Word Embeddings to Classify Types of Diachronic Semantic Shift

Dylan Koldenhof
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
d.koldenhof@student.utwente.nl

ABSTRACT

Languages are constantly evolving, in many ways. One of the ways they evolve is in semantics, the meaning of words. This presents an interesting challenge for automated Natural Language Processing (NLP), as a thorough manual inspection of this phenomenon is difficult. Much work has already shown promising results in detection of the semantic shift but there is little in the field investigating the nature of these shifts. This research aims to fill this gap by investigating whether different types of semantic shifts can be classified using word embeddings trained with Word2Vec. Different machine learning classifiers are trained on embeddings which are themselves trained on Project Gutenberg ebooks spanning the period 1800-1849, and embeddings trained on Wikipedia. Results show promise, but with a top accuracy of 0.5 when validated on another time period, there is room for improvement in future work.

Keywords

word embedding, diachronic semantic shift, language evolution, semantic change, word2vec, computational linguistics, natural language processing

1. INTRODUCTION

The phenomenon of words changing meaning over time is something that becomes rather obvious looking at any text published some centuries ago. Many words used will sound completely out of place with their modern meaning. Hence, this phenomenon has been noticed and researched for quite some time [15, 14, 4], but has seen new light with the advent of Natural Language Processing (NLP) tools, allowing for new ways to detect and analyze it. This phenomenon is called by many names in literature, but keywords generally are diachronic (change across time) and semantic (meaning of words), thus the full term that will be used in this paper is "diachronic semantic shift", shortened as "semantic shift" or simply "shift".

Generally, diachronic semantic shift is incremental and happens over the span of multiple generations. An example often cited in the literature is the word "gay" which over the past century slowly went from having a meaning of happy, to nowadays almost exclusively being associated with homosexuality. Another somewhat older example is the word "car", initially referring to a horse-drawn vehicle, which changed into having the meaning of automobile.

Looking back far enough, these changes can add up to the point where it might make some sentences incomprehensible only knowing the modern meaning. Due to their incremental nature, comprehending the exact process and nature behind these changes can get very complicated. This means lots of content over long periods needs to be parsed in order to make complete sense of every semantic shift. Thus, advances in NLP have sparked interest in applying this to semantic shifts. With these techniques, large volumes of text (corpora) can be analyzed, much quicker than any human could, with promising results in detecting shifts [17, 7, 5].

The most popular means of achieving this in current research is by comparing what are known as word embeddings. Word embeddings are vectors of a word which are derived based on the context the word appears in. The idea behind it is that semantics can be revealed based on the context of a word, and thus the vectors represent a word's meaning. Consequently, word embeddings trained on different time periods can also reveal a change in word meaning.

Going back to the initial linguistic research on semantic shift, many linguists such as Bloomfield [3] developed categorizations of semantic shift.

So far in research on semantic shift using NLP these types of categorizations do not seem to have been covered, whether using word embeddings or any other approach. An automatic categorization could provide interesting insights for historical linguistics, hence the goal of this research is to provide an exploration into this topic. Specifically, the main research question that will be asked is:

RQ0: *Can types of diachronic semantic shift be reliably automatically classified?*

Which further leads into these three questions:

- **RQ1**: *What classification methods provide the best classification results?*
- **RQ2**: *What types can be most clearly classified?*
- **RQ3**: *Do the classifiers yield consistent results across corpora?*

These research questions will be answered using word embeddings trained with the SGNS (Skip-Gram with Negative Sampling) method, using the Word2Vec implementation of the Python Gensim library[1]. For the training and

[1] https://radimrehurek.com/gensim/models/word2vec.html

35th Twente Student Conference on IT July 2nd, 2021, Enschede, The Netherlands.

initial testing of the classifiers, two corpora will be used. The first consists of Project Gutenberg[2] ebooks spanning approximately the first half of the 19th century, while the second consists of recent Wikipedia articles. Two versions of the corpora are used, one using the words as they appear in the texts, except lowercased and with punctuation removed, while the other versions contain lemmatized and part of speech-tagged tokens. These two approaches will henceforth be referred to as "raw" and "lemma".

After the models are trained on the corpora, they are aligned using the Orthogonal Procrustes (OP) method, after which the vectors for the same word across the different time periods can be compared. From the words that have most significantly shifted on the lemmatized models, an annotated list is made which will be used to categorize shifts using different supervised machine learning classifiers, with the differences between the vectors as features. Cross-validation will be performed on the annotated list with the accuracy metrics used for an initial evaluation, both on the raw and lemma embeddings.

To determine the generalizability of these classifiers, two different corpora will be used for validation, composed of time periods 1750-1799 and 1900-1949. Both of these are composed of Project Gutenberg texts and will be aligned the same way as the training corpora. The classifiers will then be used on another annotated list composed in the same manner, with the classifier predicting the categories. The performance across these corpora and the training corpora can then be compared and the metrics of the classifiers that show the best results on both corpus pairs will be used to answer the main research question.

2. RELATED WORK

As mentioned before, Bloomfield [3] proposes a categorization of semantic shifts, which is popular and used as a basis for more recent studies [6].

Bloomfield initially proposed nine shifts, which are as follows:

- Narrowing: A meaning going from a wider to a narrower scope, an example is the word "hound", which originally was the word for dog in general and later evolved to mean hunting dog specifically.
- Widening: Opposite of narrowing, and the word "dog" itself is actually an example, which used to refer to a specific sort of dog.
- Metaphor: A word used as a metaphor becoming its primary meaning, e.g. "broadcast" as described before.
- Metonymy: A meaning relating to a place or time shifting to another nearby place or time, e.g. the word "cheek", which had the meaning of jaw in older English.
- Synecdoche: Shift from a part to whole, or vice versa. An example is the word "commercial" in the sense of advertisement. Commercial is just one trait of what it is, but it shifted into the word for it entirely.
- Hyperbole: A stronger meaning shifting into a weaker one, the word "quite" is an example, which used to mean "completely" or "wholly".
- Litotes: Opposite of hyperbole, e.g. the word "kill" which in its original Germanic root had meanings in the sense of tormenting or vexing.
- Degeneration: A meaning shifting into a lower status, for instance the word "silly", which has gone through many meanings, starting in the sense of happy or blessed to weak and the meaning it has now.
- Elevation: Opposite of degeneration, the word "knight" went from a low servant to someone exalted.

Although there were some earlier methods, the advent of neural word embedding algorithms led to a large increase in research into diachronic semantic shift [12]. The first popular implementation of this method is the Word2Vec framework proposed by Mikolov et al. [10]. This is a neural network model that consists of two different training methods: Continuous Bag of Words (CBOW) and Skip-Gram. CBOW is generally more appropriate for small corpora while Skip-Gram is more suited for large ones [12], and thus Skip-Gram will be used for this research.

The Word2Vec Skip-Gram model works by sliding a window of a given size across the input text. For every word, pairs of the word and another word within the window are formed. These pairs then form samples for training a neural network with a single hidden layer. The task of the network is, given an input word, to evaluate the probablity of all words within the vocabulary to be in the window of the input word. This task itself is not the goal of the method at all, but the weights of the neurons in the hidden layer after training are what make up the vector of the input word [8].

Along with this Negative Sampling is used in the training process. Normally, when given a training sample pair, the weights for all input words are updated with the goal of the probablity of all words outside the pair having probablity 0, while for the input word in the pair the probablity of the other word in the pair will be 1. With a large vocabulary this means a lot of unnecessary computation, since most words do not share much context, and thus the weight updates will be rather insignificant. With Negative Sampling, only a limited amount of "negative" words (random words from the vocabulary) are chosen for every training sample for which the weights will be updated [9]. Skip-Gram with Negative Sampling (SGNS) was generally deemed most effective by extensive evaluation studies [18, 19].

In order to identify diachronic semantic shift however, there needs to be a way of measuring these word embeddings across time. There have been two approaches to this: static and dynamic embeddings.

The static approach relies on separating a corpus in different time slices, and training embeddings for these time slices independently. One approach is to train the model on each of these time slices separately. The problem with this is that the embeddings cannot be compared directly. Due to the stochastic nature of the model, embeddings trained on different slices will not be directly comparable, only their relative distances (provided meanings stay similar) will be [2, 12] (see Figure 1).

Hence, one way to compensate for this is to align the different vector spaces using some sort of method. The best [18] of these is found to be the Orthogonal Procrustes (OP) method proposed by Hamilton et al. [5].

The other approach is dynamic embeddings. The key difference between these and static embeddings is that dif-

[2] https://www.gutenberg.org/

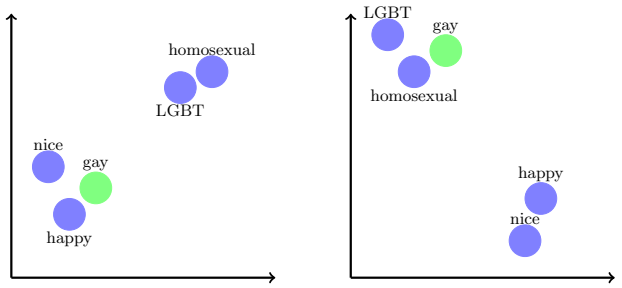

Figure 1. Illustrating what happens when word embeddings are trained on two different time periods and not aligned, using as an example the word *gay*. Relative distances are roughly the same between unchanged words, but their absolute positions in the vector space are completely different and hence they cannot be compared directly. Based on figures by Bianchi et al. [2].

ferent time slices are not modeled independently of one another, thus word embeddings at a given time slice are based on their position at previous ones. Taking this into account yields more directed and smoother shift in some cases, but a disadvantage is that time slices flow into one another too much [12]. Some examples of dynamic embeddings are those proposed by Bamler and Mandt [1] and Rudolph and Blei [16].

There have been some papers inquiring into the nature of semantic change, but none with categorizations like the one proposed by Bloomfield [3]. Mitra et al [11] use a different approach from word embeddings to derive sense clusters. The shifts observed in these sense clusters are then assigned four categories: split, join, birth and death. Hamilton et al. [5] derived statistical laws of shifts, such as that frequent words shift at slower rates.

3. METHODOLOGY

3.1 Corpora and Preprocessing

Project Gutenberg texts from approximately 1800-1849 were chosen for the older corpus. As Gutenberg metadata does not list publishing date, nor any other means of retrieving it easily (for example, ISBN number), the average between author birth and death year, which is in Gutenberg's metadata, was used to estimate publishing date. This yielded 4536 texts, consisting of around 341.8M words.

Due to the large amount of proper nouns that will otherwise dominate the most changed words, along with some words having only changed as one function of the word (for instance, the verb "to power" relates to power in the physical sense, while before it had the sense of might)[3], it was decided to also prepare a lemmatized and part of speech (POS)-tagged version of the corpora. Lemmatization means words are reduced to their simplest grammatical forms. Such as plural nouns turned into their singular forms, and verbs to the first person present. POS-tagging consists of marking the part of speech, such as verb, noun, and adjective, for a given word. For the corpora used, all words are processed into strings consisting of the lemmatized form and the POS tag, separated by an underscore. The lemma forms were retrieved using the Python SpaCy library[4].

For the raw Wikipedia corpus, the archive from May 2021 was used, only including articles with more than 5000 words. This consists of around 1.63B words. Due to the long processing time, the lemma Wikipedia corpus is from a slightly older archive from 2017, which was pre-trained on Word2Vec[5].

For validating the generalizability of the classifier and answering **RQ3**, two different corpora are used, spanning the time periods 1750-1799 and 1900-1949. These are both retrieved from Gutenberg and preprocessed in the same manner as the 1800-1849 corpus. The 1750-1799 corpus consists of 765 texts containing 66.4M words, while the 1900-1949 corpus consists of 10838 texts containing 646M words.

3.2 Training the model

The corpora (except the pre-trained one) are subsequently trained with Gensim's Word2Vec model, which includes an implementation of Skip-Gram with Negative Sampling (SGNS). An exception is the lemma Wikipedia corpus, which was pre-trained, also using SGNS. An overview of all parameters used can be found in Appendix A.

3.3 Comparing embeddings

The alignment of the different models was done using the Orthogonal Procrustes (OP) method. To illustrate further, the Orthogonal Procrustres problem is, given two matrices A and B, to find the orthogonal matrix most closely mapping A to B. This mapping can be used on the two embedding spaces to map the older period to the space of the newer one. Specifically, when $\mathbf{W}^t \in \mathbb{R}^{d \times |\mathcal{V}|}$–where d is the number of dimensions in an embedding and $\mathcal{V}$ the shared vocabulary of the models–is a matrix of all word embeddings at period t, the following equation needs to be solved:

$$\mathbf{R}^t = \arg\min_{\mathbf{Q}} ||\mathbf{W}^t\mathbf{Q} - \mathbf{W}^{t+1}||_F \quad (1)$$

With the constraint that $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$ (orthogonality). The resulting transformation $\mathbf{R}^t \in \mathbb{R}^{d \times d}$ can then be applied to $\mathbf{W}^t$ to map it to the space of $\mathbf{W}^{t+1}$, allowing the spaces to be compared [5]. The assumption of this method is that most words retain their meaning, because the alignment seeks to minimize the distance between the periods. Shifts are detected because they are the outliers that do not fit with the alignment.

Before this however, the embeddings matrix was mean-centered along columns, i.e. the means of all embedding dimensions are zero. Then the vectors were L2(Euclidean)-normalized. These steps improve the per formance of the alignment method [18].

The cosine distance (CD) between embeddings is then used to determine the magnitude of the detected shift.

3.4 Classification

From words that were above a threshold of 0.75 CD in the lemma models, words were selected that could reasonably be manually classified. The reason the lemma models were used is that proper nouns could be filtered out, which dominate the words above the threshold in the raw corpora. The selection process generally consisted of inspecting a random selection of 250 words above 0.75 CD and then verifying candidates and determining a category for them using the Online Etymology Dictionary[6].

This process yielded a list of 88 words, annotated with three different categories. The full list can be found in

[3] https://www.etymonline.com/word/power
[4] https://spacy.io/
[5] From http://vectors.nlpl.eu/repository/
[6] https://www.etymonline.com/

Appendix B.1. These are a reduction of Bloomfield's categories as described in Section 2, which was done because of the small size of the list along with many of Bloomfield's categories being quite rare in the selection process. The categories are as follows:

- Scope change (narrowing and widening)
- Metaphor and metonymy
- Other types/cannot be determined

For each word in this list, the difference between the vectors of the word from the different time periods in the aligned embedding space is computed. The elements of this delta vector are then used as features for training different machine learning classifiers, with the categories functioning as labels.

Before training the classifiers, the features are scaled to have zero mean and unit variance. This reduces the effect of outliers, along with potentially having training data that can be more generalized across corpora.

To validate the performance of the models, another annotated list is used, created in the same manner as the other. The difference is that this list is using vectors from the 1750-1799 and 1900-1949 corpora, and only contains 22 words. It can be found in Appendix B.2. The best performing classifiers on the 1800-1849/wiki pair will be tested on this list in order to determine whether they still perform similarly. The best performing classifier on both of these can then be concluded to be the most reliable for categorizing semantic shift on Word2Vec embeddings.

4. EVALUATION

4.1 First results on training corpora

The best classifiers are determined based on average prediction accuracy across a Leave-One-Out(LOO) cross-validation (CV) on the list. LOO-CV is when a model is trained on all but one of the elements in the data set, with the remaining single element used for testing. This training is then repeated with a different test element every time, until all elements in the data have been tested. Given the small size of the list, this allows the classifier to be trained as much as possible, as opposed to other forms of CV, such as 10-fold, where different selections of 10 samples are taken out and used as testing sets.

The results of this first step can be found in Appendix C. The classifiers are referred to by their class names in the Python scikit-learn library[7], which is used for all of them.

What can initially be noted is that these initial accuracy numbers are quite low. Crucially however, some outperform random or majority guessing. In Table 1 the distribution of the labels is shown. As a benchmark, the accuracy of a ZeroR classifier, which is when only the label which forms the majority of training data is predicted, the accuracy would be $33/(28+27+33) = 0.375$.

Label	Count
Scope (0)	28
Metaphor/metonymy(1)	33
Other(2)	27

Table 1. Distribution of labels

[7] https://scikit-learn.org/stable/index.html

Furthermore, the differences between the raw and lemma models appear to be fairly small except for two classifiers: Random Forest, which performs much better on the raw models, and Stochastic Gradient Descent (SGD) with log loss, which is much better on the lemma models. Upon further analysis of parameters, it showed that these differences are only due to the random elements of these classifiers, which gives them vastly different results every run. This makes them not very generalizable and thus it is more beneficial to look at methods with less randomness. Excluding SGD and Random Forest, in Table 2 the best classifiers for raw and lemma models are shown.

Classifier	Accuracy	Tokens
BernoulliNB	0.443 ± 0.163	raw
KNeighborsClassifier	0.397 ± 0.151	raw
SVC	0.386 ± 0.053	raw
SVC	0.432 ± 0.097	lemma
LogisticRegression	0.432 ± 0.125	lemma
GaussianNB	0.420 ± 0.138	lemma

Table 2. Best classifiers for raw and lemma models.

4.2 SVC grid-search

With most of these classifiers there is little room for improvement except for the Support Vector Classifier(SVC), since it allows a lot of parameter tweaking. Thus a grid-search was performed on this classifier, testing many combinations of parameters to determine the best combination. The best results for both corpora are shown in Table 3.

Kernel	Parameters[8]	Accuracy[9]	Tokens
Sigmoid	`C = 100, gamma = 0.1`	0.477 ± 0.125	raw
Sigmoid	`C = 1000, gamma = 0.1`	0.477 ± 0.136	raw
Sigmoid	`C = 10000, gamma = 0.1`	0.477 ± 0.119	raw
Sigmoid	`C = 10, gamma = 0.3`	0.614 ± 0.146	lemma
Sigmoid	`C = 1, gamma = 0.4`	0.614 ± 0.150	lemma
Sigmoid	`C = 10, gamma = 0.9`	0.591 ± 0.141	lemma

Table 3. Top 3 results from parameter tweaking on raw and lemma models.

The sigmoid kernel is the only one that showed up in the top 3 for both the raw and lemma models, so it is clearly best. However, the best parameters for the lemma and raw models are different and on the lemma models the accuracy goes up to much higher values than on the raw models. The confusion matrix of the classifier with the best accuracy is shown in Figure 2.

4.3 Validation performance

However, although the sigmoid classifier seems ideal, it performs poorly on the validation set, as can be seen in Table 4.

[9] Here and in subsequent tables parameters not given are the default used in Gensim.
[9] Here and in subsequent tables the values after $\pm$ represent the standard deviation as evaluated from a 10-fold cross-validation.

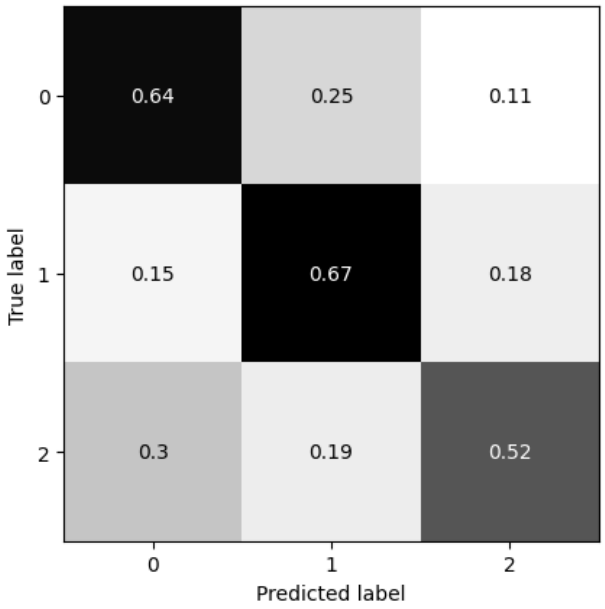

Figure 2. Confusion matrix for the cross-validated lemma models on sigmoid SVC with $C = 10$ and $\gamma = 0.3$. The values are the proportion of the true label that is predicted for a given label. Labels are as given in Table 1.

Kernel	Parameters	Acc. CV	Acc. val.
Sigmoid	`C = 10, gamma = 0.3`	0.614 ± 0.146	0.23
Sigmoid	`C = 1, gamma = 0.4`	0.614 ± 0.150	0.23
Sigmoid	`C = 10, gamma = 0.9`	0.591 ± 0.141	0.27

Table 4. Sigmoid kernel compared on the cross-validated training model pair and the validation pair.

Since this classifier is not able to categorize semantic shift on different models, it is rather useless for a general purpose. Lower values for gamma, which also performed decently on the lemma model, yield more positive results on the verification set. Along with this, the other kernels that performed better than zeroR were also tested on the other pair. The best results from this evaluation can be seen in Table 5.

Kernel	Parameters	Acc. CV	Acc. val.
Sigmoid	`C = 1, gamma = 0.1`	0.500 ± 0.166	0.36
Sigmoid	`C = 1, gamma = 0.05`	0.375 ± 0.128	0.41
Sigmoid	`C = 1, gamma = 0.01`	0.352 ± 0.091	0.50
Linear	`C = 1`	0.432 ± 0.097	0.41
RBF	`C = 100, gamma = 0.001`	0.420 ± 0.086	0.36
RBF	`C = 100, gamma = 0.0001`	0.432 ± 0.074	0.41
RBF	`C = 10000, gamma = 0.00001`	0.432 ± 0.097	0.41

Table 5. Comparison of kernels across both model pairs

Though a sigmoid kernel shows the highest performance on the validation model pair, its accuracy on the training/CV pair is poor. Hence the most generalizable classifiers seem to either be the RBF or linear kernels, with near equal

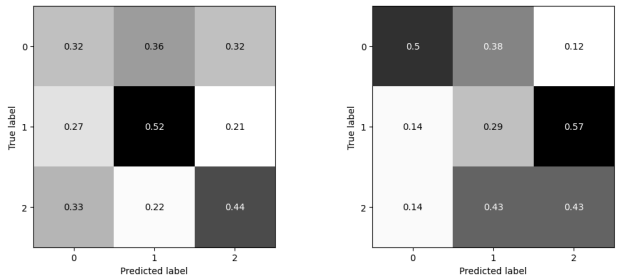

Figure 3. The confusion matrices for the SVC with RBF kernel, $C = 100$ and $\gamma = 0.0001$. On the left the performance on the training/CV pair and on the right the validation pair.

results, including equal confusion matrices. However, the RBF kernel with $C = 100$ and $\gamma = 0.0001$ is slightly better on the raw models than the linear kernel, and has a slightly lower standard deviation, so this can be seen as the best classifier. The confusion matrices using the RBF kernel with $C = 100$ and $\gamma = 0.0001$ are shown in Figure 3. It can be seen that despite the similar accuracy, the classification results are still vastly different, with category 1 being predicted well on the training pair, but not at all on the validation pair. The other classifiers that performed well on the training/CV pair showed much worse performance than the SVCs on the validation pair, so these are not covered further.

4.4 Aligning the pairs

Due to the different confusion matrices seen in Figure 3, it was thought that this was because the differences of the two pairs of vectors were not aligned and thus not comparable, similarly to two trained embeddings from different time periods. Thus it was thought an alignment of the two spaces of delta vectors, in the same manner as the alignment of the embeddings, could yield better results. Unlike with aligning the embeddings, a shared vocabulary is not necessary, because there is no need to subtract the aligned vectors for the same word. While the transformation matrix is based on a shared vocabulary, this transformation can be applied to the entire space, including words not in the other pair. All the classifiers in Tables 4 and 5 were again validated with this new alignment and the results are shown in Table 6, along with the best result found by parameter tweaking at the top.

As can be seen in the table, this best result with a sigmoid kernel and $C = 5$ and $\gamma = 0.8$ performs fairly well on both CV and the aligned deltas, with a substantial improvement coming from the alignment. With some other classifiers, the opposite effect is shown, and the RBF kernel performs better unaligned, perhaps as a result of the classifier only being effective on the particular difference existing between the pairs before alignment. The confusion matrices of the classifier for the cross-validation and the aligned validation pair on the best result are shown in Figure 4.

From the confusion matrices it can be seen that the classification is a lot more consistent across the pairs compared to the RBF kernel (Figure 3). The main difference here is the worse performance in predicting label 1, but otherwise the performance is very similar across both.

5. DISCUSSION

In the end, the results suggest that some accuracy can be

Kernel	Parameters	Acc. al.	Acc. no al.	Acc. CV
Sigmoid	`C = 5, gamma = 0.8`	0.50	0.36	0.557 ± 0.109
Sigmoid	`C = 10, gamma = 0.3`	0.41	0.23	0.614 ± 0.146
Sigmoid	`C = 1, gamma = 0.4`	0.36	0.23	0.614 ± 0.150
Sigmoid	`C = 10, gamma = 0.9`	0.45	0.27	0.591 ± 0.141
Sigmoid	`C = 1, gamma = 0.1`	0.23	0.36	0.500 ± 0.166
Sigmoid	`C = 1, gamma = 0.05`	0.32	0.41	0.375 ± 0.128
Sigmoid	`C = 1, gamma = 0.01`	0.36	0.50	0.352 ± 0.091
Linear	`C = 1`	0.36	0.41	0.432 ± 0.097
RBF	`C = 100, gamma = 0.001`	0.36	0.36	0.420 ± 0.086
RBF	`C = 100, gamma = 0.0001`	0.36	0.41	0.432 ± 0.074
RBF	`C = 10000, gamma = 0.00001`	0.36	0.41	0.432 ± 0.097

Table 6. Comparison of kernels across both model pairs with alignment.

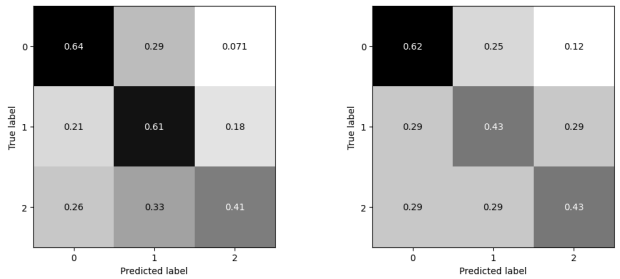

Figure 4. The confusion matrices for the SVC with sigmoid kernel, $C = 5$ and $\gamma = 0.8$. On the left the performance on the training/CV pair and on the right the aligned validation pair.

found in classifying semantic shift across different corpora, after aligning them. Though this is far from perfect, an accuracy above 0.5 for both pairs is clearly better than the zeroR benchmark of 0.38, and it goes above this threshold in predicting every label, as can be seen in Figure 3. However, the aligned validation pair still shows poorer performance than on the cross-validation, and the best performance on cross-validation does not necessarily reflect the best performance on the validation pair. This is firstly most likely the result of the small size of the training and test word list, and secondly the limitations of the alignment, which is ultimately susceptible to some inaccuracy, as it relies on the shared vocabulary between the pairs. Many words could have changed differently from 1750 to 1949 then from 1800 to the present. Furthermore on all confusion matrices it is clear that classifying label 2 (other) is less accurate. This is most likely a result of it being a grouping of different categories, so there can be many differences within it. Label 1 (metaphor/metonymy) is only more inaccurate on the validation pair, which might just be due to the small size of the validation word list.

6. CONCLUSIONS

Overall, **RQ0** has to be answered somewhat inconclusively, but with promising directions for future research. 0.6 accuracy is possible on the Gutenberg 1800-1849 and Wikipedia corpora, but it is somewhat less effective across different corpora, and even 0.6 accuracy can still not be considered that reliable.

The answer to **RQ1** seems quite clearly a form of a Support Vector Classifier, which is the only classifier that could be tweaked to yield an accuracy well above the zeroR benchmark consistently on both raw and lemma tokens as well as on the validation pair.

RQ2 can be answered by the reduction of Bloomfield's types that was performed, scope, metaphor/metonymy and other, but this was mostly done due to the imbalance on the annotated list. Within these types, the best result on the training pair seemed to perform equally well on scope change and metaphor/metonymy (see Figure 2), with some reduced performance in the other category. On the validation pair scope change performs equally well as on the training pair, so it could be said that this type can be predicted most easily.

Finally, **RQ3** has to also be answered somewhat inconclusively. With the right parameters some classifiers show decen results after alignment, but not as good as the cross-validation shows (Table 6).

There were quite some limitations in this research, so there is much room for future work here. Firstly, Gutenberg and Wikipedia have texts from quite different domains, and many detected shifts were actually noise as a result of this. For example the word "bye" as a shortening of "goodbye", which would only appear in a conversational text, rarely appears on Wikipedia, but its sports sense does, which is rare in the books. Thus this was detected as a shift despite both senses having existed fora long time. There is a balanced corpus of English text spanning the years 1810-2009, known as COHA[10], but this is sold for a rather high price and was thus not available to me.

Furthermore, this also ties into the small size of the annotated list, which was time-consuming to create due to the many words that only seemed to be detected as shifts due to the differences in corpora. It was also complicated to classify many words, since they seemed to involve aspects from multiple categories or simply not enough information could be found on them. I am also not an expert on linguistics by any means, so having a group of linguists come to a consensus over a bigger list could potentially greatly improve results.

Another method that has not been attempted in this research is to use a dynamic embedding method instead of alignment, with more time periods. These could give a more detailed look into shifts across time, with a word that has undergone multiple shifts being able to be separated into its different shifts due to the gradual process being explicit in the dynamic embeddings.

And finally, there is the method of Word2Vec itself. In recent years a different type of embedding method has emerged resulting in contextualized embeddings. These methods produce different embeddings for every context a word appears in, and in grouping these embeddings together different senses that a word has can be represented as different vectors. This could make the detected semantic shift more precise for polysemous (having multiple meaning) words, as the vector would be free from the influence of other senses that have not undergone a semantic shift. However, this method is not perfect in detecting polysemy [13] and not always better than non-contextual methods [19].

7. ACKNOWLEDGEMENTS

[10] Available from https://www.corpusdata.org/

I would like to thank my supervisor, Shenghui Wang, for giving me important directions and advice for this research, and the Intelligent Interaction track chair, for the useful info and feedback sessions.

8. REFERENCES

[1] R. Bamler and S. Mandt. Dynamic word embeddings. In *ICML*, 2017.

[2] F. Bianchi, V. Di Carlo, P. Nicoli, and M. Palmonari. Compass-aligned distributional embeddings for studying semantic differences across corpora. *ArXiv*, abs/2004.06519, 2020.

[3] L. Bloomfield. *Language*. George Allen & Unwin, 1933.

[4] A. Darmesteter. *La vie des mots*. Delagrave, 1887.

[5] W. L. Hamilton, J. Leskovec, and D. Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. *CoRR*, abs/1605.09096, 2016.

[6] A. Kutuzov, L. Øvrelid, T. Szymanski, and E. Velldal. Diachronic word embeddings and semantic shifts: a survey. In *COLING*, 2018.

[7] M. Martinc, P. K. Novak, and S. Pollak. Leveraging contextual embeddings for detecting diachronic semantic shift. *ArXiv*, abs/1912.01072, 2020.

[8] C. McCormick. Word2vec tutorial - the skip-gram model. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/, Apr. 2016. Accessed: 27-06-2021.

[9] C. McCormick. Word2vec tutorial part 2 - negative sampling. https://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/, Jan. 2017. Accessed: 27-06-2021.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[11] S. Mitra, R. Mitra, M. Riedl, C. Biemann, A. Mukherjee, and P. Goyal. That's sick dude!: Automatic identification of word sense change across different timescales. In *ACL*, 2014.

[12] S. Montariol. *Models of diachronic semantic change using word embeddings*. Theses, Université Paris-Saclay, Feb. 2021.

[13] S. Montariol, E. Zosa, M. Martinc, and L. Pivovarova. Capturing evolution in word usage: Just add more clusters? *Companion Proceedings of the Web Conference 2020*, 2020.

[14] H. Paul. *Prinzipien der Sprachgeschichte*. Niemeyer, 1880.

[15] K. Reisig. *Professor K. Reisig's Vorlesungen über lateinische Sprachwissenschaft*. Lehnold, 1839.

[16] M. Rudolph and D. Blei. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1003–1011, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[17] E. Sagi, S. Kaufmann, and B. Clark. Tracing semantic change with latent semantic analysis. *Current Methods in Historical Semantics*, pages 161–183, Dec. 2011.

[18] D. Schlechtweg, A. Hätty, M. D. Tredici, and S. S. im Walde. A wind of change: Detecting and evaluating lexical semantic change across times and domains. In *ACL*, 2019.

[19] D. Schlechtweg, B. McGillivray, S. Hengchen, H. Dubossarsky, and N. Tahmasebi. Semeval-2020 task 1: Unsupervised lexical semantic change detection. *ArXiv*, abs/2007.11464, 2020.

APPENDIX

A. PARAMETERS WORD2VEC

Corpus	Parameters[a]
1800-1849 (raw)	`vector_size = 300, window = 5, min_count = 50, sg = 1`
1800-1849 (lemma)	`vector_size = 300, window = 5, min_count = 100, sg = 1`
Wikipedia (raw)	`vector_size = 300, window = 5, min_count = 200, sg = 1`
Wikipedia (lemma)	`vector_size = 300, window = 5, sg = 1`[b]
1750-1799	`vector_size = 300, window = 5, min_count = 30, sg = 1`
1900-1949	`vector_size = 300, window = 5, min_count = 100, sg = 1`

[a]Parameters not given are the default in Gensim.
[b]Other parameters unknown (pretrained).

B. WORD LIST

B.1 Training list (1800-1849 - today)

Word	Label	Word	Label
tally_NOUN	0	campaign_VERB	1
intriguing_ADJ	2	devious_ADJ	1
lap_NOUN	2	franchise_NOUN	2
virtual_ADJ	0	milestone_NOUN	1
serving_NOUN	0	calculator_NOUN	1
fatality_NOUN	2	episode_NOUN	0
screen_VERB	1	garner_VERB	1
stereotype_NOUN	1	rack_VERB	2
rap_NOUN	2	picket_VERB	1
power_VERB	2	documentary_NOUN	0
craft_VERB	1	gig_NOUN	2
gay_ADJ	2	definitely_ADV	2
commercial_NOUN	2	cockpit_NOUN	1
dramatically_ADV	2	rendition_NOUN	1
animated_ADJ	1	shrink_VERB	1
catcher_NOUN	0	impact_NOUN	1
outstanding_ADJ	1	sampler_NOUN	2
guy_NOUN	0	film_NOUN	2
destroyer_NOUN	1		
squad_NOUN	1		
moonshine_NOUN	2		
album_NOUN	2		
retarded_ADJ	0		
hectic_ADJ	0		
quite_ADJ	2		
clinch_VERB	1		
assist_NOUN	0		
sedate_VERB	0		
brand_NOUN	1		
closet_VERB	0		
trans_ADJ	0		
untitled_ADJ	0		
party_VERB	0		
ejaculate_VERB	2		
concourse_NOUN	0		
unseasonably_ADV	0		
stint_NOUN	1		
urge_NOUN	1		
task_VERB	1		
portmanteau_NOUN	1		
installation_NOUN	1		
annexed_ADJ	0		
peak_VERB	1		
acoustic_ADJ	0		
presently_ADV	1		
expectancy_NOUN	0		
commitment_NOUN	2		
scribe_VERB	0		
cameo_NOUN	1		
coach_NOUN	1		
ill_ADV	0		
trend_NOUN	1		
alongside_ADV	1		
unavailable_ADJ	0		
focus_VERB	1		
figure_VERB	1		
caller_NOUN	2		
home_VERB	1		
bumper_NOUN	2		
operative_NOUN	0		
chair_VERB	1		
fag_NOUN	2		
cartel_NOUN	2		
shaver_NOUN	2		
jade_NOUN	2		
chum_NOUN	2		
unused_ADJ	2		
famously_ADV	2		
merchandise_VERB	1		

B.2 Validation list (1750-1799 – 1900-1949)

Word	Label
radical_NOUN	1
abstractedly_ADV	2
divan_NOUN	2
gamut_NOUN	0
deplete_VERB	0
denizen_NOUN	0
bored_ADJ	2
civilian_NOUN	1
experimentally_ADV	0
monitor_NOUN	1
projector_NOUN	1
electorate_NOUN	0
obstreperous_ADJ	0
awfully_ADV	2
exploit_VERB	2
reclamation_NOUN	2
salamander_NOUN	1
exponent_NOUN	2
sporadic_ADJ	1
outfit_NOUN	0
recording_NOUN	1
lot_NOUN	2

C. INITIAL CLASSIFIER RESULTS

Classifier	Parameters[a]	Accuracy	Tokens
KNeighborsClassifier	`n_neighbors = 5`	0.397 ± 0.151	raw
KNeighborsClassifier	`n_neighbors = 10`	0.330 ± 0.137	raw
KNeighborsClassifier	`n_neighbors = 20`	0.375 ± 0.121	raw
KNeighborsClassifier	`n_neighbors = 40`	0.397 ± 0.151	raw
RandomForestClassifier	-	0.466 ± 0.138	raw
GaussianNB	-	0.352 ± 0.129	raw
BernoulliNB	-	0.443 ± 0.163	raw
LogisticRegression	-	0.386 ± 0.067	raw
SGDClassifier	`loss = "log"`	0.386 ± 0.093	raw
SGDClassifier	`loss = "hinge"`	0.409 ± 0.109	raw
SGDClassifier	`loss = "modified_huber"`	0.386 ± 0.066	raw
SGDClassifier	`loss = "squared_hinge"`	0.409 ± 0.147	raw
SGDClassifier	`loss = "huber"`	0.398 ± 0.121	raw
SGDClassifier	`loss = "perceptron"`	0.432 ± 0.068	raw
SVC	`kernel = "linear"`	0.386 ± 0.053	raw
KNeighborsClassifier	`n_neighbors = 5`	0.318 ± 0.087	lemma
KNeighborsClassifier	`n_neighbors = 10`	0.318 ± 0.115	lemma
KNeighborsClassifier	`n_neighbors = 20`	0.364 ± 0.145	lemma
KNeighborsClassifier	`n_neighbors = 40`	0.386 ± 0.124	lemma
RandomForestClassifier	-	0.398 ± 0.157	lemma
GaussianNB	-	0.420 ± 0.138	lemma
BernoulliNB	-	0.363 ± 0.123	lemma
LogisticRegression	-	0.432 ± 0.125	lemma
SGDClassifier	`loss = "log"`	0.432 ± 0.125	lemma
SGDClassifier	`loss = "hinge"`	0.455 ± 0.088	lemma
SGDClassifier	`loss = "modified_huber"`	0.398 ± 0.068	lemma
SGDClassifier	`loss = "squared_hinge"`	0.363 ± 0.122	lemma
SGDClassifier	`loss = "huber"`	0.307 ± 0.112	lemma
SGDClassifier	`loss = "perceptron"`	0.420 ± 0.076	lemma
SVC	`kernel = "linear"`	0.432 ± 0.097	lemma

[a]Parameters not given are the default in scikit-learn.