I', 'love', 'natural', 'language', 'processing'], if delimited on whitespace, but other methods can be used to define where the demarcation point should be, such as defining the structure of a word with a regular expression.

Tokenisation plays heavily into the fields of text modelling, particularly in the models described later in this chapter: bag of words modelling, and vector space modelling. For both models, it is required that a vector of all words in a text be produced, which can be done through tokenising the string.

Delimiting a string for tokenisation in most cases can be problematic however, as there is no guarantee that a lexer will make meaningful splits, as some word and sentence boundaries are harder to isolate due to "ambiguous punctuation" (Grefenstette & Tapanainen, 1994). Examples include abbreviations such as "m.p.h"., hyphenated words, and terms that consist of multiple words. This has the capacity to cause problems in this project, particularly for triples with multi-word *object*s, such as "United States of America" for a *nationality relation*. Under whitespace-delimited tokenisation, this string would become ['United', 'States', 'of', 'America'], and each word would then become a feature of the sentence vector, not the whole term.

2.2.2 Part of Speech Tagging

Part-of-speech tagging is a process whereby individual tokens in a text are labelled with the grammatical role they play within the text, or their 'part-of-speech', e.g. noun, verb, pronoun, etc.

Part-of-speech tagging may prove useful for this project in identifying sections of a text that are more relevant than others. For instance, if identifying parts of text related to the *profession relation*, it would be more useful to only consider nouns and verbs, and to strip out the remainder. It is also useful for the process of identifying synonymic language, as it will be important to know what context a word is in, so the right synonyms can be found. Brill (2000, p. 404) posits this same problem but from a view of machine translation, by considering the English word "record" and its possible translations to French as "enregistrer" and "disque", depending on whether the part of speech of the word was verb or noun.

We can apply this same view to the realms of synonymic language. For instance, if the word "reading" was encountered in a normalised text, it could be seen to either represent the verb "to read", in which case synonyms exist, or it could represent a noun form "Reading", a town in the U.K., in which case synonyms don't exist. For this reason, part-of-speech tagging could be used to find synonyms, by identifying from context whether the word is a verb or noun.

2.2.3 Stemming and Lemmatisation

Stemming is defined as the removal of a word's derivational and inflectional affixes, with an aim to reduce words down to the same root (Lovins, 1968). Most simple stemming algorithms require the use of lookup tables containing the inflected form of a word, linking it to its root form. This is problematic however in that all inflected forms of words must be listed in the lookup table, which becomes harder to do for languages with a lot of inflected forms. To compensate, certain techniques are used to negate the need for a lookup table. For example, a stemmer may try to invent inflected forms by appending known suffixes to a word, and generating a lookup table, or it may try to strip known suffixes away from words it suspects are inflected. Stemmers of this

form mostly deal with suffixes to a word, as in the case of English, inflectional affixes are always suffixes, and only one can ever be present (Brinton, 2000).

Both methods however encounter problems, particularly with exceptions to language rules. For example, the verb "to read" in English is conjugated to "read" in the past tense, but a stemmer may try to conjugate it as "readed". Similarly, the past tense conjugation "ran" will not be detected as the past tense of "to run", as it does not end in a suffix.

To attempt to solve this problem, words in a text can undergo lemmatisation, which is the process of converting an inflected form of a word to its dictionary form, or lemma. This method typically requires knowledge of the target word's part of speech and tries to select the correct lemma to represent the inflected form using that contextual knowledge. For example, a word with a gerund form such as "meeting" could be dropped to the root form "meet" by a stemmer, regardless of whether it's POS was noun or verb. Under lemmatisation, "meeting" would be dropped to "meet" if the POS is tagged as a verb or dropped to "meeting" if the POS was a noun. As per the previous example, a lemmatiser working on the word "ran" would drop it to the root form "run", whereas a stemmer would not.

Applying either of these methods to the dataset in this project may prove useful, as it will allow words related to possible triple *object*s to be found within the text. For example, phrases such as "x acted in" and "x is acting" could be stemmed to produce two phrases both containing "act", which shows that words common to possible triple *object*s from different *predicate* sets can be found easier. However, it would be better to lemmatise instead, as this would allow better handling of irregularly inflected forms (i.e. "run" -> "x ran …", "x was running …").

2.3 NATURAL LANGUAGE PROCESSING

2.3.1 Bag of words Modelling

Bag of words modelling is a method of representing texts for information retrieval purposes, involving generating a bare tokenised form of the original text. This method preserves no grammatical structures about the text, but preserves the counts of each term, making it useful for analysis of different statistics about the text, such as term frequencies.

For example, a text such as "modelling as a bag of words" would be modelled as a simple list of all the terms, with a count of 1 for each: {'modelling': 1, 'as': 1, 'a': 1, 'bag': 1 , 'of': 1, 'words': 1}[1]. A sentence with multiple occurrences of a term would only update the attached count value, instead of appending a duplicate. ("Bag of words model, vector space model" would be modelled as {'bag':1, 'of':1, 'words':1, 'model':2, 'vector':1, 'space':1}).

Bag of words modelling does however pose some problems. For most English language corpora, the feature vectors created from the texts under this scheme will mostly be dominated by the

[1] {…} notation in this dissertation is used to represent sets. No ordering information is preserved. The words are displayed in-order for simplicity's sake.

most common terms in the English language, i.e. “a”, “I”, “the”, etc. unless techniques such as stop-word removal are performed, where a list of non-content words are removed.

This system of modelling texts is initially very powerful for the bag-of-words-based solution created in this project, as it allows features from which to generate feature vectors from to be pinpointed, however it is incredibly limited in its scope for expansion, and more than likely won’t be usable in other models.

2.3.2 Vector Space Modelling and Term Weighting

Vector space modelling is a method of representing corpora as vector spaces for use in indexing and information retrieval systems. Terms in tokenised strings map to weights of vector elements depending on a term weighting scheme, creating a vector space containing every document in the corpus. Document similarity can be measured between representations like this by using vector distance metrics on the normalised vectors, such as the L_2 distance or cosine similarity, shown in Equations 2.1 and 2.2.

$$d(A,B)=\sqrt{(A_1-B_1)^2+(A_2-B_2)^2+\ldots+(A_i-B_i)^2}$$

Equation 2.1 L2/Euclidean vector distance equation

$$\cos(\theta)=\frac{A\cdot B}{\|A\|\|B\|}$$

where $\boldsymbol{x}\cdot\boldsymbol{y}$ represents the dot product of $\boldsymbol{x}$ and $\boldsymbol{y}$
$||\boldsymbol{x}||$ represents the L_2 norm/magnitude of $\boldsymbol{x}$

Equation 2.2 Cosine similarity equation

The value that an element of a document vector takes is dependent on a term weighting scheme. Basic schemes include binary weighting, which sets each term’s element to 0 or 1, depending on whether the term is present in the document, disregarding the frequency; and term frequency weighting (TF) which sets a term’s element to the number of times it occurs in the document.

In practice however, both schemes are too basic, as they lack context about the corpus the document is contained in. A resolution to this is TF.IDF weighting. Salton & Buckley (1988) find that a term frequency parameter enhances the recall of information retrieval systems but will heavily weight common terms over the entire corpus, such as words common in the English language. Salton & Buckley (1988) resolve this by introducing an *inverse document frequency* parameter, or IDF, defined as a proportion between the number of documents containing a term (n) and the number of all documents in the corpus (N); a commonly used IDF factor is $\log\left(\frac{N}{n}\right)$ (Salton & Buckley, 1988; Sparck Jones, 1972). Salton & Buckley (1988) conclude that a TF.IDF weighting scheme would capture the most important terms, defined as terms with high term frequencies and low colletion frequencies.

In the context of this project, modelling the corpus as vectors may be useful, but not without introducing problems. Triple scores may be calculated by using a distance metric between a query vector containing the *object* of the triple and any related words, and all relevant document vectors. This however may introduce problems regarding performance if TF.IDF weighting is used, as the calculation of IDF values requires iterating the entire corpus per unique term in the collection of relevant documents. For large corpora, this is a very inefficient process, and would most likely require using smaller samples of the corpus, at the expense of data completeness.

2.3.3 Topic Modelling and Latent Dirichlet Allocation

Topic modelling is the process of extracting terms from a corpus that represent the common themes, or topics, expressed in the texts. The rationale behind this is that often a search through a corpus will be based solely on the conceptual topic of the corpus, and exact words will not be sufficient in achieving an accurate retrieval (Deerwester, et al., 1990). The first method of topic modelling was proposed by Deerwester, et al. (1990) in the form of latent semantic analysis, a method which seeks to analyse the latent semantic structure in the data by using statistical techniques to remove the randomness of word choice.

Deerwester, et al. (1990) also consider the usage of LSA in overcoming two problems in natural language processing, one of which has been eluded to prior in this chapter, namely *synonymy* and *polysemy*, which they define respectively as the fact that there are multiple words that can refer to one concept, and the fact that most words have multiple meanings. To overcome this problem, Deerwester, et al. (1990) discuss using latent semantic analysis, and by extension topic modelling, to discover the true implied meaning of queries against corpora and remove any ambiguity from the corpus' vocabulary.

While latent semantic analysis/indexing will not be used in this project, topic modelling as a concept is advantageous to use in an application such as this, as it allows sentences in a corpus to be distilled down to their most prominent terms, filtering out words that don't describe the concept. The semantic relation model in Section 4.4 uses a form of topic modelling known as Latent Dirichlet Allocation (LDA) to distil sentences down to topic words.

Latent Dirichlet Allocation is a statistical model of a corpus, where "documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words" (Blei, et al., 2003, p. 996).

The process works by backtracking on an assumed document generation process; the assumption being that a document is generated by first creating a mixture of topics that the document will be about, and then picking words from these topics (Blei, et al., 2003). Each word in a topic has an associated probability, and each topic has a document-wise probability. LDA assumes that all documents in a corpus are generated according to this process and tries to backtrack from the output text to a mixture of topics, each of which is a mixture of every single word in the corpus (Blei, et al., 2003).

For example, LDA may find two topics in a corpus: (0.5*“football”, 0.3*“goal”, 0.3*“club”) and (0.6*“basketball”, 0.1*“ball”, 0.05*“net”). These topics are mixtures of vocabulary words in the corpus, each with their own probability of occurring in document belonging to that topic. A document therefore is a mixture of these topics. So, for example if the first topic is dubbed “football” and the second “basketball”, a document of a corpus may be a percentage of either, one, or none of these topics. In this project, this method is used as a form of dimensionality reduction, by reducing corpora down to a restricted number of topics consisting of a restricted number of terms and extracting the vocabulary from these mixtures.

3 REQUIREMENTS AND ANALYSIS

In this chapter, details about the project as set by the WSDM Cup 2017 task specification are outlined, along with the goals of the system implementation. Technical aspects of the project are discussed, such as choice of programming language and the reason and rationale behind library usage, along with a detailed analysis of how the WSDM Cup 2017 provided evaluation method works.

3.1 SYSTEM REQUIREMENTS

The system will take a type-like relation triple of the form (*subject, relation, object*) and score it in the range [0, 7] based on the extent to which a corpus agrees that the *subject* belongs to the type (*relation* and *object*) (Bast, et al., 2015). The project is based on the work of Bast, et al. (2015) who define the previous triple format and discuss the methods by which the scores are measured in the interval [0, 1]. The actual project parameters however are defined by the WSDM Cup 2017 task, which defines the output interval to be [0, 7] (Wsdm-cup-2017.org, 2017).

The system created in this project will parse a large corpus of texts related to a number of well-known people, provided by WSDM Cup 2017 and described in Appendix A, and seek texts from the corpus that corroborate the fact that the triple represents. The number of texts found will be directly proportional to the output score, and the system will scale the output into the target interval.

The system should be able to process input triples in a timely and performant manner, as the evaluation step described in Section 3.3 runs a substantial number of triples through the system. To do this, the system must be written in such a way to cope with the size of the corpus, meaning that basic iteration of the file is not tractable for a final solution; the final model should be able to learn properties about the input data and recall a score based solely on that knowledge.

3.2 PROGRAMMING LANGUAGE AND LIBRARIES

For this project, the coding aspects will be written in Python. Reasons for this include the large repository of code libraries available, including natural language processing related libraries such as spaCy and Gensim.

spaCy is an NLP toolkit for Python, which provides standard tools such as tokenisation, stop-word removal, POS tagging, stemming and lemmatisation, named entity recognition, and word vector support.

Gensim is another NLP library providing support for topic modelling and word vectors. The primary use for Gensim is to provide implementations of popular vector space models, such as LDA, and LSI (Řehůřek & Sojka, 2010). It also allows a multitude of different vector definition formats to be loaded, such as GLoVe vectors (Pennington, et al., 2014) and FastText vectors (Bojanowski, et al., 2017).

Alternative languages may have been appropriate, such as object-oriented languages like Java or C#, as both have NLP based libraries (Apache OpenNLP and Stanford CoreNLP for .NET respectively) and could potentially provide better support for high volumes of data, but these languages could also potentially add complications surrounding compilation, which may make the solution less portable to different device configurations. Python is also advantageous in that evaluation scripts described in Section 3.3 are written for it, so it will be easier to integrate the evaluation step provided by WSDM into the codebase.

3.3 EVALUATION AND TESTING

The system is evaluated through an evaluation script provided by WSDM Cup 2017 and listed in Appendix A. The script runs on two files, one containing that the triples and their generated scores in tab-separated value (TSV) format, and another containing the "ground truth" triples with values judged by a panel. These ground truth files are also provided by WSDM Cup 2017.

The evaluation script runs three different metrics to assess the difference between scores, including an accuracy function, average score difference, and average Kendall's tau.

To evaluate the systems created, the models will be run multiple times with different parameters and training methods, and their outputs will be compared to the ground truths using his evaluation script. The full list of results generated over the entire course of the project can be found in Appendix B.

3.3.1 Scoring Groups

The evaluation script collects scores from TSV score files and represents them as scoring groups. The *read_files* function takes a TSV file consisting of a subject, value, and score column, and creates sub-arrays of scores, or scoring groups, for each sequential block of subjects.

Plato	Philosopher	7
Plato	Mathematician	4
George H. W. Bush	Politician	7
George H. W. Bush	Military aviator	2
Lyndon B. Johnson	Teacher	0
Lyndon B. Johnson	Politician	7

Table 3.1 Example ground truth triples

For example, ground truth values in Table 3 .1 would be read into scoring groups as the Python array [[7, 4], [7, 2], [0,7]]. For evaluation to work correctly, the run output must exactly match the sequential grouping of the ground truth files, to ensure that these scoring groups represent the same triples across both files.

3.3.2 Accuracy (ACC)

The ACC metric used in the evaluation script is defined in the file as "the percentage of scores that differ by at most the given delta" (Bast, 2016). The calculation function takes two lists of scoring groups, one consisting of calculated values from the triple scorer run, and the other from the ground truths. The accuracy calculation function flattens the scoring groups into 1-dimensional lists of values and *zip*s the values from the run with the truth values. For each pairing, if the values differ by less than or equal to a delta (defined by default as 2 in the evaluation script), the function counts the pair as accurate. The accuracy returned is the ratio of accurate values to the number of evaluated pairs, as shown in Equation 3 .3.

$$ACC = \frac{1}{N} \sum_{i=1}^{N} \left\| run_i - truth_i \right| \leq \delta \left| (where\ \delta = 2)$$

Equation 3.3 Accuracy calculation from evaluator.py

The higher this value, the better.

3.3.3 Average Score Difference (ASD)

The average score difference performs a similar calculation to the accuracy calculation, except instead of averaging Boolean values based on the delta comparison, the differences themselves are averaged, as shown in Equation 3 .4.

$$ASD = \frac{1}{N} \sum_{i=1}^{N} \left| run_i - truth_i \right|$$

Equation 3.4 Average score difference calculation from evaluator.py

The lower this value, the better.

3.3.4 Average Kendall's Tau (TAU)

The Average Kendall's Tau metric computes the p-normalised Kendall's tau ranking, described by Fagin et al. (2004), between each scoring group, and averages it across the scoring groups (Bast, 2016). Bast's code for this metric works as follows:

1. For each scoring group across both sets:
    a. Compute a list of ranks by distributing the scores across buckets of the same score and averaging ties.

       For example, if the values [3, 2, 2, 1] were to be ranked, the ranking without tie averaging would be [4, 3, 2, 1], as this is the order the list appears in when sorted.

       With tie averaging, like this algorithm implements, elements that tie are ranked as the average of their positions in the ranking, so the previous example would be ranked as [4, 2,5, 2.5, 1].

b. As an example run-through of this metric's algorithm, consider the scoring groups [7, 3, 3, 4] and [5, 7, 2, 3], producing ranking groups [4, 1.5, 1.5, 3] and [3, 4, 1, 2].

2. For each ranking group across both sets (with example ranking groups from step 1c):

   a. Combine all possible combinations of indices within the ranking group into pairings $\{i, j\} : i < j$.

      For this example, this would produce the zero-indexed pair set [(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)].

   b. Take the *i*th and *j*th rank from the run and truth ranking group and calculate a penalty.

      i. Case 1: If they differ in both lists, the penalty is 1 if the ordering does not match, else there is no penalty if the ordering does match.
      ii. Case 2: If ranks are the same in both lists, the penalty is 0.
      iii. Case 3: If ranks are the same in one list, but not in the other, the penalty is 0.5.

   c. The TAU value for this ranking group is the average penalty across all pairs.

   For example, the pair (0, 1):

   - Produces combinations (4.0, 1.5) and (3.0, 4.0) from the two ranking groups.
   - As 4.0 > 1.5, but 3.0 < 4.0, the ordering does not match, hence this pair falls under Case 1, incurring a penalty of 1.

   For the pair (0, 2):

   - Produces combinations (4.0, 1.5) and (3.0, 1.0) from the two ranking groups.
   - As 4.0 > 1.5, and 3.0 > 1.0, the ordering matches, incurring no penalty under Case 1.

   For the pair (1,2):

   - Produces combinations (1.5, 1.5) and (4.0, 1.0) from the two ranking groups.
   - As one of the combinations consists of equal ranks, this pair falls under Case 3, incurring a penalty of 0.5.

3. The average TAU values from each ranking group are then averaged again across the set of ranking groups, giving the final Average Kendall's Tau metric.

The lower this value, the better.

3.3.5 Comparisons of metrics

ACC focuses purely on proximities of scores instead of considering the general ranking. This method of evaluation raises a problem of whether truth can ever be assigned an integer score, or whether it is the case that some statements are truer than others. For example, say a particular triple scorer scored the triples (Guillermo del Toro, profession, Director) and (Guillermo del Toro, profession, Actor) as 5 and 2 respectively, but the ground truth scored them 7 and 5. In this situation, the first triple would count towards the ACC metric, as the absolute difference between the scores (|5-7| = 2) is less than or equal to the delta, which is 2 in the default case in *evaluator.py*. However, the second triple's difference (|2-5| = 3) exceeds the delta and therefore is not counted as accurate, even though the ranking itself is the same. In both cases, the triple scorer has ranked Guillermo del Toro as a director over actor, despite varying scores. It is for this reason that it should be strongly considered that ACC is not actually a very useful metric in this scenario, as all it is doing is measuring how close a score is to the ground truth files, not whether the triples are ranked by relative truth.

It is also worth considering what ASD represents. Equation 3 .4 defines the metric as the absolute difference in score for each triple between run and truth, however this also suffers the same problem as ACC, in that it mostly measures proximity to ground truth files, not ranking of truth. Take for example the same triples as in the earlier discussion on accuracy: (Guillermo del Toro, profession, Director) and (Guillermo del Toro, profession, Actor). Say these triples scored the same values as earlier, 5 and 2 in score run, and 7 and 5 in truth. In this case, their ASD would be 2.5. Now let's say that the scorer run swapped the values and rated del Toro as more of an actor than a director (2 and 5 respectively). In this new case, the ASD is still 2.5, even though the triple scorer has now incorrectly ranked the triples. From this, it is clear to see that ASD is yet another metric for measuring the proximity to the ground truth file, and not a metric that can be used to measure relative truth.

4 MODEL DESIGN AND IMPLEMENTATION

In this chapter, each model created in this project is described in detail and explained, along with results of experimentation and discussion regarding evaluation performance of the models. Two models are discussed here: the bag-of-words model, and the semantic relation model. Other points regarding the engineering of the solutions are also discussed.

4.1 INITIAL BASELINES

To form an initial baseline to work from, a constant model was created which returns 5 as the score for every triple, regardless of truth. For this model, the constant value chosen was the same value posited by Bast, et al. (2017) in order to have comparable results to the winning team's solutions.

It is important to note that the results given by Bast, et al. (2017) are calculated from both *profession relation* and *nationality relation* triples as part of the same run, however due to the design of the systems outlined in this chapter, the models described here perform *profession* and *nationality* classifications separately.

To counter this, the ground truth files are concatenated into one overall truth file, and the same concatenation is done to the *nationality* and *profession* run outputs in the same order, creating a single run file for the entire model setup, instead of two separate run outputs per *relation*. Table 4 .2 shows the results of the concatenated model results alongside the baseline results to show that the methodology produces consistent results. It is worth pointing out however, that the *evaluator.py* script formats the output scores to two decimal places. Results tables later in this document will use the two decimal place outputs, padded to three to match the baseline.

	ACC (↑)[2]	ASD (↓)	TAU (↓)
Bast et al.'s baseline	0.721	2.070	0.460
Constant model (concatenated)	0.720	2.070	0.460

Table 4.2 Accuracy, ASD, and Tau results for a constant triple score model

[2] As a reminder, arrows will be used to denote whether a higher or lower score in a metric is better.

4.2 COMMON MODEL SETUP

During initial runs, solutions were unable to be evaluated fully using the WSDM Cup 2017 evaluator script due to efficiency problems with loading the corpus. The provided *wiki-sentences* corpus contains over 33 million sentences, totalling over 4.4GB. As such, loading this file into memory for parsing is relatively intractable on consumer grade hardware.

To help combat this for evaluation purposes, a cache mechanism was implemented to store sentences found matching the *subject*, using Python's *pickle* format. Runtimes from both cached and non-cached runs of the bag-of-words model, described in Section 4.3, are listed below. To show the behaviour of the model, two names were chosen to evaluate: one with a large volume of sentences listed in the corpus, and one with a sparse amount. Both *profession* and *nationality relation* sets have been tested, including various synonyms for the *profession relation* set and other possible *object*s of interest.

	Time taken without cache (s)	**Time taken with cache (s)**
(Ayn_Rand, profession, Author)	174	0.144
(Ayn_Rand, profession, Novelist)	188	0.133
(Ayn_Rand, nationality, Russian)	163	0.145
(Ayn_Rand, nationality, American)	174	0.130
(Costas_Kadis, profession, Minister of Health)	197	0.0170
(Costas_Kadis, profession, Minister)	185	0.00552
(Costas_Kadis, nationality, Cyprus)	194	0.00451
(Costas_Kadis, nationality, Athens)	193	0.00301

Table 4.3 Bag of Words model cache and cacheless running times

As the results in Table 4 .3 suggest, most of the time taken to evaluate a triple is I/O bound, with the non-cached results taking orders of magnitude longer than cached results. Other measures taken to try and improve speed include:

- splitting the dataset into files of one million lines each with the UNIX *split* command and using a multithreaded approach for reading the files concurrently using Python's *ThreadPoolExecutor* class
- storing each individual line as a record in a SQLite3 database and retrieving them with an SQL SELECT statement

Out of both approaches used, the solution currently uses the threaded dataset search method as it provided a slight performance increase, whereas the SQLite3 solution hindered performance. This model setup is common to all models described in this chapter, except the initial constant baseline which doesn't require access to any data. The data stored in the cache has not been normalised or mutated in any way, such transformations are left for each model to perform.

4.3 BAG-OF-WORDS MODEL

The bag-of-words model is based on counting occurrences of the *object* in all documents in the corpus containing the specified *subject*.

The model works as follows, for an example triple (Guillermo del Toro, profession, Director):

1. Under the common model setup, load the corpus consisting of the cached data for the specified triple's *subject* parameter. In the case of the example, the data under cache key "Guillermo_del_Toro" will be loaded.

2. Load the relevant list of candidate *object*s, from the dataset matching the specified *relation*; in this case, as the *relation* was "*profession*", a list of *profession*s will be loaded as candidates.

3. Tokenise all sentences found in step 1, turning them into document vectors. Do the same for all candidate *object*s found in step 2. This step produces a list of document vectors, and a single vector containing every candidate *object*, called the 'candidate vector'.

   For the example, this process will create a list of tokens for every sentence in the corpus containing "Guillermo del Toro", and a list of every profession the system knows about.

4. To prevent a Python *KeyError* in a later step, the requested *object* is appended to the candidate vector in case it doesn't exist. The candidate vector is treated as a *set* datatype by Python, so this extra step does not cause duplication.

   As "Director" is more than likely to already exist in the dataset, it will not appear as a duplicate, however had the model been requested to look for a more esoteric profession, like "Lighthouse Keeper", the model would append it here.

5. The model then iterates over every document vector and calculates a set intersection between it and the candidate vector, producing a list of every candidate found in the document. Each candidate is added to a term frequency dictionary if it doesn't exist, or its value is incremented if it is already present.

   For example, if "Guillermo del Toro" returned 20 sentences from the corpus, and the

word “Director” was found 15 times within them, the term frequency dictionary would contain a {director: 15} key-value pair. Other professions the system knows about will have their occurrences counted too by nature of using a set intersection.

6. At this point, all witnesses to the specified triple have been found and processed, forming what will be dubbed for this project as a ‘witness vector’. This witness vector represents the counts of all *object*s known to the system in its *relation* set found in the document collection, including all 0-values.

7. Three key elements are extracted from the witness vector: the witness minimum value W_{min}, witness maximum value W_{max}, and the instance value $W_{instance}$. These values represent the minimum count found for a candidate element in the witness vector, the maximum count found, and the count found for the requested *object* respectively.

   For example, if the maximum value in the witness vector for “Guillermo del Toro” was 15 (for director), and the lowest was 0, $W_{min,}$ $W_{max,}$ and $W_{instance}$ would be 0, 15, and 15 respectively.

8. To produce a final score for the triple, Equation 4.5 is applied to scale the instance value between the witness maximum and minimum. S_{max} and S_{min} represent the maximum and minimum values to scale the value between, which will be taken as 0 and 7 respectively for this project.

$$score = \lfloor \frac{(S_{max} - S_{min})(W_{instance} - W_{min})}{W_{max} - W_{min}} + S_{min} \rceil$$

Equation 4.5 Simple domain scaling function

9. As per the example, the score would be calculated as follows in Equation 4.6.

$$score(Guillermo\ del\ Toro, profession, Director) = \lfloor \frac{(7-0)(15-0)}{15-0} + 0 \rceil = 7$$

Equation 4.6 Example scoring function for example triple (Guillermo del Toro, profession, Director)

10. If for example another triple was to be scored, such as (Guillermo del Toro, profession, Actor), that found 4 witnesses, Equation 4.7 shows the scoring process.

$$score(Guillermo\ del\ Toro, profession, Actor) = \lfloor \frac{(7-0)(4-0)}{15-0} + 0 \rceil = 2$$

Equation 4.7 Example scoring function for example triple (Guillermo del Toro, profession, Actor)

4.3.1 Results and Analysis

Two iterations of the bag-of-words model were evaluated: a version using binary term weighting, and a version using term frequency weighting.

	ACC (↑)	ASD (↓)	TAU (↓)
Bast et al.'s baseline	0.721	2.070	0.460
Bag-of-words (TF Weighting)	0.500	2.780	0.380
Bag-of-words (Binary Weighting)	0.460	3.080	0.450

Table 4.4 Accuracy, ASD, and Tau results for the Bag of Words triple score model

As shown by Table 4.4, the bag-of-words model is flawed in all metrics except TAU across both weightings. There may be any number of reasons for this, however the most likely reason is the granularity of the witness capture method.

The bag-of-words model captures witnesses as exact matches of the *object* word in the corpus. At no point are synonyms or related vocabulary considered, making the solution work similarly to a naïve full-text search engine. By capturing witnesses this way, words that are evocative of what is being searched for are ignored, which vastly restricts the scope of valid witnesses, for instance a profession such as "scientist" would only be captured if the term "scientist" appeared exactly in the corpus, even if terms such as "physicist" or "biologist" also appeared.

There is also the issue of how the terms detected are weighted in the score. Of the three methods of term weighting discussed in Section 2.3.2, term frequency weighting performs better and hence is chosen to be the main scheme used in this model, as it represents a midpoint between complexity and accuracy. Using binary weighting negatively affects ACC and causes a large increase in ASD, but is by far the simplest computationally, as iteration over the corpus stops when an instance is found, or when the end of the corpus has been reached. This however has large implications when used with the scaling function in Equation 4.5, as W_{min}, W_{max}, and $W_{instance}$ will always be either 0 or 1, meaning every score will either be 0 or 7.

TF.IDF weighting could in theory provide the best ACC as it weights terms proportionally to their significance in the text, however this method would cause significant performance problems due to the size of the corpus. Therefore, an appropriate middle ground would be to use term frequency, which is still computationally simple, but provides more information on the significance of terms.

Given these flaws, it is clear that a number of changes need to be made to this model to improve its performance. To compensate, a model is needed that can capture semantically related vocabulary, and detect and weight occurrences appropriately.

4.4 Semantic Relation Model

To fix the problems discovered in the previous model, a model is created that analyses the underlying concept of associated texts instead of the raw surface vocabulary. The model shares a

similar workflow as the bag-of-words model, with some differences regarding the data that the model takes as input and how witnesses are captured.

4.4.1 Feature Reduction and Re-expansion

The semantic relation model requires a specific form of input data that needs to be generated from the source text prior to scoring, a process that will be referred to as 'learning', despite no links to machine learning or learning in general. As part of the initial learning step, the source text needs to be normalised to ensure only relevant data is learned; this involves standard NLP normalisation techniques such as capitalisation and punctuation removal, stop-word removal, and lemmatisation of all remaining words, as defined in Section 2.2.

At this point, the semantic relation model distils the remainder of the corpus down to the most significant vocabulary via Latent Dirichlet Allocation. Using topic modelling at this stage can be seen to be akin to feature/dimensionality reduction, where a term in the corpus is approximated as a feature, and allows the core meaning of the text's content to be maintained, but in markedly less words.

The concept of 'significant terms' is not defined in the context of LDA, because topics and documents in an LDA model of a text are mixtures of terms by probability of occurrence. Instead, this dissertation defines 'significant terms' to be all of the non-zero probability terms in an LDA model where the number of topics and terms per topic has been restricted. In the implementation of this model used to generate results, each person's corpus LDA model is restricted to only find 5 topics, each consisting of 5 terms. The 5 terms from each of these 5 topics constitute the model's 25 significant terms.

After reducing the corpus with LDA, the refined corpus is expanded to include synonymous vocabulary. To do this, a vector space model is used, trained on a different corpus than the original to allow new vocabulary to be introduced into the model. For each word in the reduced corpus, synonyms are found by measuring the cosine distance between the target word's vector and the vector of every other word in the vector space model. The words with the largest cosine similarity are counted as synonyms and are added to the thesaurus. Error: Reference source not found shows an example thesaurus generation from a raw unnormalised corpus, with significant LDA topics detected shown in bold text.

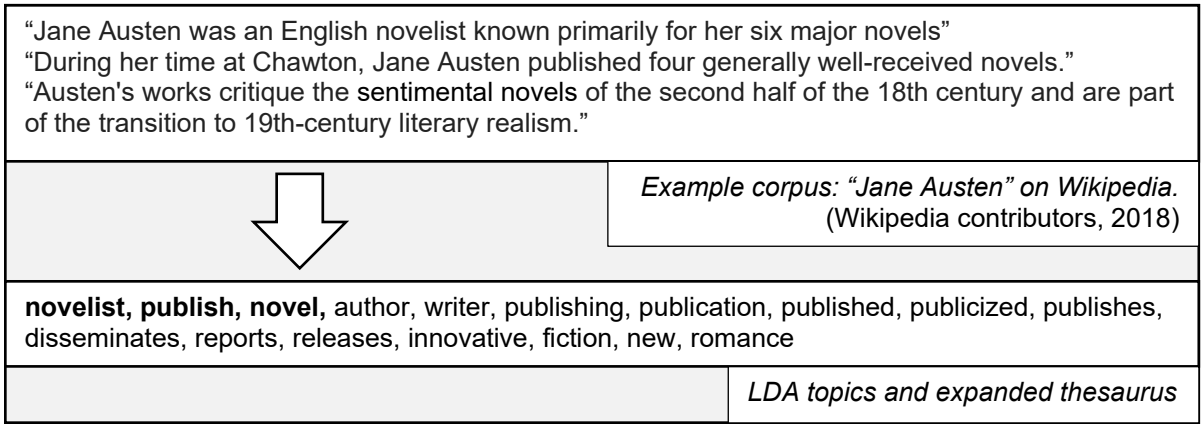


Figure 4.1 An example corpus reduction and expansion (CVM generation)

In addition to expanding each person's LDA topic vocabulary, the *relation*al vocabulary itself is also expanded with a vector space model, to increase the number of *profession* or *nationality* words that count as witnesses. These synonyms are structured in a map-like data structure, where each *relation*al term maps to a vector of its synonyms, described in Error: Reference source not found.

4.4.2 Witness Capture Methodology

In the model described in Section 4.3, witnesses were captured by a simple set intersection between *relation*al vocabulary and corpus text, which proved to be too simple a capture method for such a small search space. However, in a thesaurus-based model such as this, with a much larger search space, set intersection can be seen to be a fairly performant method.

Additionally, however, other methods of capturing witnesses exist. As the model has access to a vector space model, use of a cosine similarity metric between the vectors of each pairing of *relation*al terms and expanded thesaurus terms can be experimented with. This will be dubbed "vectorial capture" for this dissertation. For this kind of capture method, a similarity threshold is defined experimentally, by analysing the cosine similarities between the term pairs and using trial-and-error to find a suitable cut-off point where words become irrelevant.

threshold = 0.65
('novelist', 'write') 0.4231099481635837
('novelist', 'editor') 0.6015073739230115
('novelist', 'journalist') 0.7515744523310355
('novel', 'writer') 0.6618862733492705
('novel', 'screenwriter') 0.6350188759017672
('novel', 'author') 0.8129903556154872

Figure 4.3 Example of cosine similarity analysis for WC term pairs

Error: Reference source not found shows an example of such analysis, using GLoVe 6B vectors as the vector space model and a similarity threshold of 0.65. At this threshold, the term "novel" counts as a witness to the professions "writer" and "author", but not "screenwriter", whereas the term "novelist" counts only towards "journalist", which may be interpreted as a false capture, as human judgment would show that the two words aren't generally evocative of each other.

For the implementation of the model discussed in this project, the vectorial capture similarity threshold was set to 0.55.

4.4.3 Handling multiple word terms

The methods described in Sections 4.4.1 and 4.4.2 work for single word terms, but a problem arises when handling *relation*al terms that consist of multiple words. For instance, the term "United States of America" in the *nationality* relation will not expand into synonyms, by limitation of the GLoVe vectors. This leads to decreased capture rates for multiple word terms, which is unacceptable for this model, as it reverts to the behaviour of the bag-of-words model.

To remedy this, multi-word phrases are treated during thesaurus generation as their own separate set of *sub-terms*. During thesaurus generation, single word terms' synonyms are referred to as *super-terms*, so for instance the super-terms of the term "author" from Error: Reference source

not found are {author, writer, creator, …}. When a term consists of multiple sub-terms, the model treats them as their own terms, and generates a set of super-terms for each sub-term. These terms are concatenated into a single list, which is then treated as the term-as-a-whole's synonyms.

A problem however does arise from this process, in that multiple word terms will have a list of synonyms much larger than any other term, allowing multiple word terms to dwarf single terms, leading to false captures. To counter this further, at the witness capture stage, the weighting added to the running total of a term's witnesses is divided by the *sub-term* count. For instance, the term "United States [of] America" would have 3 *sub-terms* expanded, leading to a list of synonyms 3 times larger than others, therefore the weighting that adds to the final total will be *weighting/3*, to prevent the term from dwarfing over others.

4.4.4 Model Workflow

To outline the model's workflow, an example triple is used to demonstrate: (Nicolas Cage, profession, Actor):

1. For each person's corpus, train a Latent Dirichlet Allocation model and from the 'most significant' topics: extract the 'most significant' terms, creating a reduced corpus that represents the underlying concept of each person's full corpus.

   For the example triple, the LDA model may return terms such as ["actor", "movie", "film", "performance", "theatre", …], etc.

2. For each term in the reduced corpus, expand each into a set of semantically related vocabulary using a vector space model trained on a corpus other than the initial training text. This flat set of words is dubbed the 'corpus vocabulary model' (CVM).

   For this implementation, Stanford NLP's GLoVe 6B vectors were used, trained on a 2014 crawl of Wikipedia, and Gigaword 5, an archive of international newswire text (Pennington, et al., 2014; Parker, et al. 2011). This vector space model provides a number of different dimensionality sets, and all of these were experimented with.

   The example LDA 'significant' terms from the previous step creates the following CVM using 50-dimensional GLoVe 6B vectors: ['starring', 'starred', 'actress', 'comedian', 'filmmaker', 'screenwriter', 'comedy', 'film', 'entertainer', 'actors', 'movies', 'film', 'films', 'comedy', 'hollywood', 'drama', 'sequel', 'animated', 'remake', 'show', 'movie', 'films', 'documentary', 'drama', 'comedy', 'directed', 'movies', 'acclaimed', 'adaptation', 'comic', 'performances', 'success', 'best', 'impressive', 'quality', 'excellent', 'dramatic', 'achieved', 'performing', 'remarkable', 'theater', 'opera', 'ballet', 'cinema', 'orchestra', 'productions', 'repertory', 'theatres', 'studio', 'broadway'].

3. For the list of *relation*al vocabulary loaded from the triple's *relation* (either *profession* or *nationality*), concatenated with the *object* of the triple if it is not already present in the list, create a mapping between each term and a list of semantically related words, generated from the same vector space model used to create the model from step 2. This will be referred to as the *relation*al vocabulary model (RVM).

   For the *profession relation*, the RVM generated with 50-dimensional GLoVe 6B vectors would be similar to:

   {"actor": ['starring', 'starred', 'actress', 'comedian', 'filmmaker', 'screenwriter', 'comedy', 'film', 'entertainer', 'actors'],
   "director": ['executive', 'assistant', 'chief', 'associate', 'managing', 'consultant', 'dr.', 'co', 'vice', 'noted'],
   "producer": ['producers', 'production', 'produced', 'songwriter', 'distributor', 'singer', 'co', 'musician', 'co-executive', 'reynolds']}

4. For a given (*subject, relation, object*) triple, retrieve the *relation*'s RVM, and the *subject*'s CVM, and generate the Cartesian product of the two, representing the witness candidate pairs, as follows in Equation 4.8.

$$WC = RVM_{rel}[obj] \times CVM_{subj} \; for \; each \; obj \in RVM_{rel}$$

Equation 4.8 Witness candidate relation definition

   For the example, pairings would include:

   {"actor": [('starring', 'starring'), ('actress', 'starred'), ('film', 'actress'), ('comedian', 'comedian'), …], …}

5. For each witness candidate in **WC**, measure and sum the similarity between both words (methods discussed in Section 4.4.2) and assign it to its corresponding *object*.

6. Apply a scaling function, such as defined in Equation 4.5, over all scores found to restrict them to the correct range and return the score for the specified *object*.

4.4.5 Results and Analysis

Analysis of this model is more complex than bag-of-words. The scope of experimentation with this model was so large that multiple runs were performed with a variety of different parameters. For instance: varying dimensionality vector models, and absence of LDA.

	ACC (↑)	ASD (↓)	TAU (↓)
Bast et al.'s baseline	0.721	2.070	0.460
Bag-of-words (TF Weighting)	0.500	2.780	0.380
Semantic relation model (50d vectors, intersection capture)	LDA: 0.530 Raw: 0.520	LDA: 2.610 Raw: 2.670	LDA: 0.480 Raw: 0.560
Semantic relation model (100d vectors, intersection capture)	LDA: 0.540 Raw: 0.520	LDA: 2.580 Raw: 2.660	LDA: 0.470 Raw: 0.560
Semantic relation model (200d vectors, intersection capture)	LDA: 0.550 Raw: 0.520	LDA: 2.530 Raw: 2.630	LDA: 0.490 Raw: 0.560
Semantic relation model (300d vectors, intersection capture)	LDA: 0.560 Raw: 0.510	LDA: 2.520 Raw: 2.610	LDA: 0.450 Raw: 0.550
Semantic relation model (50d vectors, vectorial capture)	LDA: 0.450	LDA: 3.040	LDA: 0.440

Table 4.5 Accuracy, ASD, and Tau results for the Semantic Relation triple score model

From the data in Table 4 .5, a few interesting results can be seen. First, there is a dramatic loss of precision in general when Latent Dirichlet Allocation is disabled, indicating that the more terms there are in the corpus, the less precise thesaurus generation becomes. Deerwester et al. (1990) attribute this type of precision loss to the polysemy effect, inherent in unsupervised thesaurus generation, where terms in one context will have a different meaning than intended when taken into a different context. While terms in the initial corpus would be in the correct context, they become ambiguous after normalisation, therefore the corpus must be restricted to as fewer terms as possible, while retaining the core topics of the text.

This reduction step has large effects on performance and accuracy of the generated corpus. By reducing the number of source features, a much smaller thesaurus is generated, thus creating a much smaller set of witness candidates to measure the similarity of, which may be advantageous when used in conjunction with more computationally expensive similarity measures such as vectorial capture. The step also has the advantage of removing words that are irrelevant that

could be expanded out in the thesaurus generation stage, which would expand the scope of capture towards wrong vocabulary, leading to false captures.

Certain terms in the expanded thesaurus in Error: Reference source not found show the impact of the polysemy effect, where the term “novel”, out of the context of authorship, has been interpreted as a synonym of ‘new’, leading to words such as “new” and “innovative” being generated.

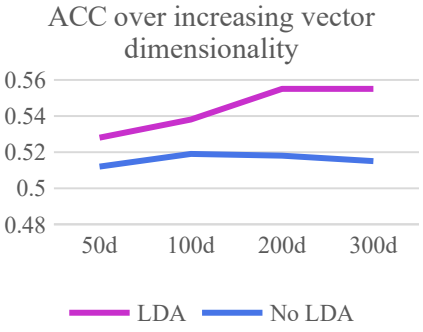


Figure 4.1 Graph showing ACC changes in LDA and non-LDA model with respect to vector dimensionality

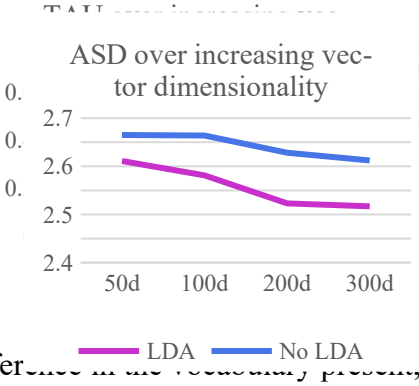


Figure 4.2 Graph showing ASD changes in LDA and non-LDA model with respect to vector dimensionality

Secondly, an increase in ACC is seen, along with a decrease in ASD and TAU as the dimensionality of the GLoVe vectors increases, as emphasised in Figures 4.4, 4.5, and 4.6. As the vector n the same dataset there is no differ meaning higher dimensionality word vectors provide a more accurate distance between the vectors of alike terms. This effect can be seen as an analogue of dimensionality reduction techniques in other fields of computer science, where lossy conversion of higher dimensional data points into lower dimensions causes a loss of precision.

A drawback of using high-dimensional vectors however is the computational power needed to perform calculations with them. Due to this system being run on consumer-grade hardware, training processes such as LDA generation and thesaurus generation occurred in a stop-start nature, therefore exact timings are not available, however a run of the 300-dimensional vectors without LDA took roughly 40 hours to finish training over all test triples, whereas 50-dimensional vectors without LDA took around an hour.

It should also be noted, from Table 4 .5, that there is only one instance of a run using the vectorial capture method as described in Section 4.4.2. As evident by its results, it performs phenomenally poorly compared to the set intersection methods, achieving a vastly lower ACC and higher ASD. There may be any number of reasons as to why this has occurred, considering that on paper, measuring semantic similarity between words instead of comparing their strings is a much better system. It may be that the similarity metric was either too high, leading to a complete lack of captures, or too low, leading to too many captures causing too many irrelevant terms to be detected as witnesses, which dampens the impact correct captures have on the mapping function.

Other reasons may include the effect of pairing the CVM and RVM as a Cartesian product. By doing this, there are |CVM|*|RVM| pairs for each relation, causing a large number of measurements to be accumulated. This accumulation means that a relation with pairs that score

consistently mid to low similarities has the same weight overall as another pairing with consistently low to zero similarities that contains one or two pairs that spike higher.

4.5 REFINEMENTS

While both models created so far fail to reach the baseline as set by Bast, et al. (2017), there are some improvements that can be experimented with without drastically altering the models. These improvements can be implemented at score-time, so affect only the witness to score mapping, not the model itself.

4.5.1 "The 2-5 Trick"

Bast, et al. (2017) describe a method known as the "2-5 Trick", where all scores 0 and 1 are replaced with 2, and all scores 6 and 7 are replaced with 5, restricting the score interval from [0,7] to [2,5] without scaling scores linearly. Bast, et al. (2017) write that the ACC measure can only increase with application of this method, however other measures suffer, which is still advantageous since the conditions for winning the competition was based on the ACC metric. However, as seen by Table 4 .6, loss in other metrics is not actually the case for the models described here.

	ACC (↑)	ASD (↓)	TAU (↓)
Bast et al.'s baseline	0.721	2.070	0.460
Ding et al.'s BOKCHOY scorer	0.868	1.630	0.327
Bag-of-words (TF Weighting)	0.500	2.780	0.380
Bag-of-words (TF Weighting, 2-5)	0.660	2.210	0.380
Semantic relation model (300d vectors, intersection capture, LDA)	0.560	2.520	0.450
Semantic relation model (300d vectors, intersection capture, LDA, 2-5)	0.660	2.140	0.450
Semantic relation model (50d vectors, vector capture, LDA)	0.450	3.040	0.440
Semantic relation model (50d vectors, vector capture, LDA, 2-5)	0.570	2.430	0.440

Table 4.6 Results of all models compared to baseline and best WSDM Cup 2017 solution with the "2-5 trick" applied

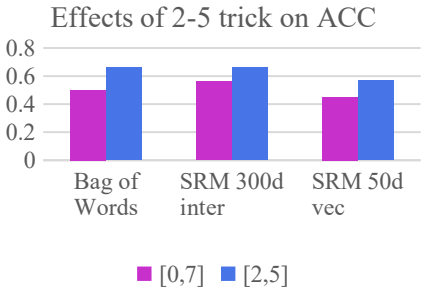


Figure 4.4 Graph comparing ACC metric for all models with and without 2-5 trick applied

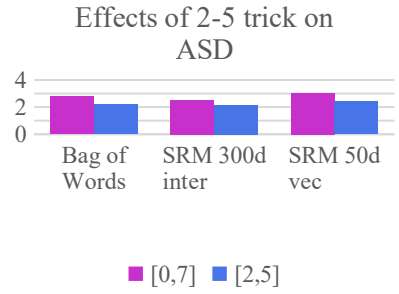


Figure 4.5 Graph comparing ASD metric for all models with and without 2-5 trick applied

Figure 4.6 Graph comparing TAU metric for all models with and without 2-5 trick applied

As seen by the data in Table 4 .6, and Figures 4.7, 4.8, and 4.9, all metrics in all model runs either improved (ACC and ASD) or remained the same (TAU). For the most performant semantic relation model (300d GLoVe 6B vectors, intersection capture) and the bag-of-words model, the accuracies make a sudden jump to become equal, to the point where from an accuracy point of view, there is actually no benefit to using the semantic relation model over bag-of-words. This point is further exacerbated by the difference in TAU; the bag-of-words model provides more accurate ranking than the semantic relation model with a lower TAU metric. With the 2-5 trick applied, the bag-of-words model provides comparable ACC to the best performing SRM run, better TAU ranking, near equal ASD, and uses a much simpler algorithm with less learning steps.

In terms of ASD however, the 300-dimensional semantic relation model outperforms bag-of-words, indicating that the 2-5 trick moves values closer to their ground truths, but not close enough to warrant an increase in ACC.

4.5.2 maplin, maplog, and mapscale

Bast, et al. (2015) and Ding, et al. (2017) posit three score mapping strategies that differ from the mapping function defined in Equation 4 .5. Ding, et al. (2017) employ these functions in different steps of the model run depending on the *relation* being tested.

$$maplin(s) = \lfloor \frac{s}{s_{max}} * 7 \rfloor$$

Equation 4.9 Bast, et al. (2015)'s maplin scaling function

$$maplog(s) = \lfloor max \left\{ 0, \log_2 \left(\frac{s}{s_{max}} * 2^7 \right) \right\} \rfloor$$

Equation 4.10 Bast, et al. (2015)'s maplog scaling function

$$mapscale(s) = \lfloor 8s - \epsilon \rfloor$$

Equation 4.11 Ding, et al. (2017)'s mapscale scaling function

Ding, et al. (2017) describe these three mapping strategies, and their respective uses in various steps of their own model.

maplin (Equation 4.9) is a linear scaling function and is used in Ding, et al. (2017)'s 'word counting' step as described in Section 2.1.1. Bast, et al. (2015) describe this strategy as plain division of all scores or probabilities by the maximum, followed by scaling into the [0, 7] range.

maplog (Equation 4.10) is a logarithmic scaling function that Ding, et al. (2017) use in their 'word MLE' step, and all *nationality* relation steps except 'word counting'. This function has the caveat that it can sometimes drop below zero, in which case it is rounded back up to zero, hence the need for $max\{0, x\}$ (Ding, et al., 2017).

The final function *mapscale* (Equation 4.11) is used by Ding, et al. (2017) for *profession* relation 'word classification' and 'path ranking' steps. This function is used on probabilities only, not on count-based scores as the models described in Chapter 4 use. For this reason, to attempt to experiment with these functions, *mapscale* experiments will average the counts to create probabilities.

For brevity's sake, only models of interest are experimented on: the bag-of-words model, and the semantic relation model with 300-dimensional GLoVe 6B vectors and intersection capture. As it has provided significant improvements across all metrics so far, the 2-5 trick as described in Section 4.5.1 will be used.

	ACC (↑)	ASD (↓)	TAU (↓)
Bast et al.'s baseline	0.721	2.070	0.460
Ding et al.'s BOKCHOY scorer	0.868	1.630	0.327
Bag-of-words (TF Weighting, 2-5)	0.660	2.210	0.380
Bag-of-words (TF Weighting, 2-5, maplin)	0.680	2.170	0.380
Bag-of-words (TF Weighting, 2-5, maplog)	0.760	1.930	0.420
Bag-of-words (TF Weighting, 2-5, mapscale)	0.690	2.140	0.380
Semantic relation model (300d vectors, intersection capture, LDA, 2-5)	0.660	2.140	0.450
Semantic relation model (300d vectors, intersection capture, LDA, 2-5, maplin)	0.660	2.180	0.460
Semantic relation model (300d vectors, intersection capture, LDA, 2-5, maplog)	0.720	1.990	0.460
Semantic relation model (300d vectors, intersection capture, LDA, 2-5, mapscale)	0.670	2.130	0.460

Table 4.7 Results of maplin, maplog, and mapscale experimentation

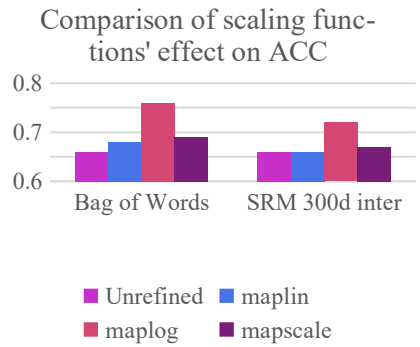


Figure 4.7 Graph comparing ACC metric for all models under different score scaling functions

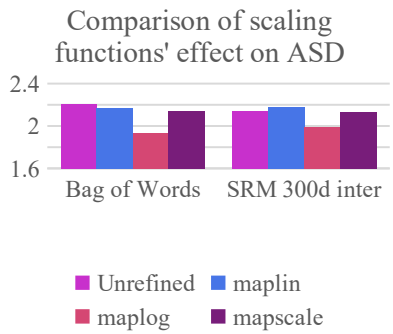


Figure 4.9 Graph comparing ASD metric for all models under different score scaling functions

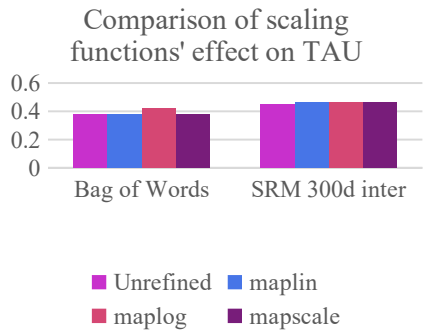


Figure 4.8 Graph comparing TAU metric for all models under different score scaling functions

Table 4 .7, and Figures 4.10, 4.11, and 4.12, show some interesting results from the mapping function experimentation. For the most part, metrics seem to improve when other functions to map to scores are applied, particularly in the bag-of-words model. When using this model alongside Bast, et al. (2015)'s *maplog* function, an increase in ACC well over the baseline is seen, pushing the solution into the realms of being a competitive model, as this solution now outperforms Yahoo! Japan's 'Cauliflower' triple scorer at 0.752 ACC (8th place) (Bast, et al., 2017). In terms of the semantic relation model, a marked increase in ACC is seen again, however it does not meet the baseline, albeit by an incredibly small amount. TAU does not suffer as much under this model and mapping scheme as bag-of-words did. Both models saw an effect to TAU under *maplog*, however bag-of-words exhibits a very large hit to TAU, whilst still being competitive against semantic modelling.

4.6 FINAL MODEL DESCRIPTION

The final model is the bag-of-words model, using Bast, et al. (2015)'s *maplog* score mapping function, as well as their "2-5 trick". This model and setup performs well over baseline, to the point where it could feasibly take 8th place out of 21 submissions to WSDM Cup 2017 in terms of ACC. In terms of the other metrics, this model does not place in the top 10, however it still beats the baseline in ASD, and matches it in TAU.

This model operates as so:

1. Load a corpus containing sentences relevant to the triple's *subject*, and tokenise, creating a set of document vectors.
2. Load a list of candidate *objects* from a *relation*al *object* set, append the *object* if it is not already part of the set, and transform into a single vector (*candidate vector*).
3. Iterate the set of document vectors and calculate an intersection between each and the *candidate vector*. Increment the value of each found term in a term frequency dictionary keyed by candidate, producing the *witness vector*.
4. Extract two values from the *witness vector*: s, the count of occurrences of the requested *object* in the *candidate vector*, and s_{max}, the largest value in the *witness vector*.
5. Calculate the score with *maplog,* restricting to the range [2, 5], as shown in Equation 4 .12.

$$score(s, s_{max}) = max\left\{2, min\left\{5, \lfloor \log_2\left(\frac{2^7 s}{s_{max}}\right)\rfloor\right\}\right\}$$

Equation 4.12 Hybrid 2-5+maplog mapping function

4.6.1

5 DISCUSSION

In this chapter, the two models created are discussed, and results are compared against the baseline and winning solution.

5.1 ANALYSIS OF THE UNREFINED MODELS

In general, without refinement, both models performed reasonably well, and in some cases better than expected, despite not beating or even matching the baseline.

	ACC (↑)	ASD (↓)	TAU (↓)
Bast et al.'s baseline	0.721	2.070	0.460
Ding et al.'s BOKCHOY scorer	0.868	1.630	0.327
Bag-of-words (TF Weighting)	0.500	2.780	0.380
Semantic relation model (300d vectors, intersection capture, LDA)	0.560	2.520	0.450
Semantic relation model (50d vectors, vector capture, LDA)	0.450	3.040	0.440

Table 5.8 Results of unrefined models compared to baseline and best WSDM Cup 2017 solution

As shown by Table 5 .8, both models created as part of this project matched neither the baseline ACC nor ASD in their unrefined state, however both solutions did achieve significantly better results in the TAU metric.

In terms of the bag-of-words model, it was expected initially that it would not be the most performant of models, given how limited it is in its capabilities. The model uses a very restrictive capture function and a very small *relation*al vocabulary set, so it can be expected that it does not capture witnesses very often. However, it is surprising to see how poorly the semantic relation model performs especially when a vectorial capture method is applied as detailed in Table 4 .5 and Table 5 .8.

Theoretically, the semantic relation model should improve ACC significantly, as the search space in which witnesses can be searched for has increased drastically, with introduction of new vocabulary from the vector space models that might not necessarily be present in the original corpus. Applying a vectorial capture function should further increase performance in the evaluation step, as the system would be comparing words based on their probabilities of appearing together in real text, as opposed to comparing their strings. However, it's clear that this is not the case.

5.2 ANALYSIS OF THE REFINED MODELS

With refinements, such as Bast, et al. (2017)'s 2-5 trick and Bast, et al. (2015)'s *maplog*, the models perform significantly better, despite no changes being made to the underlying process or data.

	ACC (↑)	ASD (↓)	TAU (↓)
Bast et al.'s baseline	0.721	2.070	0.460
Ding et al.'s BOKCHOY scorer	0.868	1.630	0.327
Bag-of-words (TF Weighting)	0.500	2.780	0.380
Bag-of-words (TF Weighting, 2-5)	0.660	2.210	0.380
Bag-of-words (TF Weighting, 2-5, maplog)	0.760	1.930	0.420
Semantic relation model (300d vectors, intersection capture, LDA)	0.560	2.520	0.450
Semantic relation model (300d vectors, intersection capture, LDA, 2-5)	0.660	2.140	0.450
Semantic relation model (300d vectors, intersection capture, LDA, 2-5, maplog)	0.720	1.990	0.460

Table 5.9 Results of refined models compared to baseline and best WSDM Cup 2017 solution

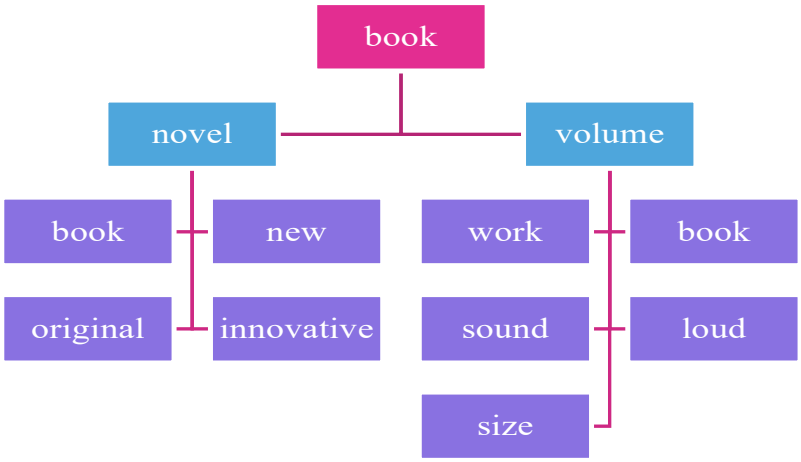
As shown in Table 5 .9, the mapping function from raw data to an integer score has a huge effect on all metrics. Restricting the range of potential scores from [0, 7] to [2, 5] with Bast, et al. (2017)'s 2-5 trick increases ACC and reduces ASD with no effect on TAU. After this restriction, both bag-of-words and the semantic relation model become equally as accurate, however the semantic relation model becomes more performant in ASD, and bag-of-words becomes more performant in TAU. At this stage, the decision between which model is best falls to whether truth ranking (TAU) or proximity to ground truth (ASD) is considered more important for the use case.

Further refinements such as application of Bast, et al. (2015)'s *maplog* mapping function improves ACC and ASD again for both models, however both models take a performance hit in the TAU metric, indicating that the function rearranges the scores in such a way that the ranking is degraded, but scores on average move closer to their ground truths. In terms of ACC, at this stage, the bag-of-words model overtakes the semantic relation model, to the point of becoming a competitive solution over the baseline. As the WSDM Cup 2017 competition measures the success of a submission by its ACC result, the bag-of-words model would have been a successful submission, achieving 8th place out of 21 submissions.

5.3 ISSUES

The two models both suffer from large efficiency problems that make the models ineffective for any sort of real world use. In both cases, the caching process required to be able to even access the 4.4GB of corpus data in a reasonable time required hours of multi-threaded searching and caching over a set of split data files. For the semantic relation model, additional generation steps were needed per cached person to generate usable datasets. A single 'learning' step for a person consisted of: tokenisation of each sentence, stop-word removal, normalisation, lemmatisation, Treebank tag filtering, LDA model generation, and thesaurus generation. While some of these steps are simple, LDA generation and thesaurus generation cumulatively took over 48 hours to complete for the set of persons needed for evaluation.

There is also the problem of the effects of polysemy and synonymy in the English language. These effects can be seen as one-to-many and many-to-one relationships between concept and words respectively, and as the English language exhibits both these properties, English can be seen as a many-to-many mapping between concepts and words, making it hard to follow a synonymic path between words without getting lost in a different conceptual view of the word



altogether.

Figure 5.10 Effect of uncontextualised synonym generation

Figure 5 .10 shows the possible effects of generating synonyms without context. Without knowledge of the concept a term represents, synonyms generated could be representative of an entirely different concept that the same word represents. This has the effect of cluttering the set of semantically related terms with words that have no relevance to the original terms, where the only criterion for the term being generated is a tenuous polysemic link. An actual example of this is in the LDA term model for the *subject* "Lady_Gaga". Given that the term "monster" is so prevalent in the individual *subject* corpus, as many aspects of the artist's career such as albums and tours are named after the term, the LDA reduction step produced the term as a prominent topic parameter. This term undergoes expansion with the vector space model, leading to terms

such as “frankenstein”, “godzilla”, and “dragon” being included in the CVM, as shown in Figure 5.2.

```
>>> lda_terms
['video', 'wear', 'dress', 'include', 'music', 'black', 'feature', 'costume', 'celebrity', 'appear', 'song', 'album', 'ar
tist', 'number', 'feature', 'single', 'include', 'record', 'music', 'release', 'performance', 'vocal', 'song', 'live', 'f
an', 'stage', 'perform', 'producer', 'version', 'piano', 'song', 'say', 'sing', 'video', "n't", 'feel', 'music', 'love',
'pop', 'good', 'way', '2011', 'bear', 'tour', 'album', 'perform', 'monster', 'release', '2014', 'concert']
>>> cvm
['hollywood', 'really', 'demon', 'bore', 'tunes', 'monster', 'touring', 'always', 'other', 'musical', 'updated', 'voice',
 'broadcast', 'includes', 'might', 'songs', 'clips', 'winning', 'just', 'piano', 'some', 'animated', 'say', 'saxophone',
'wo', '2008', 'performances', 'original', 'art', 'impressive', 'modified', 'living', 'perform', 'ballad', '2004', 'consec
utive', 'shows', 'loved', 'orchestral', 'lives', 'bears', 'gray', 'improved', 'creature', 'why', 'pianist', 'white', 'tou
rs', 'something', 'toured', 'do', 'cello', 'tour', 'fan', 'even', 'history', 'pga', 'third', 'stages', '2018', 'artist',
'oprah', 'video', 'chorus', 'musicians', 'anything', '2010', 'rabid', 'performers', 'song', 'very', 'dark', 'felt', 'show
', 'romantic', 'costume', 'celeb', 'going', 'overall', 'better', '2015', 'guitar', 'composer', 'good', 'creatures', 'pain
ter', 'vocal', 'these', 'movies', "'re", 'well', 'grizzly', 'bearing', 'quality', 'things', 'recital', 'many', 'recorded'
, 'stage', 'announced', 'clarinet', 'concert', 'how', 'teddy', 'definitely', 'feel', 'believe', 'fans', 'second', 'produc
ers', 'did', 'celebrities', 'participate', 'distributor', 'pants', 'villain', 'footage', "n't", 'vocals', 'variant', 'num
bers', 'gowns', "'m", 'introduced', 'sung', 'chef', 'monstrous', 'appear', 'cosi', 'feature', 'gala', 'although', 'sings'
, '2014', 'ep', 'producer', 'addition', 'outfits', 'digital', 'black', 'enthusiastic', 'not', 'place', 'dvd', 'including'
, 'notably', 'me', 'painting', 'featured', 'stearns', 'wears', 'lion', 'beast', '2011', 'various', 'tune', 'next', 'so',
'2006', '2003', 'performs', 'each', 'album', 'harpsichord', 'produced', 'rock', 'singles', 'career', 'audience', 'lovers'
, 'indie', 'dog', 'record', 'those', 'blue', 'lover', 'remake', 'another', 'musician', 'wore', 'skirt', 'wolf', 'profile'
, 'tasks', 'freed', 'hats', 'required', 'soundtrack', 'year', 'way', 'releases', 'superstar', 'breaking', 'performance',
'arranger', 'deer', 'reworked', 'self-titled', 'camera', 'crowd', 'favorite', 'prisoners', 'dressed', 'love', 'film', 'dr
esses', 'single', 'such', 'seem', 'several', 'compilation', 'yellow', 'features', 'tournament', 'colored', 'fashion', 'li
ved', 'worn', 'number', 'pop', 'versions', 'affection', 'orchestra', 'featuring', 'costumes', 'excellent', 'theater', 'pr
oducing', '2009', 'violin', 'best', 'released', 'exporter', 'staging', 'godzilla', 'artists', 'besides', 'wear', 'product
ion', 'illustrator', 'glamour', 'celebrity', 'instrumental', 'wardrobe', 'red', 'live', 'youtube', '2007', 'passion', 'fi
nal', 'performing', 'harmonies', 'bear', 'passionate', 'singers', 'albums', 'r&b', 'shirts', 'one', 'latest', 'notable',
'rap', 'first', 'because', 'brown', 'edition', '2017', 'gossip', 'screenwriter', 'music', 'sculptor', 'festival', 'record
ing', 'what', 'wearing', 'jazz', 'version', 'loves', 'uniforms', 'know', 'make-up', '2012', 'clothes', 'double', 'appeare
d', 'include', 'only', 'monsters', 'films', 'though', 'videos', 'concerto', 'singer', 'appears', 'able', '2013', 'but', '
feels', 'performed', 'likely', 'videotape', 'feeling', 'we', 'band', 'audio', 'movie', 'you', 'appearing', 'attire', 'pre
miere', 'singing', 'trip', 'donning', 'event', '2005', 'phase', 'dance', 'release', 'dragon', 'voices', 'have', 'it', '20
16', 'they', 'avid', 'dress', 'songwriter', 'lpga', 'frankenstein', 'singer-songwriter', 'think', 'releasing', 'previous'
, 'supporter', 'creator', 'where', 'people', 'included', 'gown', 'flute', 'concerts', 'sang', 'lyrics', 'sing', 'routine'
, 'want', 'obviously', 'records', 'are']
```

Figure 5.11 Effects of polysemy and synonymy on a real world corpus example for triple scoring subject “Lady_Gaga”

Despite these issues, it is worth noting that every refined model created as part of this project was able to beat the baseline score in terms of TAU, meaning that while the models were not able to match the ground truth scores exactly, they were able to give the triples scores that ranked them more appropriately than the baseline constant scoring. As discussed in Section 3.3.5, it should be considered that these first two metrics are only checking the proximity of a run’s results to the ground truth files, not the ranking, which is covered by average Kendall’s tau. In this case, it may be the case that these models did in fact succeed, as they were able to appropriately rank triples by how true they actually were, despite not scoring them perfectly. It is also worth noting that the ground truth scores were judged by a panel, and that for this reason, the truth represented in these files is wholly subjective, and not based on any sort of scientific metric for quantifying truth.

5.4 IMPROVEMENTS

There are a number of different improvements that could be made across both models. The main issue surrounding both models is the effective capture of semantically related words, which the semantic relation model tried and failed to do. Further experimentation can be done with this

model, including usage of different or even multiple vector space models to increase the range of vocabulary that can be introduced into a model. Methods could be developed to guide the synonymic path the thesaurus generation step takes, to lessen the number of irrelevant synonyms that are generated and to quell the problems caused by synonymy and polysemy.

Different ways of weighting terms could be used, such as TF.IDF, to add heavier weighting to more prominent terms, which could be used in both models. In bag-of-words, this would replace the term frequency weighting scheme, and would allow the model to consider the prevalence of the word in the overall corpus. In the semantic relation model, TF.IDF could be used to guide thesaurus generation in addition to weighting terms. Terms with a higher TF.IDF value can be considered to be more ‘on-topic’ than words with lower, making them better candidates for synonym expansion.

Finally, the two models could even be run in unison, and combined as base learners in an ensemble-like approach akin to Ding, et al. (2017)’s BOKCHOY triple scorer, which could potentially allow the shortcomings of each model to be mitigated.

6 CONCLUSION

In this chapter, the results of this project are outlined, and the work done is summarised. Problems inherent in the task are discussed, and the project is evaluated one last time as to how well it achieved what it set out to accomplish.

In this project, an attempt was made to utilise natural language processing techniques to produce a system capable of scoring the truth of factual triples within a [0, 7] range, where 0 is complete falsity, and 7 is complete truth. These triples consisted of a *subject*, a *relation*, and an *object* parameter, and while they could theoretically represent any fact possible, the project focused on triples of the form (person, *profession* or *nationality*, object). The project was centred around the work of Bast, et al. (2015), who posed this project as a WSDM Cup 2017 competition, of which the winning solution, the BOKCHOY triple scorer created by Ding, et al. of the Chinese Academy of Sciences, won with an 0.868 accuracy overall.

As part of this project, two triple scoring models were created: a bag-of-words model, which naïvely searches for all potential objects in a *relation*al set in the corpus, and counts exact occurrences, and a semantic relation model, which performs unsupervised thesaurus generation over a corpus using latent Dirichlet allocation as a means of corpus reduction, and Stanford NLP GLoVe vectors as a means of expanding the reduced corpus with synonyms.

In creating this system, problems were uncovered that make this task exceptionally hard. For one, loading and parsing a corpus of a necessary size required a considerable amount of engineering. The Wikipedia corpus supplied as part of the project by WSDM Cup 2017 exceeded 4.4GB of purely text, totalling over 33 million sentences, an amount of data that cannot be feasibly loaded into memory on consumer grade hardware. To counter this, a caching mechanism was designed that splits the corpus into smaller pieces, and concurrently searches all pieces in parallel for sentences that relate to a certain person, collecting them into smaller key-specific corpora for later use.

Other issues include the effects of synonymy; the fact that the English language specifically exhibits a one-to-many relationship between concept and vocabulary, meaning that words evocative of a certain triple might not be captured in a model based on the bag of words information retrieval paradigm. For instance, if a triple's *object* was "Actor", and a model only searches for this term, witnesses to the triple with words such as "starred", or "act" won't count, leading to potential corroboration being missed. Conversely, it's also not enough to just collect synonyms of these terms, as the English language also exhibits polysemy, a many-to-one relationship between concept and vocabulary, such that a single term in English may have a multitude of different meanings, and therefore a multitude of sets of valid synonyms, some of which wouldn't be useful at the time.

In addition, there is also the issue of how to classify a term as a witness to a triple. There are many ways of comparing strings, from character-by-character comparison, to methods not talked about in this dissertation like Hamming distance or Jaccard similarity. However, these methods

aren't useful for measuring the similarity of words. For this, it's possible to use vector space modelling techniques to measure the cosine distance between words, a method implemented in this project as 'vectorial capture'. Theoretically, this methodology should have improved the system on evaluation, however it was shown that this was not the case.

Finally, there is an issue about how best to scale raw figures into the [0, 7] range. Bast, et al. (2015) and Ding, et al. (2017) propose multiple mapping functions, all of which were experimented with alongside a simple linear scaling function.

Evaluation and testing of this project was done using a script provided by WSDM Cup 2017, *evaluator.py*. The script used a ground truth file containing triples and their scores as judged by a panel, and calculated various similarity metrics between the two files: *accuracy*, *average score difference,* and *average Kendall's tau.* Bast, et al. (2017) judge the correctness of a submitted solution by the accuracy metric, however this draws issue in terms of how valid a decision that is. The accuracy metric measures whether each triple differs by less than a specified delta (default: 2), meaning it only measures how similar a solution is to the ground truth, not how congruent the ranking of the triples is. For this, a better metric to use would be average Kendall's tau, which measures a ratio between pairings in agreement and disagreement.

Various refinements were performed on each model, including usage of a method from Bast, et al. (2017) dubbed "the 2-5 trick", where the accuracy metric could be increased by simply replacing all scores below 2 with 2, and all above 5 with 5, effectively reducing the interval from [0, 7] to [2, 5]. In the end, it was found that the best model in terms of accuracy was the bag-of-words model, using intersection capture, Bast, et al. (2017)'s 2-5 trick, and Bast, et al. (2015)'s *maplog* mapping function, which achieved a 0.760 accuracy, which would've made the solution competitive as it beat not only the baseline, but the 8th place solution out of 21 submissions: Yahoo! Japan's 'Cauliflower' triple scorer, with a 0.752 accuracy.

In conclusion, given the complexity of the task and the time and hardware constraints imposed, the project can be deemed a success, as it not only produced a model and method of scoring triples, but it did so in a way that could have been competitive had it had been submitted to WSDM Cup 2017. It is clear that more work and more refinements can be done to make a much more accurate solution, and indeed should be done, as the task has potential real-world use cases, from fact checking to search engines, and in such a time when false and fake data can be used for malicious means, the means to verify it are more important than ever.

REFERENCES

Allcott, H. & Gentzkow, M., 2017. Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives,* 31(2), pp. 211-236.

Bast, H., 2016. *Evaluation script for the WSDM Cup 2017 Triple Scoring Task..* [Online] Available at: http://broccoli.cs.uni-freiburg.de/wsdm-cup-2017/evaluator.py [Accessed 11 February 2018].

Bast, H., Buchhold, B. & Haussmann, E., 2015. Relevance Scores for Triples from Type-Like Relations. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval,* pp. 243-252.

Bast, H., Buchhold, B. & Haussmann, E., 2017. Overview of the Triple Scoring Task at the WSDM Cup 2017. *arXiv preprint arXiv:1712.08081 [cs.IR]*, 21 December.

Bird, S., Klein, E. & Loper, E., 2009. *Natural Language Processing with Python.* 1st ed. s.l.:O'Reilly Media Inc..

Blei, D. M., Ng, A. Y. & Jordan, M. I., 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research,* 3(Jan), pp. 993-1022.

Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T., 2017. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606v2 [cs.CL]*, 19 June.

Brill, E., 2000. Part-of-speech tagging. In: R. Dale, H. L. Somers & H. Moisl, eds. *Handbook of natural language processing.* New York: Marcel Dekker, Inc., pp. 403-414.

Brinton, L. J., 2000. *The Structure of Modern English: a linguistic introduction.* Amsterdam, Philadelphia: John Benjamins.

Deerwester, S. et al., 1990. Indexing by latent semantic analysis. *Journal of the Association for Information Science and Technology,* 41(6), pp. 391-407.

Ding, B., Wang, Q. & Wang, B., 2017. Leveraging Text and Knowledge Bases for Triple Scoring: An Ensemble Approach. *arXiv preprint arXiv:1712.08356 [cs.IR]*, 22 December.

Fagin, R. et al., 2004. Comparing and aggregating rankings with ties. *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems,* pp. 47-58.

Grefenstette, G. & Tapanainen, P., 1994. *What is a word, what is a sentence?: problems of Tokenisation.,* s.l.: s.n.

Lovins, J. B., 1968. Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics,* 11(1-2), pp. 22-31.

Parker, R. et al., 2011. *English Gigaword Fifth Edition. LDC catalogue ref. LDC2011T07,* s.l.: Philadelphia: Linguistic Data Consortium.

Pennington, J., Socher, R. & Manning, C. D., 2014. Glove: Global vectors for word representation. In: A. Moschitti, B. Pang & W. Daelemans, eds. *Proceedings of the 2014 conference on empirical methods in natural language processing.* Doha, Qatar: EMNLP, pp. 1532-1543.

Řehůřek, R. & Sojka, P., 2010. Software framework for topic modelling with large corpora. In: R. Witte, et al. eds. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.* Valletta, Malta: ELRA, pp. 45-50.

Salton, G. & Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal,* 24(5), pp. 513-523.

Salton, G., Wong, A. & Yang, C., 1975. A vector space model for automatic indexing. *Communications of the ACM,* 18(11), pp. 613-620.

Sparck Jones, K., 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation,* 28(1), pp. 11-21.

Wikipedia contributors, 2018. *Jane Austen.* [Online]
Available at: https://en.wikipedia.org/w/index.php?title=Jane_Austen&oldid=833276199
[Accessed 11 04 2018].

Wsdm-cup-2017.org, 2017. *WSDM Cup 2017.* [Online]
Available at: http://www.wsdm-cup-2017.org/triple-scoring.html
[Accessed 1 12 2017].

Yamada, I., Sato, M. & Shindo, H., 2017. Ensemble of Neural Classifiers for Scoring Knowledge Base Triples. *arXiv preprint arXiv:1703.04914v2 [cs.CL]*, 15 March.

APPENDICES

APPENDIX A: LIST OF WSDM PROVIDED MATERIALS USED

- wiki-sentences: A text file containing 33 million lines of text from Wikipedia.
- professions: A text file containing 200 different professions that can occur in the corpus.
- nationalities: A text file containing 100 different nationalities that can occur in the corpus.
- profession.train: A TSV file containing panel-judged ground truths for profession triples.
- nationality.train: A TSV file containing panel-judged ground truths for nationality triples.
- evaluator.py: A Python script containing evaluation procedures to test the system against the ground truth data.

APPENDIX B: TABLE OF ALL RESULTS

	ACC (↑)	ASD (↓)	TAU (↓)
Perfect model	1.000	0.000	0.000
Bast et al.'s baseline	0.721	2.070	0.460
Ding et al.'s BOKCHOY scorer	0.868	1.630	0.327
Bag-of-words (Binary Weighting)	0.460	3.080	0.450
Bag-of-words Model (TF Weighting)	0.500	2.780	0.380
Bag-of-words (TF Weighting, 2-5)	0.660	2.210	0.380
Bag-of-words (TF Weighting, 2-5, maplin)	0.680	2.170	0.380
Bag-of-words (TF Weighting, 2-5, maplog)	0.760	1.930	0.420
Bag-of-words (TF Weighting, 2-5, mapscale)	0.690	2.140	0.380
Semantic relation model (50d vectors, intersection capture)	LDA: 0.530 Raw: 0.520	LDA: 2.610 Raw: 2.670	LDA: 0.480 Raw: 0.560
Semantic relation model (100d vectors, intersection capture)	LDA: 0.540 Raw: 0.520	LDA: 2.580 Raw: 2.660	LDA: 0.470 Raw: 0.560
Semantic relation model (200d vectors, intersection capture)	LDA: 0.550 Raw: 0.520	LDA: 2.530 Raw: 2.630	LDA: 0.490 Raw: 0.560
Semantic relation model (300d vectors, intersection capture)	LDA: 0.560 Raw: 0.510	LDA: 2.520 Raw: 2.610	LDA: 0.450 Raw: 0.550
Semantic relation model (300d vectors, intersection capture, LDA, 2-5)	0.660	2.140	0.450
Semantic relation model (300d vectors, intersection capture, LDA, 2-5, maplin)	0.660	2.180	0.460
Semantic relation model (300d vectors, intersection capture, LDA, 2-5, maplog)	0.720	1.990	0.460
Semantic relation model (300d vectors, intersection capture, LDA, 2-5, mapscale)	0.670	2.130	0.460
Semantic relation model (50d vectors, vectorial capture, LDA)	0.450	3.040	0.440
Semantic relation model (50d vectors, vectorial capture, LDA, 2-5)	0.570	2.430	0.440