Using WordNet and a Short-Term Memory model for Contextual Disambiguation

William T. F. Strachan

Wordcount - 9191
(Wordcount produced using wc - w)
Supervisor - Dr Dimitar Kazakov
Department of Computer Science
University of York
20 November 2016

Abstract

Semantic Analysis is the process of giving meaning to the tokens of a sentence. This process has a multitude of applications from speech recognition to unsepervised knowledge aquisition. Current solutions do not provide enough accuracy to be relied upon fully for these puposes, making it a heated area o research. It is known that humans are very good at semantic analysis, and multiple psychologial models describing this exist, such as the Working Memory Model. The focus of this report is a computational model, derived from these theories, with the intention of providing a more reliable solution to the problem of semantic analysis. A subset of the working memory model is used, in conjunction with hyponymy base activation and disambiguation. The model developed provides an acccuracy greater than some approaches, though, without further development, fails to reliably identify word meanings across a large input corpus. The results implicate the need for the use of a greater number of semantic links in order to correctly derive meaning.

Contents

1 Introduction

1.1 Context

Natural language is the language spoken and written by humans. It is very powerful, though its key downfall is its ambiguity. The study of natural language processing aims to allow a computer to understand natural language, and formulate a relevant response based upon its input. Within this, the problem of input text analysis has traditionally been broken down into smaller sub-problems [1]:

- **Text Preprocessing**
  Before any analysis can take place, the input raw text must be converted into a usable format [1].
- **Lexical Analysis**
  One word can have multiple forms, for example judge (the lemma) has the forms judge, judges, judging, judged (morphological variants). The job of Lexical Analysis is to replace all morphological variants of a word, with their corresponding lemma, a process known as stemming [1].
- **Syntactic Parsing**
  When deriving meaning from sentences, the grammatical structure can provide important insight. The Syntactic parsing technique extracts this information using Part-of-Speech tagging (detecting syntactic role) and Chunking (detecting noun and verb phrases) [1].
- **Semantic Analysis**
  The derivation of meaning from the sentences tokens [1].

Semantic Analysis is a problem which continues to draw attention from researchers, due to the lack of a truly capable solution [1]. Most would agree that, though a good computational solution is not available, the human brain performs impressively well when confronted with the same problem. Given the example:

The party led government

it is known by the reader that party refers to a political party, due to the fact that it is leading government. The meaning of the word is taken from its surrounding context.

It can be seen that the study of the brains processes of semantic analysis, and their application to a computational model, could provide a satisfying solution. A great deal of research is directed at studying these process, largely influenced by the working memory model proposed by Baddeley [2]. This memory model describes a system of structures, which work in conjunction with one another to form a representation of the brains input.

The Working Memory model has many more applications than the one described in the coming report, requiring more structures than are relevant to the process of semantic analysis. For the purposes of this application, we will restrict our focus to three of these memories, short-term, long-term, and the Episodic buffer [2]; each of which is described in greater detail in the Literature review.

The following report will discuss and define a potential computational model of the psychological theories surrounding the process of Semantic Analysis. As part of the project, a system will be built, and tested against an input corpus. Due to time, and performance constraints, a level of abstraction will be required. I is the purposes of this testing to decide whether the level of abstraction used is provides a satisfactory performance.

1.2 Ethics

The project conducted as part of this report requires no outside participants, nor does it directly contribute to any research or action that could lead to harm. The purposes of this project are purely research based, with the intention of leading to further research in the same field. It is assumed that previous research used by this project was conducted following generally accepted ethical practices.

2 Literature Review

2.1 Psycholinguistics

Language understanding is a problem which, it can be stated, is solved by the human brain. From this satatement, it can be derived that a computational solution could be effectively built around knowledge of the processes at work in the brain. The process of language comprehension can be described using the working memory model [2].

According to Baddely et al. [2] there exist multiple, special purpose, memory structures within two main categories, the Short-term Store and the Long-term Store.

2.1.1 Long-term Store

The long-term store (LTS) contains semi-permanent information. Within the LTS, there exist Explicit and Implicit memory structures. The contents of the Implicit memory describe skills and methods of doing things, whereas the Explicit memory contains factual information [2]. When considering these structures, it can be seen that Explicit memory is of greater interest in the context of NLP.

Within the Explicit memory exists knowledge of semantics [2]. The information held here not only defines concepts (meanings of word forms), but also their attributes and rules of use. In 1966, M. Quillain proposed a model of Semantic Memory [3]. The model consists of a graph of nodes, each representing a concept, connected by edges of differing types, each representing a different syntactic feature (for example, hypernym).

2.1.2 Short-term Store

The short-term store (STS) is a structure of limited capacity, used to store items for periods usually of no more than a few seconds [2]. In 1955, G. Miller, based upon previous experimental results, concluded that the size of the STS existed in the realm of 7 ± 2 items of information [4].

In 1971, R. Atkinson and R. Shiffrin proposed a model of the STS [5]. In this model, the STS can both send information to, and draw information from the LTS. Inputs from the sensory registers (memory structures holding information relating to inputs from senses) are also sent to the STS. Atkinson and Shiffrin proposed that, over time, the activation of items in the STS decreased; they went on to theorise that items could be lost from the STS, only when a new, more highly activated item could take its place. To counter this loss of activation, the authors discussed the control process, rehearsal. This process makes use of repetition to increase the activation of items in memory, decreasing their chance of loss.

2.1.3 Disambiguation Models

In some cases, when assigning meaning to words, ambiguity can arise. Some words have multiple concepts, for example, bank can refer to a building, or a sloped surface alongside a body of water. In such cases, the brain uses some process to select the correct concept. One such model of this disambiguation is the Multiple-access model [6].

According to the multiple-access model, when presented with an ambiguious word, initially all corresponding concepts are activated [7]. The most appropriate concept is then chosen using context and frequency.

Previously, we established that the process of disambiguation begins with the activation of all possible word concepts. The context-sensitive model deals with the use of context and frequency in the selection of the most appropriate of these [6]. In cases where the context is strong, i.e. the correct concept can be chosen using it's surrounding context, the context is primarily relied upon for disambiguation. In the opposite case, i.e. when context gives little indication of which is the correct concept, the most frequenly used concept is used, assuming it fits with the available context.

2.2 Wordnet

In 1990, it was noted by G. Miller et al. that current attemps to organise the english lexicon, i.e. conventional dictionaries, offered few benefits when used in conjunction with computers [8]. Wordnet was an effort to produce a dictonary, containing more information than a conventional dictionary, that could be useful for computational applications.

Central to the design of wordnet, is the idea of synsets [8]. The authors began using the assumption that all concepts can be uniquely defined by their set of synonyms (words which share like meaning). In most cases, this assumption holds true, though, in cases where more detail is required, a "gloss" was added [8].

Wordnet builds upon models of the semantic memory, such as that discussed in the Long-term store section [8]. The overall structure relies on four main semantic relations:

- Synonymy
  - If two words are to be called synonyms, they must share at least one like meaning.
- Atonymy
  - Conceptually, Atonymy can be seen as the opposite of Synonymy. Atonymy is difficult to define, as not all words which share opposite meaning can be called atonyms, for example, {up, down} is an atonym pair, but {up, fall} is not.
- Hyponymy
  - If we consider a a synset to be a object-oriented class, its hypernym can be considered its parent class, for example, birch is a type of tree.
- Meronymy
  - Meronymy is relationship between two synsets where one is a part of another, for example, a goat has horns, therefore horn is a meronym of goat.

It is common knowledge that words can fall into one of a number of categories, nouns, adjectives, verbs and adverbs. G. Miller et al. note that, due to the differences in the relations between words in these categories, each type has differs in the structure they produce and are therefore held in different files [8]. The proceeding subsections will go into each of thesecategories in more detail.

2.2.1 Nouns

G. Miller et al. note that a noun can be defined using only its immediate hypernym, and how it differs from its hypernyms other hyponyms [9]. From this, it can be seen that hyponymy is perhaps the most important relation in the organisation of nouns. For this reason, nouns form a hierachical structure in wordnet.

Wordnet's designers stated the assumption that all nouns can be contained in a single hierachial structure [9]. The issue with having a single word, of which all other words are hyponyms, is that this hypernym is relatively meaningless. It was instead decide to divide all words into 25 separate files, each containing a hierachical tree beginning with one of the following synsets [9]:

{act, action, activity}

{animal, fauna}

{artifact}

{attribute, property}

{body, corpus}

{cognition, knowledge}

{communication}

{event, happening}

{feeling, emotion}

{food}

{group, collection}

{location, place}

{motive}

{natural object}

{natural phenomenon}

{person, human being}

{plant, flora}

{possession}

{process}

{quantity, amount}

{relation}

{shape}

{state, condition}

{substance}

{time}

Other than synonymy, nouns have three other important features [9]:

- Attributes
  - The attributes of a noun consist of adjectives which distinguish it from other hyponyms of its hypernym, for example {huge, green, fluffy}.
- Parts
  - The parts of a noun consist of its meronyms, described previously.

- Functions
  - The functions of a noun consist of verbs which are associated with its actions, for example chair has the functions {sit, rest}.

2.2.2 Adjectives

Adjectives in can be divided into four distinct groups, each implying a different structure of semantic links [10]:

- Descriptive Adjectives
  - As stated in the previous section, nouns have attributes. Descriptive adjectives act as modifiers for these attributes: for example, a building has a height, by saying "tall building", the height attribute is given a value [10]. Atonymy, defined previously, is considered by wordnet's designers to be the most important relation between descriptive adjectives. Unfortuantely, not all adjectives have atonyms, leading to the designer's addition of an indirect antonym" semantic link, between synonyms of a word and its atonym [10]. These semantic links give rise to a structure made up of pairs, linked to one another by their synonyms.
- Reference-Modifying Adjectives
  - Reference-modifying adjectives have an adverb form which can be used to convey the same meaning [10]. For example, the noun-phrase "the former manager", can be modified to become "the man who was formerly a manager", without diverging from its original meaning. There exist relatively few examples of this category, so no overarching structure emerges, that being said, in some cases, the atonym relation does occur [10].
- Colour Adjectives
  - As their name suggests, colour adjctives concern the value of the colour attribute. This definition implies that these words should, in fact, fit into the descriptive adjective category. Their separation is given by colour adjectives lack of true atonym (excluding modifiers such as "light" and "dark") [10]. The lack of clear semantic relations between these words poses a problem for their organisation, leading wordnet's designers to link them using their definitions, i.e. using hue, lightness and saturation.
- Relational Adjectives
  - In the phrase "maternal instinct", it can be seen that the adjective is derived from a noun, in this case "mother"; this is the defining

feature of relational adjectives [10]. In wordnet, relational adjectives are linked to their noun form, meaning they don't posess their own structure, instead falling into that described in the Noun section.

2.2.3 Verbs

In C. Fellbaums 1990 paper, "English Verbs as a Semantic Net", she discussed the lack of true synonymy across the verb category [11]. This is an issue for Wordnet's designers, with their reliance on synsets. The author goes on to describe the solution, periphrases, the use of verb phrases to give more meaning to a simple verb. In the paper, the example synset, {swim, travel through water} was given.

The relationships between verbs follow a hierachy shown in Figure 1, with each type elaborated upon in the following paragraphs.

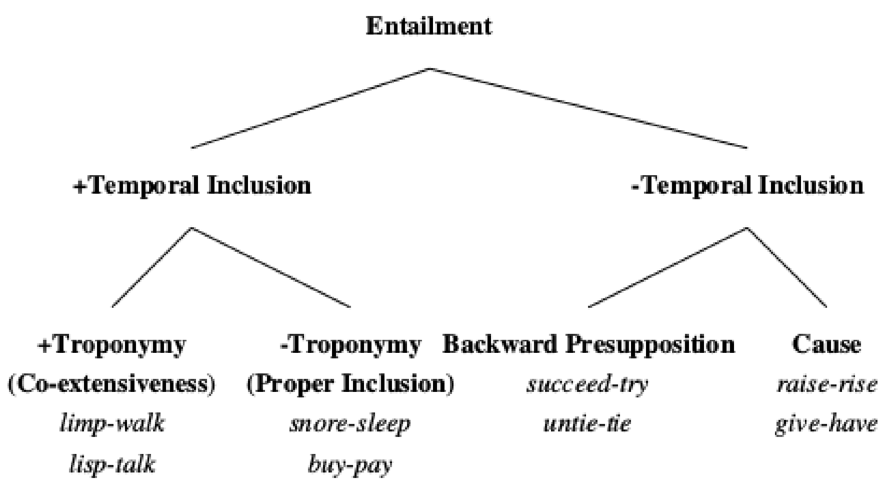

Figure 1: HIERACHY OF SEMANTIC RELATIONS BETWEEN VERBS [11, p. 15]

Entailment is a relation, simlar in nature to hyponymy. A verb (a) is said to entail another (b) if, when b is substituted for a in a sentace, the truth value of the sentance remains the same [11]. For example, "run" entails "move", "she ran" can also be described by "she moved". If a entails b, and b entials a, a and b are synonyms.

Temporal Inclusion is a form of entailment where one verb (a) is temporally included in another (b) if, b can only occur during the same time period as a [11]. For example, "swallow" is temporally included by "consume". If a can also occur without b, the entailment can be called **Proper Inclusion**.

Troponomy is a type of entailment where a verb (a) can be said to be a way of doing another verb (b) [11]. For example, "to nap" is a troponym of "to sleep".

Backward Presupposition is a relation which occurs when one verb (a) is a precondition of a verb (b) [11]. For example, one must "play" before they can "win".

Cause is a relation where one verb (a) is considered the causative verb, and another (b) is considered the resultative verb, i.e. a causes b to occur [11]. For example, "to teach" is related to "to learn".

The entailment relation leads to a similar structure to that of nouns, a hierachical tree. This structure differs though in its relative lack of depth, caused by the increased average number of concepts per word, when compared to nouns [11].

2.3 Previous Work

2.3.1 Latent Semantic Analysis

In 1997, T. Laundaur and S. Dumais proposed a purely statistical method of semantic analysis, called Latent Semantic Analysis (LSA) [12]. The model can be visualised as a set of nodes, each representing a word, existing in semntic space. Words placed close together can be said to have similar meaning, and when close enough together, can be called synonyms.

When training the model, words which exist in the same context (i.e sentence, paragraph) are linked by a distance derived from their distance in the text. This distance is then adjsted according to similar occurences in more training data. Given enough training data, the distances between each node can be used to give the most likely synonyms of a word, given the context [12].

The authors found that, when presented with a synonym test, the model's best performance gave 64.4% correct answers , which, when compared to US undergraduates for whom English is not their first language (scoring 64.5% on average), could be considered to be reasonably effective [12]. This model only considers likelyhood, which, as discussed in the Disambiguation Models section, is only one of the two methods the brain is theorised to use for this process. By using a similar method in conjunction with a context focused model, a more accurate and useful model could be produced.

2.3.2 Extended TF-IDF

TF-IDF is an algorithm which, when given a document, outputs its most important words, often used to organise a set of documents into categories. This is done by calculating the importance of each word in the document [13], using the formula:

$$t_i = f_t * (log_2(n) - log_2(f_d) + 1)$$

Where t_i is the importance of the word, f_t is the frequency of the word within the document, n is the number of documents in the corpura, and f_d is the frequency of the word across all documents in the corpura [14]. The words with the highest importance are then outputted.

In 2004, J. Sedding and D. Kazakov proposed an extension to the TF-IDF algorithm, using Wordnet. The aim of the paper was to include extra information provided by wordnet (synsets and hypernyms) in conjuncton with PoS tagging, to provide an output more suited to the categorisation of documents [14]. The authors ran multiple experiments, in order to compare different usage of additional information. unfortunately, it was found that their method was less effective than TF-IDF alone. As the authors stated, this is likely due to the lack of disambiguation, i.e. all synsets were used for each word, adding noise.

2.3.3 Neural Networks

In recent years the use of Neural Networks, more specifically Deep Neural Networks, in the solving of computational problems has dramatically increased. More recently, this technique has been applied to the problem of natural language processing. As discussed in the Natural Language Processing section, it was shown that assigning varyng labels to words and phrases (i.e. categorisation) made up a large proportion of the processes, a job neural networks are wellsuited to, and often associated with [15].

The first process requrired in the use of neural networks, is the conversion of raw text into a vector-base representation, giving a sentance as a set of vectors. The sentence is then passed into the neural network [15]. The network proposed by R.Collobert and J. Weston in 2008 contained multiple layers, each serving its own purpose [15]. Initially the network identifies word features, before using this data to calculate the probability of each synset of each word.

It was found by the authors that their model provided good results, though, in order to get these results, a large amount of time was deicated to training [15].

2.3.4 Previous use of Wordnet and Short-term Memory for Disabiguation

In 2007, a University of York student, M. Burke, produced a project with a simlar aim to the one this report describes. This project builds upon the work and findings of my predecessor. The model developed made use of memory structures based upon those discussed in the Psycholinguistics section, i.e. Short-term (STM) and Semantic memories [16]. The Short-term memory, was a list, containing synsets, each with its own activation, and Wordnet was used as the Semantic memory, once again each synset has its own activation.

As words are encountered, their activation, and the activation of their hypernyms is increased. The activation increase of the initial synset was found through experimentation, though the activation increase of hypernyms differed according to the below equation [16].

$$H = \sum_{i=1}^{N} S_i \times A$$

Where A is the attenuation, found through experimentation, S_i is the activation of the hyponyms. This model was used to prevent more general synsets dominating the short-term memory, whilst boosting hypernyms of synsets more if a similar synset also occurs in close proximity [16].

The author noted that, highly activated synsets could remain in the short-term memory indefinitely. To counter this issue, and to remain in line with the memory models discussed in the Short-term store section, the process of forgetting was added to the model, by multiplying the activation value by a number found by experimentation [16]. Forgetting occurs over time, decreasing the activations of items in the Short-term memory and the Semantic memory, the latter by a greater degree.

In the proposed model, the corpus is processed sentence by sentence. Two methods of disambiguation were proposed, with both cases beginning by removing non-useful words, and converting all words into their base forms (e.g. "flies" ⇒ "fly"), each model differs in its use of the contents of the short term memory.

Hypernym-first - The hypernyms of all words are activated, with the synsets with the highest activation bieng used to disambguate each word [16].

STM-first - The contents of the Short term memory's hyponyms are searched until one of the synsets present in the sentence are found [16].

As mentioned previously, the values of some variables were found using experimentation. In these experiments, four variables were altered [16]:

- Disambiguation Method

  The author found that STM-first was marginally better (1% difference)

- Short-term memory size

  It was found that a STM size of 5 was optimal, with a 67% accuracy.

- The Attenuation value

  The author found that an attenuation value of between 0.7 and 0.9 gave the best result, with an accuracy of 67%.

- Forgetfulness

  It was found that a small amount of forgetfulness (multiply activation by 0.95) in conjunction with a small difference in forgetfulness between Semantic and Short-term memories (multiplied by 1.05), produced the best result [16].

Unfortunately, the model failed to produce the correct synset significantly more accurately than the author's baseline (selecting the most common synset) [16]. As suggested in the report, this may be due to semantic links, which occur in the brain, not being available or utilsed by the model. Though, it may also be the case that the mathematical models used were inaccurate, with less linear functions being required.

3 Problem Analysis

3.1 Problem Definition

Semantic analysis is the process of extracting meaning from a sentence's tokens. Each individual token in a sentence has potentially many possible meanings; an effective tool should select the correct meaning, based upon each word's surrounding context. Throughout the previous work section of the Literature Review, it is clear that Semantic Analysis is a problem which remains unsolved.

Semantic analysis has many applications, these include:

- **Voice Recognition** has, over recent years, become an increasingly useful feature in consumer electronics. Conceptually, it can be seen that a system's ability to respond to voice input would be improved by a better semantic analysis.
- **The Summarisation of Documents** is a problem that has been partially solved by TF-IDF. The summarisation could be improved by the aquisition of information regading text meaning.
- **The Aquisition of Information for knowledge based systems** could occur automatically,using an effective semantic analysis system in conjunction with a large corpura.

This problem can be seen to be exceptionally relevant to current technologies, making it a worthy subject of this research.

Previously, statistical methods have been used to solve this problem. These methods, as stated previously only consider frequency in their definition, and so are limited in their ability to handle corpura where a single word can occur multiple times throughout, with differing definitions accross these instances.

It is well known that the human brain's process of semantic analysis, is very effective. The brain's disambiguation process most likely uses the memory structures proposed by Baddeley [2], containing information about relationships between word meanings. For example, given the following sentence:

$$The\ party\ led\ government.$$

the reader knows that "party"refers to a political party, likely, as discussed in the Disambiguation Models section, due to its pairing with the word "government".

Previous attempts at the problem of semantic analysis, don't necessarily correlate with our knowledge of the human brain. Given this knowledge, it could be possible to produce an equivalent computational model, which may be more effective than purely statistical methods. For this reason, this project will attempt to make use of a model of the brain's disambiguation processes to build a system, capable of semantic analysis.

3.2 Project Aims

The overall aim of this project is to build a system capable of finding the correct definitions of each word in a corpus. This problem is divisible into a set of smaller problems.

1. **A computational model of memory structures**
   This model must based upon the memory structures, as described by Baddeley [2]. An effective implementation should include both short and long term stores, as well as an episodic buffer. The LTS includes information regarding semantic links between words, the STS should contian information about the local context of a word, and the Episodic Buffer should contain information about the wider context of a word.

2. **A system for reading in a corpus**
   An input corpus must be processed in such a way that the context of a section of text (i.e. a small set of sentences) is represented by the STS, and the context of the corpus read so far is represented by the Episodic Buffer. This should be done using activation and forgettingin the STS, so as to comply with R. Atkinson and R. Shiffrin's model [5].

3. **A computational model of disambiguation**
   This model should make use of the both the local context of a word and knowledge of its frequency, so as to comply with the Multiple Access Model [6]. The contextual disambiguation of a word should make use of the semantic links between its own, and other local words definitions. This would be most appropriately done using the most apparent relation, hyponomy [9, 11].

The completed project will be evaluable according to the above critera, with a completely successful system meeting all of these.

3.3 Project Organisation

4 Design and Implementation

The application was implemented using the Python programming language, chosen for the availability of the Natural Language Toolkit (NLTK) [17]. NLTK provides an interface for use with Wordnet and a number of useful corpura (collections of tagged raw text), to be used as test data.

4.1 Corpus Analysis

Using the corpus reader built into NLTK, an entire document can be analaysed as a series of hierachical lists [17]:

- **Document**, is a list of sentences,
- **Sentences**, each of which is a list of words.

The analyser traverses each of these lists, through a series of loops, processing them a sentance at a time.

4.2 Activation

The corpus is read a sentence at a time, for each word in the sentence a series of activations takes place. Initially, the synsets of each word are activated, followed by their hypernyms following the algorithm outlined in Listing 1.

```
FUNCTION activateHypernyms(synset, depth):
    activationModifier = hypernymModel(depth)
    IF activationModifier < 0 THEN
        RETURN
    ELSE
        FOR hypernym in synset.hypernyms() LOOP
            activateHypernyms(hypernym, depth + 1)
        END LOOP
        RETURN
    END IF
END FUNCTION
```

Listing 1: Hypernym Activation

The function recursively traverses the tree of hypernyms an example of which is shown in Figure 2, activating each synset until, the model gives a result of less than zero. As the function traverses through each hypernym, the amount they are activated by decreases, favouring less general synsets.

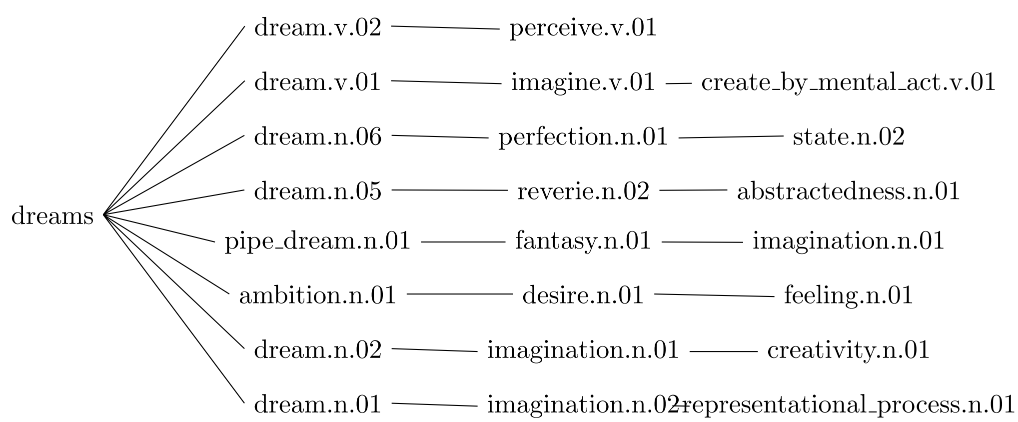

Figure 2: HYPERNYM TREE OF "DREAMS"

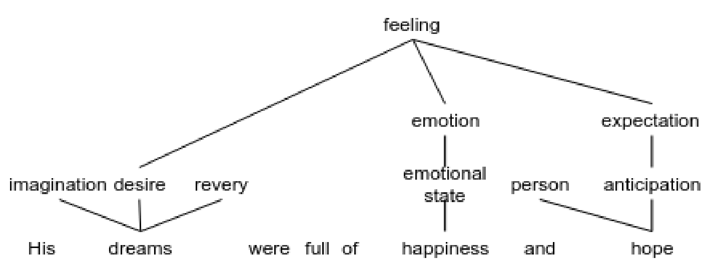

Figure 3: HYPERNYM ACTIVATION

Upon activation, the synset in question's activation is increased by an amount, dependant upon the hypernym model used (discussed later in this section). The same synset can be activated multiple times within the same sentence, an example of which is shown in Figure 3. In this case, a similar model to that proposed by M. Burke [16] is used:

$$H = \sum_{i=1}^{N} S_i$$

The total acivation increase is equal to the sum of all synset activations within the sentence.

It can be seen that, in Listing 1,there exists a function, hypernymModel.

This function is responsible for reducing the activation increase of hypernyms as they become more general (closer to the top of the tree). This function should favour most heavily, less general hypernyms, i.e. those activated by the first few recursions of activateHypernyms.

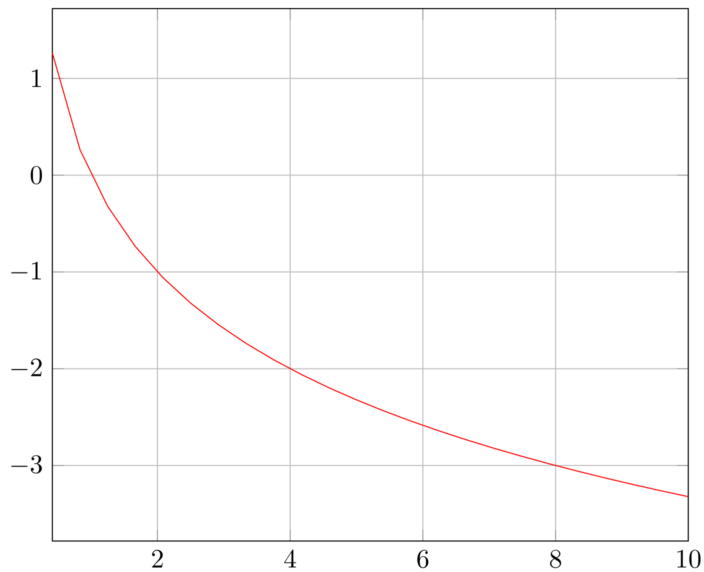

Figure 4: GRAPH OF $-\log(x)$

One function which fits this criteria is $-\log(x)$, shown in Figure 4. The values given by the function, for higher values of x are relatively high, favouring more general hypernyms heavily. For lower values of x, the values given are relatively low, causing more general hypernyms to have a lower activation increase.

It may be necessary to tune this function during experimentation, to provide greater accuracy, meaning more variables must be added. With these additions, the full model is:

$$f(x) = -b\log(\frac{x}{a})$$

where a and b are varied during experimentation, and x is the depth of the hypernym. a defines where the function passes through the x axis, i.e the maximum hypernym depth, and b defines how steep the function is.

When the depth $= 0$ (i.e. the activation of the root synset) the model would require the calculation of $\log(0)$, which is undefined. For this reason, an offset of 0.001 was added, making the final model:

$$f(x) = -b\log(\frac{x}{a} + 0.001)$$

The size of the offset needed to be small to reduce its impact on the overall model.

4.3 Memory Structures

Upon activation, the synset in question may enter the memory structures, if it posesses a high enough activation. Both an STS and Episodic Buffer have been implemented, using python classes for each. Interactions between the two memory structures are handled by a Memory Controller. As discussed in the WordNet section, WordNet provides a good model of the semantic memory [8],described in the Long-term Store section. For this reason it has been used heavily.

4.3.1 STS

The STS (Stm in psuedocode) can, in a simplified sense, be described as a list containing memory items. These memory items have been implemented in python as an object with two attributes, synset (immutable) and activation (mutable), based upon the contents of the STS in the working memory model. These memory items can be replaced, if a synset with greater activation appears. The process of synset replacement is handled by the swapLowestItem function shown in Listing 2.

```
FUNCTION swapLowestItem(newItem):
        IF Stm.size < self.maxStmSize THEN
            Stm.add(newItem)
            RETURN None
        ELSE
            self.lowestItem = self.getLowestActivation()
            IF newItem.Activation < stm.lowestActivationItem THEN
                RETURN None
            ELSE
                Stm.remove(lowestActivationItem)
                Stm.add(newItem)
                RETURN stm.lowestActivationItem
            END IF
        END IF
END FUNCTION
```

Listing 2: the swapLowestItem function

A notable feature of the STS is its limited capacity, as described by G. Miller in 1955 [4]. It is for this reason that the ability to swap memory items is required. In the case that the STS isn't full, the new memory item is simply added with no swap taking place.

4.3.2 Episodic Buffer

The purpose of the episodic buffer, is to maintain a list of synsets which have occurred in the STS previously. This allows future activations of synsets which have already occured to receive a boost when activated in the future. Interactions with the Episodic buffer only occcur during the activation of synsets, therefore, all interactions are handled by the Memory Controller.

4.3.3 Semantic Memory

The Semantic Memory is responsible for maintaining all the semantic knowledge of the system. It was decided that WordNet provided the best solution, with its in depth knowledge of synsets and their relations to eachother [9, 11].

NLTK's built in WordNet reader [17] was used to interact with WordNet. Throughout the system, synsets are used. Each of these is a reference to a synset present in WordNet, allowing their semantic relationships (accessed through their methods) to be used.

4.3.4 Memory Controller

A memory controller has been implemented to handle interactions between the STS and episodic buffer. The most notable of its uses is the activation of synsets. In order to activate a synset, the Memory Controller must take into account the whether the synset is present in any of the afformentioned memory structures, and handle the activation accordingly. The activateSynset function, shown in Listing 3 is responsible for handling this process.

```
FUNCTION activateSynset(self, synset, activationModifier):
        IF stm.inContents(synset) THEN
            synset.activate(activationModifier)
            RETURN
        ELSE IF episodicBuffer.inContents(synset) THEN
            newMemItem = MemItem(synset, 1)
            newMemItem.activate(activationModifier)
            sendToStm(newMemItem)
            RETURN
        ELSE
            newMemItem = MemItem(synset, 0)
            newMemItem.activate(activationModifier)
            sendToStm(newMemItem)
            RETURN
        END IF
```

Listing 3: the activateSynset function

If the activateSynset finds the synset in the STS, the synset is simply activated using, and remains there. If the synset is not present in the STS, it is activated, and sent to the STS, through the sendToStm function, shown in Listing 4, recieving a boost if it has previously existed in the STS (i.e. it is in the Episodic Buffer).

```
FUNCTION sendToStm(inputSynset):
        returnedItem = self.stm.swapLowestItem(inputItem)
    IF returnedItem is None THEN
        RETURN
    ELSE IF NOT episodicBuffer.inContents(returnedItem) THEN
        episodicBuffer.addSynset(returnedItem)
        RETURN
    END IF
END FUNCTION
```

Listing 4: the sendToStm function

The sendToStm function, shown in Listing 4, is responsible for changing the contents of the STS and updating the Episodic buffer accordingly. The swapLowestItem function, shown in Listing 2 is used to edit the STS. As discussed previously, this function can return a synset, if the swap is successful. In this case, sendToStm will add the synset previously in the STS, to the episodic buffer (if it is not already present there).

Throughout the analysis of a corpus, the activation of a specific memory item can change by either being forgotten or activated.

4.4 Forgetting

After each sentence is read, all items in the STS must be forgotten (have their activation reduced), so as to comply with Atkinson and Shiffrin's STS model discussed previously [5]. The forgetting of the entire contents of the STS is done using a loop, as shown in Listing 5.

```
FOR item IN Stm LOOP
        item.forget()
END LOOP
```

Listing 5: Forget loop

The forgetting process occurs after every sentence, giving synsets in the next sentence the opportunity to enter the STS.

When a synset is forgotten, its activation may fall below a threshold, varied in experimentation. If this occurs, the synset is removed from the STS and, as with the sendToStm function, is added to the Episodic Buffer. This prevents synsets irrelevant to the current context from existing in the STS.

4.5 Disambiguation

Up to this point, we have only discussed the processes involved with changing the contents of the memory structures. The purpose of these memory structures is to provide the surrounding context of a word, to aid in its disambiguation. There exists a delay between the processes described previously (i.e. the initial reading of the sentence) and the disambiguation phase. This delay allows inappropriate synsets to leave the STS before they are used. The duration of this delay is to be adjusted during experimentation.

As discussed in the Disambiguation Models section, the multiple access model relies upon context and frequency, using whichever is stronger for disambiguation [6]. The system discussed in this section makes use of both of these, through the disambiguate function, shown in Listing 6.

```
FUNCTION disambiguate(synsetList):
    FOR item in stm LOOP
        IF item in synsetList THEN
            RETURN item
        END IF
    END LOOP
    FOR item in stm LOOP
        returnedSynset = hyponymSearch(synsetList, item)
        IF returnedSynset is not None THEN
            RETURN returnedSynset
        END IF
    END LOOP
    RETURN mostLikelySynset(synsetList)
END FUNCTION
```

Listing 6: The disambiguate function

4.5.1 Context

The context used for disambiguation its taken from two sources, the STS and knowledge of noun-verb relationships (the latter is discussed in the Sanity Checking Section).

Initially, the context contained in the STS is used. Two algorithms for disambiguation using this context have been implemented, one using hyponyms and one using hypernyms.

As discussed in the Previous Works section, M. Burke found that hyponym based searching (previously called STM-First), using the Stm is the most effective method of disambiguation [16]. This method works upon the assumption that the contents of the Stm are relatively general (i.e. exist high up in the Wordnet hierarchy). The implementation of this search is given by the hyponymSearch function, shown in Listing 7.

```
FUNCTION hyponymSearch(synsetList, searchItem):
    hyponymList = searchItem.hyponyms()
    IF len(hyponymList) == 0 THEN
        RETURN None
    END IF
    FOR item in hyponymList LOOP
        IF item in synsetList THEN
            RETURN item
        END IF
    END LOOP
    FOR item in hyponymList LOOP
        returnedItem = hyponymSearch(synsetList, item)
        IF returnedItem is not None THEN
            RETURN returnedItem
        END IF
    END LOOP
    RETURN None
```

Listing 7: The hyponymSearch function

This function traverses through all hyponyms of a given synset (searchItem) until either, a match with a synset in the synsetList is found, or the function reaches the base of the tree. If a match is found, the matched synset is returned as the correct word meaning.

Another algorithm, shown in Listing 8, for disambiguation has also been implemented, using the lowest_common_hypernym method implemented in NLTK's WordNet reader [17]. This algorithm instead searches the hypernyms of both the word to disambiguate and the item in the STS, to find whether a common hypernym exists. If a common hypernym does exist, it is checked to ensure it is not too general (e.g. the common hypernym could be "entity" which is a hypernym of all nouns), and the synset is returned as the correct meaning.

```
FUNCTION hypernymSearch(synsetList, searchItem):
    FOR synset in synsetList LOOP
        common_hypernym = synset.lowest_common_hypernym(searchItem)
        IF common_hypernym.depth > 4 THEN
            RETURN synset
        END IF
        END LOOP
```

Listing 8: THE HYPERNYMSEARCH FUNCTION

Using either method, a meaning will only be found if the surrounding context of a word gives enough indication of its meaning, complying with the multiple access model, where if strong context exists, it is used for disambiguation [6].

4.5.2 Frequency

The previously described hyponym-based system will only find a meaning if a word's surrounding context is strong. When this is not the case, another system must be used. The multiple access model, described in the Disambiguation Models section, states that in such cases, the likelihood of a proposed word meaning is used [6].

In order to calculate the likelihood of each synset in the synsetList, Wordnet's lemmas are used. Lemmas in Wordnet are the word forms a synset can take. For each lemma, there exists a frequency value, which describes how common that form is, relative to other Wordnet lemmas. For each synset, the sum of all its lemmas' frequency values, is used to calculate the most likely synset, as shown in Listing 9.

```
FUNCTION synsetFrequency(synset):
    outputFrequency = 0
    FOR lemma in synset.lemmas() LOOP
        outputFrequency += lemma.count()
    END LOOP
    RETURN outputFrequency
END FUNCTION

FUNCTION mostLikelySynset(synsetList):
    outputSynset = synsetList[0]
    FOR synset in synsetList LOOP
        IF synsetFrequency(outputSynset) < synsetFrequency(synset) THEN
            outputSynset = synset
        END IF
    END LOOP
    RETURN outputSynset
END FUNCTION
```

Listing 9: The synsetFrequency and mostLikelySynset functions

As stated previously, in cases where context cannot provide a viable synset, the above is used for disambiguation.

4.5.3 Sanity Checking

Given the sentence:

He fixed the bug

The reader knows that "bug" refers to a computer bug, due to its use with the verb "fixed". So far, the system has no way of modelling the contextual effects of nouns on verbs, so the system is likely to produce invalid sentences. For this reason, post-disambiguation sanity checks occur.

Though a proposed feature, WordNet does not currently contain relationships between nouns and verbs [11]. For this reason, the information had to be extracted from the available corpus. It was decided that half of the test corpus would be used for the extraction, so that test data used for evaluation would be completely new to the system.

Listing 10 shows the algorithm used for extracting this information. Nouns which occur in the same sentence as a specific Verb are considered valid partners to the Verb, and are added to a dictionary for quick lookup.

```
FUNCTION verbDistance(verb, sentence):
        FOR word in sentence LOOP
                IF word is a noun THEN
                        outputList.append(word.synset)
                END IF
        END LOOP
        RETURN outputList
END FUNCTION

FOR sentence in inputCorpus LOOP
        FOR every verb in sentence LOOP
                update contents of verbDict[verb.synset]
                to include verbDistance(verb, sentence)
        END LOOP
END LOOP
```

Listing 10: The noun-verb relationship learning algorithm

Initially, the sentence is disambiguated, using the method described previously. The outputted sentence is then checked for correctness by the function sanityCheck, as shown in Listing 11. If a synset is found to be incompatible with others in the sentence, it is added to a blacklist, and its corresponding word is disambiguated again, ignoring the previous meaning.

```
FUNCTION sanityCheck(inputSentence, nounDict, verbDict):
    sane = False
    FOR word in inputSentence LOOP
        IF word is a noun THEN
            nounList.append(word)
        ELSE IF word is a verb THEN
            verbList.append(word)
        END IF
    END LOOP
    WHILE not sane LOOP
        sane = True
        FOR verb in verbList LOOP
            IF verb in verbDict THEN
                plausibleNouns =  verbDict[verb]
```

```
                IF nounList and plausibleNouns share common words THEN
                    verbList.remove(verb)
                ELSE
                    blackList.append(verb)
                    verbList.remove(verb)
                    Run disambiguation again with blacklist
                END IF
            END IF
                END LOOP
        END LOOP
END FUNCTION
```

Listing 11: The sanityCheck function

It may be noted that sanityCheck only considers verb incorrectness. As stated in the WordNet section, for each word, there is likely to be a larger set of synsets when compared to nouns. With this knowledge, we can assume that the likelyhood of an incorrect result for a verb is higher than that of a noun.

5 Results and Evaluation

Throughout this section, the system described previously, in section 4, will be evaluated. Each part of the system will be evaluated, with their parameters being adjusted, to produce the optimal performance. The criteria described in the Project Aims subsection will also be used to evaluate the system as a whole.

In order to test the system, the SEMCOR corpus, described in subsection **??**, will be used. With the sanity checking system requiring data to be extracted from the corpus, a subsection of the corpus will need to be used for testing alone. The rest of the corpus can then be used for data extraction, without skewing the results of testing in the positive direction.

For the purposes of parameter adjustment, the system will be tested on a single document, to reduce the time taken for each test. This will be taken into account when evaluating the results of this process. The system will then be tested across 5 documents ranging across a variety of topics. This will give a more thorough indication of the system's actual performance.

Two critera will be used to evaluate each part of the system, accuracy and the amount of synsets directly seen in the STS. As many would expect, a higher value for the accuracy would imply a more successful system. The number of directly seen synsets is a less clear method for evaluation. A high number of synsets directly seen would imply that the either, the local context of a word is not strong, or the STS has failed to correctly represent the local context. An effective STS would contain hypernyms of the words in each sentence, meaning the more general concepts of the context are represented.

5.1 Random Chance

The most basic form of disambiguation system is one which selects a synset for each word based upon random chance. When tested, the system gives an accuracy of 0.7%, a very poor performance. This is not unexpected, given the number of possible synsets a word can have, a random choice is unlikely to select the correct one, especially in cases where a word has many possible synsets (for example, verbs).

It can be seen, that a system based upon random chance alone is not useful. The process of finding the correct synset could be improved by including knowledge about each possibility.

5.2 Frequncy

The next step up from the Random Chance system would be one which relies upon the frequency of each synset. This system makes use of the synsetFrequency function, shown in Listing 9, to find the most likely synset for each word.

When tested, this system performs significantly better than random chance, with an accuracy of 12%. Even though it is an improvement, it can be seen that Frequency alone does not produce a satisfying result. From this, we can tell that frequenncy only plays a small part in the disambiguation of text, and contextual information is required for a more effective system.

5.3 STS

In order to introduce contextual information, the STS, described in Section 4.3.1 can be used. The STS system has many parameters which can be varied, so the process of testing will have to be broken down.

5.3.1 STS Size

In section 2.1.2, it was identified that the size of the STS existed in the range of 7±2 items. In order to find the optimal value for this STS system, each size in the range will need to be tested.

A larger STS could give a broader impression of the local context, with more synsets being present. The ability to hold more synsets could also lead to ambiguity existing in the STS, with multiple synsets relating to different possible definitions of each word being present. Reducing STS size would decrease the possibility for ambiguity, but it could also reduce the amount of context the system is able to represent.

As can be seen in table 1, an increased STS size leads to a minimal decrease

STS size	Accuracy (%)	Directly Seen (%)
5	47	48
6	47	51
7	46	54
8	46	55
9	46	58

Table 1: The effect of STS size on system performance

in the accuracy of the system. Contrary to this, the number of synsets directly seen increases significantly, implying the STS's ability to represent the local context, is hindered by a larger size.

Given the results of this test, it has een decided that 5 is the optimal size for the STS. When presented with a larger number of test documents, the system performs with an accuracy of 35%, having 45% of synsets directly seen in the STS. This is significantly better performance than a system based soley on frequency.

5.3.2 Activation Function

The activation function, described in section 4.2, is responsible for how synsets are activated. Previously, a linear activation function was proposed by Matt Burke, where the activation of hypernyms was reduced according to a constant factor. This function was used in the previous test,and will be used as a benchmark, in which the logarithmic function proposed by this paper will be tested against.

$$f(x) = -b\log(\frac{x}{a} + 0.001)$$

In the function described in section 4.2, shown above, there exist two variables, a and b, which can be changed. The maximum hypernym depth, given by a, will be varied in a range from 1 to 5, and the value of b will be varied between 1 and $\frac{1}{16}$.

For the first test, a will be varied, and b will remain constant, with value 1. A small value for a would lead to no hypernyms being activated, causing a loss of generality in the STS, meaning poor representation of the local context. With large values, the contents of the STS could become too general, giving the same problems as a small a.

Given the results shown in table 2, the best performance is given when $a = 2$. For this reason, the value of 2 will be used when testing different values for b.

A high value of b would favour less general synsets significantly more than their hypernyms. Smaller values would reduce this effect, though more general

a	Accuracy (%)	Directly Seen (%)
1	45	40
2	46	39
3	45	44
4	46	40
5	43	40

Table 2: The effect of changing a on system performance

b	Accuracy (%)	Directly Seen (%)
1	46	39
$\frac{1}{2}$	47	45
$\frac{1}{4}$	46	51
$\frac{1}{8}$	47	52
$\frac{1}{16}$	46	52

Table 3: The effect of changing b on system performance

synsets may come to dominate the STS, leading to poor representation of local context.

Given the results shown in table 3, it can be seen that a value of 1 for b, achieves the best balance between accuracy and the number of synsets directly seen. With more testing data, the system produces an accuracy of 35%. This is no imporvement from the previously used method. Though the accuracy has remained consistent, only 39% of synsets were directly seen, implying more generality in the STS, without loss of performance.

5.4 Episodic Buffer

The episodic buffer, described in section 4.3.2 aims to provide a more global context than the STS. With the functionality of the buffer being relatively simple, only one varibale exists, the boost provided by a previously seen synset, with a greater boost giving global context a larger impact on disambiguation. The size of this boost will be varied between 0 (i.e. no episodic buffer) and 2.

The results in table 4 show that the use of an episodic buffer dramatically improves performance, with an optimal boost of 1.5. When provided with 5 documents, as opposed to 1 when tuning parameters, the system produces an accuracy of 39%, with 23% of synsets directly seen, giving an increase in both accuracy, and the generality of the STS.

Boost	Accuracy (%)	Directly Seen (%)
0	46	39
0.5	48	35
1	51	21
1.5	51	19
2	51	22

Table 4: The effect of changing the episodic buffer boost on system performance

Algorithm	Accuracy (%)	Directly Seen (%)
Hyponym	51	19
Hypernym	50	19

Table 5: The effect of changing the disambiguation algorithm on system performance

5.5 Disambiguation

5.5.1 Algorithm

Up until this point, the hyponym based algorithm has been used for testing, described in section 4.5. Another algorithm, hypernymSearch, has also been proposed. Table 5 show the result of this testing.

As can be seen, the hypernym based algorithm produces no benefit over the usage of hyponyms. For this reason hyponyms will continue to be used, and the benchmark given by the previous section still stands.

5.5.2 Delay

In section 4.5, a delay between reading and disambiguation is proposed. This delay is measured in sentences read, and will be varied between 0 and 2. A larger delay could cause the local context of a word to be underrepresented when disambiguation takes place, though it may also give the system chance to resolve ambiguity in the STS, before its contents are used.

Delaying disambiguation by 2 sentences has a positive effect on the overall performance of the system, as shown in table 6. The most notable effect of the delay, is the reduction in synsets directly seen. This shows that the delay gives the system time to remove less general synsets from the STS. When provided with more input data, the improvement is still present, with an accuracy of 40%, and 15% of synsets directly seen.

Delay	Accuracy (%)	Directly Seen (%)
0	51	19
1	53	13
2	53	12

Table 6: The effect of changing the disambiguation delay on system performance

Sanity Check	Accuracy (%)	Directly Seen (%)
On	53	12
Off	50	11

Table 7: The effect of sanity checks on system performance

5.5.3 Sanity Checking

The sanity checking system, described in section 4.5.3, makes use of information regarding noun-verb relationships to check the disambiguation output, and feed-back to the algorithm. The relatively limited noun-verb relationship data may hinder the disambiguation algorithm, by dismissing valid results, though it could equally improve performance, especially in cases where a large number of possibilities are present.

As it can be seen in table 7, sanity checking reduces the accuracy of the system. For this reason, it can be seen that it is not useful in its current state, and can be eliminated from the system.

5.6 Overall Evaluations

The system as a whole has offered an improvement over frequency analysis, with an accuracy of 40% compared to 12%. All but one aspect of the system provided some benefit to the process of semantic analysis, with Saity Checking unfortunately failing to offer any improvement.

It can be seen, when comparing testing data (one document analysed) to evaluation data (five documents analysed), that the system performed significantly better over some documents than others. This difference is likely due to the level of ambiguity in the input text, with a news piece being used for testing, and topics covering opinion and fiction being used for evaluation.

6 Conclusions

This project has aimed to utlise previously established psychological models of disambiguation, and apply them to the problem of Semantic Analysis. The models used are those originally developed by Atkinson and Shiffrin [5], and then later further studied by Baddeley [2].

6.1 Project Aims

In section 3.2, three aims for the project were established. All three aims have been met, with varying degrees of success.

1. **A computational model of memory structures**
   As part of this project, three memory structures have been produced, STS, LTS and the Episodic Buffer.

   - The STS is a memory structure capable of holding a finite amount of data regarding the local context of a sentence. In the implementation of this project two STSs were used, one for verbs and the other for nouns. This decision does not follow the Badeley's model, in which there exists only one STS, containing data regarding all word types [2]. With that said, words are able to move to and from the STS, each with its own activation level.
   - The LTS, provided by WordNet, contains a large number of synsets connected to one another by semantic links. Hyponomy was the most heavily used of these links, though others such as meronymy, were not used. The functions of nouns were represented using separate dictionaries, due to the fact that WordNet does not yet contain this information, though these dictionaries were far from complete.
   - The Episodic Buffer was shown in testing to have a significant effect on accuracy. This is likely due to its ability to correctly represent the global context of a document. In this implementation, items would never leave the episodic buffer, meaning that synsets that appear early in the document, can still affect activation, long after context has changed.

2. **A system for reading in a corpus**
   The system uses atctivation and forgetting to model the same prrocesses which exist in Atkinson and Shiffrin's model [5]. As the input corpus is read, these processes occur for each word, causing context to be represented by the STS and Episodic Buffer.

3. **A computational model of disambiguation**
   The disambiguation model makes heavy use of the contents of the STS, in conjunction with the hyponomy relation. Frequency is also used in cases

where the correct synset can not be identified using context alone, fitting with the Multiple Access Model [6]. Unfortunatly, the disambiguation system only produced an accuracy of 40%, which, though significantly better than frequency analysis alone (12%), is not good enough to be directly useful.

The seemingly low accuracy of the system does not imply that it is is of not of use. We can see that it is capable of finding the correct synset in some cases. In others, it is not unlikely that the system reduced the number of possibilities to a smaller subset, and from then, failed to select the correct option. In this case, the described implementation could be used to reduce the number of possibilities, which in turn could be made smaller still by some future extension.

6.2 Future Extensions

6.2.1 Semantic Links

In this implementation, some semantic links have not been utilised. One of such links is Meronymy. In the Problem Definition section, the following example was given:

The party led government.

It was stated that the brain can identify that "party" refers to a political party, due to its use with the word government. This is an example of where meronymy is used, with a government containing multiple parties. "Party" and "government" share no common hypernyms in WordNet (excluding "Synsetentity"), meaning the system, as described in section 4, would fail on this sentence.

Another semantic relation partially excluded from this implementation is functions. An attempt at using this semantic link was made, with the inclusion of Sanity Checks. Unfortunately, The Sanity checking system was detrimental to the system as a whole, most likely due to the limited amount of data available to it.

Functions are a proposed feature of WordNet [9], though due to the scale of the inclusion, they are yet to be added to the available database. Inclusion of noun functions would reduce the need for independent memory structures for nouns and verbs. This relation would allow locally used nouns to contribute to the disambiguation of verbs, and vice versa, an especially useful process in cases where noun/verb based context is low.

6.2.2 Improved Usage of Syntactic Parsing

Syntactic parsing, more specifically chunking, provides information about verb and noun phrases. These phrases provide a more specific local context than a set of sentences, as used by the implemented system.

Noun phrases contain a noun and its related adjectives. In this case, the adjectives used can provide some indication of the meaning of the noun, for example:

The white cloud.

In this noun phrase, we know that "cloud" does not refer to the internet, because the internet has no colour. This extends to verb phrases, where the set of possible meanings could be reduced using functions, as discussed in the previous subsection.By using knowledge of which words directly relate to one another in a sentence, we know which context is relevant, and which is not.

References

[1] F. Damerau. N. Indurkhya, *Handbook of Natural Language Processing*, 2nd ed. Taylor and Francis Group, 2010.

[2] M. Eysenck. A. Baddeley and M. Anderson, *Memory*, 2nd ed. Psychology Press, 2015.

[3] M. Quillain, “Semantic memory,” *BBN Report*, vol. 675, no. 2, pp. 227–270, Oct 1966.

[4] G. Miller, “The magical number seven, plus or minus two some limits on our capacity for processing information,” *Psycological Review*, vol. 101, no. 2, pp. 343–352, May 1955.

[5] R. Shiffrin. R. Atkinson, “The control processes of short term memory,” *Technical Report*, vol. 173, Apr 1971.

[6] T. Harley, *The Psychology of Language: From Data to Theory*, 3rd ed. Psychology Press, 2008.

[7] D. Swinney. W. Onifer, “Accessing lexical ambiguities during sentence comprehension: Effects of frequency of meaning and contextual bias,” *Memory & Cognition*, vol. 9, no. 3, pp. 225–236, May 1981.

[8] G.Miller. et al., “Introduction to wordnet: An on-line lexical database,” *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, Aug 1990.

[9] G.Miller, “Nouns in wordnet: A lexical inheritance system,” *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, Aug 1990.

[10] C. Fellbaum. et al., “Adjectives in wordnet,” *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, Aug 1990.

[11] C. Fellbaum, “English verbs as a semantic net,” *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, Aug 1990.

[12] S. Dumais. T. Landauer, “A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge,” *Psychological Review*, vol. 104, no. 2, pp. 211–214, Apr 1997.

[13] J. Ramos, “Using tf-idf to determine word relevance in document queries,” *Proceedings of the first instructional conference on machine learning*, vol. 1, Dec 2003.

[14] D. Kazakov. J. Sedding, “Wordnet-based text document clustering,” *Proceedings of the 3rd Workshop on RObust Methods in Analysis of Natural Language Data*, vol. 3, pp. 104–113, Aug 2004.

[15] J. Weston. R.Collobert, "A unified architecture for natural language processing: deep neural networks with multitask learning," *Proceedings of the 25th international conference on Machine learning*, vol. 25, pp. 160–167, Jul 2008.

[16] M. Burke, "Contextual disambiguation using wordnet and a short-term memory model," BEng dissertation, Dept. of Computer Science, University of York, York, UK, 2007.

[17] E. Klein. S. Bird and E. Loper, *Natural Language Processing with Python*, 1st ed. O'Reilly Media, 2009.