2011

An analysis and comparison of predominant word sense disambiguation algorithms

David J. Craggs
Edith Cowan University

An Analysis and Comparison of Predominant Word Sense Disambiguation Algorithms

A thesis for a dissertation submitted in partial fulfilment of the requirements for the degree of

Computer Science Honours

By: David Justin Craggs

Student ID: 1006 8589

Faculty of Computing, Health and Science

Edith Cowan University

Supervisor(s): Dr. Mike Johnstone

Date of submission: 24th June 2011

COPYRIGHT AND ACCESS DECLARATION

I certify that this thesis does not, to the best of my knowledge and belief:

(i) Incorporate without acknowledgment any material previously submitted for a degree or diploma in any institution of higher degree or diploma in any institution of higher education;

(ii) Contain any material previously published or written by another person except where due reference is made in the text of this thesis; or

(iii) Contain any defamatory material.

(iv) Contain any data that has not been collected in a manner consistent with ethics approval.

The Ethics Committee may refer any incidents involving requests for ethics approval after data collection to the relevant Faculty for action.

Signed……………………………………..

Date……………………………………….

Table of Contents

Table of Figures

1. Abstract

This thesis investigates research performed in the area of natural language processing. It is the aim of this research to compare a selection of predominant word sense disambiguation algorithms, and also determine if they can be optimised by small changes to the parameters used by the algorithms. To perform this research, several word sense disambiguation algorithms will be implemented in Java, and run on a range of test corpora. The algorithms will be judged on metrics such as speed and accuracy, and any other results obtained; while an algorithm may be fast and accurate, there may be other factors making it less desirable. Finally, to demonstrate the purpose and usefulness of using better algorithms, the algorithms will be used in conjunction with a real world application.

Five algorithms were used in this research: The standard Lesk algorithm, the simplified Lesk algorithm, a Lesk algorithm variant using hypernyms, a Lesk algorithm variant using synonyms, and a baseline performance algorithm. While the baseline algorithm should have been less accurate than the other algorithms, testing found that it could disambiguate words more accurately than any of the other algorithms, seemingly because the baseline makes use of statistical data in WordNet, the machine readable dictionary used for testing; data unable to be used by the other algorithms. However, with a few modifications, the Simplified Lesk algorithm was able to reach performance just a few percent lower than that of the baseline algorithm.

It is the aim of this research to apply word sense disambiguation to automatic concept mapping, to determine if more accurate algorithms are able to display noticeably better results in a real world application. It was found in testing, that the overall accuracy of the algorithm had little effect on the quality of concept maps produced, but rather depended on the text being examined.

2. Introduction

2.1 The background to the study

Word sense disambiguation is the task of automatically determining the correct sense of a word within a text. In ordinary English, or any other language for that matter, a word may be used in a variety of contexts with a variety of meanings. Each of these meanings is called a word sense. Accurately determining the correct sense of a word has been the subject of a great deal of research for several decades now.

The potential applications of word sense disambiguation are numerous and wide ranging. For example:

- a search engine may use it to determine what a user wants to search for more accurately
    - Someone who searches for 'Java programming' is probably not looking for results on coffee.
- a program that translates text in one language to another can find the correct translation of a homonym
    - Translating *bill* from English to Spanish would be either *pico* or *cuenta*, depending if the user means a bird jaw, or an invoice, respectively.
- a program that converts speech to text can use it to determine the correct spelling of a word, where multiple spellings of the word exists, or properly interpret easy-to-miss words

    - If a user said 'the accelerator and brake pedals broke on the Porsche', and the computer heard 'the accelerator and *break*[1] pedals broke on the *Porch'*, the user would quickly become annoyed.
- a program making a concept map from a transcript of a meeting can use it to identify central topics more clearly
    - One meeting may have three separate people saying 'bank' in three different contexts:
        - A financial institution
        - A river bank
        - A banked curve
    - An analysis should determine these are all different concepts
    - Another meeting may involve three separate people saying
        - Police
        - Cops
        - Rozzers
    - The resulting analysis should determine that these are all the same topic.

2.2 The significance of the study

A major difficulty in natural language processing is the complexity of human language. In the WordNet dictionary, a "large lexical database of English" (Princeton University, 2006), developed by Princeton University, the average number of senses for the 121 most common nouns in the English language is 7.8, and the average number of senses for the 70 most common verbs is 12.0. This set of 191 words makes up approximately 20% of regular English text (Ng & Zelle, 1997). With so many different meanings for any given word, it is no wonder that

[1] Typing this in Microsoft Word 2007, the spell checker suggests this word should be 'brake', which would suggest some form of word sense disambiguation is being used.

automatic language processing is difficult. Another difficulty is the rate at which a language can change. For example, within the past fifteen years, a whole new meaning of the word *green* has formed: an adjective to mean environmentally friendly.

One possible solution is to use more coarsely grained lexical resources; lexical resources with fewer, more general word senses for each word in the resource. This is akin to simplifying human language. With fewer possible word senses to choose from when disambiguating a target word, and larger differences between each word sense, the application has a greater chance of determining the correct word sense.

The more clearly expressed information is, the better people are at understanding, remembering, and using the information. There are many tools to help in this regard. Mind maps, concept maps, PMI Charts, or even basic note taking are existing methods. While humans are perfectly capable of analysing text, using automated tools may be more effective. With better performing word sense disambiguation algorithms driving these types of applications, the output of these applications will improve.

Word sense disambiguation is usually performed as a part of a larger application; it is rarely performed on its own. Word sense disambiguation is used in a huge range of applications, and is a substantial component of natural language processing. As such, it plays a large role in applications that involve processing human language. By improving the accuracy of word sense disambiguation, the quality of all applications that utilise word sense disambiguation can improve.

2.3 The purpose of the study

It is the aim of this research to compare a selection of predominant word sense disambiguation algorithms, and also determine if they can be optimised by small changes to the parameters used by the algorithms. For example, most algorithms determine word senses by examining a number of words *n* around the target word. A smaller word window may not give enough clues of the correct sense of the target word. However, a larger word window also increases the computational burden, and may make an algorithm consider words that are not related to the target word, negatively affecting the accuracy of the algorithm.

This research will also address two issues in previous word sense disambiguation research. The first issue is that much research has been done on a small number of target words in a given corpus, typically less than a dozen words. By focussing on just these words, it is difficult to predict how accurate the algorithm or algorithms used would be in a real world application, where all the words in a corpus would almost certainly need to be disambiguated. The second issue from previous research in the field is a lack of testing in a real world application of word sense disambiguation. This research will address these issues by examining every word in the corpora used for testing, and also using the output of the algorithms used in a practical real world application.

2.4 Research questions

1. Which algorithm tested disambiguates words most accurately?
    a. Which algorithm tested is the fastest?
    b. Does accuracy come at the cost of high computational resources?
    c. Does the most accurate algorithm depend on factors such as the corpus being disambiguated or the complexity of the corpus?

2. Can any existing algorithms be improved by small changes in the parameters used?
    a. Can the word window be improved?
        i. Does increasing the word window come at significant computational cost?

3. How much difference does the accuracy of an algorithm make to the quality of an automatically generated concept map?

3. Review of the literature

One of the difficulties in comparing works on word sense disambiguation is the number of different foci any given paper can have on the subject. In regards to comparing algorithms, some algorithms are only used on a few specific words, whereas others will disambiguate a set of discourses from a particular domain or source. Still others may attempt to disambiguate a wide range of texts. Focusing on a small number of words or discourses can often result in an algorithm correctly disambiguating close to 90% of words, but that can drop dramatically once applied to a wider range of texts. Similarly, using a very coarsely-grained lexical resource often achieves much better results than using a finely-grained alternative. Navigli (2008) argues that coarsely-grained resources are sufficient, while others argue the opposite (Wilks et al., 1988). Ultimately, the level of granularity necessary will vary depending on how the lexical resource is being used. Tasks such as machine translation require a high level of granularity. For example, the word *German*, meaning the German language, translates to *deutsch*. German meaning nationality translates to *deutscher*. The difference is subtle, but important; word sense disambiguation can have significant benefits in such instances (Chan, Ng, & Chiang, 2007). Other tasks, such as Text-to-Speech software, need only determine high level sense distinctions, such as the difference between 'I *live* for the theatre', and 'Some fishermen use *live* bait'.

During the 1970s, when there were no large scale external lexical resources available, AI methods were used to perform word sense disambiguation (Ide & Veronis, 1998). However, this was almost completely unsuccessful. The major problem was that the algorithms were confined to a very narrow problem domain. The problem of applying word sense disambiguation to a variety of domains is the inherent difficulty of manually organising the massive amounts of linguistic information necessary to perform accurate word sense disambiguation. This has been referred to as the "knowledge acquisition bottleneck" (Gale, Church, & Yarowsky, 1993). Humans are able to disambiguate word senses very accurately due to the way our brains are able to relate stored information. While machines can also do this, having machines that can use this information to interpret natural language accurately would be a significant undertaking. Expert systems have been successfully created and implemented, but these are limited to very narrow problem domains. An expert system with the necessary level of knowledge to perform human-level word sense disambiguation consistently across a broad range of domains and texts has not been achieved with current technology.

3.1 Knowledge-Based Methods

Knowledge based methods are methods that rely on external lexical resources to disambiguate word senses. The most common external lexicons used are machine readable dictionaries, thesauri, and computational lexicons.

3.1.1 Machine-Readable Dictionaries

During the early 1980s, machine-readable dictionaries became a popular source of information for word sense disambiguation algorithms. Unlike the AI methods of the 1970s, algorithms using these external lexical sources could be applied to a much wider range of corpora. Initially, the goal of many researchers was to "automatically extract lexical and semantic knowledge bases from [machine readable dictionaries]" (Ide & Veronis, 1998). However, this has not fully come to fruition. The major machine readable dictionaries usable by word sense disambiguation algorithms are almost entirely made by hand, including the *Oxford English Dictionary* (OED) and the *Longman Dictionary of Ordinary Contemporary*

English (LDOCE)[2]. Despite the difficulties in creating and maintaining machine readable dictionaries, they are prevalent in works in natural language processing.

Perhaps the most predominant machine readable dictionary is WordNet, a freely available lexical database created by Princeton University. Much of the popularity of WordNet comes from being free to use for research purposes, and its size in terms of the number of words and individual word senses it contains. The latest version at time of writing, WordNet 3.0, contains over 150,000 different words (Laparra & Rigau, 2009). Although containing a huge number of words is not an issue, the number of senses for each word has been criticised. It has been argued that the granularity of WordNet is detrimental to the performance of word sense disambiguation tasks, and that having more coarsely grained definitions would be beneficial (McCarthy, 2006; Palmer, Dang, & Fellbaum, 2007). Of course, there are other, similar dictionaries that can be used in word sense disambiguation research, such as FrameNet, a freely available lexical resourced created at Berkeley University (Lonneker-Rodman & Baker, 2009). However, the word coverage of FrameNet is far smaller than that of WordNet. A number of attempts have been made to rectify this, by combining WordNet and FrameNet together, showing generally positive results (Chow & Webster, 2010; C. Fellbaum, 2010; Laparra & Rigau, 2009).

One of the earliest and most popular algorithms utilising machine readable dictionaries is the Lesk algorithm. Published in 1986, the idea behind this algorithm is to measure the overlap between sense definitions of words in a context. Lesk found with "some very brief experimentation...yielded accuracies of 50-70% on short samples of *Pride and Prejudice* and an Associated Press news story" (Lesk,

[2] Note that several machine readable dictionaries are existing dictionaries, converted and modified to a machine-readable format

1986). Because Lesk only used two discourses in an arguably narrow domain[3], it is somewhat difficult to gain a clear picture of the performance of the algorithm, Lesk used three different dictionaries in his testing, finding the results to be "roughly comparable".

A number of variations to the Lesk algorithm have also been implemented and used, a popular example being the Simplified Lesk algorithm. Where the original Lesk algorithm counts the number of word overlaps between the definitions of a target word and each word in context, the Simplified Lesk counts the number of times a word in context appears in the definition of the target word (Vasilescu, Langlais, & Lapalme, 2004). A major advantage the Simplifed Lesk algorithm has over the original Lesk algorithm is that the Simplified Lesk algorithm is much faster to run, as it has a significantly lower computational time complexity. The Simplified Lesk only needs to get each word in context and compare it to a definition, the Original Lesk must get each context word, its definition, and compare each word in the context word's definition to the target word definition. The Simplified Lesk clearly requires less computation. Furthermore, the work of Vasilescu, Langlais, & Lapalme found that the Simplified Lesk algorithm is also more accurate in disambiguating word senses; up to 15% in some circumstances.

Counting the number of word overlaps between word definitions is one way to determine how closely words are related but there are alternatives (Gelbukh & Torres, 2009). WordNet connects words with a number of relationships, such as synonyms, antonyms, *is-a*, and *is-a-part-of* relationships (Banerjee & Pedersen, 2010). These relationships can be utilised to aid measuring word overlap, or potentially replace counting word overlap completely. This approach was taken by Banerjee & Pedersen. Testing their adapted Lesk algorithm utilising the relationship

[3] On the other hand, neither Lesk's algorithm nor the lexical resources used were optimised for a given domain

data already in WordNet, better accuracy was observed compared to a standard Lesk algorithm. This testing was performed on the Senseval-2 data, where the result for the original Lesk algorithm was 16%. By comparison, the result for the Adapted Lesk algorithm was 32% accurate overall (Banerjee & Pedersen, 2010). While the conditions tested under did not appear to favour the Lesk algorithm, there was still a twofold increase in performance.

3.1.2 Thesauri

During the 1950s, Roget's *Thesaurus* was converted into a machine readable format, and since has been used in a number of different types of applications, including information retrieval, machine translation, and word sense disambiguation. Much of the appeal of Roget's *Thesaurus* comes from the way in which words are organised into categories. A word can appear in any number of different categories, although each of these categories is usually a distinct word sense. This forms the basis of Yarowsky's algorithm.

Yarowsky's algorithm (1992) is based on three observations:

1. Different conceptual classes of words, such as ANIMALS or MACHINES tend to appear in recognisably different contexts.
2. Different word senses tend to belong to different conceptual classes (crane can be an ANIMAL or a MACHINE).
3. If one can build a context discriminator for the conceptual classes, one has effectively built a context discriminator for the word senses that are members of those classes.

Yarowsky also found that other words within a Roget category were good context indicators for other words in the same category. While this approach is crude, it is rather effective. Yarowsky managed to achieve 92% accurate disambiguation on 12 polysemous words (Yarowsky, 1992). However, due to the huge difference in what is being disambiguated, it is difficult to compare the Lesk and Yarowsky algorithms.

One problem Yarowsky found with using Roget's *Thesaurus* were word senses that were finely grained to be distinct word senses, but existed in the same Roget category. For example, both the illicit and medicinal sense of the word *drug* were under the *Remedy* category (Yarowsky, 1992). This ultimately comes back to the question of how finely grained an external lexical resource should be. Coarsely grained categories may make word sense disambiguation faster, although important, if subtle, differences in word senses may be lost.

3.1.3 Computational Lexicons

One attempt to improve WordNet by automatic means was by Hearst (1992). By running an algorithm through a large corpus, Hearst found hyponym relationships could be identified. For example, a encyclopaedia would contain part of a sentence like "works by such authors as Herrick, Goldsmith and Shakespeare" (Hearst, 1992). The algorithm could then determine that Herrick, Goldsmith and Shakespeare are authors. This would be expressed in a form similar to Figure 1. Hearst found that the results from running this algorithm through an encyclopaedia could indeed be a viable way of improving a computational lexicon such as WordNet. While the results from this study are not sufficient to create an entire computational lexicon with no manual work, this could be evidence of a proof of concept. Perhaps an optimised algorithm, that looked for more than hyponym relationships, applied to a much larger corpus, may be able to produce a lexical resource as large, and as comprehensive, as WordNet.

```
hyponym("author", "Herrick")
hyponym("author", "Goldsmith")
hyponym("author", "Shakespeare")
```

Figure 1: Hyponym relationships as expressed in a machine-readable form (Hearst, 1992)

3.2 Supervised Methods

Supervised methods are similar to AI methods of the early 1970s (Ide & Veronis, 1998). Such methods use a manually created set of annotated corpora to train an algorithm. A supervised algorithm will typically identify patterns and rules concerning word senses in the pre-annotated corpora, which can then be applied to new corpora. For example, the pre-annotated corpora may contain the word *bank* in several texts. The supervised algorithm will find certain words that appear around the occurrences of *bank*, creating a "bag of words" for each word sense (Mihalcea & Pedersen, 2005). When this algorithm is run on a new corpus, it will use these bags of words to infer the correct sense for each word. This information is stored as information vectors.

Once the text is in the form of information vectors, a number of different learning algorithms can be used. These algorithms are often used in other problem-domains, typically those in which artificial intelligence-related solutions are found. One such algorithm is the Naïve Bayesian Classifier. This algorithm will determine, given observed features, which result is most likely correct. When applied to word sense disambiguation, features are usually words in the context of the target word present in the bag of words. The result is usually a particular word sense (Mihalcea & Pedersen, 2005). This algorithm has been compared to numerous different algorithms, including neural networks, context vectors, decision trees, probabilistic models, and several other algorithms. Based on the literature (Leacock, Towell, & Voorhees, 1993; Mooney, 1996; Navigli, 2008; Pedersen, 1998), the Naïve Bayesian Classifier was amongst the highest performing algorithms tested.

3.3 Unsupervised Methods

Unlike supervised methods, unsupervised methods do not require a set of manually annotated corpora to train an algorithm. Due to the significant time and effort needed for a human to annotate a large text, unsupervised methods appear to be a better alternative (Yarowsky, 1995). Unfortunately, unsupervised methods tend to disambiguate word senses with less accuracy than their supervised counterparts. Furthermore, these methods are only able to distinguish between different senses and uses of words, not what that difference is. For example, an algorithm may identify there are two different senses of *tank* in one discourse. However, an unsupervised method-based algorithm cannot determine one of these senses is a military vehicle, and the other is a container. This is because an unsupervised algorithm will typically determine different word senses based on the words surrounding different uses of a target word. For example, if an unsupervised algorithm was to examine the word *plant*, it would likely determine that one sense tended to be surrounded by words such as *life*, *environment*, or *flora*; whereas another sense would be surrounded by words such as *industrial*, or *machinery*. The algorithm can determine there is a difference, but with no external lexical knowledge, cannot tell what the difference is.

A predominant work concerning unsupervised methods[4] is that of Yarowsky (1995). In this paper, he described an algorithm based on unsupervised methods that could almost match, or in some cases even exceed, the accuracy of algorithms based on supervised methods. Yarowsky applied this algorithm on 14 random words that had been studied in previous literature. The data was extracted from a "460 million word corpus containing news articles, scientific abstracts, spoken transcripts, and novels" (Yarowsky, 1995), and the algorithm proved to perform extremely well; discriminating 96% of words correctly. However, while a large corpus was used, this algorithm focused on only 14 different words. Yarowsky claims to have done this to provide a better comparison with existing works discussing supervised methods, although the 96% result would likely drop if all words were attempted to be disambiguated.

[4] Although arguably, this algorithm is technically a semi-supervised method

3.4 Recent Trends

3.4.1 Hybrid Methods

In addition to supervised, unsupervised, and knowledge based methods, combinations of these approaches have been used to make hybrid methods. This is based on sound logic. A major weakness with unsupervised methods is the lack of ability to place a label on each discriminated word group. Combining an unsupervised method with a knowledge based method, particularly a machine-readable dictionary, could overcome this weakness.

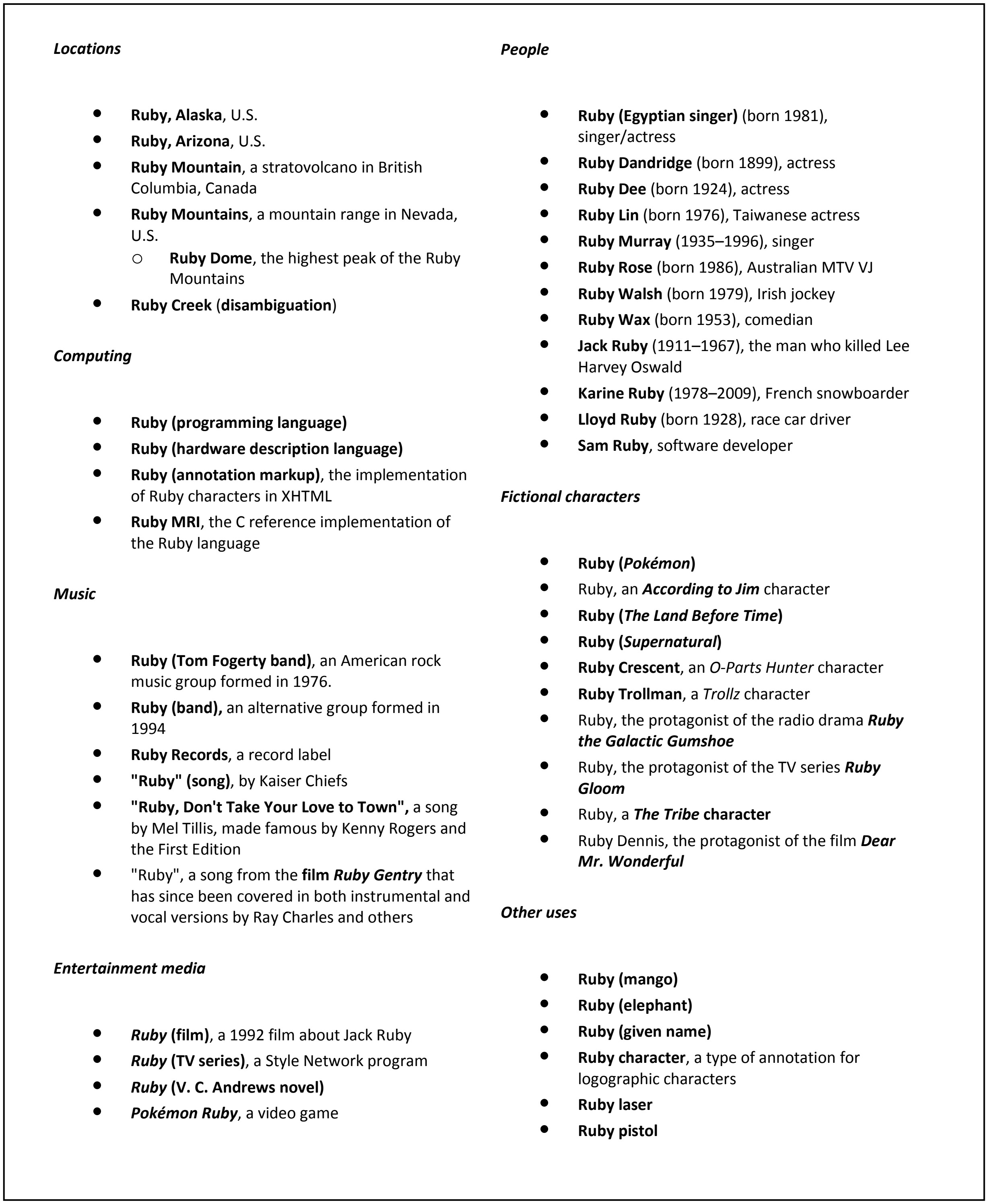
An example of a hybrid method is the approach of Legrand and Pulido (2004). This approach involved the combination of a neural network and the WordNet database to improve automatically "classifying documents on the web into different categories" (Legrand & Pulido, 2004). While the results of this algorithm were extremely promising, correctly labelling all the items in the dataset, the researchers recognise the dataset used was very small, and only contained nouns. However, this can be seen as a proof of concept that hybrid methods can be implemented, and can be extremely accurate.

Hybrid methods have also been applied to the field of machine translation. In the case of English to Brazilian Portuguese translation, Specia (2005) combined a supervised learning approach with a knowledge based approach. This system was able to correctly translate the verbs *come*, *get*, *give*, *go*, *look*, *make* and *take* 81.7% of the time on average. Interestingly, the system could only disambiguate the word *make* 68% of the time, whereas it could disambiguate the term *give* 91% of the time. Specia did not mention the level of granularity used in this system, although WordNet lists 33 word senses on average for the words tested. If a granularity level similar to WordNet was used, 81.7% is an impressive level of accuracy.

Another hybrid of knowledge based and supervised methods was created for disambiguating corpora in the Italian language. This system used a knowledge based method to substitute for a lack of training data, before the supervised method refined the results, resulting in a system that was more accurate than either system individually (Basile, de Gemmis, Lops, & Semeraro, 2008).

3.4.2 Utilising the Internet

More recent attempts at word sense disambiguation have used the World Wide Web as a lexical resource. Wikipedia is an example. Wikipedia already contains "rich, many-to-many mapping between terms (names, words, and phrases) and concepts (things and ideas)" (Gregorowicz & Kramer, 2006). Wikipedia contains approximately 3.6 million articles in English at the time of writing (Wikipedia, 2011). This has the potential to be a huge lexical resource for a number of areas, including information retrieval and natural language processing (Medelyan, Milne, Legg, & Witten, 2009). To compare an example with WordNet, take the term *ruby*. WordNet lists 3 senses; A gemstone, a mineral, and the colour, and the adjective describing colour (Fellbaum, 1998). These are reasonably fine grained senses; a more granular resource would most likely reduce the senses to have the gemstone and mineral as one sense, and colour as the second sense. Wikipedia, on the other hand, lists 48 possible uses or meanings of *ruby* (See Figure 2). Using Wikipedia, of course, has potential problems. Wikipedia, by its very nature, is open to editing by anyone, regardless of his/her credentials. Also, as Figure 2 shows, many articles on Wikipedia are about popular culture. This may be beneficial to certain applications of word sense disambiguation, or to certain audiences, but probably detrimental to businesses. Finally, using Wikipedia as an encyclopaedia to give a computer human level intelligence would also suffer from the aforementioned knowledge acquisition bottleneck.



Locations

- **Ruby, Alaska**, U.S.
- **Ruby, Arizona**, U.S.
- **Ruby Mountain**, a stratovolcano in British Columbia, Canada
- **Ruby Mountains**, a mountain range in Nevada, U.S.
  - **Ruby Dome**, the highest peak of the Ruby Mountains
- **Ruby Creek** (**disambiguation**)

Computing

- **Ruby (programming language)**
- **Ruby (hardware description language)**
- **Ruby (annotation markup)**, the implementation of Ruby characters in XHTML
- **Ruby MRI**, the C reference implementation of the Ruby language

Music

- **Ruby (Tom Fogerty band)**, an American rock music group formed in 1976.
- **Ruby (band),** an alternative group formed in 1994
- **Ruby Records**, a record label
- **"Ruby" (song)**, by Kaiser Chiefs
- **"Ruby, Don't Take Your Love to Town",** a song by Mel Tillis, made famous by Kenny Rogers and the First Edition
- "Ruby", a song from the **film *Ruby Gentry*** that has since been covered in both instrumental and vocal versions by Ray Charles and others

Entertainment media

- ***Ruby* (film)**, a 1992 film about Jack Ruby
- ***Ruby* (TV series)**, a Style Network program
- ***Ruby* (V. C. Andrews novel)**
- ***Pokémon Ruby***, a video game

People

- **Ruby (Egyptian singer)** (born 1981), singer/actress
- **Ruby Dandridge** (born 1899), actress
- **Ruby Dee** (born 1924), actress
- **Ruby Lin** (born 1976), Taiwanese actress
- **Ruby Murray** (1935–1996), singer
- **Ruby Rose** (born 1986), Australian MTV VJ
- **Ruby Walsh** (born 1979), Irish jockey
- **Ruby Wax** (born 1953), comedian
- **Jack Ruby** (1911–1967), the man who killed Lee Harvey Oswald
- **Karine Ruby** (1978–2009), French snowboarder
- **Lloyd Ruby** (born 1928), race car driver
- **Sam Ruby**, software developer

Fictional characters

- **Ruby (*Pokémon*)**
- Ruby, an ***According to Jim*** character
- **Ruby (*The Land Before Time*)**
- **Ruby (*Supernatural*)**
- **Ruby Crescent**, an *O-Parts Hunter* character
- **Ruby Trollman**, a *Trollz* character
- Ruby, the protagonist of the radio drama ***Ruby the Galactic Gumshoe***
- Ruby, the protagonist of the TV series ***Ruby Gloom***
- Ruby, a ***The Tribe* character**
- Ruby Dennis, the protagonist of the film ***Dear Mr. Wonderful***

Other uses

- **Ruby (mango)**
- **Ruby (elephant)**
- **Ruby (given name)**
- **Ruby character**, a type of annotation for logographic characters
- **Ruby laser**
- **Ruby pistol**

Figure 2: Other Meanings of *Ruby*, According to Wikipedia

One such attempt to use Wikipedia for word sense disambiguation yielded promising results. Using Wikipedia as a sense-tagged corpus for training in a supervised method, Mihalcea (2007) found this method to be superior to a baseline algorithm that assigned the statistically most frequent sense of a word and Lesk algorithms. This was performed on the nouns from the Senseval-2 and Senseval-3 workshops, with the Most Frequent Sense baseline scoring 72.58% on average, the Lesk algorithm 78.02% on average, and the supervised algorithm trained by Wikipedia 84.65% on average. This shows that Wikipedia, often thought to be inappropriate for use in academia due to issues with accuracy, has real potential as a lexical resource.

3.4.3 Identifying Emotion

An emerging application of word sense disambiguation is to identify the emotion or tone behind a text, as “recognizing the emotive meaning of text can add another dimension to the understanding of text” (Aman & Szpakowicz, 2008). Similar to the assigning a word with a particular word sense, Aman and Szpakowicz use Roget’s Thesaurus with a machine learning algorithm to assign one of eight emotion labels to a sentence: happiness, sadness, anger, disgust, surprise, fear, mixed emotion, or no emotion. Also like word sense disambiguation, results were measured by precision, recall, and F-measures. The results of this were somewhat positive, with F-measures ranging from 0.493 to 0.751 between the various emotions.

3.5 Current Evaluation

An important question is how good does an algorithm needs to be for widespread application. This was briefly addressed by Gale, Church and Yarowsky. In answer to "Should we be happy with 70% performance", they stated "70% really isn't very good" (1992), which is somewhat disheartening, seeing as no algorithm discussed above can achieve 70%. Of course, it is possible for a slightly modified version of the above algorithms to score better, or an entirely different algorithm may be able to achieve above 70%.

So if 70% is not good enough, what is? 80%? 90%? Will anything less than perfect be good enough? Gale, Church and Yarowsky estimated the upper bounds of accuracy by "trying to estimate the limit of our ability to measure performance. We assumed that this limit was largely dominated by the ability for the human judges to agree with one another" (1992). The upper bound was found to be approximately 95%, which was "imposed by the limit for judges to agree with one another. Unfortunately, this does not really translate to 95% is good enough for real world applications. However, it has been observed that algorithms "seem to need near-100% accuracy in order to be useful in real applications" (Sánchez-de-Madariaga & Fernández-del-Castillo, 2008). In other words, not only do word sense disambiguation algorithms need to be almost 100% accurate, it may be difficult to determine if an algorithm is that accurate.

3.6 Workshop Evaluations

There have been several competitions for evaluating word sense disambiguation algorithms. The first of these was Senseval, which took place in 1998. A variety of tasks were available to participants, covering a variety of topics related to natural language processing, and a number of different languages. The main task was the English all words task: a straightforward task of disambiguating word senses on a set of English corpora. Participants were provided a machine readable dictionary of 35 words, training data, and later test data. They were then tasked with creating a word sense disambiguation system that would disambiguate the test corpus as accurately as possible (Kilgarri, 1998). As each team had identical dictionaries and test data, results of precision and recall could be compared directly. Results were also compared against a number of baseline algorithms, including a system that simply assigns the most common word sense to a target and ignoring context, a system that assigned word senses randomly, and two Lesk algorithms. A summary of the results can be seen in Figure 3. Human judges scored exceedingly well, as would be expected. The best system scored well, over 75% accurate on fine grained word senses. The best baseline also performed well, scoring better than the average of the systems. The worst system performed very poorly, only 33% accurate under the best circumstances. As discussed, Gale, Church and Yarowsky found "70% really isn't very good" (1992), suggesting anything other than the best system would not be good enough. Also, the test circumstances were ideal; participants had a good idea of what the test data would be like while developing a system, and only had to disambiguate 35 different words. A real-world application would not have these benefits, and would certainly decrease performance.

	Fine-grained precision (recall)	Mixed-grained precision (recall)	Coarse-grained precision (recall)
Human	**0.965** (0.963)	**0.968** (0.967)	**0.970** (0.968)
Best system	**0.771** (0.771)	**0.797** (0.797)	**0.814** (0.813)
Average of systems	**0.550** (0.376)	**0.632** (0.410)	**0.661** (0.426)
Worst system	**0.205** (0.162)	**0.315** (0.248)	**0.338** (0.267)
Best baseline	**0.691** (0.689)	**0.720** (0.719)	**0.741** (0.739)

Figure 3: Summary of percentage results from SenseEval (Kilgarriff & Rosenzweig, 2000)

In Senseval-2, the format of the English all words task set for participants was largely identical to that of the first Senseval, but different words and corpora were used (Edmonds & Cotton, 2001), a set of three articles covering three different genres (Agirre & Edmonds, 2006). This makes it difficult to compare the results between the two tasks: one set of words or corpora could be much easier to disambiguate than another. Overall, participants scored worse in SenseEval-2 than the first Senseval, with the best scoring system only disambiguated words with 69% precision and recall (Edmonds & Kilgarriff, 2002). Interestingly, the fine grained analysis performed no worse than the coarsely grained analysis.

Senseval-3 was very similar to Senseval-2, although an option to mark words as fitting no definition in the WordNet dictionary was added (Snyder & Palmer, 2004). This meant that each system had two scores; one with allowing systems to tag words as untaggable ('With U'), and another score where untaggable senses were skipped ('Without U'). When calculating the 'With U' score, "the instance would be scored as correct if the answer key also marked it as I, and incorrect otherwise". As untaggable words were simply skipped when calculating the 'Without U' score, "precision was not affected by those instances, but recall was lowered" (Snyder & Palmer, 2004). Figure 3 shows a summary of Senseval-3 scores. As WordNet 1.7 was used as the lexical resource, the scores should be considered as working on fine grained word senses. The average of all the systems is 57% for precision, and 52% for recall. A baseline algorithm, which simply assigned the first WordNet sense to each word, achieved a score of 61%. As this was using different test corpora, and focussing on different words than the previous Sensevals, results between them are not entirely accurate. This is fortunate, as the best scoring system of Senseval-3 scores slightly worse than the best system of Senseval-2. However, Snyder and Palmer note that human annotators only agreed of sense definitions 70-75% of the time, due to how finely grained some of the WordNet sense definitions are. It could be argued that the best system was only 5-10% worse than human level disambiguation.

System	**'With U' Precision/Recall (%)**	**'Without U' Precision/Recall (%)**
GAMBL-AW-S	.652/652	.651/.651
SenseLearner-S	.646/.646	.651/.642
Koc University-S	.641/.641	.648/.639
R2D2: English-all-words	.626/.626	.626/.626
Meaning-allwords-S	.624/.624	.625/.623
Meaning-simple-S	.610/.610	.611/.610
Upv-shmm-eaw-S	.609/.609	.616/.605
LCCaw	.607/.607	.614/.606
UJAEN-S	.590/.590	.601/.588
IRST-DDD-00-U	.583/.583	.583/.582
University of Sussex-Prob5	.572/.572	.585/.568
University of Sussex-Prob4	.554/.554	.575/.550
University of Sussex-Prob3	.551/.551	.573/.547
DFA-Unsup-AW-U	.548/.548	.557/.546
IRST-DDD-LSI-U	.501/.501	.661/.496
KUNLP-Eng-All-U	.500/.500	.510/.496
Upv-unige-CIAOSENSO-eaw-U	.481/.481	.581/.480
Merl.system3	.458/.458	.467/.456
Upv-unige-CIAOSENSO2-eaw-U	.452/.452	.608/.451
Merl.system1	.450/.450	.459/.447
IRST-DDD-09-U	.446/.446	.729/.441
autoPS-U	.436/.436	.490/.433
Clr04-aw	.434/.434	.506/.431
Merl.system2	.359/.359	.480/.352
autoPSNs-U	.359/.359	.563/.354
SLSI-UA-all-Nosu	.280/.280	.343/.275

Figure 4: Summary of Senseval-3 system scores. A -S or -U after the system name indicates that the system was reported as supervised or unsupervised, respectively (Snyder & Palmer, 2004)

After the low rate of human annotators agreeing on sense definitions, Senseval-4 (also known as Semeval-2007) had two separate all-words English challenges: one for coarse grained definitions, and another for finely grained definitions. For the coarsely grained challenge, a new lexical resource needed to be used. To create one, a combination of WordNet and the OED were used. Inter-annotator agreement of word senses rose to 94% on the test data; far more than that of the Senseval-3 data (Navigli, Litkowski, & Hargraves, 2007). As a result of the more coarsely grained sense definitions, the performance of the 12 systems submitted improved, as seen in Figure 5. The highest scoring achieved a score of 82.5%, far greater than previous scores on finely grained sense definitions. A baseline system was also used, simply assigning the most frequent sense to a word. This system scored 78.89% precision and recall. The average of all the systems was 72.20% precision and 68.00% recall. Clearly, performance can be increased by using coarsely grained lexical resources. The same task using the regular, finely grained WordNet dictionary did not achieve such high results. The best performing system achieved only 59.1% precision and recall, the most frequent sense baseline scored 54.1%, and the average was 48.09% (Pradhan, Loper, Dligach, & Palmer, 2007). All the scores were approximately 20% lower than the equivalent system using a coarsely grained lexical resource. Human annotators agreed on 72% of word senses for nouns, and 86% of word senses for verbs, also 15-20% lower than that of the coarsely grained equivalent.

System	**Precision/Recall (%)**
NUS-PT	82.50/82.50
NUS-ML	81.58/81.58
LCC-WSD	81.45/81.45
GPLSI	79.55/79.55
BASELINE	78.89/78.89
UPV-WSD	78.63/78.63
TKB-UO	70.21/70.21
PU-BCD	69.72/62.80
RACAI-SYNWSD	65.71/65.71
SUSSX-FR	71.73/52.23
USYD	58.79/56.02
SUSSX-C-WD	54.54/39.71
SUSSX-CR	54.30/39.53
UOR-SSI	83.21/83.21

Figure 5: System scores for coarsely grained word sense disambiguation from Senseval-4. The last system listed was created by one of the task organisers (Navigli, et al., 2007)

3.7 Word Sense Disambiguation and Concept Mapping

It is not surprising that research into combining word sense disambiguation and concept maps has been done before. One such example is the work of Cañas, Valerio, Lalinde-Pulido, Carvalho and Arguedas (2003). This research used CmapTools, a software tool "that empowers users, individually or collaboratively, to represent their knowledge using concept maps, to share them with peers and colleagues, and to publish them" (Cañas et al., 2004). CMapTools also has a client-server architecture, to facilitate user collaboration, and to link concept maps in the same server. CMapTools was modified so that as a concept map was being constructed, possible related concepts were suggested. This technique has a major advantage over trying to perform word sense disambiguation over plain text; it is clear what words are related to the target word and which are not. In plain text, a word window may contain words that give no clues of the correct sense of the target word, providing false clues of the correct sense. Results of this study were promising, with their algorithm proving 75% accurate, using an average word window of six words.

3.8 Concept Map Analysis

There have been a number of different methods in order to judge concept maps proposed, with varying degrees of quantitative and qualitative analysis involved. Often, attempts to define a completely quantitative framework to assess concept maps still have some element of qualitative analysis. One such example is the work of Calafate, Cano, and Manzoni (2009). Calafate, Cano, and Manzoni propose a framework in order to quantitatively assess concept maps created by students, using measures of whether all the essential concepts of the topic were identified, whether secondary concepts were identified, the degree of 'meshness' and relationship accuracy, and other quality factors. These measures are weighted, and a final percentage score is calculated. However, there are problems with this approach. The first step in this framework is determining the number of essential concepts - those that the concept map must contain. Different people can disagree on what the most essential concepts of a topic or text are. The next step in the framework to assess a concept map is count the number of essential and secondary concept identified in the map. The distinction between an essential and secondary concept can be blurry. Furthermore, the next steps of assessing the overall 'meshness' of the concepts, and weighting the components of evaluation are completely subjective. While there is nothing necessarily wrong with the framework, it is by no means objective.

One class of metrics that could be used to judge concept maps could be social network analysis metrics. However, there are a few problems with this idea, due to the differences between a concept map and a network. One important difference is in a network, all nodes are treated equally. In a concept map, concepts are not all equal; a few concepts, usually the general overarching ones, will be key to the network. Social network analysis metrics do not allow for ensuring that certain nodes are present in a network. Furthermore, while it is possible to calculate the social network analysis metrics for a concept map, such as the network density, a concept map does not necessarily improve with more connections between concepts. Some concepts in a concept map will not really be directly related, only indirectly related. A completely dense concept map, where every concept is related to every other one, does not help in showing how concepts are really related.

Another method of judging concept maps could be to use human experts in the field relating to the text being disambiguated. However, there are problems with this approach as well. Due to the wide variety of topics that could involve concept maps, finding experts in all the necessary fields would be difficult. Furthermore, experts in a field would not necessarily agree on what a good concept map should look like, nor agree on what the central topics in a text are. To address this, multiple experts in each field would be required. Finally, for some concept maps created from texts, it is unclear on who an expert in the field would be. For most concept maps created from journal articles, determining if someone could be considered an expert in the field would not be difficult. However, for other texts, this is less clear. What qualifications would make someone an expert in fictional text excerpts, or the area of humour?

Another method of assessing concept maps was proposed by McClure and Bell (1990). This method focussed on the links between concepts within a concept map, rather than the concept themselves. When scoring a concept map, the assessor would examine each proposition. A proposition was defined as two concepts linked by an arrow, with a text label to describe the relationship between the two concepts. Each proposition would be given a score between zero and three inclusive, depending on the correctness of the link. The guide for assigning a score is shown in Figure 6. The sum of these scores would then become the final score for the map. While this method is reasonably quantitative, there is still room for subjectiveness when deciding a score for a proposition. Another potential issue is that there is little room to decide whether a concept belonged on a map. For example, if a concept map was produced from a text discussing alternative energy sources, important concepts may be coal, solar power, geothermal power, or wind power. The article may never mention nuclear power, or tidal power. While these are related concepts, they should not be included in the map, assuming the map should be limited to only what the text discussed.

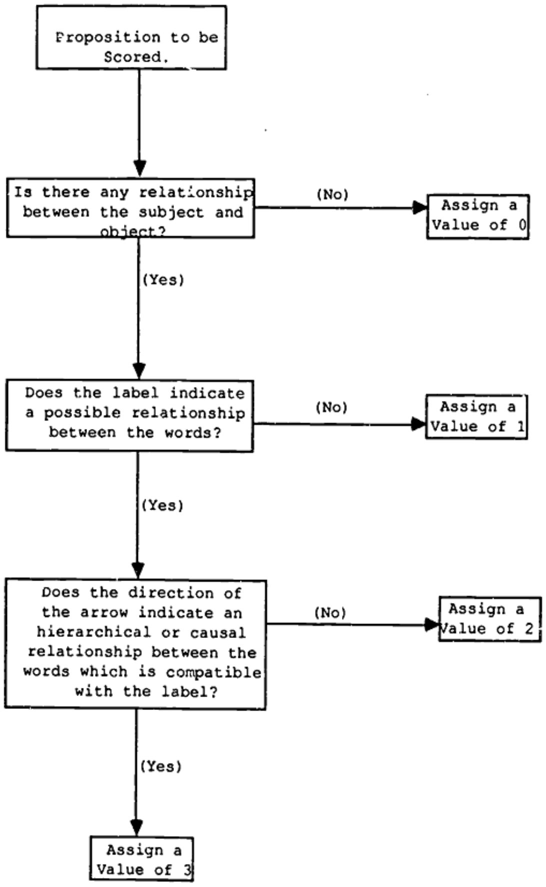

Proposition to be Scored.
Is there any relationship between the subject and object?
(No)
Assign a Value of 0
(Yes)
Does the label indicate a possible relationship between the words?
(No)
Assign a Value of 1
(Yes)
Does the direction of the arrow indicate an hierarchical or causal relationship between the words which is compatible with the label?
(No)
Assign a Value of 2
(Yes)
Assign a Value of 3

Figure 6: Method for Scoring Propositions (McClure & Bell, 1990)

Another method could be the structural scoring method proposed by McClure, Sonak, and Suen (1999), adapted from a method proposed by Novak and Gowin (1984). This method assigned score not only to propositions, but also for concepts arranged in a hierarchical structure, for links between branches of a hierarchy, and for examples of a concept provided. Unlike the method proposed by McClure and Bell, propositions were assigned one point each. Hierarchy levels are assigned five points, and cross links are assigned ten points. Examples are only given one point each. The major issue with this framework is that concepts in a concept map do not necessarily fall into neat hierarchial levels.

3.9 Conclusion

A common trend throughout work on word sense disambiguation algorithms is a lack of testing on a large scale text corpus. Algorithms are often tested on a small set of corpora, or only test select words (typically less than a dozen). While algorithms are often claimed to achieve high levels of disambiguation, often 80-90% or more, it is unlikely these algorithms would score as highly if they were used on large scale corpora. More testing needs to be done on these algorithms to determine how well they scale up on larger corpora.

Clearly, there is no shortage of different algorithms to tackle the problem of word sense disambiguation. Most of these have advantages and disadvantages, which are summarised in Figure 7. Supervised methods are accurate, but are reliant on pre-annotated corpora to be effective. This can be overcome using unsupervised methods; although those methods have difficulty in determining why and how word senses are different. Knowledge based methods can solve this problem, although the external lexical resources are difficult to create manually. It is unclear what it will take in order to create an algorithm that can disambiguate finely grained word senses with greater than human level accuracy. It is possible it will be a new type of algorithm, unlike the methods described above.

Algorithm Category	Strengths	Weaknesses	Predominant Algorithm(s)
AI Methods	Some ideas formed the basis of all further work on the subject e.g. word window	Very domain specific	Expert Systems, as described by Small (1981) Semantic Networks, as described by Dahlgren (1988)
Knowledge Based	Accuracy	Rely on precompiled lexical knowledge resources	The Lesk algorithm, as described in Lesk (1986) Yarowsky's algorithm, as described in Yarowsky (1992)
Supervised	Accuracy	Dependent on pre-annotated corpora for training data	Naïve Bayesian Classifier, as described by Gale et al. (1993)
Unsupervised	No pre-training necessary Works on multiple languages with no modification to the algorithm	Merely discriminates between word senses; not disambiguate word senses	Yarowsky's algorithm, as described in Yarowsky (1995)

Figure 7: Summary of Word Sense Disambiguation Approaches

4. Research Design

4.1 Selection of methodology

There are numerous approaches for information systems research, which according to Galliers (1990), can be placed in a continuum between quantitative and qualitative research. Galliers offers a summary of these approaches and defines a number of objects of interest for research. These are society, organisation/group, individual, technology, methodology, theory building, theory testing and theory extension. Narrowing potential choices for this research is not difficult. Research that looked at people, how people interact, their behaviour, or a similar topic would fall into the categories of society, organisations/group or individuals, depending on the size of the group being investigated. Research focused on investigating a certain methodology or technology could use a range of approaches, both quantitative and qualitative. Technology could arguably be the object of interest in this research, however technology usually refers to the application of a tool and how it can be utilised for certain purposes or tasks. The final objects of interest all concern theories. Theory building is concerned with creating a new theory, which is not involved in this research. Theory extension looks at how existing theories can be improved. This research will likely contain a small element of theory extension in terms of changing parameters of existing algorithms; however it is not the focus of the research. The last remaining object of interest is theory testing, which involves examining pre-existing theories, and possibly applying them to new areas or comparing them. This research is focused on examining and comparing various algorithms that perform word sense disambiguation in terms of accuracy and speed. This fits exactly with theory testing.

Galliers' summary table, replicated in Figure 8, shows the possible approaches to theory testing are laboratory experiments, field experiments, case studies, surveys, simulations, descriptive/interpretive, and action research. Some of these methodologies can be eliminated immediately. Methodologies such as surveys can be eliminated, as this research is not focussed on the public's perception of word sense disambiguation algorithms. Also, methodologies such as action research and case studies can also be eliminated. As this research is not focussed on how word sense disambiguation algorithms are used in the real world, there is no client or client group involved in this research, making case studies or action research impossible. Furthermore, the testing of this research does not warrant the use of simulations. Simulations are often defined as a method for using computer software to model the operation of 'real-world' processes, systems, or events, that offer some, but not all of the characteristics of the environment being modelled (Lave & March, 1993; Law & Kelton, 2000). However, this research is simply running implementations of algorithms; there is no real-world environment that the algorithms are dependent on to function. Therefore, the only option left is experiment, either in the field or a laboratory.

This research is looking at three main areas: which algorithm tested disambiguates words best, can any of the algorithms be improved, and how do the differences in algorithms affect a real world application. These can all be investigated thoroughly using laboratory experiments; therefore there is no need to use field experiments.

Object	Modes for traditional empirical approaches (observations)							Modes for newer approaches (interpretation)		
	Theorem Proof	Laboratory Experiment	Field Experiment	Case Study	Survey	Forecasting and Futures Research	Simulation and Game/Role Playing	Subjective/ argumentative	Descriptive/ Interpretive (inc Reviews)	Action Research
Society	No	No	Possibly	Possibly	Yes	Yes	Possibly	Yes	Yes	Possibly
Organisational/ group	No	Possibly (small groups)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Individual	No	Yes	Yes	Possibly	Possibly	Possibly	Yes	Yes	Yes	Possibly
Technology	Yes	Yes	Yes	No	Possibly	Yes	Yes	Possibly	Possibly	No
Methodology	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Theory Building	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Theory Testing	Yes	Yes	Yes	Possibly	Possibly	No	Possibly	No	Possibly	Possibly
Theory Extension	Possibly	Possibly	Possibly	Possibly	Possibly	No	No	No	Possibly	Possibly

Figure 8: Information Systems Research Approaches (Galliers, 1990)

To ensure proper testing of algorithms and repeatability, a controlled environment is needed. If the environment used to test the algorithms is not controlled, other factors could alter the results. However, maintaining a controlled environment within a computer is not difficult. As long as the same computer is used to test the algorithms, and there are no changes to the software or hardware, the environment can be considered controlled. As the algorithms tested in this research do not make any changes to the computer environment, any algorithm can be run any number of times to ensure the results obtained are consistent. This repeatability ensures that the effect of any external factors is minimised. Computer science has a unique advantage that other fields do not have; performing an experiment is very cheap, it only costs computer cycles and time. Other fields, such as chemistry, often use consumable materials in research, and are limited by the quantity of materials available.

4.2 Research Procedure

The algorithms used in testing will be the standard Lesk algorithm, the Simplified Lesk algorithm, a Lesk variant using hypernyms, a Lesk variant using synonyms, and a baseline algorithm that assigns the first found sense of a word, ignoring context. Each of these algorithms will use WordNet 2.1, the latest stable version of WordNet for Microsoft Windows at the time of performing this research.

The pseudocode of the Lesk algorithm, as seen in Appendix C, shows four embedded loops. Also, the operation to count the number of words that occur in the gloss of both word senses is $O(n^2)$, giving the Lesk Algorithm a time complexity of $O(n^6)$. The Lesk variants using hypernyms and synonyms will have an even higher time complexity, as obtaining the hypernyms and synonyms for each word sense requires another embedded loop. The psuedocode of the Simplified Lesk algorithm, however, shows only three embedded loops. Furthermore, counting the number of overlapping words between the phrase and the gloss of the target word also has a lower time complexity, resulting in a total time complexity of $O(n^4)$. However, as the baseline does not take into account context, it has a time complexity of just $O(n)$.

In order to test the algorithms, the SemCor 2.1 corpus was used. This is a freely available selection of texts from the Brown Corpus, that have been manually annotated with WordNet senses. SemCor 2.1 consists of 186 texts with all words tagged with word senses, and 166 additional texts with only verbs annotated. As this research is only focussed on all words disambiguation, the texts with only verbs will be excluded. Finally, due to errors in text parsing, 8 texts are also excluded. This leaves a total of 174 texts, each approximately one thousand words in length.

As the texts in SemCor are already manually tagged, determining how accurate an algorithm is simple. Each algorithm will be run on the 174 texts, producing a set of answers. The answers obtained can then be compared against the manual sense annotations in each text. To score each algorithm, the precision is gained by dividing the correct answers by the number of words attempted. The recall is gained by dividing the correct answers by the total number of words in the text.

A graphics framework will be used to display the results for this research. The framework will produce concept maps based on the results of the various word sense disambiguation algorithms. This framework will be based on JUNG; the Java Universal Network/Graph Framework (O'Madadhain, Fisher, White, & Boey, 2003). This is an open source framework that will be modified to suit the purposes of this research.

In order to judge the concept maps, a number of methods of judging concept maps were considered in section 3.8. Finally, the work of Calafate, Cano, and Manzoni (2009) was selected. While the framework produced arguably still relies on subjective measures, such as what the essential and secondary concepts of a topic are, and how each measure is weighted, the quantitative measures are useful in providing some amount of repeatability for other researchers, as scoring a concept map using the framework uses mostly quantitative measures. The equations for this framework are as follows:

$Score\ (\%) = \alpha \cdot \frac{S_1}{M_1} + \beta \cdot MS \times RA + \gamma \cdot Q$ where

$S_1 = \frac{n_e}{N} \cdot \log_N(n_e + n_s)$,

$M_1 = \log_N(N + 4 \times N)$,

$DM = \frac{r}{R_{min}} = \frac{r}{n-1}$,

$MS = \begin{cases} 0.7 \leftarrow DM < 1.04 \\ 0.85 \leftarrow 1.4 \leq DM < 1.08 \\ 1 \leftarrow DM \geq 1.08 \end{cases}$, and

$(\alpha, \beta, \gamma) = (0.6, 0.35, 0.05)$.

In these equations, N is the total number of essential concepts that should appear in the map, n_e is the number of essential concepts identified, n_s is the number of secondary concepts identified, and M_1 is the ratio between n_e and n_s. The 'degree of meshness' in a concept map was defined as DM, the number of relationships as r, the minimum number of relationships possible as R_{min}, the total number of concepts identified n, and the meshness score as MS. The Relationship Accuracy (RA) is a subjective score between zero and one, for the "overall correctness and accuracy of the relationship proposed" (Calafate, et al., 2009). The quality parameter (Q) is another subjective measure, for "other quality details...including segregating the most important concepts from the rest through highlighting (font, color, box shape etc.)". Finally, (α, β, γ) are weighting parameters, which the values of 0.6, 0.35 and 0.05 were used by Calafate, Cano, and Manzoni.

5. Materials and Methods

5.1 Equipment

The specifications of the computer used for testing is as follows. Several of these specifications will have no bearing on the performance of the algorithms, but have been included for completeness.

Laptop: MSI GX620
Motherboard: MSI MS-1651
CPU: Intel Core 2 Duo P8600 @ 2.8GHz
RAM: 4GB DDR2 800MHz
GPU: nVidia 9600M GT
HDD: Western Digital 320GB 7200rpm SATA

Operating System: Windows 7 Home x64
Java Version: Version 6 Update 20

5.2 Procedure

A selection of predominant word sense disambiguation algorithms will be implemented in Java, and run on a selection of test corpora. As many algorithms will be used as practical, within the time constraints. These algorithms will be a variety of knowledge based methods. Due to a lack of suitable freely available training corpora supervised methods will not be used. Unsupervised methods will not be used because these methods are not suitable for all words disambiguation tasks. Running the algorithms will produce results that can be measured: the accuracy of the algorithm, the speed of the algorithm, and the usefulness of the algorithm.

The accuracy of the algorithm will be results based on the percentage of recall and precision achieved; a higher result is desirable. The speed of the algorithm will be measured in seconds. Finally, the usefulness of the algorithm will be determined by using a graphics framework to create a concept map from the output of each algorithm, and scoring each concept map using the framework proposed by Calafate, Cano, and Manzoni (2009). While a given algorithm may be accurate and fast, there may be a reason it is not as useful than another algorithm. Measuring these metrics is the focus of research question 1.

Many algorithms have different parameters that can be adjusted to alter the performance of the algorithm. Most algorithms depend on a word window; the words either side of the target word being disambiguated. Increasing the size of the window may improve the performance of the algorithm; more clues will be available to determine the correct sense of the target word. However, this will increase the computational cost of running the algorithm, increasing the amount of time to run the algorithm. Furthermore, a larger word window may permit an algorithm to consider words that will deter it from determining the correct sense of the target word. The effects of changes to these algorithms is the focus of research question 2.

Developing algorithms that are able to disambiguate word senses more accurately has been the subject of research for several decades now. However, there is little research on how the accuracy of an algorithm affects practical application of an algorithm. To test this, concept maps will be generated from the results of the algorithm and evaluated qualitatively. Running algorithms with different accuracy levels on the same corpus should produce different concept maps. Whether the more accurate algorithms produce better concept maps than the less accurate algorithms is the focus of research question 3.

5.3 Data analysis

After judging the algorithms used, variations to the parameters will be made, in an attempt to find the optimal value of the parameters for each algorithm. This will be done manually, by trial and error in the case of boolean parameters. Where a parameter can be a numeric value, such as the word window, the algorithm will be run numerous times, slowly incrementing the parameter after each successful run. This will continue until an optimum value has been found, or higher values have a negligible effect on the precision and recall of an algorithm. It is almost certainly possible for this to be automated, although it is beyond the scope of this research.

5.4 Limitations

There are, of course, limits to this research. Only a few word sense disambiguation algorithms are able to be tested. This is due to time constraints. Only knowledge based methods will be tested. Testing more algorithms could be the subject of further research.

Large improvements to algorithms beyond changes in the parameter values, such as external lexical resources used, are beyond the scope of this research. The aim of this research is to find the best existing word sense disambiguation algorithm(s) for automatic concept mapping. Should any major potential alterations be identified, they will of course be identified and explained. This may be the subject of future research.

Testing the practical application of the programs created in this research in a business environment is also out of scope of this research. This would doubtless be interesting to test; this research is concerning the application of word sense disambiguation to automatic concept mapping. Whether or not this research is useful to businesses is not an insignificant matter.

This research is limited to the English language. It has been shown that certain word sense disambiguation algorithms can be applied to multiple languages with little or no modification (Dagan, Itai, & Schwall, 1991). In the case of algorithms using lexical resources, using an equivalent resource in another language will not cause any problems in regard to the algorithm running. However, using another language is outside the scope of this research.

The implementations of the algorithms used in this research will likely be less optimal than what is possible. With better implementations, more accuracy and speed will be obtainable.

6. Results and Evaluation

In this chapter, the results of applying the algorithms are evaluated both quantitatively, examining the raw performance figures of the algorithms; and qualitatively, examining the effects of different levels of performance with concept mapping software. Unexpected results were obtained, with the baseline outperforming the best algorithm in terms of recall and precision. This was investigated, and strategies for examining the issues were formulated and tested. This included modifying the best algorithm in order to gain improvements in precision and recall.

6.1 Expected Results

Predictions of the relative performance of each algorithm in terms of accuracy and speed can be made with an understanding of how each one works. For example, the baseline should be less accurate compared to the other algorithms, as the context of the target word is not taken into consideration.

As discussed in section 4.2, the Lesk algorithm will likely run much slower than the Simplified Lesk algorithm, due to having a much higher time complexity. The Lesk algorithms using hypernyms and synonyms will likely run even slower than the Lesk algorithm, as to obtain the hypernyms and synonyms for each word sense requires another embedded loop. Of course, no algorithm will run nearly as fast as the Baseline, with a time complexity of O(n).

6.2 Quantitative Evaluation

The two metrics to be evaluated quantitatively are accuracy in terms of precision and recall, and the time taken to run the algorithm. Precision is defined as "the percentage of correctly disambiguated words, out of all the words disambiguated" in a text. Recall is defined as "the percentage of correctly disambiguated words, out of all the words in the discourse" (Rada Mihalcea & Moldovan, 2000).

6.2.1 Accuracy

The algorithms tested gave varying levels of performance in terms of precision, recall, and speed. To test each algorithm, and to address the issue of optimal word window size, each algorithm was run on a subset of the SemCor corpus 25 times: starting with a word window size of two (one word either side of the target word) and incrementing by two words with each successive run.

Word	Baseline		Lesk		Simplified Lesk		Hypernym-Lesk		Synonym-Lesk	
Window	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
2	50.57	39.20	32.67	25.11	41.25	21.59	28.01	15.22	34.28	26.57
4	50.57	39.20	32.51	24.99	40.64	21.89	28.12	15.28	34.28	26.57
6	50.57	39.20	32.45	24.95	39.99	21.99	28.17	15.31	34.28	26.57
8	50.57	39.20	32.39	24.91	39.40	22.06	28.21	15.33	34.28	26.57
10	50.57	39.20	32.36	24.89	38.89	22.11	28.25	15.35	34.27	26.56
12	50.57	39.20	32.33	24.87	38.51	22.20	28.28	15.36	34.28	26.57
14	50.57	39.20	32.30	24.85	38.12	22.28	28.30	15.38	34.28	26.57
16	50.57	39.20	32.30	24.84	37.84	22.38	28.32	15.39	34.28	26.57
18	50.57	39.20	32.31	24.86	37.58	22.46	28.32	15.39	34.27	26.56
20	50.57	39.20	32.31	24.86	37.44	22.61	28.35	15.40	34.28	26.57
22	50.57	39.20	32.29	24.85	37.29	22.72	28.38	15.42	34.28	26.57
24	50.57	39.20	32.29	24.85	37.08	22.78	28.39	15.42	34.28	26.57
26	50.57	39.20	32.28	24.84	36.94	22.88	28.40	15.43	34.28	26.57
28	50.57	39.20	32.29	24.85	36.77	22.94	28.41	15.44	34.28	26.57
30	50.57	39.20	32.28	24.85	36.67	23.03	28.43	15.45	34.28	26.57
32	50.57	39.20	32.28	24.85	36.58	23.11	28.44	15.45	34.27	26.56
34	50.57	39.20	32.26	24.84	36.46	23.17	28.44	15.45	34.27	26.56
36	50.57	39.20	32.26	24.84	36.35	23.22	28.44	15.45	34.27	26.56
38	50.57	39.20	32.27	24.84	36.27	23.30	28.43	15.45	34.26	26.56
40	50.57	39.20	32.27	24.85	36.20	23.38	28.43	15.45	34.26	26.55
42	50.57	39.20	32.27	24.85	36.11	23.42	28.43	15.45	34.25	26.55
44	50.57	39.20	32.25	24.84	36.05	23.48	28.44	15.45	34.26	26.55
46	50.57	39.20	32.25	24.84	36.02	23.57	28.44	15.46	34.26	26.55
48	50.57	39.20	32.25	24.84	35.95	23.62	28.46	15.46	34.25	26.55
50	50.57	39.20	32.26	24.85	35.89	23.67	28.48	15.47	34.26	26.55

Figure 9: Percentage precision and recall of the tested algorithms over a varying word window

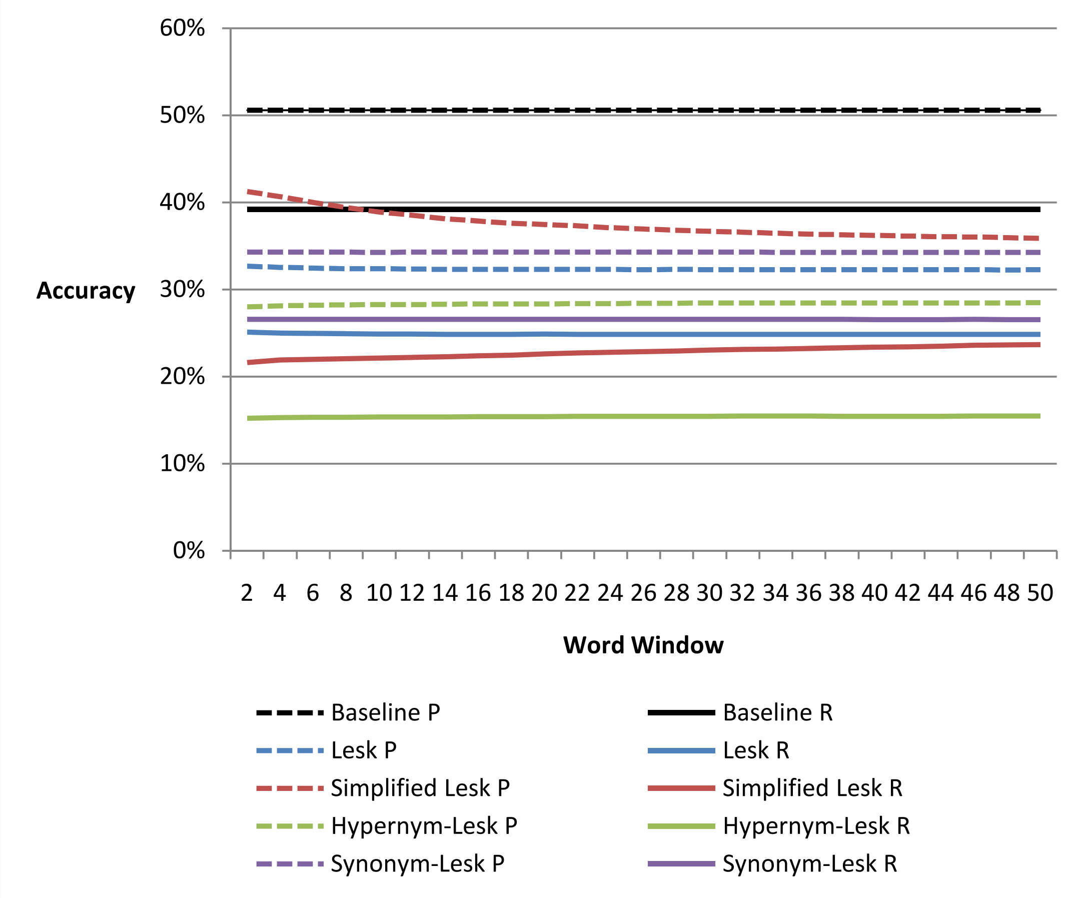


Figure 10: Graph of data from Figure 9

The most frequent sense baseline algorithm was the most accurate algorithm tested, with a recall of 39%, and a precision of 51%. As this algorithm ignores context, the results are the same regardless of word window size. As the baseline is unaffected by the size of the word window, the graph is straight horizontal lines.

The baseline algorithm shows an interesting statistic: only 135,981 words of the 175,444 words in the SemCor subset were assigned a sense label. As the only way for the baseline to not assign a sense is the program not finding a word in the WordNet dictionary (often due to proper nouns, or words such as 'a', 'the' and 'to'), no algorithm could reach a recall of more than 77%. Modifying the recall score based on 135,981 words, instead of 175,444 words, increased the precision of all algorithms, as seen in Figure 11 and Figure 12. As precision is calculated based on the number of words attempted, it is unaffected by this change. This change in scoring causes any algorithm that assigned a sense to every possible word to have an equal precision and recall. This was the case for the baseline and synonym algorithm. The Lesk algorithm attempted to assign a sense to almost every word, not assigning a sense to just 1,017 words, resulting in a difference between precision and recall of just 0.24%, not shown in the table due to rounding. While each algorithm gained a different increase in recall, the relative positions between the algorithms are unchanged.

Algorithm	Precision (%)	Old Recall (%)	New Recall (%)
Lesk	33	25	32
Simplified Lesk	41	22	28
Hypernym Lesk	28	15	20
Synonym Lesk	34	27	34
Baseline	51	39	51

Figure 11: Table showing the results of modifying how recall is calculated

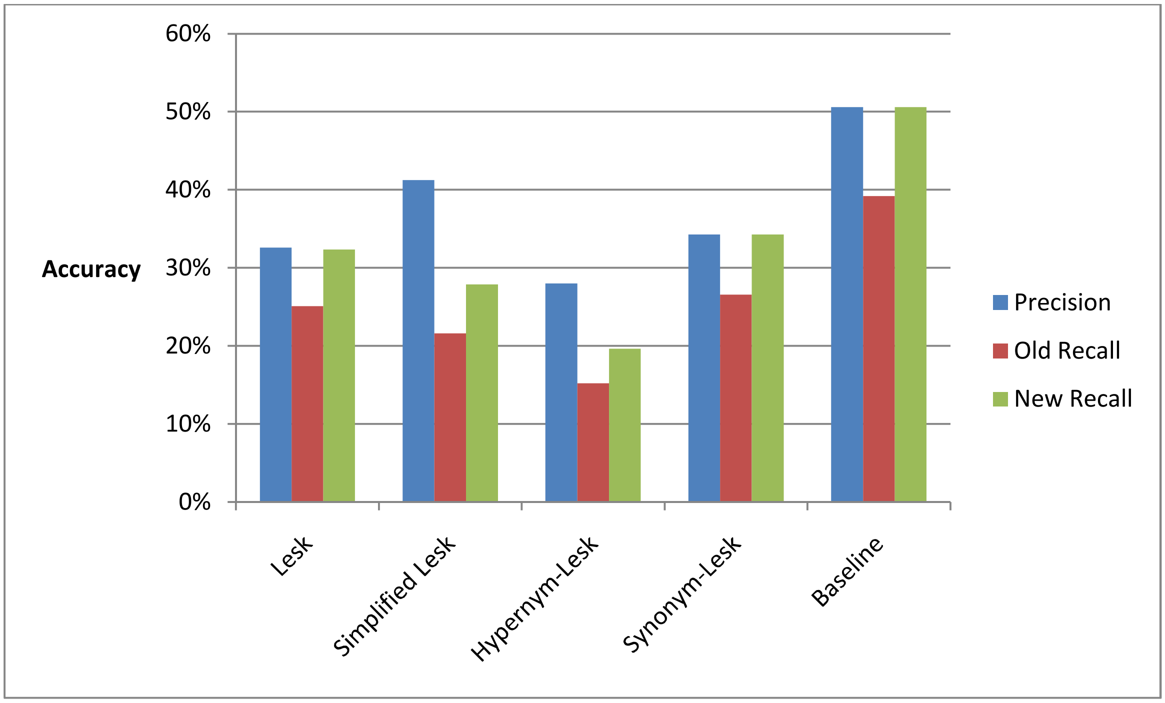


Figure 12: Graph showing the results of modifying how recall is calculated

The accuracy of the Lesk algorithm was unaffected by changing the size of the word window. The precision attained was 32%, with a recall of 25%. Figure 10 also shows the Lesk algorithm is around the average of the algorithms tested. The Simplified Lesk algorithm had higher results. This algorithm scored a precision and recall of 41% and 22% at best, considerably lower than the baseline. Figure 10 shows the precision varied by 24%, and the recall varied by 21%. The results also show an interesting curve; as the word window increases in size, the precision drops while the recall rises. However, the precision drops more than the recall rises; the precision falls nearly 6%, whereas the recall rises only 2%. This suggests that when the word window increases, words unrelated to the target word are considered, and are negatively impacting the results.

However, to determine the full extent of this trend, the Simplified Lesk algorithm was re-run on the corpora, with word windows of 200, 500, and all the words in the text. With a word window of 200, the precision dropped to 35%, with an 'old' recall of 25% and a 'new' recall of 33%. With a word window of 500, the precision fell to 34%, with 'old' recall reaching 26%, and new recall scoring 34%. With the entire text being considered for every word being disambiguated, 34% precision was achieved, 26% recall using the 'old' method, and 34% recall using the 'new' method. However, the difference between the precision and 'new' recall was 0.33%, which was lost in rounding. From these results, it could be argued that with a larger word window, the Simplified Lesk algorithm sacrifices higher precision for recall. If a certain application was more dependent on recall than precision, a huge word window would be preferable. However, with a word window this large, the Synonym-Lesk algorithm with a small word window has equal precision, and 1% more recall than the Simplified Lesk.

Unlike the Simplified Lesk Algorithm, changing the size of the word window has a negligible effect on the precision and recall on the Lesk algorithm using synonyms, at 34% and 27% respectively. The Lesk Algorithm using hypernyms also showed no overall benefit of a larger word window. Overall, the precision was 27%, where the recall was 15%, the least accurate of the algorithms tested.

6.2.2 Speed

The time taken to process the SemCor corpus was measured in seconds, the results of which are graphed in Figure 13 and 14. As no algorithm tested saw any substantial difference in performance with an increased word window, only the time taken with a word window of two is displayed here. Graphs showing the time taken over a varied word window can be seen in Appendix E.

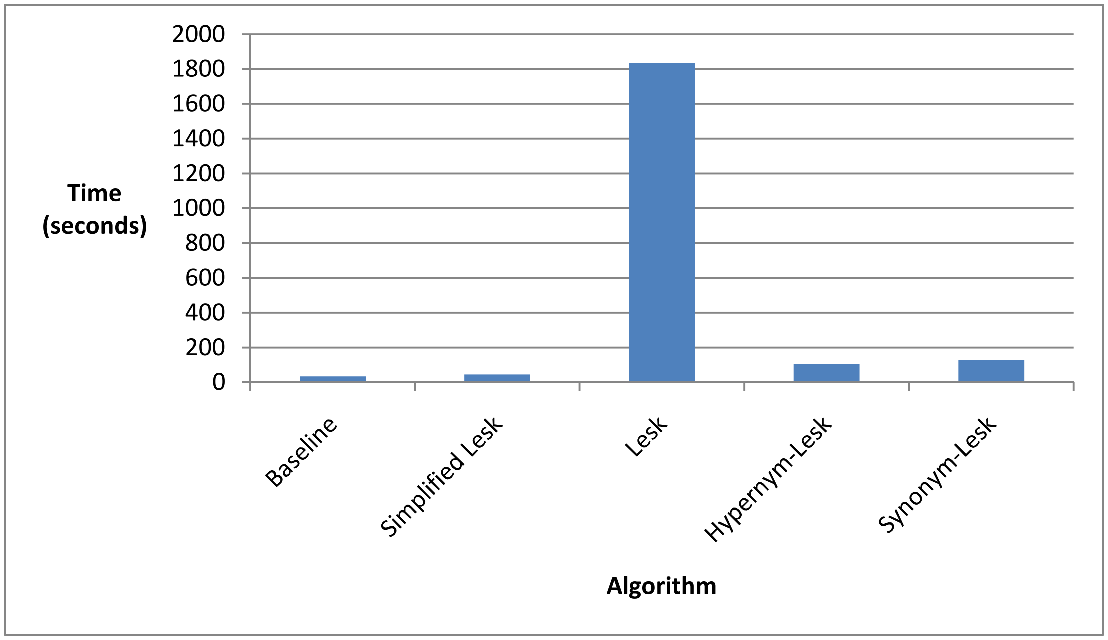


Figure 13: Comparison of the speed of the algorithms

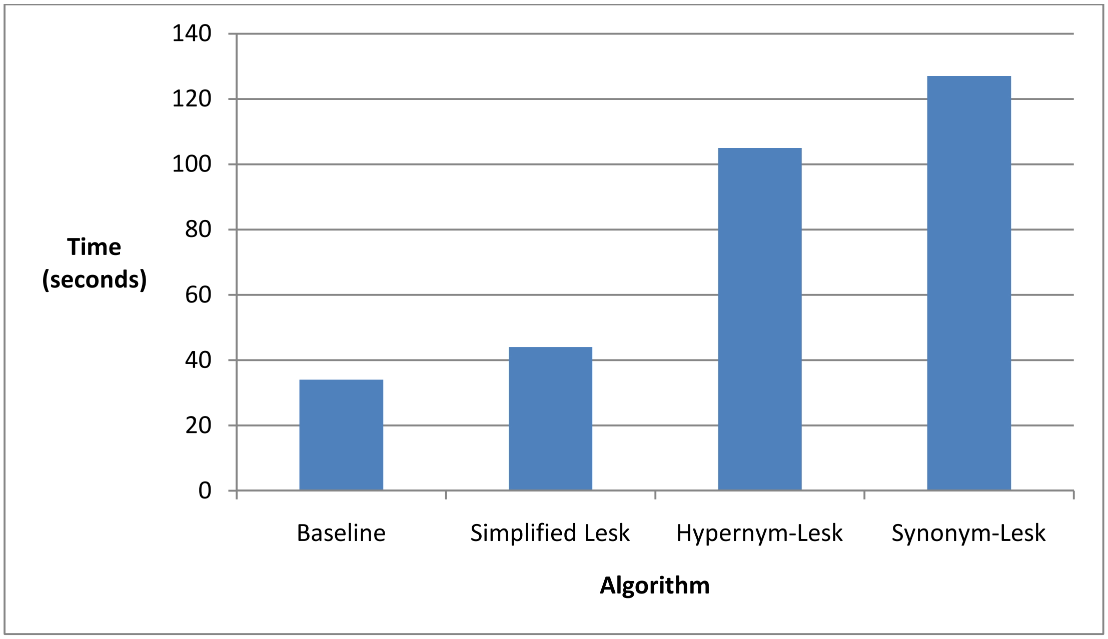


Figure 14: Comparison of the speed of the algorithms, excluding the Lesk algorithm

The baseline was unsurprisingly the fastest algorithm, taking only 34 seconds to process all 175,444 words. The simplified Lesk algorithm was not much slower, taking only 44 seconds with a word window of two. The Lesk algorithm using hypernyms was next quickest, taking 105 seconds to complete with the smallest word window. The Synonym-Lesk algorithm was not far behind, finishing in 127 seconds. However, the Lesk algorithm was far slower than all the other algorithms tested, taking 1837 seconds, or a little over 30 minutes to complete the SemCor corpus.

There are two unexpected trends with the speed of these algorithms. The first is the time taken for the Lesk algorithm to complete the corpus. While it is unsurprisingly the slowest algorithm, it does not come close to the speed of the other algorithms. To test if this was an odd quirk of the test system, every algorithm was rerun on a second, faster system, with an updated version of Java. Unsurprisingly, each algorithm completed the SemCor corpus quicker on the more powerful machine, but the Lesk algorithm was still disproportionally slower. It would appear that the Lesk algorithm simply does far, far more operations than the other algorithms.

The second odd result is the time taken to run the Baseline algorithm. As it only examines each word in the text once, it should be much faster than the Simplified Lesk algorithm, which examines each word numerous times, and must also access WordNet far more times. However, the Simplified Lesk algorithm is only ten seconds slower than the Baseline. On the second system, this difference was reduced to just six seconds. It is possible the mechanical hard drive is forming a bottleneck - even with more power the Java program could not read in the text files quickly enough. To test this, both WordNet and the corpus files were moved to a 10,000 rpm Velociraptor hard drive, the fastest drive that could be obtained. However, the speed difference between the drives were not evident, with the only algorithm showing a measureable difference was the Lesk algorithm, performing just 7 seconds faster on the faster drive. It is possible that the corpus files and WordNet are being cached, or that a faster solid state drive would show a difference. Unfortunately, determining if the files were being cached effectively was unable to be determined, and a solid state drive was unable to be procured for this research.

CPU: Intel Core i7 2600K @ 4.2GHz
RAM: 8GB DDR3 1600MHz
HDD1: Western Digital 2000GB 5400rpm SATA2
HDD2: Western Digital 600GB 10000rpm SATA3

Operating System: Windows 7 Home x64
Java Version: Version 6 Update 24

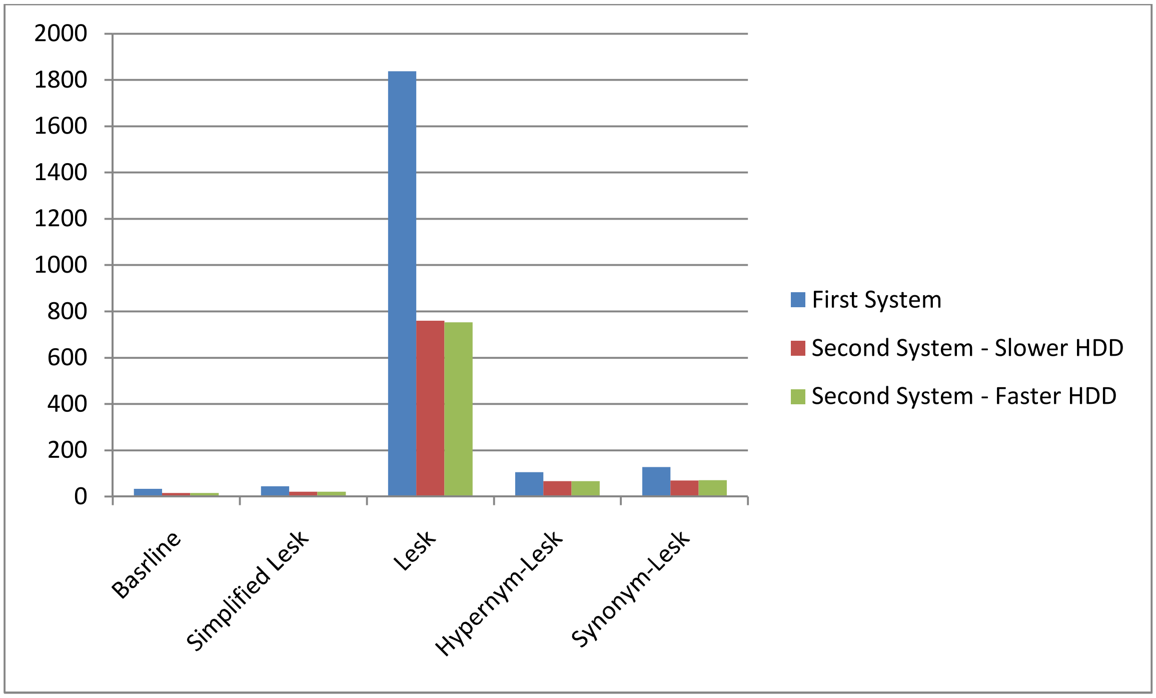


Figure 15: Comparison of the speed of the algorithms on the different test systems

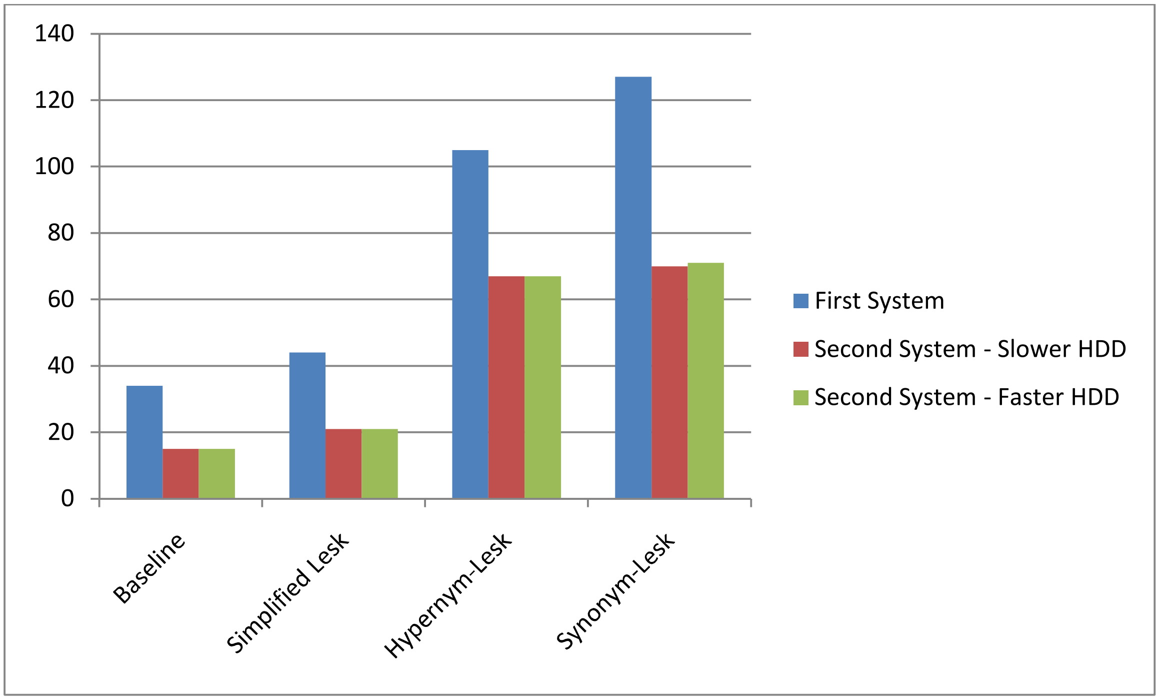


Figure 16: Comparison of the speed of the algorithms on the different test systems, excluding the Lesk algorithm

Based on the quantitative analysis, the Lesk algorithm could not be considered the best. The accuracy achieved was approximately average, but took far longer than any other algorithms to complete the corpus. While the corpus was not small, and issues with speed could conceivably be reduced by simply adding more processing power, this algorithm was still 15 times slower than the next slowest algorithm. The Simplified Lesk algorithm performed much better. This algorithm had high precision, was very fast, but had average recall. The Hypernym-Lesk algorithm did not perform particularly well. This algorithm had below average precision, recall, and not particularly fast. It is difficult to see why this algorithm would be used. The Synonym-Lesk algorithm was a stronger performer, with above average precision and recall. However, it was slightly slower than the other algorithms, other than the Lesk algorithm. Ultimately, it is hard to ignore the performance of the baseline algorithm. Not only did it complete the corpus quicker than the other algorithms, it was 10% more accurate in terms of both precision and recall. An investigation into how the baseline was so accurate is detailed in section 6.4.1.

From these results, research questions one and two can be answered. The algorithm that disambiguated words most accurately was the Simplified Lesk algorithm. The Simplified Lesk algorithm appears to be slightly more accurate than the other algorithms, though it does sacrifice some recall for precision. Conveniently, accuracy does not come at the cost of high computational resources, as the Simplified Lesk algorithm was also the quickest performing. Regarding the complexity of the corpora affecting the accuracy of an algorithm, algorithms generally did not experience wild fluctuations in accuracy between different corpora. As can be seen in Appendix D, a few corpora tended to favour one algorithm while possibly giving a lower than average result with another. However, anything more than a few percentage points were exceptions.

Regarding research question two, it can be determined that the performance of algorithms can be improved by using a very small word window; the best results obtained in this research used only one word either side of the target word. However, the exception to this is the Simplified Lesk algorithm: with a word window large enough to cover the entire text, the Simplified Lesk algorithm could gain more recall at the expense of precision. Should that be preferable, the word window should be as large as possible.

6.3 *Qualitative Analysis*

To determine what effect the accuracy of algorithms has on real world applications, the different algorithms were run on a random selection of articles from the SemCor corpus. The output of the algorithms was fed into an automatic concept mapper, and scored using the methodology proposed by Calafate et al (2009). The precision and recall of each algorithm on each corpus were also obtained, to determine the strength of correlation between the accuracy of an algorithm and the concept map score.

Algorithm	**Precision (%)**	**Recall (%)**	**Concept Map Score (%)**
Lesk	33	25	36
Simplified Lesk	41	19	19
Hypernym Lesk	32	18	39
Synonym Lesk	34	26	21
Baseline	51	39	42

Figure 17: Precision, recall and concept map score obtained on the BR-F10 corpus

The first article randomly selected was BR-F10, an article about doctors selling phony therapeutic devices for profit. Looking at concept maps produced by each algorithm, every algorithm with the exception of the Lesk algorithm charted 'helium' and 'World Health Organisation' as the two most predominant nodes (the

Lesk algorithm did not find 'World Health Organisation', only 'helium'). However, neither 'helium' or 'World Health Organisation' were mentioned in the article. WordNet does not include pronouns such as 'he' and 'who', though it does contain 'He', the chemical symbol for helium, and 'WHO', the acronym for the World Health Organisation. Therefore, whenever the program encountered 'he' or 'who', 'helium' and 'World Health Organisation' were considered viable senses. This could be partly improved with a refined method of comparing words in a text with words in WordNet. This could be an area for further research.

In this particular text, each algorithm tested scored a recall of between 18% and 26%, with precisions ranging from 31% and 40%. The baseline performed well, 38% recall and 50% precision. Each of the algorithms tested produced similar concept maps: each correctly identified most of the important ideas in the article, such as quack, machine, cancer, remedy, and medical. However, there were some clear mistakes, in addition to the helium and WHO errors mentioned. For example, the Hypernym-Lesk algorithm found 'Doctor' could mean 'Doctor of the Church[5]' on six occasions. The Simplified Lesk algorithm produced arguably the worst results, despite having the highest recall after the baseline. While this algorithm identified 'helium' less frequently than the other algorithms, WordNet's other sense of 'he', the 5th letter of the Hebrew alphabet, was assigned instead. Furthermore, while more of the key concepts were identified, they were identified less frequently than any other algorithm. This could be because the Simplified Lesk algorithm had a lower precision than the other algorithms, at just 19%, only 1% better than the Hypernym-Lesk algorithm.

The next article randomly selected was BR-D03, an opinion piece observing current trends of Catholic, Protestant and Anglican churches and groups in London. On this

[5] WordNet 2.1 defines as "a title conferred on 33 saints who distinguished themselves through the orthodoxy of their theological teaching" (WordNet, 2011)

corpus, each algorithm scored poorly when compared to the BR-F10 corpus. Precision ranged from 28% to 34% (with three algorithms scoring 34%), with the baseline scoring 44%. Recall ranged from 14% to 27%, with the baseline scoring 35%. With each algorithm scoring closely, no algorithm was noticeably better or worse than another.

The concept maps generated from this text, an example of which can be seen in Figure 18, scored noticeably highly, as not only did every algorithm correctly identify each main concept, but were able to link each concept. This is largely due to the concepts being related closely within WordNet. The concepts; church, religion, belief, Christian, Catholic and Protestant; are all closely linked in WordNet, making it clear that these are related concepts. In other texts, the concepts were not always related, or were related only in certain contexts. Furthermore, the main concepts in this text happen to have only a few different senses in WordNet, making it more likely they will be correctly tagged.

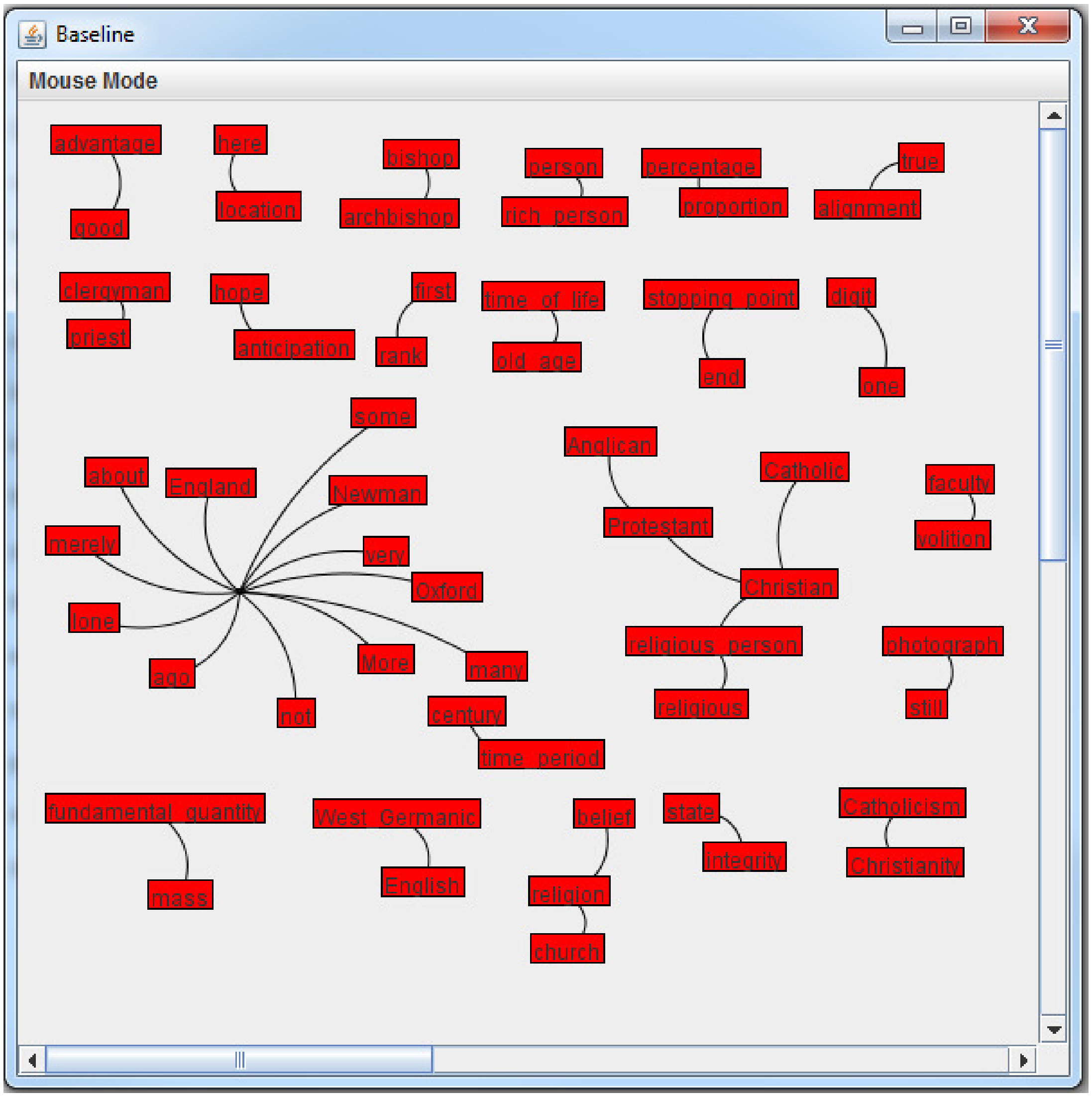


Figure 18: Concept map generated from the output of the baseline

The next article examined, BR-G28, discussed the literary works of William Faulkner, a South American writer. Recall ranged from 29% to 47%, and precision ranged from 12% to 27%. The baseline scored 54% recall and 39% precision. The most interesting results came from the Simplified Lesk algorithm, which scored 47% recall and 22% precision, but failed to correctly identify either of essential concepts of the text. As a predominant writer, Faulkner appears in WordNet. His name appears nine times in the article. However, the Simplified Lesk algorithm only assigned a sense tag to one instance of his name. Furthermore, at no point did the

Simplified Lesk algorithm assign senses to instances of the word 'south' or 'southern', another key theme in the text. However, out of the algorithms tested, this algorithm had the highest recall, and only 5% less precision than the most precise algorithm, the Lesk algorithm. This shows that greater precision and recall does not necessarily mean better concept maps can be produced.

The next article, BR-J11, discussed various methods of measuring the size of giant snakes. The algorithms tested had recall ranging from 30% to 45%, and precision ranging from 16% to 26%. The baseline scored 50% recall and 38% precision. A concept incorrectly identified in all algorithms except the Simplified Lesk algorithm was assigning the sense 'rich person' to instances of the word 'have'. Like the algorithm's tendency to label the pronoun 'he' with 'helium', this could be improved by allowing the algorithms to better identify 'have' as a verb, therefore not allowing a noun word sense to be assigned.

When processing BR-K15, a fictional text, each algorithm had difficulty identifying the main concepts. This was likely because of the written style of fictional texts: many of the key concepts require 'reading in between the lines', rather than being explicitly stated. This text was about dealing with the death of a child. A journal article on the same subject matter would be written very clearly and explicitly, unlike this text. These difficulties are shown from overall poor performance in terms of precision and recall, but especially with the concept map scores. No algorithms were able to link any concepts together, as there were no links between the few concepts that were identified. For example, the Simplified Lesk algorithm was able to identify death and child, but could not link the two. Worst performing was the standard Lesk algorithm, which could not successfully identify any of the key concepts.

BR-R05 is a humorous article discussing wit coming from (funnily enough) ambiguity in the English Language. The fine grained nature of WordNet is seen in the results of this text, with algorithms having difficulty differentiating between the two senses of 'ambiguity': "an expression whose meaning cannot be determined from its context", and "unclearness by virtue of having more than one meaning" (WordNet, 2011). Clearly the difference is subtle at best.

6.4 Discussion of Results

Unfortunately, the results gained thus far are not particularly satisfying; the baseline should be the worst performing algorithm, preferably by a large margin. The baseline does not take any context into consideration, it simply assigns a sense to the target word and goes onto the next word. For this reason it is unsurprising that it is the fastest algorithm, but it should be the least accurate. It is unlikely the corpora used for testing are the cause of this, with over 150,000 words, SemCor could easily be considered a large enough sample size to test the algorithms. An analysis of the implementation of the algorithms, and the scoring system, revealed nothing. Coming at the issue from a different approach, there could be two reasons for this trend; the baseline is unexpectedly accurate for a minimum starting point to gauge performance, or the algorithms are not disambiguating words as accurately as they should be.

6.4.1 Baseline Performance

The results show a unexpected trend of the baseline algorithm performing consistently more accurately all other algorithms, even though it should be the least accurate system. However, this could be explained by how the baseline picks a sense. In WordNet, word senses are listed in the order they are most likely to be used; the most frequent sense of a word is listed first, based on other literature studied by Princeton University. This baseline is often used in research involving

WordNet, seen in (Mihalcea & Moldovan, 1999; Pedersen, 2010; Preiss, Dehdari, King, & Mehay, 2009; Vasilescu, et al., 2004). Had WordNet not used any order in listing word senses, or another lexical resource with no sense ordering was used, this baseline could not be made.

It could therefore be argued that picking the first sense listed in WordNet is not an appropriate or fair baseline, as it makes use of predetermined statistical information not used by the other algorithms. Depending on how a baseline is defined, an algorithm that assigns the first WordNet sense to a word is not necessarily an accurate baseline. WordNet 2.1 defines this sense of baseline to mean "an imaginary line or standard by which things are measured or compared" (WordNet, 2011). The Oxford Dictionary defines a perhaps more conventional sense of baseline to mean "a minimum or starting point used for comparisons" (Oxford Dictionary, 2011). The key distinction The Oxford Dictionary makes is that it suggests a baseline should be a minimum to measure by. Using this definition, it could be argued that a completely random algorithm, that made no use of previous knowledge regarding the likelihood of word senses, would serve as a better baseline to compare performance.

To determine how a truly random system would perform, a second baseline algorithm was implemented. Rather than assigning the first sense of a word, this algorithm randomly picked a sense out of all the senses listed. As a result, this algorithm could assign obscure and rarely used senses to words, and should therefore be far less accurate. As context is still not considered, it performs as fast as the most frequent sense baseline. Finally, for completeness, a 'least frequent sense' algorithm was implemented. This algorithm was made to always pick the last sense of a word listed, which should be the statistically least likely.

To test this idea, the Random Baseline was run on the same SemCor corpora as the other algorithms, to ensure fair testing. As context is ignored, the size of the word window is irrelevant, but was set to two. As Figure 19 and Figure 20 show, the Random Baseline was slightly more accurate than the Synonym Lesk algorithm, slightly less accurate than the standard Lesk and Hypernym Lesk, and less precise than the Simplified Lesk. The least frequent sense algorithm is the lowest scoring out of all the algorithms.

Algorithm	Precision (%)	Recall (%)
Lesk	33	25
Simplified Lesk	41	22
Synonym Lesk	34	27
Hypernym Lesk	28	15
Baseline	51	39
Random	30	23
LFS	19	15

Figure 19: Table comparing previously tested algorithms and a Random Baseline

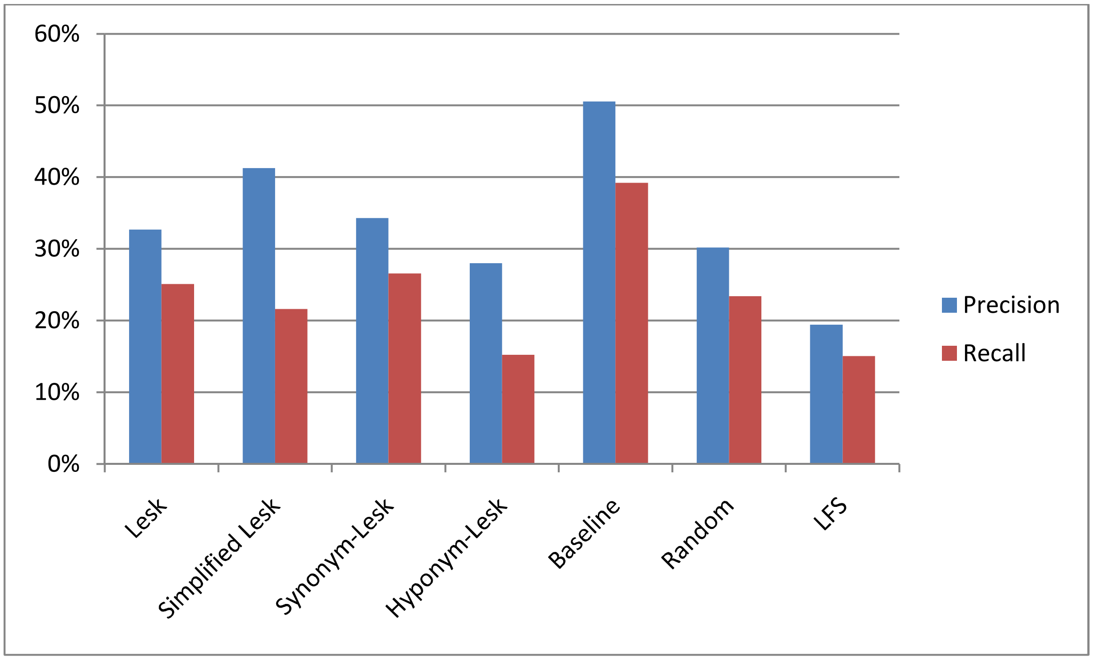


Figure 20: Graph comparing previously tested algorithms and a Random Baseline

As the Random Baseline will pick different word senses each time it is run, the precision and recall will vary. To ensure fair comparisons to the other algorithms, which will gain the exact same score each time they are run on a given corpus, the Random Baseline was run over the entire SemCor corpus ten times. This should ensure fair comparisons of the results of other deterministic algorithms to results that depend on luck, that can fluctuate wildly. After running the Random Baseline ten times over the SemCor corpus, the precision and recall varied less than 0.4%. This demonstrates that the SemCor corpus is more than large enough to provide a good sample size of data.

While the algorithms tested only perform slightly better than completely random guessing, this does suggest that guessing word senses based on predetermined statistics, like the initial baseline tested, is a viable strategy for word sense disambiguation. To test this further, the Simplified Lesk algorithm was modified to make use of this data in WordNet. An exploration of this idea is in section 6.4.2.

To test the Random Baseline on concept mapping, four texts were chosen: BR-D03, BR-G28, BR-E04, and BR-F10. Based on the concept map scores achieved by the other algorithms, these texts appear to represent a variety of difficulties, in terms of creating concept maps; BR-D03 appears quite easy, while BR-F10 appears much harder. Due to the random nature of the Random Baseline, each corpus was tested three times, with the average precision, recall and concept map scores recorded. Ultimately, these scores did not fluctuate more than a few percent between tests.

The BR-D03 text appeared to be the easiest text to create a concept map of, due to the low number of senses for the main concepts and clear links between these concepts. The Random Baseline scored an average concept map score of 71% over the three runs; a high score, but lower than the other algorithms. This was due to being unable to identify 'believe' as a secondary concept on each of the three tests, and failing to link some concepts in one run. Believe has a total of five different senses in WordNet, as compared to only two or three senses for the other main concepts. Therefore it is to be expected that the Random Baseline failed to consistently tag multiple instances of the word with the same sense.

The average recall and precision of the Random Baseline on BR-D03 was 31% and 24%, not significantly lower than other algorithms. The Random Baseline also scored highly in the BR-G28. The high score was partly from one key concept, 'Faulkner', having only one sense in WordNet. Furthermore, like BR-D03, there were clear links between the concepts of 'Faulkner', 'South', and 'literature'. This resulted in a concept maps score of 60%, equalling the Original Lesk algorithm, and far surpassing the Hypernym-Lesk algorithm. However, the precision and recall were lower than that of most of the other algorithms, at 30% and 22%.

The BR-E04 text proved to be more difficult than BR-D03 and BR-G28 for the Random Baseline. While the precision and recall achieved was not significantly lower than that of the other algorithms, the Random Baseline struggled with the 13 different possible word senses of the term 'record', and was only able to correctly link 'sound' and 'music' in one test. The final concept map score averaged out to be 27%, behind the Original Lesk algorithms score of 35%, and the 49%-50% scores of the other algorithms.

While once again, the precision and recall of the Random Baseline algorithm on the BR-F10 text were not significantly lower than the other algorithms, the concept map produced was very poor, scoring 0%. While every algorithm failed to identify 'medical' as an essential concept, every other algorithm identified at least one other essential concept (either 'treatment', or 'quack'). However, on only one test of the Random Baseline was 'quack' identified to mean a fake doctor. Furthermore, on no test were any concepts correctly linked together.

These results show, correlation between the accuracy of an algorithm and the concept map score. The Random Baseline is the lowest scoring algorithm, and the concept map scores obtained from its results are also the lowest (with two outliers - the Hypernym-Lesk and Simplified Lesk algorithms on BR-G28). However, there is stronger correlation between the score of the concept map produced from the Random Baseline results, and the concept map scores produced from the output of other algorithms. This suggests that while the accuracy of the algorithms has some effect on the concept map scores, the concept map scores depend more on the particular corpus being disambiguated.

6.4.2 Improving Performance

As the original baseline algorithm performed strongest, each algorithm was modified to guess the most frequent sense of a word when no words in the context provided clues of the correct sense. As mentioned, numerous words in the corpus could not be found in WordNet. In these cases, no word sense was assigned. The results of this are shown in Figure 21. In terms of speed, this modification had no noticeable effect on the time taken to run each algorithm. The Lesk and Synonym-Lesk algorithms saw very little difference in performance. The Simplified Lesk algorithm saw a measurable benefit, with precision rising 4%, and recall rising 14%. The Hypernym-Lesk algorithm saw even more benefit, with precision rising 9%, and recall rising 14%. While the Simplified Lesk algorithm scored precision and recall within 5% of the baseline, all algorithms should be performing better than the baseline.

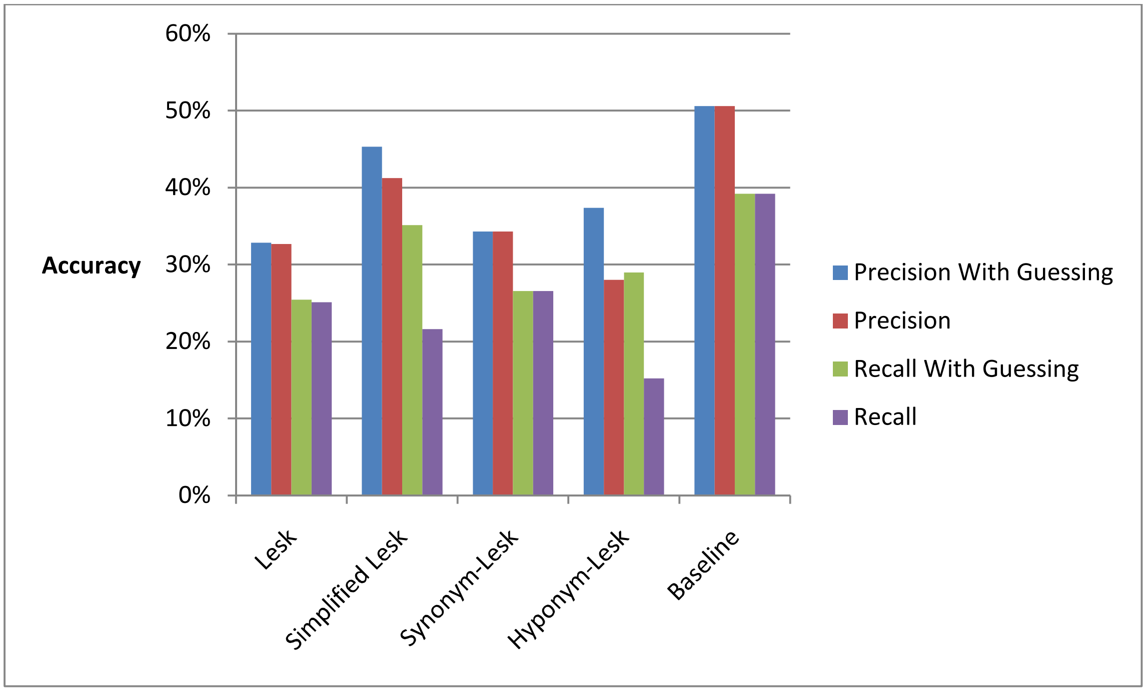


Figure 21: Comparison of the algorithms with guessing enabled

As discussed, the fine grained nature of WordNet can impede accuracy of word sense disambiguation algorithms. A way to combat WordNet's fine grained nature was devised by Mihalcea and Moldovan (1999): to only consider the first two senses listed in WordNet for any given word. This modification was applied to the Simplified Lesk algorithm, and tested in the previous manner. Testing revealed a 3% increase in precision, and a 0.2% increase in recall over the unmodified Simplified Lesk algorithm. While this is still a performance increase, albeit a small one, this modification will almost certainly automatically prevent some words from having the correct word sense applied. A better way to work around WordNet's fine grained nature would be to modify WordNet in some way to increase the granularity of word senses.

As figures 28 to 38 in Appendix D show, algorithms can sometimes score poorly in terms of concept map score compared to other algorithms, without showing a lower precision and recall. This is particularly the case with the Simplified Lesk algorithm. As the most frequent sense baseline appeared to be immune from this effect (whenever the baseline obtains a poor concept map score, all algorithms do as well, such is the case with BR-K15), perhaps using the most frequent sense information in WordNet is the solution. In order to test this, the Simplified Lesk with Guessing algorithm was re-run on the BR-G28, BR-J01, and RB-R05 corpora. The results of precision, recall, and concept map score were added to Figure 29, Figure 34, and Figure 35 respectively in Appendix D.

Without modification, the Simplified Lesk algorithm was unable to correctly identify the two main concepts of BR-G28 of 'Faulkner' and 'South'. With guessing enabled, these two concepts were identified, and both linked to the concept 'literary'. With these concepts linked together, the concept map score grew from just 6%, to 66%. The concept map produced from the Simplified Lesk output of the BR-J01 text, a journal article extract concerning radiation emitted by various planets, was also relatively low scoring without modification, at 25%. This was due to the failure to identify 'thermal' as an essential concept, as well as three out of five secondary concepts. With guessing enabled, 'thermal' was correctly identified as an essential concept, as well the remaining secondary concepts. However, like the concept maps produced by other algorithms, none of these concepts were linked together. The final concept map score for the text was 57%, more than double the score of the unmodified Simplified Lesk algorithm.

The concept map scores of the BR-R05 corpus proved interesting, with the baseline and Hypernym-Lesk algorithms scoring 67%, and each of the other algorithms scoring 15%. This can largely be explained due to the low number of essential concepts, only four were chosen in this article. The Simplified Lesk algorithm initially scored poorly due to not identifying the concepts of 'ambiguity', and 'anatomical reference'. When guessing was enabled, both these concepts were correctly identified. Furthermore, both these concepts were correctly linked in the concept map, increasing the concept map score to 67%. As the Original Lesk algorithm obtained a concept map score of 15%, the same as the Simplified Lesk algorithm, its modified version was also run on BR-R05. However, its concept scores did not increase. The modified Lesk algorithm did score a slightly higher precision and recall than its unmodified counterpart, although this difference is not shown due to rounding. While the Synonym-Lesk algorithm also had a concept map score of 15%, this algorithm never failed to assign a sense to a word. As the modification only takes affect when an algorithm cannot assign a word sense, the Synonym-Lesk algorithm is unaffected by the modification.

These results confirm the previous findings of research questions two and three. There are small adjustments to parameters that can be made to algorithms. By using the predetermined statistical data in WordNet, algorithms can become much more accurate. This was shown by setting the algorithms to guess the most likely sense of a word instead of not assigning any sense. However, if a scoring method that penalised incorrect answers was used, this may not be viable. Algorithms could also be set to only consider the first (and therefore most likely) two senses of a given word. However, as this automatically stops an algorithm finding some correct answers, this may also not be viable, depending on how the algorithm was used. Furthermore, these results reinforce what was previously found on research question three. While there is some correlation between the accuracy of an algorithm, there is still greater correlation between the results of individual corpora.

6.5 Summary of Results

Through initial evaluation of the algorithms, it was found that the Simplified Lesk algorithm was the most accurate algorithm. While the recall was slightly lower than that of the Lesk or Hypernym-Lesk algorithms, the precision was noticeably higher than the other algorithms. However, all algorithms proved to be less accurate than the baseline. The Simplified Lesk algorithm was also the fastest performing algorithm, processing 175,444 words in just over thirty seconds on the test system. The Hypernym-Lesk and Synonym-Lesk algorithms took just over one minute each to process the same corpora. However, the Lesk Algorithm took disproportionally longer, taking thirty minutes. When it came to creating concept maps with the output of the algorithms, results were varied, with average concept map scores ranging from around 80% in the BR-D03 corpus, to 10% on the BR-K15 corpus. It appeared that the concept map score depended more on the individual corpus than the algorithm used to get the results.

To explore the results further, the baseline accuracy was examined further. This examination revealed that the unusually high accuracy of the baseline could be explained by the way WordNet organises senses for each word: the most frequent senses of a word are arranged first, with the least likely senses arranged last. This meant the baseline, which always picked the first sense of a word, would always pick the most frequent sense of a word, based on research by Princeton University. Therefore, another baseline was implemented that would pick a sense completely randomly. This random baseline proved to be less accurate than the most frequent sense baseline, and each other algorithm, save the Hypernym-Lesk algorithm.

In order to boost the accuracy of the algorithms, modifications were investigated. In order to make use of the statistical data in WordNet, each algorithm was set to guess the most frequent sense of a word, if no sense could be determined for a given word. As Lesk and Synonym-Lesk would rarely not be able to assign a sense to a word, this modification had little effect. However, the Hypernym-Lesk and Simplified Lesk algorithms saw substantial increases in both precision and recall. Another modification tested was setting an algorithm to only consider the first two (and therefore the most likely two) senses of a word. However, the precision and recall gained from this modification were rather small.

7. Conclusion

This research has examined the accuracy of a number of word sense disambiguation algorithms, and used the output of these algorithms in order to automatically generate concept maps. This was to done to address two issues in previous word sense disambiguation research. The first issue was that much research was done on a small number of target words in a given corpus, typically less than a dozen words. By focussing on just these words, it is difficult to predict how accurate the algorithm or algorithms used would be in a real world application, where all the words in a corpus would almost certainly need to be disambiguated. The second issue from previous research in the field was a lack of testing in a real world application of word sense disambiguation. This research addressed these issues by attempting to disambiguate every word in a large, varied corpus, and then using the output of the algorithms to automatically generate concept maps. These concept maps would be used to determine what, if any, effect more accurate word sense disambiguation algorithms had in a real world application.

The first thing investigated was the accuracy of the algorithms, to answer research question one. By running each algorithm tested on the SemCor 2.1 corpus, it was found that the Simplified Lesk algorithm was the most accurate. While the recall of this algorithm was slightly worse than the algorithms, the precision was noticeably higher, as seen in section 6.2.1. Conveniently, this was also the fastest system to disambiguate SemCor, processing 175,444 words in just over 40 seconds on the test system, as seen in section 6.2.2. Extrapolating, it can be assumed that Tolstoy's 460,000 word long epic novel War and Peace could be disambiguated in under two minutes (Tolstoy, 1949). From these results, it is clear that accuracy does not necessarily require more processing power; the standard Lesk algorithm took far longer to process SemCor than any other algorithm, but was less accurate overall. It can be seen in Appendix D that the corpus being disambiguated does not have a great effect on the accuracy of an algorithm.

Regarding research question two, it can be determined that the performance of algorithms can be improved by using a very small word window; the best results obtained in this research used only one word either side of the target word, as seen in section 6.2.1. However, the Simplified Lesk algorithm proved to be an exception to this rule; with a word window large enough to cover the entire text, the Simplified Lesk algorithm could gain more recall at the expense of precision. However, this markedly increased the time taken to disambiguate the corpus, taking approximately half an hour to process SemCor on the test system. It was also found in section 6.4.2 that there were several ways to improve the performance of the algorithms by changes to parameters used. It was found that algorithms could become significantly more accurate if the algorithm would guess the first sense of WordNet, if no other sense could be assigned. More accuracy could also be gained if an algorithm could be set to consider just the first two senses of WordNet for any given word.

To answer research question three from these results, it does not appear that the accuracy of the algorithms correlates with better real world performance, at least in the area of automatic concept mapping. While there were measureable differences between the accuracy of the different algorithms, these differences appeared to have no effect on the concept maps produced, as demonstrated in section 6.3, and later confirmed in section 6.4.2. In fact, on several occasions, the less accurate Hypernym-Lesk algorithm produced a noticeably better concept map than the more accurate Simplified Lesk algorithm. The reason for this appears to be that a large part of the framework used to score concept maps involves determining if the main concepts were correctly identified. This does not exactly correlate with the accuracy of the algorithms; an algorithm could have 90%

precision and recall, but if the main concepts are not in that 90%, the concept map score would be zero. Of course, more accurate algorithms have a greater chance of correctly identifying the main concepts of a text than a less accurate algorithm.

7.1 Future Work

In regards to future work regarding this research, an obvious area of focus would be determining the effectiveness of different algorithms with respect to various other practical applications. While this research was wholly focussed on automatically generated concept maps, future research could focus on machine translation, or other applications of word sense disambiguation.

Future research could also focus on a greater variety of algorithms. As this research required lexical information such as definitions, synonyms and hypernyms in order to create concept maps, supervised and unsupervised methods were less suitable. Using a larger variety of algorithms could affect the results greatly.

As the algorithms tested did not rely on large amounts of sequential processing, these algorithms should be able to be run effectively on a general purpose graphics processing units (GPGPU). Whereas a modern CPU may run at 2-3 GHz and with 2-4 cores, a GPGPU may run at around 1GHz, but with hundreds or thousands of cores. This difference in architecture enables GPGPUs to have power orders of magnitude more than CPUs, but only if the code run is sufficiently parallelised. As determining the correct sense of a target word is not dependent on knowing the sense of any other word, these algorithms can be highly parallelised.

8. References

Agirre, E., & Edmonds, P. G. (2006). Word sense disambiguation: Algorithms and applications: Springer.

Aman, S., & Szpakowicz, S. (2008). Using Roget's thesaurus for fine-grained emotion recognition.

Ambiguity. (2011). WordNet Search. Princeton University. Retrieved May 22, 2011, from http://wordnetweb.princeton.edu/perl/webwn?s=ambiguity

Banerjee, S., & Pedersen, T. (2010). An adapted Lesk algorithm for word sense disambiguation using WordNet. Computational Linguistics and Intelligent Text Processing, 117-171.

Baseline. (2011). Oxford Dictionaries Online. Oxford University. Retrieved June 6, 2011, from http://oxforddictionaries.com/definition/baseline

Baseline. (2011). WordNet Search. Princeton University. Retrieved June 6, 2011, from http://wordnetweb.princeton.edu/perl/webwn?s=baseline

Basile, P., de Gemmis, M., Lops, P., & Semeraro, G. (2008). Combining knowledge-based methods and supervised learning for effective Italian word sense disambiguation.

Calafate, C., Cano, J.-C., & Manzoni, P. (2009). A Comprehensive Methodology for Concept Map Assessment. Paper presented at the 2009 Ninth IEEE International Conference on Advanced Learning Technologies.

Cañas, A. J., Hill, G., Carff, R., Suri, N., Lott, J., Eskridge, T., . . . Carvajal, R. (2004). CmapTools: A knowledge modeling and sharing environment.

Cañas, A. J., Valerio, A., Lalinde-Pulido, J., Carvalho, M., & Arguedas, M. (2003). Using WordNet for word sense disambiguation to support concept map construction.

Chan, Y., Ng, H., & Chiang, D. (2007). Word sense disambiguation improves statistical machine translation.

Chow, I., & Webster, J. (2010). Integration of Linguistic Resources for Verb Classification: FrameNet Frame, WordNet Verb and Suggested Upper Merged Ontology. Computational Linguistics and Intelligent Text Processing, 1-11.

Dagan, I., Itai, A., & Schwall, U. (1991). Two languages are more informative than one. Paper presented at the Proceedings of the 29th annual meeting on Association for Computational Linguistics, Berkeley, California.

Dahlgren, K. (1988). Naive semantics for natural language understanding (Vol. 58): Springer.

Doctor of the Church. (2010). WordNet Search. Princeton University. Retrieved May 22, 2011, from http://wordnetweb.princeton.edu/perl/webwn?s=Doctor of the Church

Edmonds, P., & Cotton, S. (2001). senseval-2: Overview.

Edmonds, P., & Kilgarriff, A. (2002). Introduction to the special issue on evaluating word sense disambiguation systems. Natural Language Engineering, 8(04), 279-291.

Fellbaum. (1998). WordNet: An Electronic Lexical Database (Language, Speech, and Communication): The MIT Press.

Fellbaum, C. (2010). Harmonizing WordNet and FrameNet. Advances in Natural Language Processing, 2-2.

Gale, W., Church, K., & Yarowsky, D. (1993). A method for disambiguating word senses in a large corpus. Computers and the Humanities, 26(5), 24.

Gale, W., Church, K. W., & Yarowsky, D. (1992). Estimating upper and lower bounds on the performance of word-sense disambiguation programs. Paper presented at the Proceedings of the 30th annual meeting on Association for Computational Linguistics, Newark, Delaware.

Galliers, R.D. (1990). 'Choosing Appropriate Information Systems Research Approaches: A Revised Taxonomy'. Pages 155-73 of: Nissen, H-E., Klein, H.K. and Hirschheim-Stuart, R., (eds), The Information Systems Research Arena of the 90's: Challenges, Perceptions and Alternative Approaches - Proc. IFIP TC8 WG8.2 Conference. Copenhagen, Denmark.

Gelbukh, A., & Torres, S. (2009). Comparing Similarity Measures for Original WSD Lesk Algorithm. 11.

Gregorowicz, A., & Kramer, M. A. (2006). Mining a large-scale term-concept network from Wikipedia Mitre Technical Report.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. Paper presented at the Proceedings of the 14th conference on Computational linguistics - Volume 2, Nantes, France.

Hyponym. (2010). WordNet Search. Princeton University. Retrieved March 12, 2010, from http://wordnetweb.princeton.edu/perl/webwn?s=hyponym

Ide, N., & Veronis, J. (1998). Introduction to the special issue on word sense disambiguation: the state of the art. Comput. Linguist., 24(1), 2-40.

John, M., & Enss, R. (2008). An Investigation of Word Sense Disambiguation for Improving Lexical Chaining.

Kilgarri, A. (1998). Senseval: An exercise in evaluating word sense disambiguation programs.

Kilgarriff, A., & Rosenzweig, J. (2000). English SENSEVAL: Report and results. LREC, Athens, 265–283.

Laparra, E., & Rigau, G. (2009). Integrating WordNet and FrameNet using a knowledge-based Word Sense Disambiguation algorithm.

Lave, C., & March, J. (1993). An introduction to models in the social sciences: Univ Pr of Amer.

Law, A., & Kelton, W. (2000). Simulation modeling and analysis: McGraw-Hill, Boston.

Leacock, C., Towell, G., & Voorhees, E. (1993). Corpus-based statistical sense resolution. Paper presented at the Proceedings of the workshop on Human Language Technology, Princeton, New Jersey.

Legrand, S., & Pulido, J. (2004, September 24). A Hybrid Approach to Word Sense Disambiguation: Neural Clustering with Class Labeling. Paper presented at the 15th European Conference on Machine Learning (ECML), Pisa, Italy.

Lesk, M. (1986). Automatic Sense Disambiguation using Machine Readable Dictionaries: How to Tell a Pine Cone From an Ice Cream Cone. Paper presented at the Proceedings of the 5th Annual International Conference on Systems Documenation, Toronto, Ontario, Canada.

Lonneker-Rodman, B., & Baker, C. (2009). The FrameNet model and its applications. Natural Language Engineering, 15(03), 415-453.

McCarthy, D. (2006). Relating WordNet senses for word sense disambiguation. Making Sense of Sense: Bringing Psycholinguistics and Computational Linguistics Together, 17.

McClure, J. R., & Bell, P. E. (1990). Effects of an Environmental Education-Related STS Approach Instruction on Cognitive Structures of Preservice Science Teachers.

McClure, J. R., Sonak, B., & Suen, H. K. (1999). Concept map assessment of classroom learning: Reliability, validity, and logistical practicality. Sci Teach, 36, 475-492.

Medelyan, O., Milne, D., Legg, C., & Witten, I. H. (2009). Mining meaning from Wikipedia. Int. J. Hum.-Comput. Stud., 67(9), 716-754. doi: http://dx.doi.org/10.1016/j.ijhcs.2009.05.004

Mihalcea, R. (2007). Using wikipedia for automatic word sense disambiguation.

Mihalcea, R., & Moldovan, D. I. (1999). A method for word sense disambiguation of unrestricted text.

Mihalcea, R., & Moldovan, D. I. (2000). A Highly Accurate Bootstrapping Algorithm for Word Sense Disambiguation. International Journal on Artificial Intelligence Tools, 10(1-2), 16.

Mihalcea, R., & Pedersen, T. (2005). Advances in Word Sense Disambiguation. Paper presented at the The Twentieth National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania.

Mooney, R. J. (1996). Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. Paper presented at the Empirical Methods in Natural Language Processing, Philadelphia, PA.

Navigli, Litkowski, & Hargraves. (2007). Semeval-2007 task 07: Coarse-grained english all-words task.

Navigli, R. (2008). A structural approach to the automatic adjudication of word sense disagreements. Natural Language Engineering, 14(4), 547.

Ng, H. T., & Zelle, J. (1997). Corpus-based approaches to semantic interpretation in natural language processing. AI Magazine, 18(4), 45-64.

Novak, J. D., & Gowin, D. B. (1984). Learning how to learn: Cambridge Univ Pr.

O'Madadhain, J., Fisher, D., White, S., & Boey, Y. (2003). The jung (java universal network/graph) framework. University of California, Irvine, California.

Palmer, M., Dang, H., & Fellbaum, C. (2007). Making fine-grained and coarse-grained sense distinctions, both manually and automatically. Natural Language Engineering, 13(02), 137-163.

Pedersen, T. (1998). Learning Probabilistic Models of Word Sense Disambiguation. Southern Methodist University.

Pedersen, T. (2010). A baseline methodology for word sense disambiguation. Computational Linguistics and Intelligent Text Processing, 29-41.

Pradhan, S., Loper, E., Dligach, D., & Palmer, M. (2007). SemEval-2007 task 17: English lexical sample, SRL and all words.

Preiss, J., Dehdari, J., King, J., & Mehay, D. (2009). Refining the most frequent sense baseline. SEW-2009 Semantic Evaluations: Recent Achievements and Future Directions, 10.

Sánchez-de-Madariaga, R., & Fernández-del-Castillo, J. R. (2008). The Bootstrapping of the Yarowsky Algorithm in Real Corpora. Information Processing and Management, 45(2009), 14.

Schutze, H. (1998). Automatic word sense discrimination. Comput. Linguist., 24(1), 97-123.

Small, S. L. (1981). Word expert parsing: A theory of distributed word-based natural language understanding. Dissertation Abstracts International Part B: Science and Engineering[DISS. ABST. INT. PT. B- SCI. & ENG.], 42(2), 1981.

Snyder, B., & Palmer, M. (2004). The English all-words task.

Specia, L. (2005). A Hybrid Model for Word Sense Disambiguation in English-Portuguese Machine Translation.

Tolstoy, L. (1949). War and peace: Winston.

Vasilescu, F., Langlais, P., & Lapalme, G. (2004). Evaluating variants of the Lesk approach for disambiguating words.

Wikipedia. (2011). Wikipedia:Size of Wikipedia Wikipedia, the free encyclopedia (Vol. 2010).

Wilks, Y., Fass, D., Guo, C.-m., Mcdonald, J. E., Plate, T., & Slator, B. M. (1988). Machine tractable dictionaries as tools and resources for natural language processing. Paper presented at the Proceedings of the 12th conference on Computational linguistics - Volume 2, Budapest, Hungry.

Yarowsky, D. (1992). Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. Paper presented at the Proceedings of the 14th conference on Computational linguistics - Volume 2, Nantes, France.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. Paper presented at the Proceedings of the 33rd annual meeting on Association for Computational Linguistics, Cambridge, Massachusetts.

Appendix A **Definitions of terms or operational definitions**

Corpus: (plural corpora) A body of text, or collection of documents.

Discourse: Document

Hyponym Relationship: (antonym Hypernym) A word that is more specific than a given word. Such relationships are also known as 'hierarchical relationships' (WordNet, 2010). For example, *dog* is a hyponym of *animal*.

Polysemous Words: Words with multiple senses

Precision: The percentage of correctly disambiguated words, out of all the words disambiguated. (Rada Mihalcea & Moldovan, 2000)

Recall: The percentage of correctly disambiguated words, out of all the words in the discourse. (Rada Mihalcea & Moldovan, 2000)

Resource-Based Approach: An outside dictionary, thesaurus or other resource is used in order to help disambiguate words

Supervised Approach: Methods that use a manually created set of annotated corpora to train an algorithm. An algorithm will typically identify patterns and rules concerning word senses in the pre-annotated corpora, which can then be applied to new corpora.

Unsupervised Approach: Also known as word sense discrimination. "The task of dividing the usages of a word into different meanings, without regard to any particular existing sense inventory" (Mihalcea & Pedersen, 2005)

Word Sense: A meaning of a word in a given context

Word Sense Disambiguation: The task of assigning sense labels to occurrences of an ambiguous word. (Schutze, 1998)

Appendix B Contents of SemCor Summary

Document	Document Type
br-a01	Newspaper/periodical article
br-a02	Newspaper/periodical article
br-a11	Newspaper/periodical article
br-a12	Newspaper/periodical article
br-a13	Newspaper/periodical article
br-a14	Newspaper/periodical article
br-a15	Newspaper/periodical article
br-b13	Newspaper/periodical editorial
br-b20	Newspaper/periodical editorial
br-c01	Newspaper/periodical review
br-c02	Newspaper/periodical review
br-c04	Newspaper/periodical review
br-d01	Essay on religion
br-d02	Essay on religion
br-d03	Essay on religion
br-d04	Essay on religion
br-e01	Article on hobbies/recreation
br-e02	Article on hobbies/recreation
br-e04	Article on hobbies/recreation
br-e21	Article on hobbies/recreation
br-e24	Article on hobbies/recreation
br-e29	Article on hobbies/recreation
br-f03	Article on history/folklore
br-f10	Article on history/folklore
br-f19	Article on history/folklore
br-f43	Article on history/folklore
br-g01	Social commentary article
br-g11	Social commentary article
br-g15	Social commentary article
br-h01	Government report
br-j01	Academic journal article
br-j02	Academic journal article
br-j03	Academic journal article
br-j04	Academic journal article
br-j05	Academic journal article
br-j06	Academic journal article
br-j07	Academic journal article
br-j08	Academic journal article
br-j09	Academic journal article
br-j10	Academic journal article
br-j11	Academic journal article
br-j12	Academic journal article
br-j13	Academic journal article
br-j14	Academic journal article
br-j15	Academic journal article
br-j16	Academic journal article
br-j17	Academic journal article
br-j18	Academic journal article
br-j19	Academic journal article
br-j20	Academic journal article
br-j22	Academic journal article
br-j23	Academic journal article
br-j37	Academic journal article
br-j52	Academic journal article
br-j53	Academic journal article
br-j54	Academic journal article
br-j55	Academic journal article
br-j56	Academic journal article
br-j57	Academic journal article
br-j58	Academic journal article
br-j59	Academic journal article
br-j60	Academic journal article
br-j70	Academic journal article
br-k01	Fiction
br-k02	Fiction
br-k03	Fiction
br-k04	Fiction
br-k05	Fiction
br-k06	Fiction
br-k07	Fiction
br-k08	Fiction
br-k09	Fiction
br-k10	Fiction

br-k11	Fiction
br-k12	Fiction
br-k13	Fiction
br-k14	Fiction
br-k15	Fiction
br-k16	Fiction
br-k17	Fiction
br-k18	Fiction
br-k19	Fiction
br-k20	Fiction
br-k21	Fiction
br-k22	Fiction
br-k23	Fiction
br-k24	Fiction
br-k25	Fiction
br-k26	Fiction
br-k27	Fiction
br-k28	Fiction
br-k29	Fiction
br-l11	Fiction
br-l12	Fiction
br-m01	Fiction
br-m02	Fiction
br-n05	Fiction
br-p01	Fiction
br-r05	Humour
br-r06	Humour
br-r07	Humour
br-r08	Humour
br-r09	Humour

Figure 22: Breakdown of the types of documents in SemCor 2.1 (John & Enss, 2008)

Appendix C Pseudocode of tested Algorithms

```
for every word w[i] in the phrase
     let BEST_SCORE = 0
     let BEST_SENSE = null
     for every sense sense[j] of w[i]
          let SCORE = 0
          for every other word w[k] in the phrase, k != i
               for every sense sense[l] of w[k]
                    SCORE = SCORE + number of words that occur
                                    in the gloss of both
                                    sense[j] and sense[l]
               end for
          end for
          if SCORE > BEST_SCORE
               BEST_SCORE = SCORE
               BEST_SENSE = w[i]
          end if
     end for
     if BEST_SCORE > 0
          output BEST_SENSE
     else
          output "Could not disambiguate w[i]"
     end if
end for
```

Figure 23: Pseudocode of the Lesk Algorithm

```
for every word w[i] in the phrase
     let BEST_SCORE = 0
     let BEST_SENSE = null
     for every sense sense[j] of w[i]
          let SCORE = 0
          for every other word w[k] in the phrase, k != i
               SCORE = SCORE + number of words that occur
                               in the gloss of both
                               sense[j] and phrase
          end for
          if SCORE > BEST_SCORE
               BEST_SCORE = SCORE
               BEST_SENSE = w[i]
          end if
     end for
     if BEST_SCORE > 0
          output BEST_SENSE
     else
          output "Could not disambiguate w[i]"
     end if
end for
```

Figure 24: Pseudocode of the Simplified Lesk Algorithm

```
for every word w[i] in the phrase
     let BEST_SCORE = 0
     let BEST_SENSE = null
     for every sense sense[j] of w[i]
          let SCORE = 0
          for every hypernym hypernym[k] of every sense[j] of
                                   w[i] in the phrase, k != i
               for every word w[l] in the phrase
                    for every sense sense[m] of w[l]
                         for every hypernym hypernym[n] of
                                             every sense[m]
                              SCORE = SCORE + number of
                                      hypernyms in both
                                      hypernym[k] and
                                      hypernym[n]
          end for
          if SCORE > BEST_SCORE
               BEST_SCORE = SCORE
               BEST_SENSE = w[i]
          end if
     end for
     if BEST_SCORE > 0
          output BEST_SENSE
     else
          output "Could not disambiguate w[i]"
     end if
end for
```

Figure 25: Pseudocode of the Hypernym Lesk Algorithm

```
for every word w[i] in the phrase
     let BEST_SCORE = 0
     let BEST_SENSE = null
     for every sense sense[j] of w[i]
          let SCORE = 0
          for every synonym synonym[k] of every sense[j] of
                                   w[i] in the phrase, k != i
               for every word w[l] in the phrase
                    for every sense sense[m] of w[l]
                         for every synonym synonym[n] of
                                             every sense[m]
                              SCORE = SCORE + number of
                                      synonyms in both
                                      synonym[k] and
                                      synonym[n]
          end for
          if SCORE > BEST_SCORE
               BEST_SCORE = SCORE
               BEST_SENSE = w[i]
          end if
     end for
     if BEST_SCORE > 0
          output BEST_SENSE
     else
          output "Could not disambiguate w[i]"
     end if
end for
```

Figure 26: Pseudocode of the Synonym Lesk Algorithm

```
for every word w[i] in the phrase
     let BEST_SCORE = 0
     let SCORE = 0
     if senses found for word
          BEST_SENSE = first sense found in dictionary
          BEST_SCORE = 1
     end if
     if BEST_SCORE > 0
          output BEST_SENSE
     else
          output "Could not disambiguate w[i]"
     end if
end for
```

Figure 27: Pseudocode of the original (most frequent sense) baseline

Appendix D Results of Algorithms on Individual Corpora

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	34	26	79
Simplified Lesk	34	19	83
Hypernym Lesk	28	14	79
Synonym Lesk	34	27	83
Baseline	44	35	83
Random	31	24	71

Figure 28: Precision, recall and concept map score obtained on the BR-D03 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	37	27	60
Simplified Lesk	47	22	6
Hypernym Lesk	29	12	34
Synonym Lesk	36	26	66
Baseline	54	39	66
Simplified Lesk with Guessing	50	36	66
Random	30	22	60

Figure 29: Precision, recall and concept map score obtained on the BR-G28 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	34	26	35
Simplified Lesk	45	20	21
Hypernym Lesk	30	16	35
Synonym Lesk	32	25	35
Baseline	50	38	38

Figure 30: Precision, recall and concept map score obtained on the BR-J11 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	25	18	0
Simplified Lesk	36	19	13
Hypernym Lesk	22	12	20
Synonym Lesk	29	21	13
Baseline	42	30	6

Figure 31: Precision, recall and concept map score obtained on the BR-K15 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	32	25	35
Simplified Lesk	43	23	50
Hypernym Lesk	27	14	49
Synonym Lesk	33	26	49
Baseline	49	39	50
Random	29	23	27

Figure 32: Precision, recall and concept map score obtained on the BR-E04 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	32	24	37
Simplified Lesk	45	22	37
Hypernym Lesk	28	14	37
Synonym Lesk	35	27	53
Baseline	49	37	53

Figure 33: Precision, recall and concept map score obtained on the BR-B20 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	44	34	53
Simplified Lesk	42	21	25
Hypernym Lesk	28	15	57
Synonym Lesk	34	27	25
Baseline	60	47	57
Simplified Lesk with Guessing	51	40	57

Figure 34: Precision, recall and concept map score obtained on the BR-J01 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	31	25	15
Simplified Lesk	41	22	15
Hypernym Lesk	32	18	67
Synonym Lesk	36	29	24
Baseline	51	41	67
Lesk with Guessing	31	25	15
Simplified Lesk with Guessing	47	37	67

Figure 35: Precision, recall and concept map score obtained on the BR-R05 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	31	24	53
Simplified Lesk	50	28	53
Hypernym Lesk	30	17	53
Synonym Lesk	35	27	43
Baseline	56	43	62

Figure 36: Precision, recall and concept map score obtained on the BR-J29 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	35	27	27
Simplified Lesk	55	25	24
Hypernym Lesk	16	10	5
Synonym Lesk	30	24	16
Baseline	56	44	46

Figure 37: Precision, recall and concept map score obtained on the BR-J13 corpus

Algorithm	Precision (%)	Recall (%)	Concept Map Score (%)
Lesk	34	26	36
Simplified Lesk	41	19	19
Hypernym Lesk	32	18	39
Synonym Lesk	34	26	21
Baseline	51	39	42
Random	31	23	0

Figure 38: Precision, recall and concept map score obtained on the BR-F10 corpus

Appendix E Results of Processing Time Over a Varying Word Window

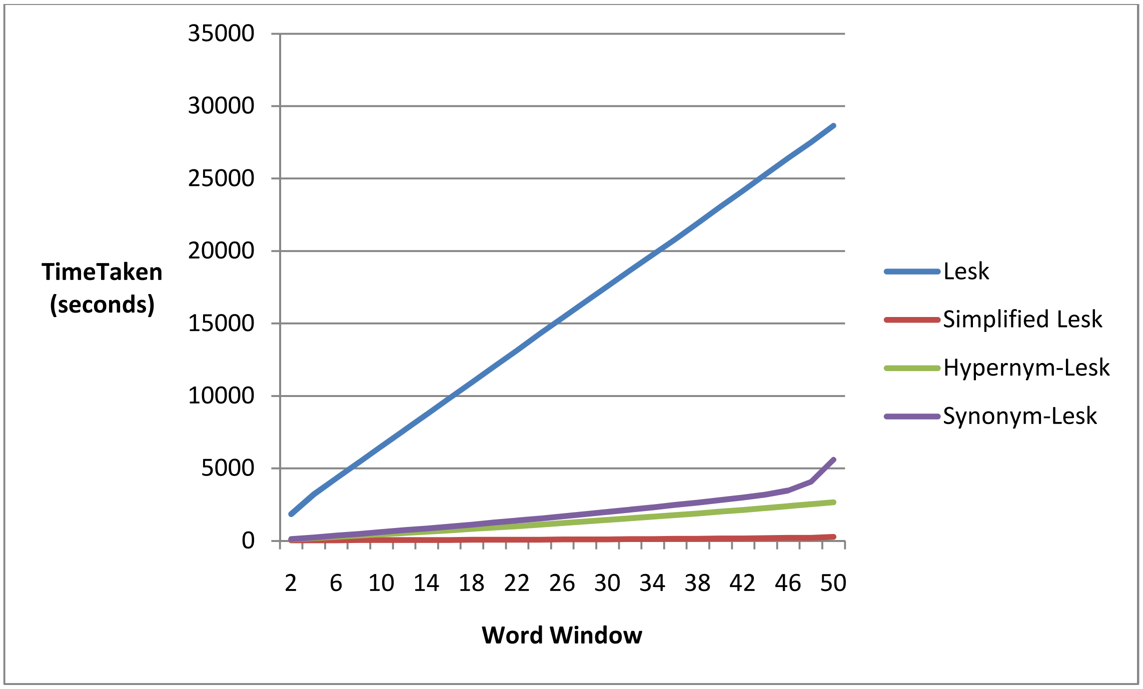


Figure 39: Comparison of the speed of four algorithms over a varying word window

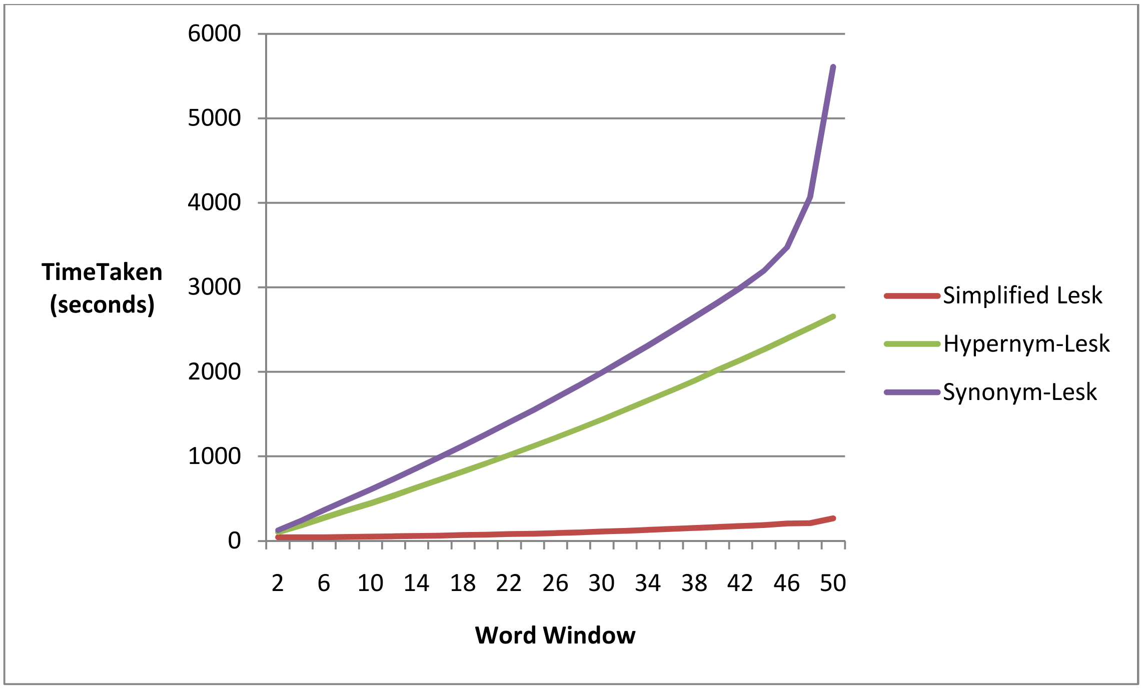


Figure 40: Comparison of the speed of three algorithms over a varying word window

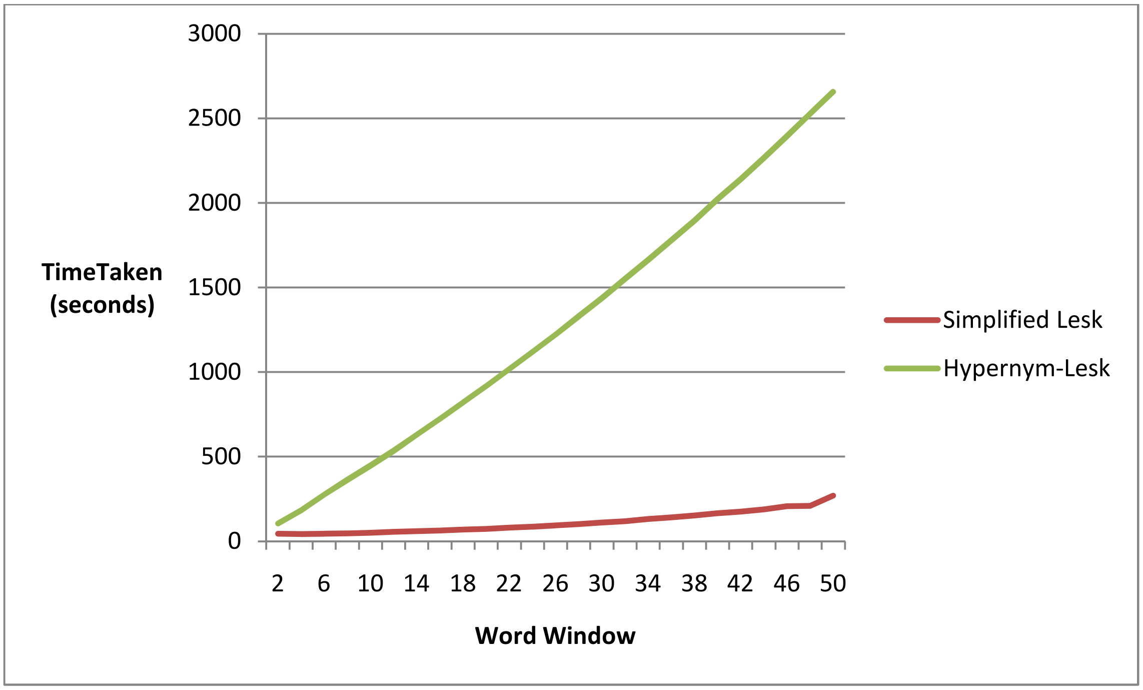


Figure 41: Comparison of the speed of two algorithms over a varying word window

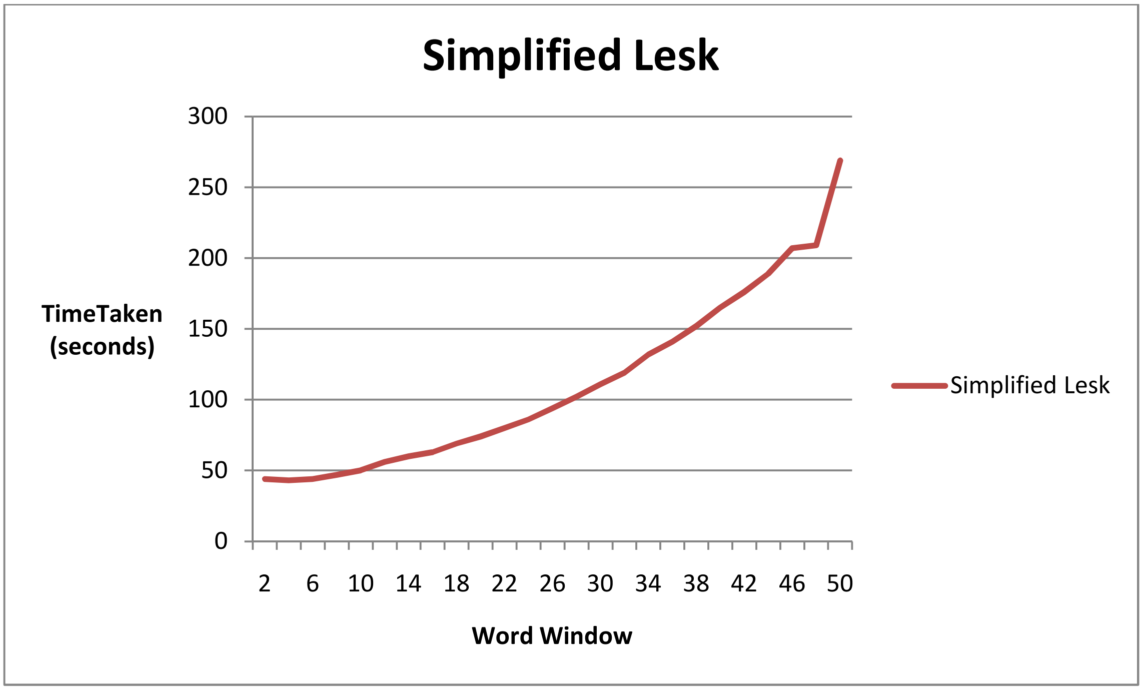


Figure 42: Comparison of the speed of the Simplified Lesk algorithm over a varying word window