Determining if this *word* is used like that *word*: Predicting usage similarity with supervised and unsupervised approaches

by

Milton King

Bachelor of Science, Mount Allison University, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Faculty of Computer Science

Supervisor(s):	Paul Cook, PhD, Computer Science
Examining Board:	Michael Fleming, PhD, Computer Science, Chair
	Christopher Baker, PhD, Computer Science
	Wladyslaw Cichocki, PhD, Linguistics

This thesis is accepted

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

August, 2017

Abstract

Determining the meaning of a word in context is an important task for a variety of natural language processing applications such as translating between languages, summarizing paragraphs, and phrase completion. Usage similarity (USim) is an approach to describe the meaning of a word in context that does not rely on a sense inventory—a set of dictionary-like definitions. Instead, pairs of usages of a target word are rated in terms of their similarity on a scale. In this thesis, we evaluate unsupervised approaches to USim based on embeddings for words, contexts, and sentences, and achieve state-of-the-art results over two USim datasets. We further consider supervised approaches to USim, and find that they can increase the performance of our models. We look into a more detailed evaluation, observing the performance on different parts-of-speech as well as the change in performance when using different features. Our models also do competitively well in two word sense induction tasks, which involve clustering instances of a word based on the meaning of the word in context.

Acknowledgements

Many people have made a positive influence on me during my time in my masters degree. Dr. Paul Cook, I thank for his guidance throughout my degree. I would also like to thank my family for continuously supporting me and sending up a dish of food every now and then. My lab mates, Waseem and Priyal, I thank for your assistance and discussions. I would also like to thank Jodi, for helping make my experience in my masters much smoother. Alina, I would like to thank for her encouragement. Finally, I would like to thank ACENET for providing me access to their hardware as well as financial support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The meaning of a word changes depending on the context in which it occurs. For example, the word *run* has different meanings in *I will run the program* and *They are going to run across Canada*. Word sense disambiguation (Navigli, 2009) attempts to assign a predetermined meaning, similar to a dictionary-like definition of a word, called a sense, to an instance of a word. Determining the sense, or meaning, of a word in context can help many downstream applications such as machine translation, plagiarism detection, and next word suggestion on smartphones (Zou et al., 2013). One important issue with word sense disambiguation is that a predefined sense inventory must be available, which can be expensive to generate and may not contain new words and new word senses. Usage similarity is a task that is able to describe the meaning of a word in context without the requirement of a sense inventory, through looking at how similar the usages of a word are on a scale from 1–5 in two different sentences (Erk et al., 2009). The guideline for the scale is 1 (completely different), 2 (mostly different), 3 (similar), 4 (very similar), and 5 (identical). The idea is that instances with

similar usages will have similar meanings. For example, in the following sentences that were present in a dataset from Erk et al. (2009), the similarity score—assigned by human annotators—for the word *strong* between a and b is 3.3, while c and d are less similar with a score of only 1.3.

(a)The chain will only be as **strong** as its weakest link.
(b) That means Norway must send a **strong** signal.

(c) Try a beer with a **strong** fruity flavor ; raspberry and peach are good choices.
(d) This race is about leadership, and I have clearly demonstrated throughout my career that I have the courage and determination needed to be a **strong** leader.

In our work we will be using vector representations of words, contexts, and sentences known as embeddings to generate vector representations of usages. These embeddings are generated through the training of artificial neural networks and have been applied in a variety of natural language processing tasks such as sentence completion (Melamud et al., 2016), semantic relatedness and paraphrase detection (Kiros et al., 2015).
The term *word* has been and still is a difficult term for linguists to define (Crystal, 1987). The term *light bulb* is one example that shows a reason why it can be difficult. It is not concrete if *light bulb* should be two words, *light* and *bulb*, or one word possibly with a hyphen, *light-bulb*. Crystal (1987) explains five different preexisting methods for determining if an item is a

word. These methods include, but are not limited to, observing if there is a pause when saying the item aloud, as well as deciding if it is possible to add another word in the item. In our work, we will be using the terms lemma, type, and token when discussing words or the meaning of the term word will be indicated by the context. A lemma represents a set of related word forms. For example, *play* is the lemma of *play, playing, played,* and *plays*. A type is any occurrence of the same word form. For example, the usages of *cat* in *the cat jumped* and *a cat ran* are of the same type. A token is more specific and is any instance of a word form, i.e., the token *cat* occurs twice in the previous exmaple.

1.1 Research Questions

The following questions have guided this research and are the questions we answer in our work.

1. Do unsupervised usage similarity models based on embeddings outperform the state-of-the-art approaches to usage similarity? The state-of-the-art was achieved by Gella et al. (2013) and Lui et al. (2012) on text from Twitter and more conventional text, respectively, using topic modelling in an unsupervised manner (Blei et al., 2003). We will develop unsupervised methods based on embeddings and compare these unsupervised methods to the same datasets that were tested on in Gella et al. (2013) and Lui et al. (2012).

2. Can a supervised model improve our results? Supervised models train on a labeled dataset and therefore often outperform unsupervised models,

which do not. We would like to see if we can achieve better results by adding supervision to our model.

3. Does our unsupervised model also perform well on word sense induction tasks? Word sense induction is a clustering task, where instances of a word are clustered based on the sense that they exhibit. Similar to usage similarity, a representation of the word in context is needed. We would like to see if the unsupervised model that we apply to usage similarity is also able to perform well in two different word sense induction tasks.

1.2 Layout

We will look at the work that is related to usage similarity and approaches applied to it in Chapter 2, including different types of embeddings, and the tasks of word sense induction and word sense disambiguation. We then discuss the different ways that we will represent a usage of a word (diamond in Figure 1.1) as a vector (rectangle in Figure 1.1), along with using this representation in both our unsupervised and supervised usage similarity models (top two ovals in Figure 1.1) in Chapter 3. The description of the setup for the usage similarity experiments is given in detail in Chapter 4, including the two different datasets—one containing text from Twitter and the other containing more conventional text—and the evaluation metric used. Our experimental results are then presented in Chapter 5. First we consider evaluations on parameter tuning for some embedding models, and then we look at detailed evaluations including the results pertaining to different parts-of-speech, as well as using different features for our super-

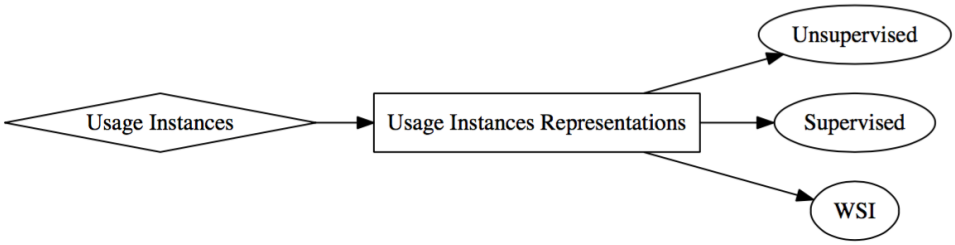

Figure 1.1: A flow chart illustrating how usage instances are represented by vectors, and then used in our experiments on unsupervised and supervised usage similarity as well as word sense induction.

vised setups. We then take our unsupervised model and apply it to word sense induction tasks (bottom oval in Figure 1.1) in Chapter 6 and compare our results with those that have previously approached these tasks. We finish with a discussion leading into possible future directions to build on our work, and a conclusion of our findings, in Chapter 7.

1.3 Contributions

In this work, we present an embedding-based model that achieves state-of-the-art results on two different usage similarity tasks—one containing text from Twitter and one containing more conventional text. We show that we can improve our model even further by applying supervised learning. We provide a detailed analysis of different features for our supervised model to show which features work best under different setups. Also, our unsupervised model's representations of words is general enough that it can be applied to word sense induction, where we found that our unsupervised model achieved the highest value on one evaluation metric, and performed competitively on other evaluation metrics.

Chapter 2

Related Work

In this section we look at previously done work on generating embeddings, and approaches to word sense induction, word sense disambiguation, and usage similarity.

2.1 Embeddings

Traditional co-occurrence-based distributional models store the co-occurrence statistics of each word in large sparse vectors that contain n dimensions, where n is the size of the vocabulary. These vectors are often mainly populated with zeros, which can give incorrect similarity measures. For example, if two words that are known to be similar, such as *cat* and *feline*, do not co-occur with many of the same words, they will not be as similar as *cat* and *dog* which might co-occur with many of the same words, as seen in the following artificial example where V_i represents the vector representation for the words in sentence S_i. (This example is only for this toy corpus of three

sentences to make the problems of this model easily visible, where normally the model would use frequency counts from a larger corpus and observe the context words of multiple instances of each target word.)

$$S_1 = \text{the } \underline{\text{cat}} \text{ ate the small mouse}$$

$$S_2 = \text{the } \underline{\text{dog}} \text{ ate the large mouse}$$

$$S_3 = \text{usually a } \underline{\text{feline}} \text{ is covered in fur}$$

Constructing vectors where the indexes pertain to co-occurrences of the words [the, ate, small, mouse, large, usually, a, is, covered, in, fur], we get the following:

$$V_1 = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]$$

$$V_2 = [1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0]$$

$$V_3 = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]$$

V_1 would be rated much more similar to V_2 than the expected V_3, which is shown by the cosine similarity, comparing $cos(V_1, V_2) = 0.75$ against $cos(V_1, V_3) = 0$, where the cosine similarity of vectors A and B is defined as $cos(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{j=1}^{n} A_j^2} \sqrt{\sum_{k=1}^{n} B_k^2}}$, where n is the length of the vectors.
Embeddings are dense vector representations that overcome this issue of sparsity. Some of the most recent embeddings are produced from artificial neural networks and achieve state-of-the-art or competitive results over a variety of NLP tasks including sentence completion (Melamud et al., 2016), semantic relatedness and paraphrase detection (Kiros et al., 2015). These

embeddings can be constructed to represent a word (Mikolov et al., 2013; Pennington et al., 2014; Neelakantan et al., 2015), context (Melamud et al., 2016), sentence (Kiros et al., 2015), or document (Le and Mikolov, 2014). The training models can project a large dimensional vector down to a vector of far fewer dimensions with the idea of preserving the important information.

2.1.1 Word Embeddings

Starting at the word type level, Mikolov et al. (2013) implemented Word2vec, which consists of two variations of a neural network that embed a single word (Figure 2.1). The first model is called Skip-gram. It takes a target word $w(t)$ in the form of a one-hot vector—a vector that contains all 0's except for a 1 at the index mapped to the given word—as an input and an artificial neural network tries to predict a fixed window of width $\pm\, i$ of context words $(w(t-i), w(t-i+1), ..., w(t-1), w(t+1), ...w(t+i))$ surrounding that target word. The projection layer is a matrix of weights where the input target word is mapped to a lower dimensionality. The next layer, which is not seen in Figure 2.1, is the hidden layer where the distribution over all vocabulary words is learned and stored as another weight matrix. The weights in this matrix are updated in training similar to the weights of a hidden layer in a regular feed forward artificial neural network. For example, given the training sentence *see the cat in the tree*, for the target word *cat*, and a window size of ± 1, then the network would attempt to predict the one-hot vector representations of *the* at $w(t-1)$ and *in* at $w(t+1)$. If the artificial neural network predicts any of these incorrectly, then the weights

involved with the incorrect prediction are updated in the hidden layer's weight matrix. The other Word2vec model is called Continuous Bag of Words (CBOW) and is similar to Skip-gram but instead, the input is the context words $(w(t-i), w(t-i+1), ..., w(t-1), w(t+1), ...w(t+i))$ and the network tries to predict the target word $w(t)$.

Mikolov et al. (2013) developed a word analogy dataset where both semantic and syntactic questions are asked in the form of *a is to b as c is to* __. Semantic questions capture the relationship in terms of meaning between sets of words such as *Paris is to France as Ottawa is to* __. There are nine syntactic relationships that are captured in the syntactic questions, such as pluralization—as in *mouse* is to *mice* as *dollar* is to __—and past tense—as in *swim* is to *swam* as *walk* is to __. Word2vec had the state-of-the-art results for joint accuracy over both the semantic and syntactic datasets. Word2vec also had the state-of-the-art results on the Microsoft sentence completion challenge (Zweig and Burges, 2012) until Melamud et al. (2016) applied their Context2vec model discussed in Section 2.1.2 to the same dataset. In sentence completion, participants are given a sentence with one word missing and are required to select the correct word from a list of options. The Microsoft sentence completion challenge dataset had five words to choose from but only one word was logical with regards to semantics, although all words would fit in terms of syntax.

Word2vec does not take into account the global information provided from the document outside of the context window. Pennington et al. (2014) implemented Global Vectors (GloVe), which gathers information from the entire training corpus in the form of co-occurrences of words. GloVe recruits the

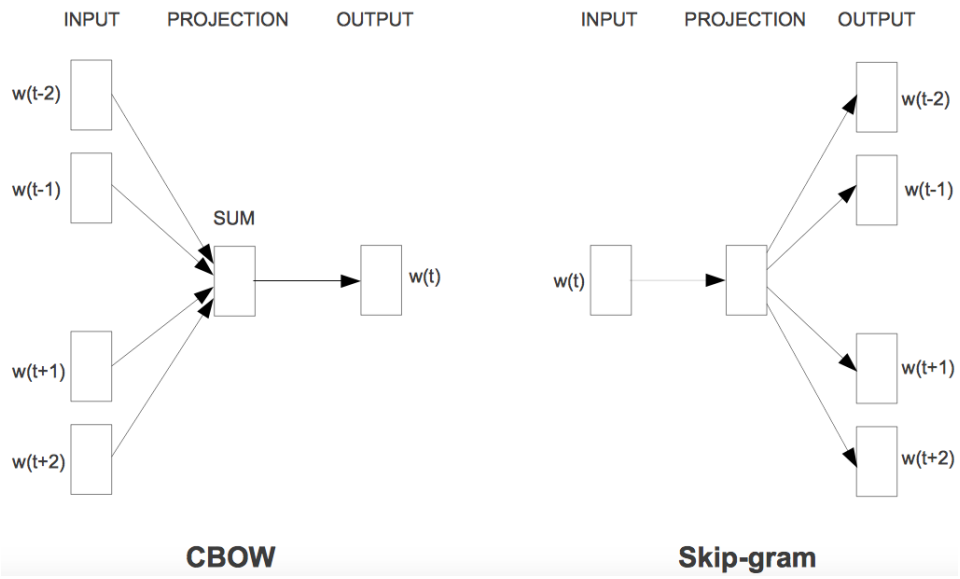

Figure 2.1: The CBOW and Skip-gram model (Mikolov et al., 2013). w_t is the target word at index t.

knowledge of the co-occurrences of words and their distance from the target word. They train a weighted least square model on the co-occurrence matrix to make similar words, according to their co-occurrence values, be similar. Formally, they use the following novel least squares regression model:

$$J = \sum_{i,j=1}^{v} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b_j} - logX_{ij})^2$$

where X_{ij} is the number of times that word i and j occur together. w_i is the vector for word i and $\tilde{w}_j$ is the vector for context word j, b_i and $\tilde{b_j}$ are

biases. $f(X_{ij})$ is a weighting function and is defined below.

$$f(x) = \begin{cases} \frac{x}{x_{max}}, & \text{if } x < x_{max} \\ 1, & \text{otherwise} \end{cases}$$

They use global matrix factorization on their word-word co-occurrences matrix to reduce the dimensionality of the vectors with the intent to maintain useful information. Pennington et al. (2014) achieved state-of-art results on the dataset for word analogy (Mikolov et al., 2013), five different datasets for word similarity (Finkelstein et al., 2001; Miller and Charles, 1991; Rubenstein and Goodenough, 1965; Huang et al., 2012; Luong et al., 2013) and two of three datasets, for named entity recognition (Tjong Kim Sang and De Meulder, 2003). Word similarity is the task of evaluating how similar two words are in context (Miller and Charles, 1991) and out of context (Finkelstein et al., 2001). Named entity recognition is the task of identifying named entities, such as the name of a town or a restaurant, as one of a set of predefined categories such as *place* or *restaurant*.

Both Word2vec and GloVe learn only one embedding for each word, and therefore treat every instance of the same word as the same sense. For example, the word *run* in the sentences *run the program* and *run across town* will have the same word embedding in both cases. Neelakantan et al. (2015) overcome this issue by allowing multiple embeddings to be learned for each word. They implemented Multi Sense Skip-gram (MSSG), a word embedding model that also performs word sense disambiguation during training. The model is similar to Skip-gram in terms of predicting the context words,

but before that stage is reached, the sense of the target word is first selected. MSSG takes the context words' vectors (where each context word now has multiple vectors representing it) and averages them to form a context vector. If the target word has never been seen, then a new sense cluster and sense vector is created; otherwise the similarity between the context vector and each sense cluster of the target word is calculated. The sense of the target word is then chosen by finding the most similar context cluster centroid only if the similarity is greater than some predefined threshold. If the similarity is less than the predefined threshold α, a new sense cluster and sense vector is formed. This allows the model to create the number of senses for each word that it deems are present, and not use a fixed number of senses, although the number of senses can vary depending on the selected similarity threshold α. There is also a variant of this embedding model where the embedding model is given a predetermined number of senses to learn. The centroid of a context cluster is the average of all context vectors that belong to that cluster. Each context cluster is assigned to a sense vector This sense vector is then used to try to predict all the context words of the given target word.

2.1.2 Context and Sentence Embeddings

Word embedding models cannot embed a given instance of a word based on its sequential context. Therefore these models—MSSG, Word2vec, and GloVe—do not preserve the order of the context words. In this section we consider models that take into account sequential context information to embed word contexts and sentences. Melamud et al. (2016) proposed and implemented Context2vec which overcomes this issue of preserving the order

of context words and embeds the context of a target word at run time and places both word types and contexts in the same vector space (Figure 2.2). Looking at Figure 2.3, it embeds the context [*John, a, paper*] by feeding in one word at a time into a bidirectional long short term memory neural network (LSTM), (blue and green ovals). Similar to CBOW, the target word is trying to be predicted from its given context. This bidirectional LSTM is split into two parts, where the forward LSTM (blue ovals) is used to embed the tokens that occur before the target word *submitted* in the example sentence. The reverse part is similar but the tokens after the target word are fed in reverse order starting at the end of the given sentence [*paper, a*]. The model is trained by feeding in the context to each LSTM part and concatenating the parts as shown in the blue and green combined rectangle. Once the concatenation of the two LSTM parts is formed, the concatenation is fed to a multi layer perceptron (MLP) which generates the embedding for the given context (purple rectangle). The model then uses the context embedding to try to predict the target word's embedding (orange rectangle), and updates the weights of the model if needed.

Melamud et al. (2016) achieve state-of-art results over the previous Mikolov et al. (2013) model on the Microsoft sentence completion challenge. With regards to lexical substitution, the task of choosing an appropriate replacement word in context, they achieved state-of-art results on the LST-07 dataset (McCarthy and Navigli, 2007a) and nearly state-of-the-art results on the LST-14 dataset from Kremer et al. (2014). The final task that they evaluated on was supervised word sense disambiguation on the dataset used in Mihalcea et al. (2004), where they achieved comparable results to the

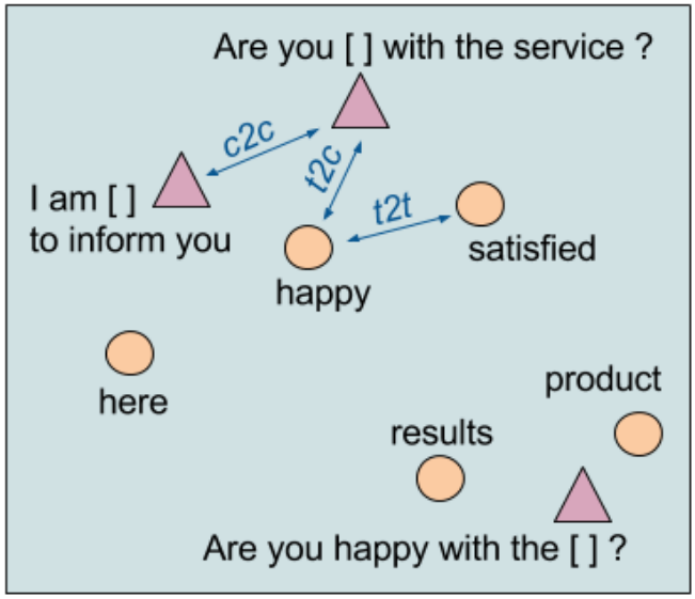

Figure 2.2: A simplified 2D vector space of Context2vec context embeddings and word type embeddings (Melamud et al., 2016).

state-of-the-art.

Instead of limiting the model to only embedding the context of a word like Context2vec, Kiros et al. (2015) proposed and implemented Skip-thoughts, which embeds a full sentence at runtime. This is trained similarly to Skip-gram, but instead of being given a target word and trying to predict the context words, Skip-thoughts is given a target sentence and tries to predict both the sentence that occurs directly before, and the sentence directly after, the target sentence. The model embeds the target sentence by using an encoder-decoder model. The model consists of three neural networks, all of which are recurrent neural networks similar to the type of neural network implemented in Context2vec (Melamud et al., 2016). In the Skip-thoughts model, one network is used for the encoder and one network is used per each

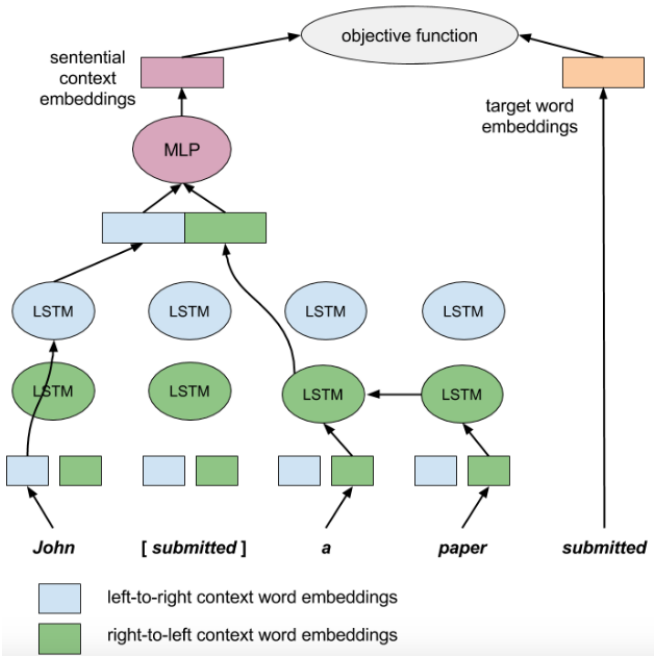

Figure 2.3: Context2vec model (Melamud et al., 2016).

decoder. The encoder neural network is given a sentence s_i as the input, which embeds it into a vector representation. This vector representation is then given to each decoder as input, where one decoder will use the encoding to attempt to build the previous sentence s_{i-1} and the other decoder will attempt to build the next sentence s_{i+1}. The sentence s_i is given to an encoder one word at a time updating the hidden state. Both the encoder and decoders are trained together.

Kiros et al. (2015) evaluate Skip-thoughts on three tasks—semantic relatedness, paraphrase detection, and image-sentence ranking. Semantic relatedness involves comparing how similar two sentences are to each other, paraphrase detection is determining if two sentences are paraphrases, and

image sentence ranking is matching images with their proper description. Kiros et al. (2015) achieved comparable results to the state-of-art results, at the time they evaluated, on the SICK dataset from the SemEval 2014 Task 1: semantic relatedness (Marelli et al., 2014), the Microsoft Paraphrase Detection Corpus (Dolan et al., 2004), and the Microsoft COCO dataset (Lin et al., 2014) for image-sentence ranking.

2.2 Word Sense Disambiguation

Word sense disambiguation (WSD) is an early-defined task of assigning the correct pre-defined sense—a dictionary-like definition representing the meaning of a word in context—from a sense inventory such as WordNet (Kilgarriff and Fellbaum, 2000) to a target lemma's instance. For example, according to Wordnet, the lemma *play* in *We played hockey all afternoon* should be assigned the WordNet sense *participate in games or sport* rather than the *play on an instrument* sense. WSD research dates back to the 1940's for machine translation (Navigli, 2009). There are two versions of WSD; the first is known as *lexical sample*, where a single target lemma is assigned a sense, and the other is *all-words*, where all open class words (nouns, verbs, adverbs, and adjectives) require their senses to be assigned.

There have been a variety of methods applied to WSD. Some methods include the use of a lexical resource. WordNet is a lexical resource in the form of a graph that connects groups of synonyms, called synsets, based on lexical semantic relations. WordNet is widely used in predicting word similarity and relatedness, identifying if a word is somehow related to another, such as

a branch is part of a tree (Pedersen et al., 2004). WordNet also provides a dictionary-style definition used to explain a given word, commonly known as a gloss. The Lesk algorithm (Lesk, 1986) is an unsupervised WSD method that assigns the sense with the most overlapping words between its gloss and the context words of the target lemma as the sense of the target lemma. There are some models that use text on web pages and recruit the use of dependency parsing (Chen et al., 2009). Chen et al. (2009) used text from web pages to find instances of words that are to be disambiguated to gather statistics based on the number of dependencies for each context word and they partially weigh the importance of context words based on their level in a dependency tree when assigning a sense. Some models use machine learning for WSD such as Lapata and Brew (2004), which uses a naive Bayes classifier with features that looked at the collocations and co-occurrences of words in a variety of locations including immediately to the left and right of the target word, all words in the sentence of the target word, and words in the sentence preceding the sentence with the target word. Given a feature set, this supervised method—meaning it requires labeled data for training—trains a naive Bayes classifier to generate the probability that a single sense applies for each sense, and selects the sense that maximizes this probability. In contrast, the unsupervised Lesk algorithm simply selects the sense that contains the most words that also exist in the context of the target word and does not require labeled training data.

Word senses try to describe how a word can be used in context. Corpus pattern analysis (CPA), which uses evidence of corpus patterns from text to show how a specific lemma can be used in text (Hanks, 2004), also strives to

evaluate how a word can be used in context. Corpus patterns are templates that display how words can and have been used with a range of semantic types. For example, the word *eat* could have the corpus pattern *[animal] eat [food]*, where *[animal]* and *[food]* are semantic types and could be any lemma that belongs to the *animal* semantic type and *food* semantic type respectively. Continuing with this example, *the cat ate the fish* would exhibit that corpus pattern. CPA parsing is similar to WSD, but rather than assigning the correct sense to a target word in context from a predefined sense inventory, the goal is to identify the subject and object of the target verb and then assign the correct semantic type from a predefined semantic type inventory (Baisa et al., 2015). CPA parsing has been approached with models that extract features, such as syntactic dependencies, from a training dataset and trained a supervised model using these features (Baisa et al., 2015) similarly to the extraction of features from a training dataset and applying supervised learning used in WSD. CPA clustering is the task of clustering instances based on their exhibited corpus pattern (Baisa et al., 2015).

2.3 Word Sense Induction

WSD utilizes sense inventories, and one way to generate simple sense inventories, although less informative than WordNet, is through word sense induction. Word sense induction (WSI) is a clustering task, where instances of lemmas are clustered based on the sense that they exhibit in their given context. For example, for the lemma *bank*, *fishing at the river bank* and *I*

caught a fish from the bank should be clustered together, while *I deposited money at the bank* should belong to an entirely new cluster.

Navigli and Vannella (2013) discuss task 11 from SemEval 2013,[1] which is a WSI task in which each instance of a lemma belongs to only one sense cluster. Many times however, instances can take on more than one sense (Jurgens and Klapaftis, 2013). Jurgens and Klapaftis (2013) presents results from task 13 of SemEval 2013 where participants are asked to cluster instances of lemmas based on their sense but in this case, a single instance can exist in multiple clusters at once and cluster membership is graded as opposed to being discrete. An example is shown below, which contains an instance from the SemEval 2013 task 13 dataset as well as the annotated WordNet senses for the target lemma *wait* and their applicability rating on a scale from 0–4, where 4 means it is the exact meaning of the target word and 0 means the sense does not apply.

And is now the time to say I can hardly <u>wait</u> for your impending new novel about the Alamo?

sense 1: stay in one place and anticipate or expect something (applicability 4)

sense 2: look forward to the probable occurrence of (applicability 2)

Embedding-based approaches have been applied to WSI that include rep-

[1]SemEval is an organization that releases a variety of NLP tasks, called shared tasks, that research groups can participate on and then be compared to other groups' models.

resenting the context of a word by averaging Skip-gram-style embeddings with the average not being weighted (Song et al., 2016) and being weighted (Kågebäck et al., 2015). Kågebäck et al. (2015) weighted their average by having context words' influence be determined by the distance from the target word and the level of semantic similarity they have with the target word, and clustered using k-means. Song et al. (2016) generate sense clusters in training by trying to predict a context vector given a sense vector. When given instances to cluster, they build context vectors—through averaging embeddings of the context words—and assign the sense cluster with the nearest centroid.

2.4 Usage Similarity

Instances of a lemma in context often exhibit multiple senses with a varying degree of applicability (Jurgens and Klapaftis, 2013), and therefore senses can not be treated as discrete entities. Erk et al. (2009) showed that simply using coarser sense inventories will not aid in the cases where there are multiple senses that are applicable. Kilgarriff (1997) even raised doubts as to the existence of word senses entirely, claiming that they are created by humans and not naturally occurring. WSD attempts to use sense inventories to establish the meaning of words. This reliance on sense inventories puts the WSD model at the mercy of the quality of the sense inventory, which is not easily generated. This means that sense inventories can be quickly outdated, especially in the area of social media where novel, out-of-vocabulary, words are a common occurrence (Baldwin et al., 2013). An example of the sense of

a word changing is present in the case of the word *cell*. It had the common sense of being a room in a prison or dungeon, but now has the common sense of being a cellular phone. Usage similarity (USim) overcomes this issue by measuring the similarity between two usages of the same lemma on a scale from 1 to 5, where 1 is not similar at all and 5 is perfectly similar. For example, the usage of the lemma *bank* in *I deposited money at the bank* is much more similar to *I have a meeting with my financial advisor at the bank* than *I'm fishing off the bank*. In contrast to USim, traditional methods for WSI and WSD treat word senses as discrete entities with no overlap among sense clusters except for the discussed version of Jurgens and Klapaftis (2013).

USim allows word senses to be treated as fluid where degrees of applicability from different senses can exist in a single instance of a lemma without the use of sense inventories. Erk et al. (2009) conducted an experiment that compared word sense similarity to usage similarity. Word sense similarity is the task of rating the applicability of senses for a target word. For example, the target lemma *strong* in the phrase *The coffee is strong* could exhibit the WordNet sense *having strength or power greater than average or expected*, as well as the WordNet sense *having a strong physiological or chemical effect*, each to a different degree. This experiment was performed with humans performing each task through an online interface. It was concluded that word sense similarity and usage similarity have a strong correlation with each other (Erk et al., 2009). This means that we can determine the similarity of word senses through obtaining the similarity of usages and therefore determine the meaning of a word in context without needing a sense inventory.

Topic modelling has been applied to the two USim datasets that we evaluate our models on. Topic modelling takes into account the statistics of words for entire documents (Blei et al., 2003). It uses statistics of words to generate the distribution of words within a topic as well as the distribution of topics within a document for each document in a corpus. A common form of topic modelling is the latent Dirichlet allocation (LDA, Blei et al., 2003). A topic is a probability distribution over words. For example, a topic containing 0.4 *peach*, 0.1 *hamburger* and 0.5 *dinner* could be representing a topic similar to *food*. The model assumes that a single document is created by choosing a distribution of topics and then a distribution of words within those topics. Topic modelling has been used in information retrieval (Wei and Croft, 2006) and usage similarity (Lui et al., 2012) and (Gella et al., 2013).

Lui et al. (2012) applied topic modelling to USim as the first attempt for this task. Given a pair of usages, they used Latent Dirichlet Allocation (Blei et al., 2003) to map each word in the context of the target word in each usage to a latent topic, which allows them to represent each usage as a distribution over topics. They compared training a model for all target lemmas and training a different model for each target lemma and found that their general model outperformed their model that was trained for a specific lemma. They tried three different training corpora for their topic model including the sentences from the testing dataset, 340 pages from English Internet Corpus which included the sentences in the testing dataset, and the full English Internet Corpus. Topic modelling uses a parameter that controls the number of topics that are used. The best results were achieved using 340 pages from English Internet Corpus and 8 topics. They used both

cosine similarity and Jensen-Shannon divergence to measure the similarity between the vector representations of the given usages, which were formed using topic modelling, and discovered that using cosine similarity produced better results.

Similarly, Gella et al. (2013) applied Latent Dirichlet Allocation to the USim task on their novel dataset, which was constructed from actual tweets that will be discussed later in Section 4.1. They experimented with a range of number of topics, from 2 to 500. They also recruited the use of three different training corpora: ORIG, which contains 1 million tweets which contain one of the target words; ORIG with 40k random tweets that contain a target lemma; and ORIG with an additional 40k tweets that contain hashtags that were present in ORIG within a frequency threshold, which achieved the best results. They too applied cosine similarity to the vector representations of the given sentences. Both Blei et al. (2003) and Gella et al. (2013) use unsupervised methods for predicting USim.

2.5 Summary

In this chapter, we looked at different embedding models and their method for generating embeddings. The models that we looked at include Word2vec, GloVe, MSSG, Skip-thoughts, and Context2vec. Previously implemented work shows that embedding-based models outperform traditional count-based models, leading to our decision to use embedding-based models for our tasks. We explained the tasks that we will be working with, which are usage similarity and word sense induction. We described word sense disam-

biguation, a precursor to usage similarity, and explained that the advantage of usage similarity over word sense disambiguation is that it does not require a predefined sense inventory.

Chapter 3

USim Models

Our methods that were applied to USim measure the similarity between two given usages of the same lemma. Each usage is represented in the form of a dense vector that is generated through the use of embeddings from either individual words, the full given sentence, or the context, in which the usage occurs. These embeddings are generated through training an embedding model on a corpus of text to capture semantic information in a dense vector representation.

3.1 Usage Representations

In the following subsections we present how each model is used to generate a vector representation of a given usage. We explore the use of Word2vec, GloVe, Skip-thoughts, Context2vec, and MSSG.

3.1.1 Word2vec

The first model that we recruited and applied is Word2vec. Using Word2vec embeddings, we sum up the embeddings for all context words in the given sentence except for frequent words known as stop words based on a predefined stop word list.[1] Averaging Word2vec embeddings is a common baseline for tasks where generating a vector representation of a sentence is required (Jabri et al., 2016). Stop words are frequent words in a language that often give little semantic information because of their frequent appearance and often with semantically different tokens. For example, the stop words *the* and *a* both co-occur with *cat, house, planet,* and *flower* so by looking at the words *the* or *a*, we can not accurately predict the word that will follow them. Stop words are typically function words such as prepositions, pronouns, articles, and conjunctions. The contrast to function words are lexical words—nouns, verbs, adjectives, and adverbs—which are the basic building blocks of a language and serve the primary role of conveying meaning. The vector sum of the context words is then normalized, which is our final vector representation of a single usage.

3.1.2 GloVe

GloVe embeddings capture global semantic information in training by observing co-occurrences. This global information of co-occurrences might help capture the usage of a target lemma since the GloVe embedding model maps words with similar co-occurrences near each other in its embedding space.

[1] `http://www.lextek.com/manuals/onix/stopwords1.html`

Summing up the embeddings for words in the context of the target, excluding stop words, and normalizing the vectors gives a representation of the usage of the target lemma. Therefore, the GloVe embedding model is used in the same manner as the Word2vec embedding model, because it provides individual word embeddings like the Word2vec embedding model.

3.1.3 MSSG

We now discuss the MSSG embedding model, which is similar to the Skip-gram version of Word2vec, but contains multiple sense vectors for a single lemma. Using multiple vectors per word provides the ability to select different senses of a word to capture a word's meaning in context. We again sum up the global embeddings of each context word. We then calculate the similarity between this context summation and all sense clusters' centroids of the target word. Each of these sense clusters is a cluster of different senses of a given word, all of which are represented as a vectors. We use the target word and not the target lemma because the model generates sense clusters for each word such as *playing* and *play*, which have different sense clusters. The sense vector associated with the nearest target lemma's sense cluster's centroid is then used as the representation for this usage of the target lemma.

3.1.4 Skip-thoughts

The use of the Skip-thoughts embedding model moves away from the idea of summing up the individual word embeddings, but rather embeds the entire sentence by passing the sentence containing the target lemma into the

already trained neural network, known as the encoder, to generate a vector representation of the sentence. The idea behind the use of an embedding of a sentence is that through embedding the sentence we also embed the context that is within the sentence and therefore embed the usage of the target word. Previously, the Skip-thoughts embedding model was used in a similar way for an idiom classification task (Salton et al., 2016).

3.1.5 Context2vec

The use of the Context2vec embedding model follows the style of the Skip-thoughts model in terms of embedding the given sentence using an already trained neural network. The difference, besides the design of the neural network, is that the Context2vec embedding model embeds the context words of the target lemma only and does not embed the entire sentence. The embedding of the context words is motivated by the idea that a word's meaning is indicated in its context. The Context2vec embedding model maps similar words and contexts near each other in the same embedding space and it is the embedding of the target word usage's context that we use in our model.

3.2 Unsupervised Model

To determine the similarity in the unsupervised setup, where we do not see any instances of the dataset before testing, cosine similarity is calculated between the two vector representations that are generated for the two sentences containing a target lemma, for each instance in the dataset. Lui et al. (2012) and Gella et al. (2013) also calculated cosine similarity on their

vector representations of usages to predict usage similarity. This results in a similarity score in the range from 0–1, where 0 indicates the vectors are orthogonal and therefore have nothing in common in terms of the meaning of the sentences they represent. A cosine similarity of 1 indicates the same vector, meaning the same sentence or usage. This scale is then translated to a range from 1–5 to match the range of the labels in the gold standard datasets (described in Section 4.1).

3.3 Supervised Model

Given the ability to train a supervised model, where we observe a portion of the dataset, we evaluate if it is worth acquiring labeled data to assist with the USim task. Given each selected pair p_i of usages, $u1_i$ and $u2_i$, where i is the identification of one selected pair and $u1_i$ and $u2_i$ are the vector representations of the usages in p_i obtained from one of the embedding models described above, we create a feature vector c_i for each given pair of usages p_i by concatenating the component-wise multiplication and the absolute difference of $u1_i$ and $u2_i$ which were the features used in Kiros et al. (2015) for semantic relatedness. This then becomes a regression problem to map the generated feature set c_i to a similarity rating. Ridge regression is quick to train and has been previously applied to a sentence similarity task (King et al., 2016), which is a similar task to USim. Ridge regression could be easily replaced with a stronger regression algorithm such as support vector regression for future work. The first setup discussed, which we will call the supervised *All* setup, uses 10 fold cross-validation for training and

testing of the ridge regressor. In 10 fold cross-validation, we evenly and randomly divide the dataset into 10 folds (sets of instances), then we train our regressor on 9 of the 10 folds and then make our predictions on the remaining fold. We perform this evaluation 10 times so that each fold is the testing fold exactly once, resulting in a list of predicted similarities for each instance from the dataset.

We also evaluate our supervised method's ability to perform on unseen target lemmas, where we follow the same method as the supervised *All* setup up to the point of splitting up the data for the 10 fold cross-validation. Instead, we split our dataset based on the target lemma so that we train our ridge regressor on all instances that do not involve a selected target lemma and then test on the instances that do contain the selected target lemma. We perform this training and testing until each lemma is selected as the test target lemma once. Formally, suppose that we have four different target lemmas. We divide our dataset into four sets $\{a, b, c, d\}$ so that all target lemmas within one set are the same. We then train our ridge regressor on three of the four sets such as $\{b, c, d\}$ and test on a. We then train another ridge regressor on $\{a, c, d\}$ and test on b. We will continue this process until all sets in $\{a, b, c, d\}$ have been the testing set exactly once.

We evaluated our supervised methods further by using only one feature set at a time to determine the quality of information that each feature set contains when being applied to USim. This means that we train and test the ridge regressor exclusively on the component-wise multiplication features or the absolute difference features. This setup only provides half as many features for the regressor.

3.4 Summary

In this chapter we described how we used each embedding model including Word2vec, GloVe, MSSG, Skip-thoughts, and Context2vec to represent a single usage of a word. We then explained how we used these usage representations in both an unsupervised and supervised setup. The use of absolute difference and product of two usage representations as our features for the supervised setup are also explained in detail.

Chapter 4

Materials and Methods

In this chapter we will discuss the datasets and training corpora used for USim along with the methods used to predict USim, including a detailed report of the training of Word2vec embeddings and how we apply each embedding model to the two USim datasets. We also discuss the metric used for evaluating our models.

4.1 Datasets

We evaluated our methods on two different datasets. The dataset from Erk et al. (2009), ORIGINAL, contains pairs of sentences with each pair sharing the same target lemma. The sentences are from *LEXSUB* (McCarthy and Navigli, 2007b), which is a dataset for lexical substitution evaluation. The sentences were originally found in a corpus of English documents built from the web (Sharoff, 2006). ORIGINAL contains 1512 pairs of usages divided over 34 target lemmas including nouns, verbs, adjectives, and adverbs. Each

target lemma has 10 sentences and each pairwise combination of sentences for a single target lemma is generated for the dataset, totaling 45 pairs of usages per lemma. This originally gave 1530 usage pairs, but due to annotators not being able to rate a select few instances, 18 were removed from the dataset, finalizing the dataset with 1512 usage pairs. The distribution of this dataset is 10 nouns, 9 verbs, 12 adjectives, and 3 adverbs.

The annotation process for ORIGINAL was performed online by three annotators, all of whom were native British English speakers. The annotators were given two sentences with the same target word and were asked to rate the similarity of the usage of the target word on a scale from 1–5. Specifically the annotators were asked the following: *"Your task is to rate, for each pair of sentences, how similar in meaning the two boldfaced words are on a five-point scale"*(Erk et al., 2009). If the annotator could not rate the given pair of sentences, they were provided with an option *"Cannot Decide"*. The average Spearman's rank correlation for pairs of annotators was 0.548 with a significance value less than $2.2 * 10^{-16}$, meaning that this task is difficult even for humans.

The dataset from Gella et al. (2013), TWITTER, contains 550 usage pairs divided over 10 target lemmas, equally splitting 55 usage pairs per target lemma. Each sentence in a usage pair is a real tweet from the TREC 2011 microblog dataset.[1] All the target lemmas are nouns and each tweet is randomly chosen given that it contains at least four words that have the part of speech of either noun, verb, adjective, or adverb.

[1] `http://trec.nist.gov/data/tweets/`

The annotations were performed using Amazon Mechanical Turk,[2] a crowd sourcing platform with outliers removed post hoc. They created Mechanical Turk HITs (Human Intelligence Tasks), where each HIT contained 5 sentence pairs. Each job was annotated by 10 annotators. There were 68 annotators that took part, all of whom were based in United States of America and have had at least 95% of their previous annotations accepted on Amazon Mechanical Turk. The inter-annotator agreement for this dataset was 0.681, which again shows that this USim task is not simple.

4.2 Evaluation Metrics

The Spearman's rank correlation, also called Spearman's rho, was used to evaluate our method, which measures the similarity of two lists on a scale of -1 (perfectly negative correlation) to +1 (perfect correlation). 0 means there is no correlation. This is the same metric used in both Erk et al. (2009) and Gella et al. (2013) and therefore, we can compare our results directly with theirs. Spearman's rank correlation takes into account only the ranking in each list, so in our case it is the ranking of the similarity scores present in our predictions compared against the ranking of the similarity scores in the gold standard. The Spearman's rank correlation is calculated by

$$r = \frac{cov(rg_x, rg_y)}{\sigma_x \sigma_y},$$

where rg_x is the list of ranks of our predicted scores, rg_y is the list of ranks of the gold standard, and σ_x and σ_y are their respective standard

[2] `https://www.mturk.com/mturk/welcome`

deviations. *cov* is the covariance. The Spearman's rank correlation is also associated with a measurement of the significance of the correlation known as the p-value. It is important to note that under all setups, Spearman's rank correlation was calculated for lists containing scores for all instances from a dataset. We perform this for our supervised setups by storing all our similarity predictions in a list as we perform the 10 fold cross-validation for the supervised *All* setup, and when we perform our testing on the supervised *Lemma* setup where we separate the training and testing based on the target lemma.

4.3 Embedding Details

Following Pennington et al. (2014), who trained GloVe on a dump of English Wikipedia, an easily accessible source for building corpora that cover a wide range of topics, we trained Word2vec embeddings on a dump of English Wikipedia from September 1, 2015, WIKI, which contains 2.6 billion tokens.[3] Unlike ORIGINAL, the TWITTER dataset contains text from Twitter. Text from Twitter is noisy, which adds an additional complexity to the dataset. Noisy text from Twitter can contain incorrectly spelled words, novel words, and novel senses. These issues appear much more frequently in Twitter than in more-conventional text such as text from Wikipedia. Misspelled words, intentional or unintentional, such as *timeeeeee* will not be in an embedding model's vocabulary and therefore will not be able to be used as a context word when building a usage representation. Novel words can appear in text

[3]A dump of Wikipedia is a file that contains all text on Wikipedia at a particular instance in time.

after an embedding model is trained and therefore the embedding model would not have had the chance to learn and embed the meaning of the new word. This occurs often when new products are released such as the name of video game consoles or cellular phones. Novel senses may have been created for a word after the embeddings have been trained and therefore the embedding model would not have been trained on the novel sense and would not be able to represent its new meaning, but rather it would be representing the other senses of the word which could be entirely different. A recent example is the new sense that the word *lit* takes on, meaning *excellent* or *exciting*. This new sense of the word was uncommon around a decade ago.[4] To help overcome these issues, we used models that were trained on a corpus that contains text from Twitter, when possible. Word2vec embeddings were therefore trained on tweets that were gathered from the Twitter API from November 2014 to March 2015 containing 1.3 billion tokens, TwitterCorpus1.3.

We trained multiple Word2vec embedding models using the Skip-gram training model on wiki and TwitterCorpus1.3 with a variety of parameters.[5] The parameters that we changed when training Word2vec embeddings include the dimension of the resulting embeddings and the training window, which is the maximum distance from the target word that the model looks at as the context. For example, if the training window was 2 for the target word *bank* in the phrase *I went to the bank to deposit money*, then the context that is used would be *[to,the,to,deposit]*. Smaller windows seem to capture more

[4] `https://www.merriam-webster.com/words-at-play/lit-meaning-origin`

[5] We trained CBOW embeddings as well, but preliminary experiments showed that Skip-gram embeddings outperform CBOW embeddings on USim.

syntactic information, while larger windows capture more semantic relations (Levy and Goldberg, 2014). We trained Word2vec embeddings using window sizes of either 2, 5, or 8 and generated embeddings with either 50, 100, or 300 dimensions. Before training on WIKI, we tokenized the corpus using the Stanford tokenizer.[6] We used a tokenizer on TWITTERCORPUS1.3 that was used in Hallsmar and Palm (2016) and is specifically designed to tokenize text from Twitter before training on it. This tokenizer is a rule-based tokenizer that preserves hashtags, URLS, emojis , abbreviations, dates, emails, dollar signs and percentages. We also ignore all words that occur in the training corpus less than 15 times. There are 5 words randomly chosen for negative sampling. Negative sampling is the process of randomly choosing incorrect output words of the neural network to allow the weights to update faster in learning the actual desired output and get further away from learning the incorrect words. This method is used to speed up the training procedure. The training method iterates over the corpus 5 times and had a starting learning rate of 0.025.

We trained our own Word2vec embeddings, but used publicly available embeddings for all of the following embedding models: GloVe, Skip-thoughts, and Context2vec. We used GloVe embeddings that were trained on a 6B token dump of Wikipedia from 2014 with text from the fifth edition of Gigaword, a corpus of newswire text (GLOVEWIKI), and a 27B token corpus consisting of actual tweets from Twitter (GLOVETWITTER). The GLOVEWIKI embeddings had a vocabulary of 400k lemmas and were composed of 300 dimensions, which are bigger than the GLOVETWITTER embeddings with

[6] `http://nlp.stanford.edu/software/tokenizer.shtml#Obtaining`

200 dimension and a vocabulary of 1.2M lemmas. Both GloVe embeddings, GLOVEWIKI and GLOVETWITTER, contain a number of embeddings that are within our range of dimensions for our trained Word2vec embeddings.

Our pre-trained MSSG embeddings were trained on the April 2010 Wikipedia snapshot. After experimenting with MSSG embeddings with different dimensions and number of senses per word, we found that the embeddings with 300 dimensions with 3 senses per word performed the best out of the different MSSG embeddings and therefore used these embeddings for all experiments involving the MSSG embedding model. The number of dimensions for this set of MSSG embeddings is the same number of dimensions in the Word2vec embeddings and the GLOVEWIKI embeddings. This allows for a direct comparison.

The number of dimensions is much larger with our pre-trained Skip-thoughts embedding model. It generates embeddings consisting of 4800 dimensions. The model has a vocabulary of approximately 931k lemmas and is trained on the BookCorpus (Zhu et al., 2015) consisting of approximately 984M words collected from approximately 11k books.

Our pre-trained Context2vec embedding model was trained on the ukWaC and generates embeddings with 600 dimensions. ukWaC is a corpus containing approximately 2 billion tokens and is comprised of a collection of text from a web crawler on .uk URLs (Ferraresi et al., 2008). We also train a Word2vec Skip-gram model on the ukWaC with 600 dimensions for a direct comparison to the pre-trained Context2vec embedding model.

Chapter 5

Results

In this chapter we will show our results for our unsupervised setup and both supervised setups. We then dig deeper into the results in these setups, observing the performance on individual parts-of-speech as well as the effects of the training parameters—training window and number of dimensions. In the supervised setups, we also look at the results in feature ablation experiments.

5.1 Unsupervised

We first look at the effects of applying Word2vec embeddings to the unsupervised setup, using different values for the two training parameters, number of dimensions and window size. The Word2vec embeddings applied to ORIGINAL were trained on a Wikipedia dump and the Word2vec embeddings that were applied to TWITTER were trained on text from Twitter. We observe the performance of both the CBOW and Skip-gram embedding models in

D	W	Original		Twitter	
		Skip-gram	CBOW	Skip-gram	CBOW
50	2	0.251	0.172	0.246	0.135
50	5	0.262	0.198	0.272	0.144
50	8	**0.286**	0.197	0.282	0.140
100	2	0.267	0.186	0.248	0.149
100	5	0.273	0.213	0.253	**0.230**
100	8	0.273	0.214	0.298	0.148
300	2	0.275	0.200	0.266	0.173
300	5	0.279	0.221	0.295	0.212
300	8	0.281	**0.225**	**0.300**	0.154

Table 5.1: Spearman's ρ on each dataset using the unsupervised approach with the Word2vec embeddings trained using the Skip-gram and CBOW embedding model under several settings for the number of dimensions (D) and $\pm$ window size (W) on their respective corpus. The best ρ for each embedding model on each dataset is shown in boldface. All values are significant with a p-value < 0.05.

Table 5.1. We immediately see that the Skip-gram embedding model outperforms the CBOW embedding model—meaning for every combination of parameter setting, on each dataset, the Skipgram embedding model outperformed the CBOW embedding model. For this reason, from here on, the Word2vec embeddings used on the different setups will have been generated using the Skip-gram embedding model.

Table 5.1 displays the Spearman's ρ for the different training parameters for the CBOW and Skip-gram embedding models used on both Original and Twitter. Looking only at Skip-gram, for both Original and Twitter, given the same number of dimensions, the embedding model always performs better by increasing the training window size. For example, an embedding trained on a window size of ± 8 with 300 dimensions outperforms an em-

bedding trained on a window size of ±5 and with 300 dimensions. Also, given the same training window size, the embedding model performs better when increasing the number of dimensions except for one case for ORIGINAL where the training window size is 8 and the number of dimensions is 50.

The Skip-gram embedding with 50 dimensions and a training window of 8 achieves the best performance, out of the different Word2vec embedding models, of 0.286 on ORIGINAL. The best performance of 0.300 on TWITTER was achieved by a Skip-gram embedding model that had a training window of 8 and 300 dimensions. Both of these results have a higher correlation than the previous state-of-the-art on ORIGINAL and TWITTER, which was 0.202 and 0.29 respectively (Lui et al., 2012; Gella et al., 2013). We beat the previous state-of-the-art on ORIGINAL with the Skip-gram embedding model using any of the training parameters tested. For TWITTER, we beat the previous state-of-the-art with three different combinations of the parameter values for the Skip-gram model. These results show that choosing any reasonable parameters for the Skip-gram embedding model, such as a training window of ±8 and 300 dimensions, will result in a relatively strong performance. Both models that previously held the state-of-the-art values were also unsupervised models (Lui et al., 2012; Gella et al., 2013).

There were four different sets of pre-trained MSSG embeddings that we experimented with in order to decide which MSSG embeddings will achieve the highest performance on our later evaluations. We evaluate the different MSSG embeddings on ORIGINAL because the MSSG embedding models were trained on a snapshot of Wikipedia which contains text that is more similar to the text found in ORIGINAL and we expect the embedding models

Model	Unsupervised	Sup (All)	Sup (Lemma)
50D-3S	0.123	0.349	0.176
300D-3S	**0.203**	**0.454**	**0.205**
50D-10S	0.150	0.358	0.120
300D-10S	0.200	0.428	0.093

Table 5.2: Applying MSSG to the ORIGINAL dataset under all three setups. D = number of dimensions. S = number of sense clusters generated for each word. All correlations are significant with a p-value < 0.05.

Dataset	Embeddings	Spearman's ρ
ORIGINAL	Word2vec	0.281
	GloVe	0.218
	Skip-thoughts	0.177
	Context2vec	**0.302**
	MSSG	0.203
TWITTER	Word2vec	**0.300**
	GLOVE	0.122
	Skip-thoughts	0.095
	Context2vec	0.122
	MSSG	0.271

Table 5.3: Spearman's ρ on each dataset using the unsupervised method for each embedding approach. The best ρ for each dataset is shown in boldface. All correlations are significant (p-value< 0.05).

to perform better on ORIGINAL than TWITTER. Table 5.2 displays the performance of the different sets of embeddings. Two embedding dimensions, 50 and 300, and two different numbers of senses, 3 and 10, were used as the training parameters for the MSSG models. The number of senses is the number of sense clusters that are generated for each word. The embeddings with 300 dimensions and 3 sense clusters per word clearly outperformed the other sets of embeddings under all three setups. We will use this set of embeddings, *300D-3S*, from here on when referring to embeddings generated from MSSG.

Having determined the best parameter settings for both Word2vec and MSSG, we will now move on to comparing our models under the unsupervised setup. The Word2vec embeddings that were applied to ORIGINAL were trained on a Wikipedia dump and the Word2vec embeddings that were applied to TWITTER were trained on text from Twitter. The Word2vec embeddings were trained on a window size of 8 and with 300 dimensions based on the findings in Table 5.1. The pre-trained GloVe embeddings that were applied to ORIGINAL and TWITTER were trained on either a dump of Wikipedia or on text from Twitter respectively. The other embeddings were only trained on one corpus regardless of which dataset they were applied to. Skip-thoughts was trained on BookCorpus and Context2vec was trained on ukWaC. First looking at the unsupervised setups applied to ORIGINAL—shown in Table 5.3—Word2vec, GloVe, Context2vec and MSSG beat the previous state-of-the-art of 0.202. Skip-thoughts was the only embedding that does not beat the previous state-of-the-art. This might be showing that embedding the full sentence does not provide a meaningful representation of a specific word in context. Context2vec achieved the new state-of-the-art with a correlation of 0.302.

In the unsupervised setup for TWITTER, the Word2vec embeddings were the only embeddings that beat the previous state-of-the-art of 0.29 with the new state-of-the-art value of 0.300. Context2vec's and Skip-thought's poor performance could be credited to their training corpus, which was not a corpus of text from Twitter. GloVe, however, was trained on a corpus of text from Twitter and still performed poorly. All results for the unsupervised setup, for both datasets, were deemed significant, p-value< 0.05.

Dataset	Embeddings	Spearman's ρ
ORIGINAL	Word2vec	0.435
	GLOVE	0.410
	Skip-thoughts	0.436
	Context2vec	0.417
	MSSG	**0.454**
TWITTER	Word2vec	0.384
	GLOVE	0.314
	Skip-thoughts	0.360
	Context2vec	0.193
	MSSG	**0.447**

Table 5.4: Spearman's ρ on each dataset using the supervised *All* method with 10 fold cross-validation with folds based on random sampling across all lemmas for each embedding approach. The best ρ for each dataset is shown in boldface. All correlations are significant (p-value$<$ 0.05).

Our unsupervised model achieved the best performance using Context2vec and Word2vec embeddings on ORIGINAL and TWITTER respectively. The Word2vec embedding model performed consistently well on both datasets—achieving the new state-of-the-art results on TWITTER and beating the former state-of-the-art results on ORIGINAL. Context2vec achieved the new state-of-the-art results on ORIGINAL but sadly performed poorly on TWITTER. This was possibly due to the type of training data and therefore suggests we could consider further evaluations with different training data in our future work.

5.2 Supervised *All*

Taking advantage of labeled data, we look at the performance of the embedding models in the supervised *All* setup using 10 fold cross-validation as seen in Table 5.4. We use cross-validation because of the relatively small

size of the datasets. Under the supervised *All* setup, we train on 9 of the 10 randomly generated evenly distributed folds and then make our predictions on the remaining fold. We perform this evaluation 10 times so that each fold is the test fold exactly once. The regression model has the chance of seeing the same target lemma in both training and testing under this setup. The predictions from each fold are stored in one list, which we use as our final predictions to compare against the gold standard. For ORIGINAL, using the supervised *All* setup, each embedding model achieved a higher score than its own score under the unsupervised setup (shown in Table 5.3). Also, each score under the supervised *All* setup beat the highest score under the unsupervised setup. All scores for the supervised *All* setup are fairly similar—ranging from $0.410 - 0.454$, as opposed to the larger range of $0.095 - 0.302$ in the unsupervised setup. The MSSG embedding model achieves the highest correlation of 0.454 on ORIGINAL.

It is worth noting the number of dimensions that the different embedding models contain because increasing the number of dimensions gives the regression model more features to train on and potentially allows the embedding models with more dimensions to have a better performance. The Skip-thoughts embeddings contain 4800 dimensions, which is 8 times more dimensions than the next largest set of embeddings, which is Context2vec containing 600 dimensions. Word2vec and MSSG embeddings only use 300 dimensions—1/16 the number of Skip-thoughts—and still perform within 0.001 of Skip-thoughts, and outperform Skip-thoughts, respectively. When training the regressor, the number of features used is double the dimension of the embedding model that is being recruited. In the unsupervised

setup, Word2vec performs worse than Context2vec, but Word2vec outperforms Context2vec in the supervised *All* setup.

Looking at the supervised *All* setup for TWITTER (Table 5.4, bottom panel), we see that the correlation of each model in the supervised *All* setup is higher than in the unsupervised setup, but unlike ORIGINAL, not all correlations for the supervised *All* setup—Context2vec's—achieve higher correlation than all models in the unsupervised setup. Skip-thoughts, Context2vec, and MSSG are not trained on a corpus made from tweets and therefore have a disadvantage, which could explain Context2vec's poor performance, although the MSSG embedding model achieved the best performance of 0.447, beating all embedding models by at least 0.063. MSSG's performance might be attributed to how the usage representations are formed, which is similar to Word2vec—which achieves the second highest value and almost the second highest value on TWITTER and ORIGINAL respectively. Skip-thoughts' large number of dimensions could be helping it overcome this issue of not training on a corpus containing text from Twitter. Again, the Word2vec word embedding outperformed the GloVe word embeddings, which might show that the method used to form Word2vec embeddings is simply superior to the method used to form GloVe embeddings for representing a word in context via averaging word embeddings. All results show that if labeled data is available, then it should be used to strengthen a model applied to USim.

Dataset	Embeddings	Spearman's ρ
ORIGINAL	Word2vec	0.220*
	GLOVE	**0.230***
	Skip-thoughts	0.099*
	Context2vec	0.172*
	MSSG	0.205*
TWITTER	Word2vec	0.196*
	GLOVE	0.134*
	Skip-thoughts	0.058
	Context2vec	0.067
	MSSG	**0.271***

Table 5.5: Spearman's ρ on each dataset using the supervised *Lemma* setup for each embedding approach. For each dataset, the best ρ for each dataset is shown in boldface. Significant correlations (p-value$<$ 0.05) are indicated with *.

5.3 Supervised (Lemma)

The supervised *All* setup allows for the regressor to see target lemmas that it will test on, giving it an advantage. We remove this advantage in the supervised *Lemma* setup by training our regressor on instances that do not include a selected target lemma, and then by testing on all instances that do contain the selected target lemma. Results for the supervised *Lemma* setup for both datasets are shown in Table 5.5. Applied to ORIGINAL, each embedding model performs worse than its unsupervised correlation except for GloVe and MSSG (Table 5.3). There is no clear reason why GloVe and MSSG performed better than their unsupervised correlations, but GloVe's and MSSG's values for supervised *Lemma* are still less than the unsupervised setup for both Word2vec and Context2vec. The poor performance by all embedding models shows that our supervised model performs poorly for unseen target lemmas.

POS	Word2vec	GloVe	Skip-thoughts	Context2vec	MSSG
nouns	0.315*	0.245*	0.240*	0.282*	**0.316***
adjectives	0.249*	0.216*	0.170*	**0.303***	0.138*
adverbs	0.361*	0.000	0.070	**0.408***	0.101
verbs	0.263*	0.260*	0.174*	**0.277***	0.208*
All	0.281*	0.218*	0.177*	**0.302***	0.203*

Table 5.6: Spearman's ρ for the unsupervised setup on ORIGINAL displaying the embedding models' performance across the different parts-of-speech. Correlations in bold represent the highest values for each part-of-speech. * represents significant correlation, p-value < 0.05.

Results from embedding models applied to TWITTER under the supervised *Lemma* setup tell a similar story to the same setup on ORIGINAL. GloVe was the only embedding model to outperform its unsupervised correlation, but MSSG and Word2vec still outperform GloVe under the supervised *Lemma* setup. MSSG was able to tie its unsupervised correlation. The supervised *Lemma* was the only setup that had values that were not deemed significant, meaning their p-value was above 0.05, which was for both Skip-thoughts and Context2vec. Again, the supervised *Lemma* setup on TWITTER backs up the claim that our model does not perform well on unseen target lemmas.

5.4 Part-of-Speech

We look into a more detailed evaluation at the performance of the different embedding models for different parts-of-speech under the three different setups. We perform this form of evaluation only on ORIGINAL because all target lemmas in TWITTER are the same part-of-speech as discussed in Section 4.1. We first look at the unsupervised setup shown in Table 5.6.

We perform the same evaluation as in Section 5.1, except we now separate our model's predictions based on part of speech. We perform four measurements with Spearman's ρ metric, one for each part-of-speech—nouns, verbs, adjectives, and adverbs. Context2vec achieves the highest correlation for all parts-of-speech except nouns, where MSSG achieves the highest values. Word2vec achieves competitive results to MSSG on nouns, which is consistent with our findings for Word2vec in the unsupervised setup where it achieves the best performance of the embedding models being applied to TWITTER, which consists of only nouns. The part-of-speech with the largest range of correlations under this setup is for the adverbs, 0.000–0.408. This range has Word2vec and Context2vec performing on the higher end with GloVe, Skip-thoughts, and MSSG performing on the lower end. Adverbs might be non-consistent in correlations and difficult for some models due to the low number of instances—only 3 out of the 34 target lemmas were adverbs. Adverbs also had the lowest inter-annotator agreement of all the parts-of-speech (Lui et al., 2012). An ensemble model based on part-of-speech for the unsupervised setup, meaning to select the predictions from models based on which part-of-speech that the target instance is, might increase performance slightly here, but not by a large margin, since Context2vec achieves the highest correlation on all parts-of-speech except one. Furthermore, for the one part-of-speech—nouns—where Context2vec does not achieve the highest correlation, MSSG only outperforms Context2vec by 0.034.

We now look at the supervised setups beginning with the *All* setup, displayed in Table 5.7, where we evaluate the embedding models' performance on

POS	Word2vec	GloVe	Skip-thoughts	Context2vec	MSSG
nouns	0.502*	0.450*	0.490*	0.428 *	**0.504***
adjectives	0.394*	**0.434***	0.409*	0.375*	0.409*
adverbs	0.272*	0.136	0.313*	**0.403***	0.093
verbs	0.461*	0.414*	0.443*	**0.450***	0.415*
All	0.435*	0.410*	0.436*	0.417*	**0.454***

Table 5.7: Spearman's ρ for the supervised All setup on ORIGINAL displaying the embedding models' performance across the different parts-of-speech. Correlations in bold represent the highest values for the given part-of-speech. * represents significant correlation, p-value < 0.05.

different parts-of-speech. We train and test our regressor using 10 fold cross-validation in the same fashion as we did for the supervised *All* setup in Section 5.2, except before we calculate the Spearman's ρ we separate our predictions and the gold standard values, preserving the order, based on the part-of-speech of the target lemma. We do not separate our folds based on the part-of-speech of the target lemma before training, so we will see the same parts-of-speech in the training and testing folds.

We see that there is much more variation in terms of which embedding model does best on each part-of-speech. The results for a single part-of-speech are fairly similar across all embedding models, except for *adverbs*, where Context2vec clearly does better than all other embedding models.

The results for the supervised *Lemma* setup, shown in Table 5.8, do not show the same characteristic as the supervised *All* setup. For example, Context2vec achieved the highest correlation on verbs in the supervised *All* setup, but achieved almost the lowest correlation on verbs in the supervised *Lemma* setup. Similarly to the supervised *All* setup, there is a substantial variation in terms of which embedding model performs the best on each

POS	Word2vec	GloVe	Skip-thoughts	Context2vec	MSSG
nouns	0.166*	**0.242***	0.107*	0.118*	0.240*
adjectives	**0.230***	0.167*	0.083	0.166*	0.170*
adverbs	0.318*	0.285*	0.036	**0.357***	0.290*
verbs	0.237*	**0.275***	0.133*	0.164*	0.183*
All	0.221*	**0.230***	0.099*	0.172*	0.205*

Table 5.8: Spearman's ρ for the supervised *Lemma* setup on ORIGINAL displaying the embedding models' performance across the different parts-of-speech. Correlations in bold represent the highest values for the given part-of-speech. * represents significant correlation, p-value < 0.05.

part-of-speech.

There were only a few consistencies in patterns across all parts-of-speech. Context2vec outperformed all other embeddings for adverbs under each setup. It is not possible to make a concrete judgment on Context2vec's ability on predicting the adverbs' similarity due to the small number of them. Skip-thoughts was the only embedding model that did not achieve the highest value on a part-of-speech for a single setup, although it did come close in the supervised *All* setup. The results on the supervised *All* setup had the most variations in terms of different embedding models performing well on different parts-of-speech. This setup might benefit from a model that predicted the similarity of an instance using an embedding model that had the highest performance on the part-of-speech of the target lemma within that instance.

	Original			Twitter		
	Abs Diff	Prod	Both	Abs Diff	Prod	Both
Word2vec	0.371*	0.385*	**0.444***	0.314*	0.373*	**0.384***
GloVe	0.318*	0.360*	**0.410***	0.189*	0.310*	**0.314***
Skip-thoughts	**0.465***	0.390	0.436*	0.292*	0.352*	**0.360***
Context2vec	0.406*	0.348	**0.417***	0.115*	**0.193***	**0.193***
MSSG	0.345*	0.445*	**0.454***	0.293*	**0.461***	0.447*

Table 5.9: Spearman's ρ for the supervised *All* setup on both Original and Twitter displaying the embedding models' performance using different feature sets. Correlations in bold represent the highest correlations for each embedding on each dataset. * represents significant correlation, p-value$<$ 0.05.

5.5 Feature sets

In this section, we look at the effect of using different feature sets. We compare using only one of the two feature sets and using both feature sets—taking the absolute difference and element-wise product of two usage representations—under both supervised setups for each dataset. Table 5.9 shows our results when comparing the different feature sets on Original under the supervised *All* setup. The absolute difference feature set, Abs Diff, outperforms the element-wise product feature set, Prod, for the two embedding models that embed the sentence and context, Skip-thoughts and Context2vec, respectively. The embedding models that involved summing up word embeddings to represent the usage of a word achieved higher results with Prod over Abs Diff—Word2vec, GloVe, and MSSG. This might be due to each feature set working better with the different methods in how the usage representations are formed either through embedding the usage (Skip-thoughts and Context2vec) or summing word embeddings (Word2vec,

	Original			Twitter		
	Abs Diff	Prod	Both	Abs Diff	Prod	Both
Word2vec	0.194*	**0.231***	0.221*	0.165*	0.164*	**0.196***
GloVe	0.128*	**0.263***	0.230*	0.015	0.128*	**0.134**
Skip-thoughts	**0.107***	0.097*	0.099*	−0.014	**0.093***	0.058
Context2vec	0.165*	**0.204***	0.172*	−0.006*	**0.083**	0.067*
MSSG	0.017	**0.250***	0.205*	0.025	0.124*	**0.271***

Table 5.10: Spearman's ρ for the supervised *Lemma* setup on both ORIGINAL and TWITTER displaying the embedding models' performance using different feature sets. Correlations in bold represent the highest correlations for each embedding on each dataset. * represents significant correlation, p-value$<$ 0.05.

GloVe, and MSSG). Using both feature sets outperforms the use of only one of the feature sets for each embedding model except Skip-thoughts, where it achieves its highest value using only Abs Diff.

Turning towards the supervised *All* setup for the TWITTER dataset (Table 5.9), the trend where the embedding models that embed the sentence or context—Skip-thoughts and Context2vec—perform better with Abs Diff than Prod does not apply. This might be due to the nature of the training corpora for these models, where Skip-thoughts and Context2vec were not trained on text from Twitter. Prod continues to outperform Abs Diff for Word2vec, GloVe, and MSSG. Using both feature sets outperforms using only one in all cases except for the Context2vec and MSSG embedding models, where using both feature sets ties or slightly beats using the Prod feature set.

The results of the supervised *Lemma* setup on ORIGINAL show some interesting traits (Table 5.10). For all embedding models except Skip-thoughts, the use of only Prod outperform only using Abs Diff, and the use of both

feature sets. Nevertheless, these results show that using only Prod usually outperform using only Abs Diff or both feature sets, which gives rise to the importance for Abs Diff in future work on the supervised *Lemma* setup. Skip-thoughts is the only embedding model that achieved its highest correlation only using Abs Diff, but all three correlations produced are fairly similar with a range of only 0.099–0.107.

The trend previously seen in the supervised *Lemma* setup for ORIGINAL is not as strong in the supervised *Lemma* setup for TWITTER (Table 5.10). The use of the Prod feature set outperforms or achieves values that are very similar to the use of the Abs Diff feature set, but using both feature sets outperforms using either feature set individually in the case of the Word2vec, GloVe and MSSG embedding models. Again, those embedding models that involve summing up word embeddings to represent the usage of a word show similar characteristics.

Between both datasets, using only Prod, or both feature sets, usually outperforms using only Abs Diff. Only using Abs Diff is not recommended for the supervised *Lemma* setup because the only time it outperforms the others, it is by a small margin. Prod more often outperforms using both feature sets, but these two feature sets often perform similarly in terms of correlations produced.

Chapter 6

Application of USim to WSI

We now look at applying our model to two word sense induction tasks available from SemEval-2013.[1] In this task, users must cluster instances of words based on their senses. For example, the following simplified examples from the SemEval-2013 Task 11 dataset with the target word *kangaroo* would be clustered together: *How to catch live kangaroo* and *A kangaroo is a marsupial from the family Macropodidae*, but the phrase *The kangaroo coin-operated videogame by Sum Electronics* would not be clustered with the other two. Since we only need to apply a clustering algorithm to our usage representations to perform WSI tasks, our unsupervised model is easily adapted to WSI. Similar to choosing embedding models, clustering algorithms can also be interchanged to attempt to achieve better performance. We will use k-means clustering in our experiments. We used the Word2vec and Context2vec embedding models in the same manner as they were applied to usage similarity. We only used these two word embeddings because

[1] `https://www.cs.york.ac.uk/semeval-2013/`

they had the highest performance in the unsupervised setup on TWITTER and ORIGINAL respectively.

6.1 SemEval Task 11

In this section we look at applying our unsupervised model to the SemEval2013 task 11 (Navigli and Vannella, 2013), using the embedding models that performed well on the unsupervised setup for ORIGINAL—Word2vec and Context2vec. We consider the embedding models that performed well on ORIGINAL because it shares a similar type of text as the dataset for SemEval2013 task 11—text from web pages. Task 11 is a non-graded word sense induction task, meaning instances are placed in clusters without a level of applicability. The dataset used was generated by first sending 100 queries to Google and gathering snippets from the top 64 results that were returned for each query. When annotating the dataset, disambiguation pages from Wikipedia were used as the sense inventory. The dataset was annotated through Amazon Mechanical Turk. The annotators were given the following explanation *The goal is annotating the search result snippets returned by Google for a given query with the appropriate meaning among those available (obtained from the Wikipedia disambiguation page for the query). You have to select the meaning that you consider most appropriate* (Navigli and Vannella, 2013).

The metrics used to evaluate the predicted clusters of a model are F1, Rand Index, Adjusted Rand Index, and Jaccard Index (Navigli and Vannella, 2013). Rand Index looks at how many pairs of instances are correctly

clustered together and how many are correctly not clustered together. Adjusted Rand Index is similar to Rand Index, but takes into account the expected value of the Rand Index and measures the degree of overlap between two clusters. The Jaccard Index looks at how many pairs of instances are correctly clustered together. F1 is simply the harmonic mean of the recall and precision. The recall measures how many of the instances in a gold standard cluster are captured by a single predicted cluster, whereas the precision measures how many instances in a predicted cluster are also together in a gold cluster.

We compare against the systems that achieved the highest values in each evaluation metric, which include the All-in-One baseline, HDP-Lemma, HDP-NOL, and UKPLLR (Navigli and Vannella, 2013). The All-in-One baseline clusters all instances of a lemma into one sense cluster and is often a difficult baseline to beat for sense clustering tasks because most words have a predominant sense that occurs far more frequently than its other senses (McCarthy et al., 2004). Both HDP systems apply topic modelling, treating the topics as the senses, (Lau et al., 2013). UKPLLR clusters based on co-occurrences with the assistance of a spanning tree (Zorn and Gurevych, 2013).

We applied our model using Word2vec and Context2vec embeddings. The clustering was performed by a k-means algorithm, where we used multiple values for k to observe the performance of our model under a different number of sense clusters. For each value of k, we calculated the average values for each evaluation metric of ten runs, using a different seed for the k-means clustering algorithm. Our model—using Word2vec or Context2vec

Model	# Clusts	F1	Rand	Adj Rand	Jaccard	Avg. Clust Size
Word2vec	2	0.565	0.508	0.049	0.311	32.000
	3	0.590	0.558	0.077	0.266	21.330
	4	0.615	0.585	0.092	0.236	16.000
	5	0.631	0.595	0.093	0.210	12.800
	6	0.643	0.601	0.093	0.189	10.670
	7	0.654	0.606	0.093	0.175	9.140
	8	0.664	0.607	0.089	0.161	8.000
Context2vec	2	0.560	0.493	0.032	0.323	32.000
	3	0.589	0.561	0.069	0.255	21.330
	4	0.600	0.578	0.071	0.218	16.000
	5	0.614	0.589	0.074	0.195	12.800
	6	0.625	0.595	0.074	0.173	10.670
	7	0.637	0.600	0.073	0.158	9.140
	8	0.647	0.601	0.069	0.143	8.000
All-in-One	1	0.544	0.399	0.000	**0.399**	64.000
HDP-Lemma	6.63	**0.683**	**0.652**	0.213	0.330	11.070
HDPNOL	6,54	0.680	0.649	**0.215**	0.338	11.680
UKPLLR	3.64	0.583	0.500	0.025	0.339	32.340

Table 6.1: Comparing our results on Task 11 obtained from using Word2vec and Context2vec embeddings against the systems that approached this task when it was released. The highest scores for each evaluation metric is represented in bold. The standard deviation was less than 0.005 for our models for each evaluation metric.

embeddings—achieves competitive results to the highest values achieved by the teams that participated in this task during the release of the shared task for all metrics except Adjusted Rand Index, where our models performed poorly. More precisely, our model comes within 0.019, 0.045, and 0.088 for F1, Rand, and Jaccard, respectively. The scores are shown in Table 6.1.

6.2 SemEval-2013 Task 13

Similar to Section 6.1, we applied our model using two embedding models that performed well on the ORIGINAL USim task and applied them to

SemEval-2013's task 13 (Jurgens and Klapaftis, 2013). This task is similar to Task 11 discussed in Section 6.1, except each instance is given a level of applicability for each context cluster that it is assigned to. For example, using a similarity scale of 0–4, the following phrase from the SemEval-2013 Task 13 dataset, *And is now the time to say I can hardly wait for your impending new novel about the Alamo?* would belong to a sense cluster representing *stay in one place and anticipate or expect something* with a complete applicability score of 4 and the sense cluster representing *look forward to the probable occurrence of* with an applicability of only 2. We show our results and compare them to the models that performed the best on the task in regards to two evaluation metrics—fuzzy B-Cubed, given as the precision, recall, and F-score, and fuzzy normalized mutual information (Fuzzy-NMI)—which are the unsupervised evaluation metrics provided by Jurgens and Klapaftis (2013). Fuzzy B-Cubed looks at how many items are in the same cluster as some item i in the gold standard and in the same cluster as some item i in the predicted clusters. Fuzzy-NMI looks at how many instances in the predicted clusters are in the same clusters as in the gold standard.

Table 6.2 shows the mean of the precision, recall and F-score of fuzzy B-Cubed over ten runs, when we use our model for SemEval-2013's task 13 using Word2vec and Context2vec embeddings, respectively, along with the results of the team that performed the best in terms of Fuzzy B-Cubed and Fuzzy-NMI in the shared task when it was held. The two models that we compare against include Unimelb (Lau et al., 2013), which is the same topic modelling based system as HDP-Lemma and HDP-NoL from Section

6.1, and AI-KU (Baskaya et al., 2013), which clusters instances based on co-occurrences generated after finding substitute words for the target word. The All-in-One baseline, which is a common baseline where all instances of a lemma are clustered into the same sense cluster, achieves the highest F-score for fuzzy B-Cubed. In terms of models that attempt this problem that are not the All-in-One baseline, our models achieve F-scores of 0.506 and 0.521 with Word2vec and Context2vec, respectively—beating the previous highest F-score of 0.483.

Our highest Fuzzy-NMI scores are 0.036 and 0.055 achieved with Word2vec and Context2vec, respectively, as seen in Table 6.3. Our models fall short of the highest F-NMI of 0.065 achieved by AI-KU.

Our model performed well on Task 11 in Section 6.1—achieving values close to the best systems for F1, Rand, and Jaccard—by only applying a simple clustering algorithm to our USim representations of a target word in context. Our model achieved the highest performance for fuzzy B-Cubed, using either Word2vec or Context2vec as the embedding model, on Task 13. This shows that our USim model can be generalized to WSI and is worth exploring further. Our models achieve a low standard deviation on both tasks, meaning that our models perform fairly similarly regardless of what seed is set for the k-means clustering algorithm.

Model	# Clusters	Precision	Recall	F-Score
Word2vec	2	0.454	0.577	0.506
	3	0.471	0.394	0.429
	4	0.473	0.304	0.370
	5	0.480	0.250	0.328
	6	0.485	0.211	0.294
	7	0.488	0.182	0.265
	8	0.490	0.160	0.242
	9	0.492	0.143	0.222
	10	0.494	0.130	0.205
	11	0.479	0.118	0.189
Context2vec	2	0.465	0.593	0.521
	3	0.476	0.402	0.436
	4	0.487	0.314	0.382
	5	0.497	0.261	0.342
	6	0.503	0.222	0.308
	7	0.512	0.196	0.283
	8	0.520	0.177	0.264
	9	0.525	0.160	0.245
	10	0.532	0.148	0.232
	11	**0.537**	0.136	0.217
All-in-One		0.455	**0.989**	**0.623**
AI-KU		0.838	0.254	0.390
Unimelb(50k)		0.524	0.447	0.483

Table 6.2: Fuzzy B-Cubed results on task 13 comparing our model using Word2vec and Context2vec embeddings against systems that approached this task during its release. The highest values for each evaluation metric is in bold. The standard deviation for each of our models is less than 0.004.

Model	# Clusters	fuzzy-NMI
Word2vec	2	0.034
	3	0.036
	4	0.030
	5	0.030
	6	0.033
	7	0.034
	8	0.034
	9	0.035
	10	0.036
	11	0.036
Context2vec	2	0.038
	3	0.042
	4	0.044
	5	0.046
	6	0.046
	7	0.049
	8	0.051
	9	0.052
	10	0.053
	11	0.055
All-in-One		0.000
AI-KU		**0.065**
Unimelb(50k)		0.060

Table 6.3: Fuzzy-NMI results on task 13 comparing our model using Word2vec and Context2vec embeddings against systems that approached this task during its release. The highest value is in bold. The standard deviation for each of our models is less than 0.007.

Chapter 7

Conclusion

We presented our embedding-based model that we applied to two different USim datasets and two different WSI tasks. We achieved state-of-the-art results on both USim datasets, which answers our first research question. It also showed that embedding-based models can be used to generate useful representations of the meaning of a word in context. We beat the previous state-of-art results on ORIGINAL, achieved by topic modelling, using Word2vec, GloVe, and Context2vec embeddings in our unsupervised setup. We beat the previous state-of-art value on TWITTER, also achieved by topic modelling, only by using Word2vec embeddings in our unsupervised setup. Our second research question asks if a supervised model can achieve better results then an unsupervised model. This is possible, although it depends on the training/testing split of the dataset. We showed that when evaluating the model with ten fold cross validation—meaning that the target lemma can be seen in training—the model always performs better than its unsupervised counterpart. However, this is untrue for the case when we split the dataset

based on the target lemma, where the target lemma is not seen in training. We also observed the change in performance of our supervised models using different features. We found that using both types of features—absolute difference and element-wise product—usually achieves the best results on the supervised *All* setup. Our last research question asks if our model can generalize to WSI tasks. Our model did well on both WSI tasks, even achieving the best results under one of the evaluation metrics—fuzzy B-Cubed—on the SemEval-2013 task 13.

7.1 Future Work

The idea of evaluating models based on parts-of-speech sparks the idea of a model where each usage would be represented by the model that performed the best on the part-of-speech of the target lemma. For example, since Context2vec achieved the highest score on verbs out of all embedding models, then all usages with a verb as the target lemma will have their vector representations from a Context2vec embedding. We would expect that an ensemble model would not greatly increase the values here because a single model, Context2vec, performs the best on all parts of speech except nouns, which was performed only slightly better on by Word2vec. If there was more of a balanced distribution across different models in terms of the best performing models on each part of speech, or if the difference for nouns was much greater, then there would be more of a cause to explore an ensemble model. We used a simple regression model in our experiments. Experimenting with a variety of regression models, such as a support vector

regression model, for our supervised setup, might increase the performance of our models. The regression model is not the only model that can be replaced; different embedding models can be applied to our model which might capture the semantic meaning of a usage. Doc2vec (Le and Mikolov, 2014) is an obvious choice for an embedding model for our model, where Doc2vec is designed to capture the semantic information in an entire document—in our case it would be the the full sentence containing the target lemma. Our supervised *Lemma* setup might be failing due to an insufficient amount of training data. Providing more instances of different lemmas might assist our supervised model to generalize across lemmas—meaning to increase performance on unseen lemmas. We briefly looked into two WSI tasks. It might be possible to better our results simply by applying different clustering algorithms instead of K-means, such as an agglomerative clustering algorithm.

Bibliography

Vıt Baisa, Jane Bradbury, Silvie Cinková, Ismaıl El Maarouf, Adam Kilgarriff, and Octavian Popescu. 2015. SemEval-2015 task 15: A corpus pattern analysis dictionary-entry-building task. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015)*. pages 315–324.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)*. Nagoya, Japan, pages 356–364.

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. AI-KU: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 300–306.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3:993–1022.

Ping Chen, Wei Ding, Chris Bowes, and David Brown. 2009. A fully unsupervised word sense disambiguation method using dependency knowledge. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 28–36.

David Crystal. 1987. *The Cambridge Encyclopedia of Language*. Press Syndicate of the university of Cambridge.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of Coling 2004*. Geneva, Switzerland, pages 350–356.

Katrin Erk, Diana McCarthy, and Nicholas Gaylord. 2009. Investigations on word senses and word usages. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Singapore, pages 10–18.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop: Can we beat Google*. Marrakech, Morocco, pages 47–54.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM, pages 406–414.

Spandana Gella, Paul Cook, and Bo Han. 2013. Unsupervised word usage similarity in social media texts. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. Atlanta, USA, pages 248–253.

Fredrik Hallsmar and Jonas Palm. 2016. Multi-class sentiment classification on twitter using an emoji training heuristic.

Patrick Hanks. 2004. Corpus pattern analysis. In *Euralex Proceedings*. volume 1, pages 87–98.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea, pages 873–882.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *European Conference on Computer Vision*. Springer, pages 727–739.

David Jurgens and Ioannis Klapaftis. 2013. SemEval-2013 task 13: Word sense induction for graded and non-graded senses. In *Proceedings of the*

7th International Workshop on Semantic Evaluation (SemEval 2013). Atlanta, USA, pages 290–299.

Adam Kilgarriff. 1997. "I Don't Believe in Word Senses". *Computers and the Humanities* 31(2):91–113.

Adam Kilgarriff and Christiane Fellbaum. 2000. WordNet: An electronic lexical database.

Milton King, Waseem Gharbieh, SoHyun Park, and Paul Cook. 2016. UNBNLP at SemEval-2016 task 1: Semantic textual similarity: A unified framework for semantic processing and evaluation. *Proceedings of SemEval* pages 732–735.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 3276–3284.

Mikael Kågebäck, Fredrik Johansson, Richard Johansson, and Devdatt Dubhashi. 2015. Neural context embeddings for automatic discovery of word senses. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Denver, Colorado, pages 25–32.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *Proceedings of the 14th Conference of the European Chapter of*

the Association for Computational Linguistics. Association for Computational Linguistics, Gothenburg, Sweden, pages 540–549.

Mirella Lapata and Chris Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics* 30(1):45–73.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic modelling-based word sense induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 307–311.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*. PMLR, Bejing, China, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*. ACM, pages 24–26.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Maryland, USA, pages 302–308.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*. Springer, pages 740–755.

Marco Lui, Timothy Baldwin, and Diana McCarthy. 2012. Unsupervised estimation of word usage similarity. In *Proceedings of the Australasian Language Technology Association Workshop 2012*. Dunedin, New Zealand, pages 33–41.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*. pages 104–113.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland, pages 1–8.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Using automatically acquired predominant senses for word sense disambiguation. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Barcelona, Spain, pages 151–154.

Diana McCarthy and Roberto Navigli. 2007a. SemEval-2007 task 10: En-

glish lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval '07, pages 48–53.

Diana McCarthy and Roberto Navigli. 2007b. SemEval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Prague, Czech Republic, pages 48–53.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. pages 51–61.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The Senseval-3 English lexical sample task. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Barcelona, Spain, pages 25–28.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the first International Conference on Learning Representations (ICLR 2013)*. Arizona, USA.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6(1):1–28.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41(2):10:1–10:69.

Roberto Navigli and Daniele Vannella. 2013. Semeval-2013 task 11: Word sense induction and disambiguation within an end-user application. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. volume 2, pages 193–201.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654* .

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::Similarity - Measuring the relatedness of concepts. In *Demonstration Papers at the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*. Boston, MA, pages 38–41.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM* 8(10):627–633.

Giancarlo Salton, Robert Ross, and John Kelleher. 2016. Idiom token classification using sentential distributed semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 194–204.

Serge Sharoff. 2006. Open-source corpora: Using the net to fish for linguistic data. *Corpus Linguistics* 11(4):435–462.

Linfeng Song, Zhiguo Wang, Haitao Mi, and Daniel Gildea. 2016. Sense embedding learning for word sense induction. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, Berlin, Germany, pages 85–90.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147.

Xing Wei and W Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 178–185.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the first International Conference on Computer Vision (ICCV 2015)*. pages 19–27.

Hans-Peter Zorn and Iryna Gurevych. 2013. UKP-WSI: UKP Lab Semeval-2013 task 11 system description. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Sev-*

enth International Workshop on Semantic Evaluation (SemEval 2013). Association for Computational Linguistics, Atlanta, Georgia, USA, pages 212–216.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, pages 1393–1398.

Geoffrey Zweig and Chris J.C. Burges. 2012. A challenge set for advancing language modeling. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Association for Computational Linguistics, Montréal, Canada, pages 29–36.

Curriculum Vitae

Candidate's full name: Milton King
University attended: Mount Allison University, 2011-2015, Bachelor of Science.
Publications:

Milton King, Waseem Gharbieh, SoHyun Park, and Paul Cook. 2016. UNBNLP as SemEval-2016 task 1: Semantic textual similarity: A unified framework for semantic processing and evaluation. *Proceedings of SemEval* pages 732-735.

Milton King and Paul Cook. 2017. Supervised and unsupervised approaches to word usage similarity. *Proceddings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications* pages 47-52.

Conference Presentations:

Milton King. Supervised and unsupervised approaches to word usage similarity. The 1st Workshop on Sense, Concept and Entity Representations and their Applications. April, 2017. Conference centre, Valencia, Spain. Poster presentation.

Milton King. Supervised and unsupervised approaches to word usage similarity. Graduate research conference. March, 2017. University of New Brunswick, Fredericton, New Brunswick. Poster presentation.

Milton King. Supervised and unsupervised approaches to word usage similarity. 13 annual research conference. Apri, 2016. University of New

Brunswick, Fredericton, New Brunswick. Poster presentation.

Milton King. Supervised and unsupervised approaches to word usage similarity. Science atlantic mathematics, statistics and computer science conference 2016. October, 2016. University of Cape Breton, Cape Breton. Conference presentation.