Trinity College Dublin
The University of Dublin

THE EFFECTS AND USES OF TARGET WORD POSITION IN UNSUPERVISED N-GRAM WORD SENSE DISAMBIGUATION

Fourth Year Project

August 2021

Supervisor: Martin Emms

Johnny Scanlon
Scanloj1@tcd.ie

Plagiarism Declaration

"I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write."

Abstract

The aim of this project was to investigate the effects of target word position in unsupervised word sense disambiguation. The EM algorithm is used to perform this word sense disambiguation on these N-grams.

To examine the effects of position the project starts by fixing target word position in these 5-grams. Using a diachronic model, the per year probabilities of detected senses can be graphed. By graphing these sense probabilities, the emergence of neologisms and relative sense probabilities can be seen. By comparing these to known values, the effect of target word position can be examined and compared. The project then examines more complex models, examining how position is influencing these results.

Pseudo-neologisms are used to make fairer comparison between different models and perform more detailed exploration of edge cases.

There was an observed difference between n-grams of different target word positions. Methods for using these differences to improve results in more complex models are then explored with positive results.

Ultimately it is shown that target word position is relevant to the success or failure of a word sense disambiguation, and knowledge of it can be used to improve the model and get better results.

Acknowledgments

I would like to thank Martin Emms for all his help and continued support throughout this project.

A huge thanks to my mother whose continued support made a difficult year that bit easier.

Thanks to my father for his unending support in getting me this far.

Contents

Table of Figures

1. Introduction

"You shall know a word by the company it keeps" ~ John Rupert Firth [1]

This quote, made by celebrated linguist John Rupert Firth in the 1950s, is something of a mission statement for the whole field of distributional semantics. The founding philosophy of this field is that the meaning of a word can be determined from the words which surround it, and that words with similar words surrounding them have similar meanings.

This concept underpins this project. Word sense disambiguation is the computational task of determining the meaning of a given target word [2]. Typically, this is done using the context words that surround the target word. For example, take the word mouse. Mouse has two senses: the longstanding sense related to the small rodent, and the newer sense related to the computer device which emerged in the 80s. An arbitrary occurrence of this word might go something like:

"…the cat caught a mouse in its mouth…"

From this sentence, one would presume that the sense in question is the rodent type. In doing so a kind of word sense disambiguation has been carried out. Had words such as computer, pointer, or monitor been there, one may have concluded that it's the computer type of mouse.

These context words have influenced the concluded sense of the word. Considering that every language has its own inherent structure, one may ask the question is there some internal structure as to which words most influence this sense?

For example, in English adjectives typically fall before the object they are describing, e.g., the *tall* tree, the *fat* cat. If one was attempting to discern the sense of a given adjective, one might presumably look for the object which it is describing. This invariably means looking to the right of this target word.

The aim of this project to investigate the positional nature of where meaning may lie. In particular, this project examines whether words closer to the target word have generally contribute more to meaning than those further away. By examining clusters of words, with the target word in various positions, it can be determined whether the position of the target word has an effect. It can also be determined which positions are most effective. In essence, this is an exploration of where best to find clues as to the sense of a word.

For humans, this may seem obvious as we excel at the sort of lateral thinking on which language is based. For machines however, this is not the case. Overcoming machines weakness for lateral thinking has been a major challenge for many computational problems. For word sense disambiguation, along with many other problems, the answer lies in constraining the problem. Being aware of whereabouts position may lie allows one to do this more effectively.

This project aims to explore the influence of position, by first explicitly restricting the position of the target word in these context clusters. After learning about how the position of the target word can affect results it then looks at more complex models and how target word position may be affecting these, either positively or negatively. Finally, it explores a possible method of explicitly using the effects of target word position to the advantage of this word sense disambiguation.

This project uses a diachronic model to graph the uses of a words sense probabilities over a time span of 150 years. In theory this should capture the emerging senses a word can take on, as sense probability rises from 0% before a certain date to some natural number representing its use. It also gives a sense of how the use of a certain word changes over time. By comparing the resulting model

depicting word sense, a model can be evaluated by comparing it to known dates of sense emergence.

1.1 Motivation

From language translation to personal home devices like Amazon's Alexa, the need for computers to understand human language has never been greater and only seems to be growing. Paired with the growing ubiquity of the Internet worldwide, the need for useful natural language processing tools has never been greater. This trend has been a constant push for development in the field of computational linguistics.

Much of the developments in recent years, like many areas of machine learning, have relied on large databases. This has of course laid bare a kind of inequality in the quality of natural language processing, between languages. Alas, not all languages are blessed with hundreds of millions of speakers and decades of digitised data to work with.

One need only look at the stark contrast in translation accuracy between Google translations from minor languages and major languages, for this to be seen in full effect. The internet and other such digital resources can be difficult to access for minority language speakers [3]. While this technology has come a long way there is still much to be done.

Central to so much of this language processing is word sense disambiguation. For a translation to be accurate it must know which mouse one is talking about. Likewise for any personal home assistant. Indeed, unsupervised word sense disambiguation is such a core problem, it has been described as an AI complete problem [2]. That is to say solving it would be akin to solving the central problem of AI.

It is with this in mind that this project seeks to explore the effect of a words position in a word sense disambiguation. The aim of the project is to get a better grasp on how position is affecting these results.

1.2 Project Goal

The goal of this project is to inspect the effects of target word position in an n-gram on the ability of an EM algorithm to accurately determine word sense and identify the date of emergence for a neologism. Following this, it examines more complex models and seeks to explore how this can be used to the benefit of better results.

1.3 Readers Guide

Here an overview of how the report is laid out:

Chapter 2: Literature Review

> This Chapter covers existing and related work upon which this project builds. As well it seeks to give some background as to the ideas, as well as give an idea of the current state of the art and how this work relates to it.

Chapter 3: Ideation

> This chapter covers to the thinking and concepts in working towards a solution.

Chapter 4: Design

This chapter covers the design of what was developed for this project.

Chapter 5: Results

This chapter covers the results of the run tests

Chapter 6: Conclusions

This chapter seeks to draw conclusions from the results.

Chapter 7: Future work

This chapter seeks to examine where further research could be done, to further flesh out the conclusions from this project

2. Literature Review

This section seeks to give some background to the concepts involved in this project. In doing so, it also helps to place this project in the context of current and past work in the field.

2.1 Unsupervised Word sense Disambiguation

Word sense disambiguation is the task of determining the meaning of a word, in a computational manner. While the conceptual idea behind this was delineated briefly in the introduction section, a better outline of its history is offered here with our focus on it as computational task.

Word sense disambiguation is a part of the wider field of natural language processing, a field of study at the crossroads between linguistics, computer science, and artificial intelligence. It was first introduced as a computational problem by Warren Weaver in 1949 [4], and has been a central problem to the field of Linguistic computation and Machine Learning ever since.

The problem poses such a core problem to artificial intelligence that it has been described as an AI-complete problem[2]. Solving this problem completely is equivalent to solving the central problems of artificial Intelligence. With these problems so intrinsically linked, it is no surprise that any advancements in AI often lend themselves to an advancement in Word Sense disambiguation. Indeed, the digital revolution proved revolutionary for WSD as it did for AI.

From its early days as a computational problem the issue of data proved significant. Much of machine learning opted for a supervised approach, which proved problematic as no large-annotated corpora existed. Significant strides began to be made in this field in the 80s, as the first large-scale, annotated, machine readable corpora became available. This opened doors for more robust techniques, supervised and unsupervised alike. The issue of Corpora is handled with more detail in section 2.3.

The advent of statistical approaches in the 90s and Deep Learning in the 2000s, opened up new avenues for word sense disambiguation. This applied for supervised and unsupervised methods alike.

The issue of supervised and unsupervised corpora still proves an issue for word sense disambiguation. Many of the current best practises rely on large, annotated corpora. Such corpora are a scant resource. The time consuming and resource draining creation of such resources is required for each new implementation, be it language, domain, or even sense. This has proven a fundamental issue to the field, known as the knowledge acquisition bottleneck [5].

For this reason, efficient design becomes important. This project uses an unsupervised approach to see how position of a target word can affect results. This is done with an aim for better understanding the process of word sense disambiguation and finding a more useful method for doing so.

2.2 The unsupervised approach and the Diachronic Model

Known as Word sense Induction, the unsupervised approach has proven a serious challenge for WSD researchers. These approaches seek to derive meaning from unannotated data.

The unsupervised approach typically seeks to derive meaning by examining the context of words, the documents they are included in, or comparing words which share similar syntactic use [5]. As such, an unsupervised approach is typically defined by its approach to clustering. Clustering, in natural

language processing sense, refers to techniques for grouping words with similar syntactic meaning. For unsupervised word sense disambiguation, this means identifying the words most linked to a given sense. Returning to the example mouse, this could mean identifying words like cheese and trap for the rodent kind, and computer and click for the computer kind.

Much of the work in this field follows the work of Schutze [6]. Since then, many approaches to clustering have been taken. This project uses an n-gram approach, as used by Schutze in his 1998 paper [7]. An n-gram refers to a small context window of length N. for example, take the snippet:

…the mouse is in the…

This project builds heavily on a paper and code base developed by Martin Emms and Arun Jayapal, which used the expectation maximisation algorithm to estimate the yearly sense probabilities [8]. In this way a diachronic model of sense is built, showing sense use over time. This project seeks to analyse the effect of target word position in these n-grams.

The diachronic model used in this project is base off of work done by Arun Kumar Jayapal and Dr. Martin Emms, where it used to detect sense emergence [9]. This is outlined below. Here Y is the year, S is the sense, and w is the word string.

$$P(Y,S,w;\pi_{1:N},\theta_{1:K},\tau_{1:N}) = P(Y;\tau_{1:N})P(S|Y;\pi_{1:N})\prod_i P(w_i|S;\theta_{1:k}) \quad \text{-(1)}$$

Simplifying this gives

$$P(Y,S,w) = P(Y)P(S|Y)\prod_i P(w_i|S) \quad \text{-(2)}$$

These parameters here are estimated using the EM algorithm.

2.3 The EM Algorithm

The expectation maximisation (EM) algorithm is an iterative technique to find *a posteriori* estimates for local maxima or minima of some parameters.

In short, the algorithm looks at finding the maximum likelihood estimate of some parameters by examining an incomplete dataset. The estimated parameters will be those most likely to return the observed data. In the case of this project the parameters in question are those of the diachronic model outlined in section 2.2. The code base used for this was developed for a 2015 paper titled "An unsupervised EM method to infer time variation in sense probabilities" by Dr. Martin Emms at Trinity College Dublin [8].

The EM algorithm functions by iteratively applying two steps: the E step and the M step.

The E-step

Given a training set D = (d_1,d_2,d_3,…,d_n), the E-step generates a virtual complete corpus by completing each individual incomplete data item, x_d, with all possible values for S, P(S=k, x_d). A conditional probability (S=k| x_d;θ_n) is then calculated for each using the current best estimates for parameters, θ^n. This conditional probability is denoted by $\gamma^d(k)$.

The M step

The algorithm then uses this $\gamma^d(k)$ value as counts and applies maximum likelihood estimation to derive new parameter estimation, θ^{n+1}.

In this way the algorithm iteratively moves towards a local maximum or minimum.

It is important to recognise that while this algorithm is guaranteed to find a local maxim or minima, this may not be the global maxima or minima. As such, it is not a perfect algorithm.

The EM algorithm arrives at these probabilities with no knowledge of what the sense means. Rather it estimates from the words around it the given probability of it being one sense or another with no deeper view as to what these senses are.

2.4 Corpora

The term corpus traditionally refers to a large collection of text. Such large collections of text are at the heart of many natural language processing problems. Word sense disambiguation is no different. Both supervised and unsupervised methods have long been helped or hindered by the availability of suitable corpora for training.

Supervised methods rely on large, laboriously annotated corpora. While much work has been done in creating suitable corpora, for example WordNet, the sheer effort needed to label every word is a serious barrier. It is for this reason that this project focuses on the unsupervised case.

This project focuses on two separate corpora: The google 5-gram corpus and the COHA corpus.

2.4.1 Google 5-gram Corpus

The google 5-gram corpus is in many ways a non-traditional corpus. Whether the term “corpus” should even apply to it, with its traditional meaning of a large collection of texts, could be a matter for debate.

Rather than supplying collections of texts, through which a program must search line after line for a particular word, the Google 5-gram corpus simply delivers a list of 5-grams and a frequency count. These 5-grams were found from the massive collection of digitised texts from google books.

Rather than providing texts from which a program must find a target word occurrence and create a 5gram from, the google corpus provides these 5-grams readymade. However, this produces an interesting result. Take an occurrence:

> “the cat and the mouse ran around the house”

Taking mouse as the target word the creators of the google n-gram corpus turned this into 5 5-grams, Listed below.

The cat and the **mouse**

Cat and the **mouse** ran

And the **mouse** ran around

The **mouse** ran around the

Mouse ran around the house.

The words right beside the target word are included in 4/5 fivegrams, while the words at the edge only occur in one each. In its creation, the google 5-gram corpus results in an overcounting of words that occur near to the target word.

There may be some unintended logic to this, as it is possible that words nearer a target word may tend to carry more meaning. Thus, this overcounting may help important words rise above the noise. This is directly tested in this project.

Unfortunately, the Google 5-gram corpus has no way of rewinding and finding the original occurrences as they occur in the text.

It is for this reason, the more traditional and flexible COHA is used as well.

2.4.2 COHA Corpus

The COHA (corpus of historical American English) corpus is exactly this; a more traditional, more flexible corpus. Made by a team out of Brigham Young University, the corpus contains about 400 million words. Far less than the trillion contained in the Google N-gram corpus, but still a considerable amount. Seeing the size of these corpora, the need for an unsupervised method should be apparent.

The COHA corpus provides lines of text which must be processed to find a given target word. The benefit is of this is that the broken up 5-gram case can be compared to the original 9-gram case. It is also open to other approaches using different weighting models. This increased flexibility, coupled with a more manageable size for tests (a run only taking about 15 minutes), makes it a very suitable corpus for this project.

Of course, this increased flexibility comes with an increased complexity in the code needed to process it, as shall be seen in the design section.

2.5 Pseudo-Words and Pseudo-Neologisms

The use of an unsupervised learning method brings with it one major challenge: the correct answers are not known. This can make it challenging to fairly assess and compare how well a given model has worked.

An answer to this was found in the use of pseudo-words. Used and described by Schultz in his 1998 paper [7], pseudo-words operate by combining two distinct words with two distinct meanings into one “pseudo-word” which covers both meanings. These pseudo-words then replace the occurrences of both original words in the corpus. This creates an ambiguous word within the corpus with two distinct senses.

An unsupervised word sense disambiguation can then be performed on these pseudo words, and the results compared to the true known values of the original words.

Perhaps best described with an example, take the word banana and door. Each of these words has an established sense. By combining them into a pseudo-word, banana/door, a single word is created which covers both original senses. In this way the algorithm can be tested on ambiguous words for which there are real values. When using pseudo-words, the computer changes all occurrences of either banana or door to this pseudo word.

A twist on these pseudo words, tailored to a diachronic model is outlined in a paper by Arun Jayapal [10]. Referred to as pseudo-neologisms, these are pseudo words, where one of the combined words in question sees no use before a certain date. As such, these pseudo-neologisms can be used to mimic the emergence of a new sense in our diachronic model.

2.6 Current best methods

The Deep Learning revolution affected many machine learning problems. Word sense disambiguation is no different. Current best approaches, like BERT and ELMo, use Deep Learning to some degree [11].

However, with this in mind it does not make work done with Bayesian methods obsolete. Deep Learning is still in many ways a black box. For future improvements it is necessary to understand why the previous models have worked so well.

It is with regard to this that the project is carried out. This project seeks to understand something of the mechanics of where meaning can be found in the words that surround a target word.

While any improvements made in this project are unlikely to take the crown from these current leaders, they should hopefully illuminate something as to the nature of word meaning and how it can be used to an advantage.

3. Ideation

This section outlines the thinking as well as some of the theory and terms used in this project.

The aim of this project is to examine the effect of target word position in this N-gram analysis by examining the effect on a diachronous model.

The project examines the effect of position in several different ways. Initially, it performs position fixing in 5-grams, and compares the results. Building on this, more complicated models are compared. There are three more complicated models tested: the "any" case, the 9-gram case, and the new model case. The exact details of these are explained later in this section.

To aid comparison Pseudo-neologisms, as described in section 2.5, are used. Pseudo-neologisms help provide tangible numbers by which to make comparisons as well as fairly evaluate performance.

In short, the project can be thought of as moving in different phases of analyses on the effects of position. Starting rough by looking directly at individual position locked n-grams, before moving on to comparing more complex models. Finally, it seeks to develop an improved model for analysis.

Roughly breaking this up into several phases:

- Does Target word Position matter? Initial rough tests into position locking and comparing models
- What is the extent of this? This is a more detailed look into the effects of position. Pseudo-neologisms are used to compare models and positions. Models explicitly compared using the resulting calculated error.
- Developing a new model. Looking to see if a better model can be developed.

By starting by small, analysing position before building onto more complex models and using pseudo-neologisms, the effects of position should hopefully become more apparent. It also helps for keeping track of results, to keep in mind what is being tested and why.

3.1 Inspecting Target position in real words

This section concerns itself with how this project goes about examining the effects of target word position for real words. Terms like position-fixing and models have been used in the project thus far. This section seeks to clearly lay out what is meant by these terms and the logic behind them.

3.1.1 Fixing Target word Position

Restricting target word position may seem a straightforward idea. However, it is important to consider the fundamental differences in the corpora being used. While the Google 5-gram corpus provides long lists of ready-made 5-grams, the COHA provides lines of text. As such the approach to restricting target word position in a 5-gram must be different for each corpus.

To lock position for the google 5-gram, only 5-grams where the target word is in the desired position are read in. There may be different numbers of 5-grams for different positions. This could lead to disparity in results and should be considered upon analyses.

When reading in the COHA the 5-grams are created by the programme. As such, the approach here is to create these with the target word in the desired position. A result of this is that the COHA corpus deals with no disparity in counts between 5-grams of different target word positions.

By performing the EM algorithm, the resulting diachronic models for 5-grams of different target word positions can be compared. As such whether certain positions prove more effective should become apparent.

3.2.1 Different Models: 9-gram vs Any

Reference has also been made to more complicated models. By model what is meant is how to programme handles this reading in of N-grams. These more complicated models contrast with the simple position locked case. They are not necessarily more conceptually complicated, but rather they include a wider variety of 5-grams.

Two of the main models used are the ANY case and the 9-gram case.

The Any Case

The ANY case refers to a model which accepts 5-grams with the target word in any position. As discussed in Section 2.4.1, this is the natural way to approach the google 5-gram. Also discussed was the overcounting effect of words nearer the target word by doing this. This "any" case is applied to the COHA corpus by similarly breaking up any occurrence into five sperate 5-grams.

The 9-Gram Case

To contrast this a 9-gram case is also used. This is most likely the most obvious way of approaching the 9-gram case. This model takes a 9gram with the target word at its centre. This is only applicable to the COHA corpus. This is useful as it can be used to directly compare the effect of breaking an occurrence up into smaller 5-grams against the original 9gram from which those 5-gram are formed.

Forming these cases

To visualise how these models work for the COHA corpus, it is best to use an example. The below table contains an occurrence and shows the resulting N-grams produced from this occurrence.

Occurrence	I was walking downstairs when I saw the mouse caught in the trap that had been left out.
9-gram	downstairs when I saw the mouse caught in the trap
ANY	downstairs when I saw the mouse I saw the mouse caught saw the mouse caught in the mouse caught in the mouse caught in the trap

Figure 1 - Forming N-gram cases

By contrasting these models, the effect of this central overcounting can be observed. By comparing with the position locked cases, which positions are most effective can be observed.

3.2 Using Pseudo-Neologisms

After these initial tests by position locking, an idea of the effects of target word position should apparent. However, making fair comparisons between models can be a challenge. The unsupervised approach means there is no “correct” answer to compare the results of our learning algorithm. To fairly compare models, tools for evaluating results are needed.

3.2.1 Pseudo-Neologisms

The theory behind pseudo-neologisms is outlined in section 2.5. This section seeks to describe how this is applied for use within this project and some of the hurdles that go with it.

Using pseudo-neologisms not only allows for better evaluation of results. They are also useful for testing edge cases. By knowing the sense probabilities beforehand, cases where one sense is significantly more common can be tested. Ultimately the goal of using these pseudo neologisms is to better evaluate results and more thoroughly test edge cases.

By measuring the error between the actual and estimated values a value for fair comparison between models can be found.

3.2.2 Pre-Processing COHA to identify words

When making pseudo-neologism, it is necessary to find suitable true neologisms as well as steady words.

To find a true neologism, a word that saw no use before a certain date is needed. Finding suitable steady words entails finding words that saw reasonable steady use throughout the span of the corpus. It is also necessary to find words with a particular total frequency count (i.e. how often it occurred in the corpus).

This results in looking for words with quite particular requirements. Given how large the COHA corpus is, processing it once for a particular word can take up to 20 minutes. To avoid a time-consuming trial-and-error approach, it is necessary to introduce some pre-processing functions.

The idea is to have some method of organising the corpus and identifying suitable words.

3.3 A new model

Having analysed the effects of position individually in the form of position locked 5-grams, as well as more complicated models like the “any” and 9-gram case,

This explores the idea of actively harnessing the observed positional effects to improve results. This is an area that could become very complex. This aim of this project is more to explore whether improving results can be done, rather than offer a comprehensive view of how results can be viewed to the maximum effect.

While more complex models featuring informed left or right bias could be implemented, this project chooses to combine the weighting of the “any” case with the 9-gram case.

The logic behind this is to combine this central weighting effect, which emphasizes words closer to the target word with the more zoomed out 9-gram approach. This idea was arrived at having observed the benefits of central weighting, but also recognising the benefit of the 9-gram in some cases.

4. Design

This section gives an in-depth view of the additional code developed for this project. Much of the code builds on a code base developed by Martin Emms and Arun Jayapal [8], as described in Section 2.2. This codebase implemented the EM algorithm on a variety of different corpus types, including the Google 5-gram and the COHA corpus.

For use in this project several key features were developed and added.

- Position Fixing Code and handling different models
- Pre-processing the corpus to identify suitable words for pseudo-neologisms
- Creating and handling pseudo-neologisms

4.1 How the codebase handles data and the EM algorithm

Before explaining the development of additional code, it helps to have some understanding of how the codebase operates.

The codebase operates by reading in data from several types of corpora and performing the EM algorithm on the resulting data to return year probabilities for sense. This programme is named dynamicEM. This programme essentially works in two phases the first is a reading in of data, the second is performing the EM algorithm on this data.

The different types of corpora require very different methods for reading in data. The EM algorithm is shielded from this by using an occurrence class object. The idea here is that regardless of corpus the data is read in and stored in this class object. The EM algorithm is then designed to work on a vector of these class objects.

The class object is named Occ, and stores details such as context words and year.

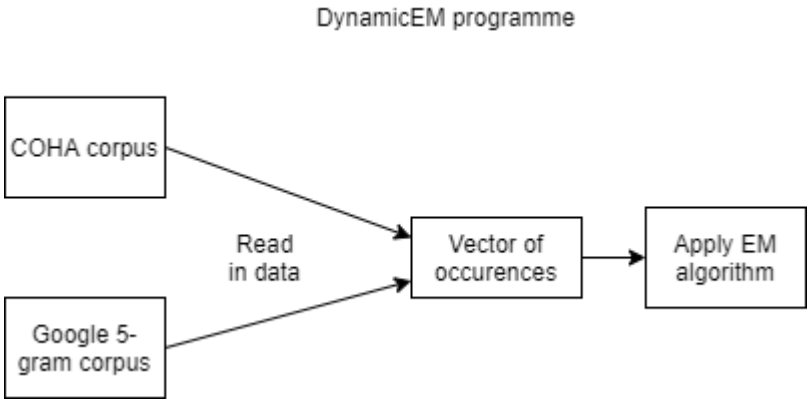


Figure 2- DyanmicEM programme

Coding efforts should be focused to before the creation of this vector. This minimises the need for change in other parts of the code.

4.2 Position Fixing and Handling Models

Position fixing refers to fixing the position of the target word in the N-gram. This way the EM algorithm only sees N-grams with a specific target word position. Handling the more complex models similarly uses code affecting the kind of N-grams that are produced for the EM algorithm to operate on.

From Section 4.1, it was seen how the code essentially isolates the EM algorithm from the details of the different corpora, instead providing it with an Occ occurrence class object. As such, efforts at position fixing and handling these different models should be focused on what occurrences are being added to this vector of Occ occurrences which is then shown to the EM algorithm.

The different approaches for the different corpora are outlined as follows.

4.2.1 Google N-gram: Position Fixing and Models

To best approach position fixing for the Google 5-gram corpus it's important to consider the nature of the corpus. Ready-made 5-grams with year and frequency counts. The target word can be in any position in these 5-grams.

The code restricts position at the reading in of the 5-gram from the document. As its read in, the target word position is checked. If it is in the desired position an occurrence is made. If it is not than that 5-gram is passed, and the next line is read in.

This ensures that at the end of reading in the data, the vector of occurrences on which the EM algorithm operates only contains those 5-grams with the desired target word position.

For the any model, all 5-grams result in a respective occurrence which is added to the vector of occurrences.

```
Read in next 5-gram line

While(document end has not been reached)
        if(model = any)
                make occurrence from line

        else if(model = position fixed)
                if(target word position equals desired position)
                        make occurrence from line

        Read in next 5-gram line
```

Figure 3 - Pseudo Code for making models in Google corpus

4.2.2 COHA: Position Fixing and Models

In contrast to the Google Corpus, the COHA provides lines of text from which occurrences of a certain word must be identified. Upon finding the given target word, the context words are extracted, and an occurrence made. The result is still a vector of occurrences.

Position Fixing

Any time an occurrence is located, a 5-gram is created with the target word in the desired position.

This means extracting the correct number of words to the left and right of the target word. For example, if the desired 5-gram position was 1, the code would take no words to the left of the target word and four words to the right of the target word when making the occurrence.

For example, given an occurrence: the orange cat caught a mouse in its mouth yesterday.

If the desired position was position 3, this would result in the following 5-gram:

caught a mouse in its

An Occ object is then made for this 5-gram and added to the final vector of occurrences.

ANY model

The any model mimics the 5-gram google corpus. This case in the COHA corpus takes any occurrence of a target word and breaks it up into five 5-grams. Each of these 5-grams is given its own occurrence in the occurrence vector. As such to the EM algorithm it will see each of these 5-grams as induvial occurrences.

For example, take the same occurrence: the orange cat caught a mouse in its mouth yesterday.

For the any case this results in five separate 5-grams, with mouse in a different position in each one.

An Occ object is made for each of these 5-grams individually and stored on the final vector of occurrences.

9-gram model

The 9-gram simply creates a 9-gram occurrence with the target word at the centre and four words to the left and right.

Below is some pseudo code which shows how the creation of these models from reading in lines of text is handled.

```
Read in next word

While(document end has not been reached)
        if(next word is target word)
                if(model = positon)
                        take position words from the left and 5 - position words from the right
                        make occurence and and add to occurence vector

                else if(model = any)
                        for(int position = 1; position < 6; position++)
                                take position words from the left and 5 - position words from the right
                                make occurence and and add to occurence vector

                else if(model = 9-gram)
                        take 4 words from left and four words from right

        read in next word
```

Figure 4 - Pseudo code for making models in COHA

4.3 Processing COHA to identify true neologisms and steady words

When creating pseudo-neologisms, its necessary to use a suitable true neologism and a steady word. The theory behind this is outlined in section 2.5, while some of the practicalities as they relate to this project are explained in Section 3.2. A single run-through of the COHA corpus can take up to 20 minutes with the DyanmicEM programme. This can make finding suitable words a time-consuming task using a trial-and-error approach.

Code was developed to run through the corpus and identify suitable pseudo words as well as steady words. Rather than building on the original code base, this programme stands alone. It makes use of some of the functions from the original code. This programme is called COHA_pseudo_word_finder.

To identify words a struct to represent a word that occurs in the corpus is created. This struct contains the details of the word, total frequency, and a map that maps a year-to-year frequency.

```
struct corpus_words {
    string word;
    int total_freq;
    map<string,int> year_frequenies;
};


map<string,corpus_words> all_corpus_words;

map<string,corpus_words> pseudo_words_map;
```

The programme begins moving through the COHA corpus and as it goes adds any new word found to a map, called all_corpus_words. This map maps words to their corresponding struct. If this word is found again then its total frequency and corresponding year frequency is increased.

```
Read in next word

while(document end has not been reached)
        if(word already exists in map)
                increase total frequency
                increase relvant year frequency

        else
                add word entry to map

        read in next word
```

Figure 5- Pseudo code to read in all COHA words

By the end of processing the corpus the code is left with a map containing every word in the corpus as well as a struct containing its total frequency and its years frequencies.

4.3.1 Finding True Neologisms

A true neologism is a word which saw no use before a given target date. This date is somewhat arbitrary and can be set to the taste of whoever is running the tests. For the sake of this project, it was set to three decades of no use. The data begins from 1850. As such a neologism will be any word that did not see any use until at least 1880s. This is performed with the find_pseudo function.

```
void find_pseudo(){
  bool one_year = false;
  bool two_year = false;
  bool three_year = false;
  string one_year_f = "1850";
  string two_year_f = "1860";
  string three_year_f = "1870";

  corpus_words new_words;
  string word;

  map<string, corpus_words>::iterator corpus_word_itr;

  for(corpus_word_itr = all_corpus_words.begin();
    corpus_word_itr != all_corpus_words.end(); corpus_word_itr++ ){

    map<string,int>::iterator year_freq_itr;

    word = corpus_word_itr->first;

    new_words.word = word;
    new_words.total_freq = corpus_word_itr->second.total_freq;
    new_words.year_frequenies = corpus_word_itr->second.year_frequenies;

    year_freq_itr = corpus_word_itr->second.year_frequenies.find(one_year_f);
    if(year_freq_itr != corpus_word_itr->second.year_frequenies.end()){
      one_year = true;
    }

    year_freq_itr = corpus_word_itr->second.year_frequenies.find(two_year_f);
    if(year_freq_itr != corpus_word_itr->second.year_frequenies.end()){
      two_year = true;
    }

    year_freq_itr = corpus_word_itr->second.year_frequenies.find(three_year_f);
    if(year_freq_itr != corpus_word_itr->second.year_frequenies.end()){
      three_year = true;
    }

    if( one_year && two_year && three_year){
      pseudo_words_map.insert(pair<string,struct corpus_words>(word, new_words));
    }
    one_year = false;
    two_year = false;
    three_year = false;
  }
}
```

Figure 6 - Find Pseudo word from COHA function

Found neologisms are printed to a file with their total frequency and individual year frequencies. A suitable neologism can be selected from this file.

4.3.2 Finding Steady words

A suitable steady word requires two things. A suitable frequency count and a relatively steady year frequency count.

A function which cycles through the map of total corpus words and finds words with a total frequency inside a specified range is designed. This function prints out the results along with total word frequencies as well as per year frequencies.

```
void find_steady(){

  int frequency_lower_bound = 10000;
  int frequency_upper_bound = 12000;

  corpus_words new_words;
  string word;

  map<string, corpus_words>::iterator corpus_word_itr;

  for(corpus_word_itr = all_corpus_words.begin();
    corpus_word_itr != all_corpus_words.end(); corpus_word_itr++ ){

    word = corpus_word_itr->first;
    new_words.word = word;
    new_words.total_freq = corpus_word_itr->second.total_freq;
    new_words.year_frequenies = corpus_word_itr->second.year_frequenies;

    if( corpus_word_itr->second.total_freq > frequency_lower_bound &&
        corpus_word_itr->second.total_freq < frequency_upper_bound){

      pseudo_words_map.insert(pair<string,struct corpus_words>(word, new_words
));

    }

  }

}
```

Figure 7 - Find Steady words code

4.4 Pseudo Neologisms

How the code creates a vector of occurrences which is then passed on the EM algorithm is explained earlier in this chapter. For handling Pseudo-neologisms this structure is kept, with a vector occurrences containing N-grams passed on to the EM algorithm. It is also important that while doing this relevant are details about the true values as well.

To handle this this code is developed that initially handles the two target words as separate. A vector of occurrences for each individual word is created. These vectors are then merged into the final vector which is passed to the EM algorithm. In the way the EM algorithm is presented with a vector containing N-grams for both words. The original target word has no effect on this, it only sees the context words.

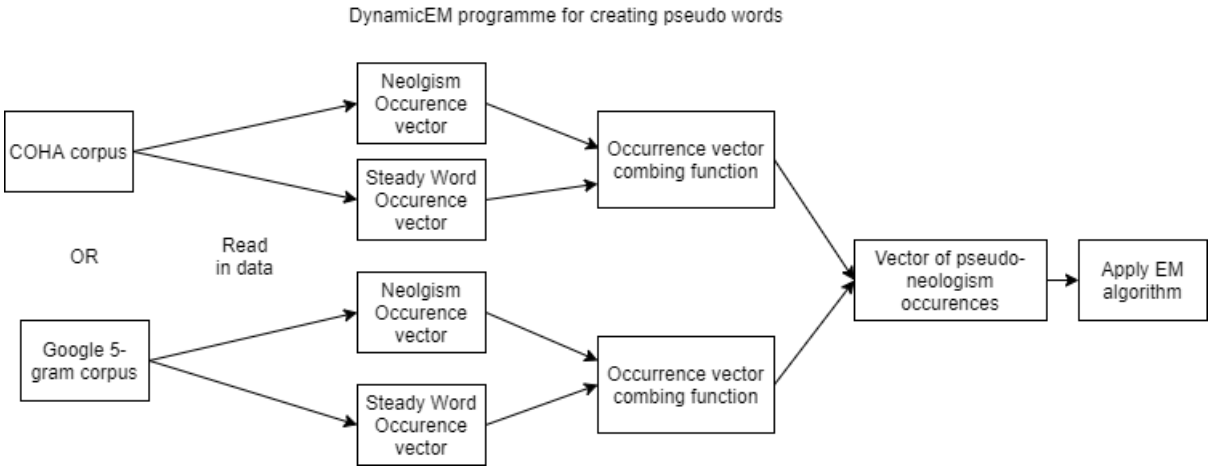


Figure 8 - DynamicEM for pseudo-neologisms

Real details about the original target words are recorded as they are read in, with real statistics being calculated as the two vectors are merged. How the code handles the reading in of data for both corpus types, as well as how the merge occurs are outlined.

4.4.1 Creating Pseudo-Neologisms

The basic theory behind how these pseudo neologisms has been detailed. This sub-section seeks to explain how this is handled for each corpus. As stated, the goal is to create two vectors of occurrence for each word. Further code is developed to combine these occurrences into a vector of pseudo-neologisms which is presented to the EM algorithms.

Google 5-gram

For the google corpus, a directory to the files is containing the 5-grams for the target word is included. To create pseudo neologisms code was added that uses two separate addresses. The code then handles these separately creating the vector of occurrences for each separately.

COHA

For COHA both target words are given as parameters. As the code searches the corpus it checks which target word has been found and adds the occurrence to its respective vector.

After the code has been searched through these vectors are then combined into the final pseudo neologism vector.

Combining vectors into a vector of pseudo words.

The individual vectors for the target word occurrences are combined into a single vector. The occurrence class contains the context words. These context words are what the EM algorithm looks at. As such to the EM algorithm there is no difference between those occurrences that came from either the neologism or the steady word.

This combine function is called combine_sense_for_pseudo, and is contained in the process_corpus file. This function combines the two vectors while also keeping the chronological structure contained in the original vectors. To do this it combines the vectors one year at a time. This is done by checking by first adding the occurrences which occur for a given year before moving onto the next year.

```
Read first neolgism occ from vector
Read first steady word occ from vector

While(not reached end of neologism vector OR not reached end of steady word vector)

        if(neolism occ year is less than steady word occ year OR not steady word vector end)
                add all neologism occ with that year to pseudo-neologism vector
                Record statistics

        else if(steady word occ year is less than neologism occ year OR  not neologism end)
                add all ateady occ with that year to pseudo-neologism vector
                Record statistics

        else if(neolism occ year equal steady word occ year)
                add all neologism occ with that year to pseudo-neologism vector
                add all ateady occ with that year to pseudo-neologism vector
                Record Statistics

        Read next neolgism occ from vector
        Read next steady word occ from vector
```

Figure 9- Combine Vectors pseudo code

4.4.2 Calculating Error

While combining these vectors real data about the per year statistics are recorded. A structure called pseudo_year_data is used to record per year details. A vector of these years is created. In this way the per year data is stored.

```
struct pseudo_year_data {
   int year;
   int year_count_1;
   int year_count_2;
   double probability_1;
   double probability_2;
};


extern map<int, pseudo_year_data> pseudo_year_data_map;
```

Figure 10 - Pseudo Year data Structure

As the two vectors are merged on a per year basis, final year probabilities are calculated before moving onto the next year. Error is then calculated by finding the Mean absolute error between the estimated probability values for each year and these real values.

This provides a single number by which to compare accuracy between models.

4.5 Using the code

This brief sub-section seeks to outline how this additional functionality is used, as well as describe some of the programmes used for analysis.

4.5.1 DynamicEM

This is the programme which performs the EM algoirhtm on the dataset. The given inputs function as described in the documentation.

The use of additional functionality is controlled with several new inputs.

-targ_2: This input takes the second target word.

-pseudo_word: This is a Boolean input. If set to “yes” then the programme forms a pseudo-neologism from the two target words

-targ_position: This input determines the position or model of the N-gram. To fix it in a given position input 1,2,3,4, or 5. To use one of the three more advanced models use any, 9gram, or model.

4.5.2 Finding suitable words for pseudo-neologisms

The values in this programme are hardcoded in. To control the length of time without use needed to be determined a true neologism, there are Booleans for each decade in the find_pseduo function.

To control the upper and lower bound for are steady value there are int variables in the find_steady function.

The address to the COHA corpus file list is included at the beginning of main.

The results of this programme are output to three separate files, one containing all the corpus words, one containing, the pseudo words, and one containing potential steady words.

4.5.3 Analysing results

To analyse results several functions are used.

The inspector programme is used to identify significant words for each sense.

The R programme is used to graph the results.

The outlines for use for both programmes are described in the documentation. To graph both measuered and real values for pseudo-neologisms, the pseudo_sense_stats file should be input to the R programme.

5. Results

This section outlines the results of this project. This is broken up into two sections. The first handles real words with ambiguous sense. The second part handles pseudo-neologisms.

5.1 Real Neologisms

The first point of testing is to examine how the position of a target word in a 5gram affects the results. This is done for both the Google 5-gram corpus and COHA corpus on words with known neologisms.

The inspector programme is used to return the most significant words for the calculated senses. In this way the graphs can be compared with some knowledge of which words are best contributing to the results.

5.1.1 Google 5-gram Corpus

Before examining the results of fixing position, it is worth keeping in mind the raw data on which this analysis is done. As previously mentioned, the google 5-gram corpus provides a long list of readymade 5-grams. The code here essentially seeks to isolate the position of the target word so only those relevant to the relevant position are examined. Naturally, this means that there will be less data for the individual positions than the "any" case. There may also be a discrepancy between individual positions as well. A table outlining the number of extracted liens and occurrences is included.

Google 5-gram: MOUSE

The word mouse is known to take on another meaning, relating to the computer tool, in the 80s. This test examines the effect of position fixing for a 2-sense disambiguation.

As well, there is not an even distribution among the positions. Although given the sheer size of the corpus there is not a major lack of data for edge cases. The below table gives some idea of this.

Position	Extracted lines	Occurrences
1	149076	758692
2	209828	1345203
3	270831	1864498
4	241853	1526918
5	221974	1257717
Any	1093562	6753028

Figure 11 - Google Mouse Year Counts

ANY model

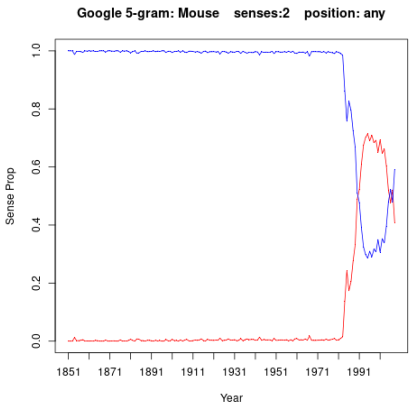


Figure 12 - Mouse Senses:2 Position:Any

Most significant words:

SENSE 0: button, pointer, left, right, release, over, move, down, your, you, drag, hold, click, use, then, Release, cursor, clicking, when, to, Move, on, position, press, Click, changes, user, moving, When, while

SENSE 1: cat, rat, cells, in, ", human, model, anti, game, -, like, embryo, as, a, mammary, house, embryos, of, brain, little, field, rabbit, such, /, for, trap, tumor, white, (, development

Position Fixed Cases

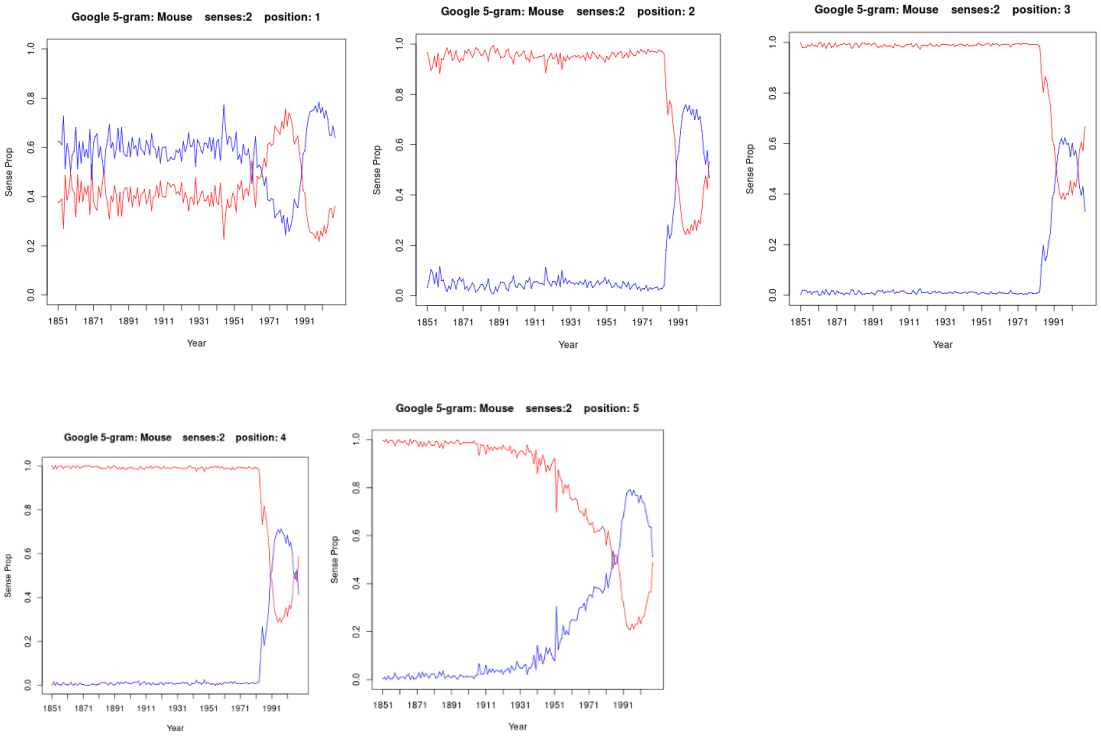


Figure 13 - Google 5-gram Mouse position fixed

Most significant words:

Position: 1

SENSE 0: _END_, cells, mammary, -, virus, tumor, ., marrow, bone, embryonic, stem, embryo, vitro, cell, trap, gene, brain, chromosome, :, ;, embryos, hole, macrophages, peritoneal, gland, skin, monoclonal, liver, L, human

SENSE 1: pointer, drag, the, you, to, over, on, move, can, changes, then, while, Mus, select, cursor, button, musculus, when, ,, is, of, one, right, left, other, not, up, an, a, display

Position: 2

SENSE 0: -, cells, '', of, rat, embryos, mammary, brain, model, embryo, human, virus, trap, tumor, in, skin, game, ', Mus, musculus, marrow, bone, (, clicks, a, by, little, liver, hole, macrophages

SENSE 1: button, pointer, left, over, right, your, drag, to, cursor, on, changes, select, move, down, you, is, pressed, while, the, when, moved, released, display, then, The, moves, clicked, position, can, press

Position: 3

SENSE 0: a, rat, _END_, in, '', model, like, of, cat, human, anti, game, keyboard, embryo, cells, by, house, little, for, field, embryos, as, In, -, brain, A, ?, but, be, mammary

SENSE 1: button, pointer, left, over, release, right, move, down, Release, when, drag, Move, cursor, to, position, changes, your, Position, hold, on, moves, select, while, use, Click, is, press, When, click, until

Position: 4

SENSE 0: in, cat, a, rat, as, game, '', keyboard, anti, like, -, house, model, cells, such, human, embryo, field, white, quiet, of, (, embryos, little, /, development, brain, IgG, was, from

SENSE 1: button, pointer, left, release, right, down, move, your, hold, Release, you, Move, drag, then, over, clicking, Click, user, press, position, cursor, Position, _START_, use, moving, holding, on, Use, moves, dragging

Position: 5

SENSE 0: cat, as, anti, such, rat, like, quiet, -, was, field, '', a, white, game, church, rabbit, goat, been, little, than, footed, poor, for, into, I, playing, --, man, said, pig

SENSE 1: click, left, release, you, right, down, move, hold, then, your, use, drag, When, user, clicking, can, Click, press, /, position, holding, when, clicks, Hold, If, dragging, using, while, moves, development

Analysis

Looking at the graphs, we see that position 1 and 5 give more erroneous results than the more central positions. As well looking at the any case we see it closely resembles the position 3 case. In this case, target word position has an effect.

With the inspector programme, most significant words are show. Some of the most significant words of the more successful position cases are missing from the 1 and 5 positions. For example, “cat” and “rat” place very highly in the more successful cases but are missing from the unsuccessful position 1 case. This would suggest that important details are not necessarily distributed evenly around the target word.

Google 5-gram: TANK

Looking now at the word tank we see that it takes on a new meaning around the 30s with the creation of the modern tank used in warfare.

Details for occurrences are left below.

Position	Extracted Lines	occurrences
1	158782	436561
2	204253	638687
3	362357	1584970
4	356996	1208475
5	274425	1002210
any	1356813	4870903

Figure 14- Google Tank Position Counts

position(any)

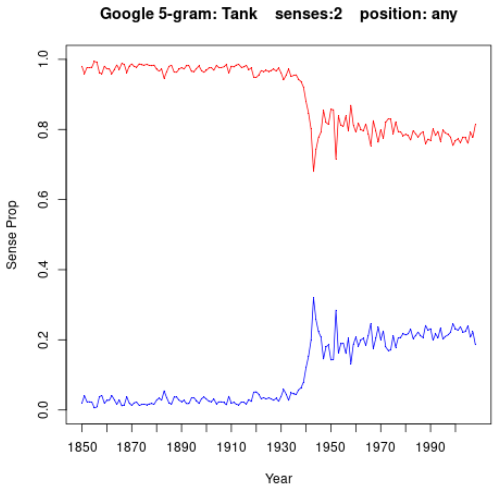


Figure 15 - Google 5-gram Tank any position

SENSE 0: bottom, into, is, the, from, gas, to, of, be, fuel, full, filled, it, in, large, side, fill, storage, circuit, should, at, septic, which, water, through, back, empty, end, placed, aeration

SENSE 1: anti, -, guns, think, gun, stirred, weapons, reactor, --, ', (,), missiles, gallon, ditch, two, artillery, continuous, shorts, guided, Washington, '', tanks, cars, fire, mines, conservative, weapon, an, T

Position fixed cases

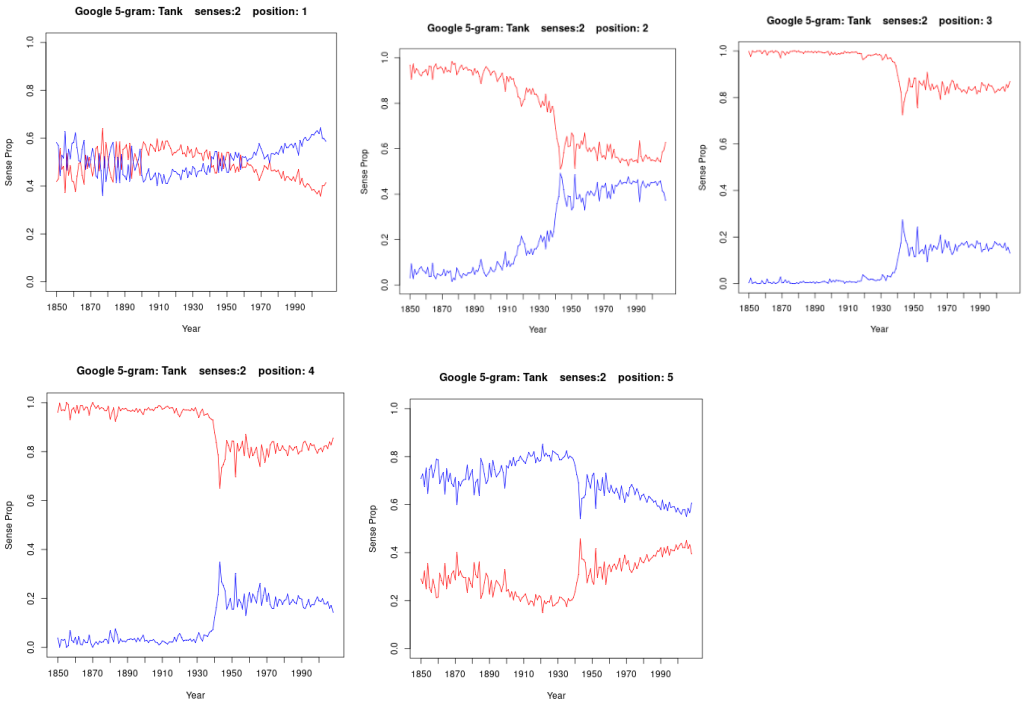


Figure 16 - Google 5-gram Tank Position fixed

Position: 1

SENSE 0: be, (, which, It, as, should,), shown, has, can, filled, may, that, where, not, capacity, so, been, there, provided, reactor, from, CSTR, will, equipped, with, used, connected, but, described

SENSE 1: _END_, -, top, gas, ., at, anti, on, full, empty, other, one, '', bottom, aircraft, time, front, or, means, back, car, end, rate, fuel, of, side, roof, cars, pump, artillery

Position: 2

SENSE 0: is, be, it, to, at, The, was, filled, which, with, water, as, should, so, where, large, full, that, can, has, not, ,, empty, from, shown, he, but, an, then, other

SENSE 1: _END_, -, guns, '', .,), gun, weapons, stirred, reactor, anti ,(, ?, --, ', missiles, CSTR, ditch, fire, top, ;, s, septic, guided, cars, reactors, her, battle, system, mines

Position: 3

SENSE 0: the, _START_, into, of, The, fuel, gas, be, _END_, water, from, is, full, an, filled, septic, in, storage, to, circuit, through, should, large, a, when, his, not, fish, ?, where

SENSE 1: anti, guns, -, weapons, gun, think, stirred, --, missiles, reactor, cars, ditch, top, guided, mines, weapon, were, ', (, missile, artillery, continuous, '', two, rifles, destroyers, fire, ditches, reactors, tanks

Position 4

SENSE 0: is, into, bottom, to, in, from, septic, the, top, large, storage, ., side, fuel, at, fill, of, was, gas, stirred, back, out, end, full, aeration, level, If, reactor, settling, through

SENSE 1: anti, guns, -, gun, think, weapons, '', ', ditch, --, missiles, gallon, two, conservative, pounder, destroyers, guided, an, T, mines, weapon, ditches, fire, cars, German, Anti, rifle, dive, artillery, trucks

Position: 5

SENSE 0: -, anti, ,, think, _START_, an, as, with, --, stirred, gallon, 's, by, and, or, tank, large, '', T, for, two, A, tanks, filled, conservative, continuous, hot, car, full, There

SENSE 1: bottom, top, side, of, end, back, the, into, level, liquid, front, sides, surface, contents, out, be, capacity, inside, edge, size, put, are, volume, aeration, part, pressure, removed, pumped, wall, from

Analysis

Position 3 with the closest resemblance to the any case and is the position which best captures sense emergence.

Position 1 and position 5 are considerably off.

Some of the most significant words for the more successful positions are missing from position 1 and position 5. Again, possibly indicating that it is missing out on crucial details. For example the word weapons is s significant word in position 3 and the Any case for SENSE 1. It does not appear for position 1 or 5.

Google 5-gram: GAY

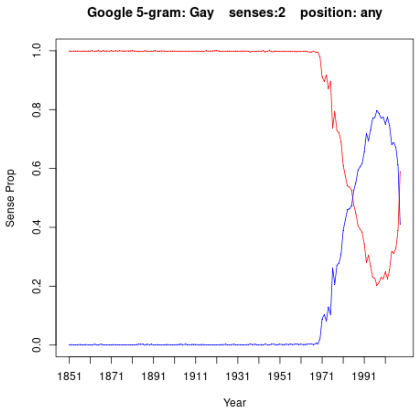
The word gay took on a new meaning in the 1900s relating to homosexuals.

Details for its occurrences are left in the table below.

Position	extracted lines	occurrences
1	226160	965340
2	330954	1474018
3	363383	1574029
4	321808	1217404
5	284767	947246
any	1527072	6178037

Figure 17 - Google Gay Positions Counts

Position: any



SENSE 0: was, he, I, so, very, grave, be, not, you, 'm, his, she, a, happy, light, it, but, world, He, they, bright, 're, It, were, ?, lively, young, all, is, flowers

SENSE 1: lesbian, men, lesbians, rights, bisexual, community, movement, /, liberation, straight, male, women, couples, for, studies, issues, people, parents, anti, marriage, Lesbian, against, communities, (, or, identity, among, relationships, families, abortion

Position Fixed cases

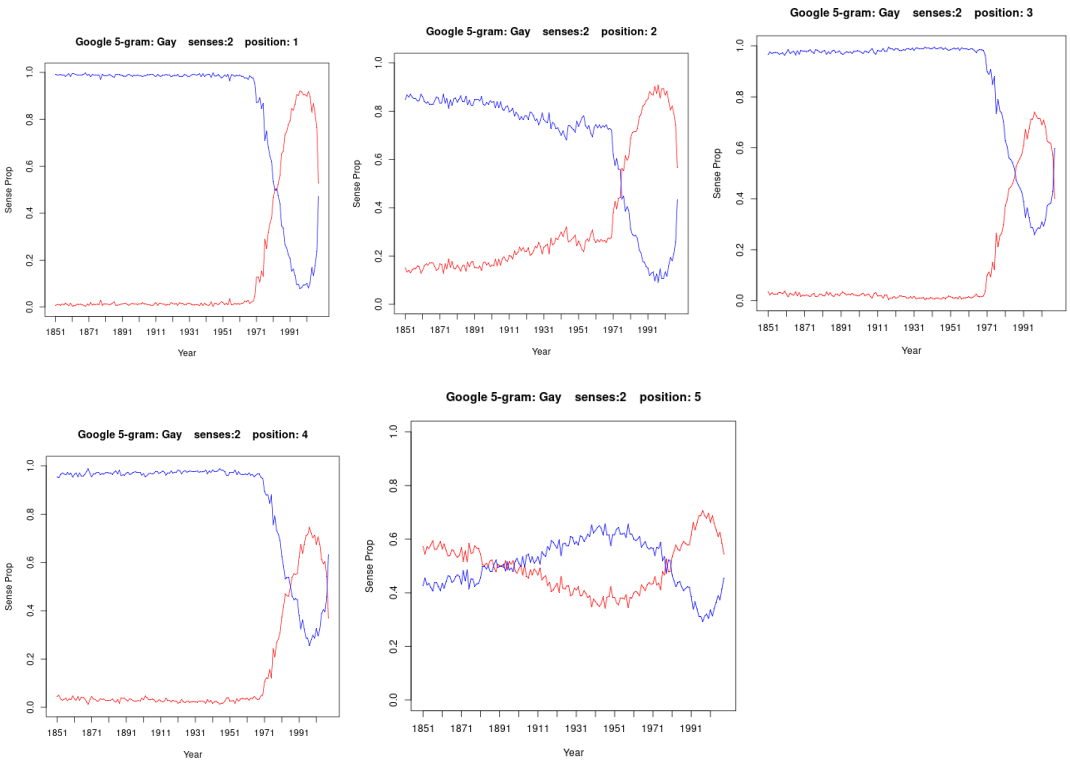


Figure 18 - Google 5-gram Gay Fixed Position

Position 1

SENSE 0: lesbian, bisexual, lesbians, men, rights, community, movement, /, straight, couples, male, women, or, liberation, issues, parents, are, identity, people, communities, studies, relationships, (, have, bisexuals, students, transgender,), families, '

SENSE 1: of, happy, he, lively, flowers, light, with, hearted, a, was, she, young, good, cheerful, his, full, bright, all, brilliant, little, as, I, lark, but, so, -, ever, said, when, ;

Position: 2

SENSE 0: _END_, men, lesbian, lesbians, movement, ., ", rights, community, ', liberation, people, couples, ?, /, male, for, among, women, identity, parents, studies, issues, bar, communities, relationships, marriage, culture, against, families

SENSE 1: so, was, as, happy, lively, he, -, I, flowers, light, they, full, but, she, ,, young, cheerful, brilliant, lark, bright, grave, witty, very, flags, it, bisexual, from, ever, sad, little

Position: 3

SENSE 0: lesbian, men, lesbians, rights, movement, bisexual, community, /, liberation, male, anti, Lesbian, people, women, studies, couples, within, parents, marriage, many, issues, straight, abortion, for, -, (, Lesbians, against, or, communities

SENSE 1: _END_, was, he, I, very, ", not, so, be, grave, man, ?, 'm, ;, a, you, but, bar, happy, 're, world, bright, she, being, light, his, life, lively, am, !

Position: 4

SENSE 0: lesbians, men, lesbian, rights, community, for, Lesbian, women, liberation, movement, ', male, /, -, against, studies, anti, of, marriage, identity, on, couples, Lesbians, among, parents, abortion, many, issues, HIV, word

SENSE 1: was, he, be, I, ., grave, not, very, were, is, so, 'm, they, He, you, ?, It, she, been, had, 're, are, bright, bar, because, It, She, if, who, have

Position: 5

SENSE 0: lesbians, lesbian, of, women, the, and, rights, on, number, anti, against, /, for, In, old, part, lives, gayest,), group, -, Lesbian, year, midst, example, children, movement, needs, development, abortion

SENSE 1: I, was, not, he, be, It, very, you, it, had, He, they, 'm, is, who, been, have, out, she, 're, know, She, were, so, If, are, There, They, there, no

Analysis

Again, the more central positions give the best estimation of sense emergence. These are the positions that the any case most resembles.

Again, significant words for the more successful cases are not seen in the edge cases. For example, the word grave.

5.1.2 COHA corpus

The COHA operates as a more conventional corpus, in that it contains lines of text through which the programme must search to find target word occurrences. This results in an even number of occurrences among all positions.

To add context to results the total number of occurrences for words is also included

COHA Mouse

There are 6626 total occurrences

Advance Models

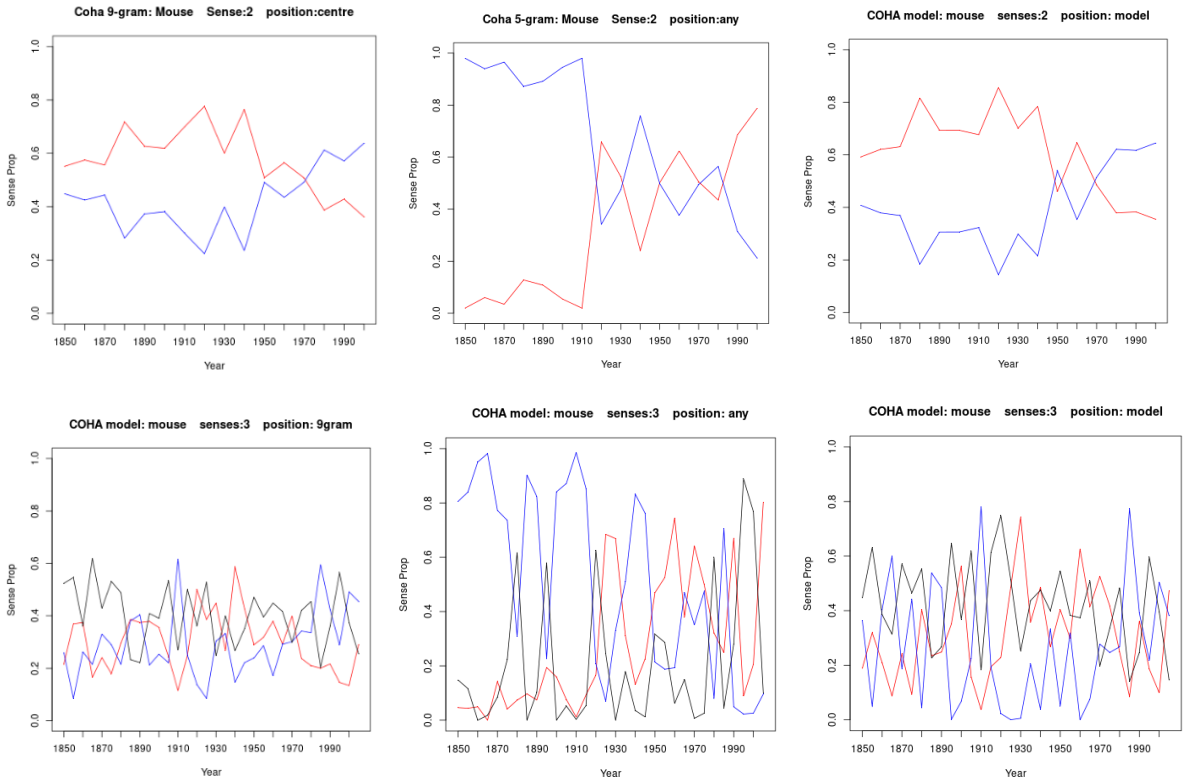


Figure 19- COHA Mouse Advanced Models

9gram

2 senses, 10 year gap

SENSE 0: , !, meadow, '', n't, mr.-, do, ?, i, ', she, say, church, man, you, as, little, catch, quiet, him, singing, not, still, see, ask, poor, what, know, he, think

SENSE 1: rat, over, click, which, use, their, bird, human, between, each, from, squirrel, pig, make, them, laboratory, cell, day, first, other, produce, dog, inject, develop, with, skin, and, guinea, rabbit, fly

Position(any)

Senses 2, 10 year gap

SENSE 0: hev, reside, estimate, mildly, rabbit, by, andy, he, live, intend, spot, 1791-1865, watch, convert, bedchamber, never, species, tearful, magnify, soundtrack, together, teach, eve, kitchen, ?, ask, beside, jerrold, assure, waste

SENSE 1: hev, rabbit, by, reside, estimate, inside, andy, species, he, mildly, live, jerrold, spot, hough, tearful, still, knee-high, thine, magnify, really, ?, gang, 1791-1865, noise, ant, tame, illustration, bel, council, crushed

Position Fixed Cases

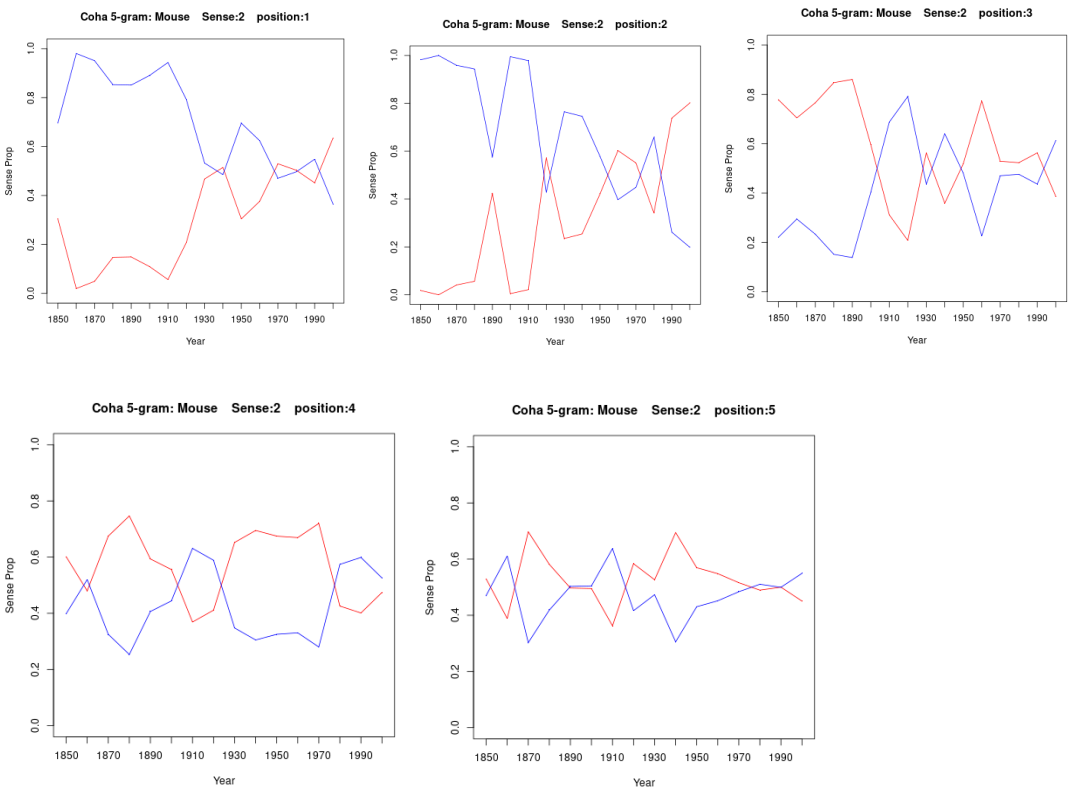


Figure 20- COHA mouse fixed positions

Position 1

SENSE 0: <p>,), (, 's, eye, with, house, click, over, your, pig, cage, turn, from, a, human, ?, their, thing, cell, two, corner, head, her, guinea, antibody, gene, test, name, three

SENSE 1: rat,, we, i, do, n't, not, could, you, him, have, ;, there, more, come, she, know, see, when, it, !, me, seem, be, if, still, rabbit, what, as, here

Position 2

SENSE 0: <p>, meadow, (, house, blind, minnie, with, click, your, human, guinea, tatty,), two, female, computer, give, grasshopper, mother, mighty, pig, more, want, vole, catnip, ..., 's, live, between, nest

SENSE 1: singing, ', --, come, !, ;, could, run, tiny, little, out, it, young, say, play, man, who, through, country, bird, what, me, gnaw, we, lady, before, she, when, mauve, another

Position 3

SENSE 0: rat, and, man, cat, which, rabbit, play, keep, ;, from, through, white-footed, by, bird, ,, :, five, every, country, keyboard, mauve, give, tatty, but, or, sing, of, among, inject, fly

SENSE 1: meadow, singing, mr.-, <p>, three, blind, he, do, have, ", not, master, gray, kind,), that, one, when, say, before, watch, just, get, old, church, no, ., show, trap, n't

Position 4

SENSE 0: like, as, a, not, little, there, i, three, gray, still, blind, than, quiet, poor, church, she, watch, find, in, see, on, you, big, n't, even, or, man, cancer, while, such

SENSE 1: meadow, singing, mr.-, say, , rat, house, and, use, kind, master, our, away, ", to, their, mouse, nest, deer, ask, squirrel, white-footed, computer, lion, <p>, keyboard, scheme, the, o, from

Position 5

SENSE 0: as, , meadow, mr.-, do, than, !, n't, i, there, quiet, still, blind, they, three, hear, poor, church, watch, about, more, we, see, look, you, big, such, have, think, know

SENSE 1: rat, singing, and, from, into, click, of, kind, or, away, say, out, squirrel, over, nest, 's, by, the, scheme, white, computer, size, bird, lion, rabbit, keyboard, cell, lay, dog, all

Analysis

Here the algorithm fails to accurately discern the senses for either the “any” or the 9gram case.

This is carried into the various position fixed options, with each failing to accurately discover the date of neologism adoption.

The advanced models here are also run with a shorter year gap as well as a third sense to see if this has an effect on senses.

COHA Gay

Total occurrences are 12966.

Advanced Models

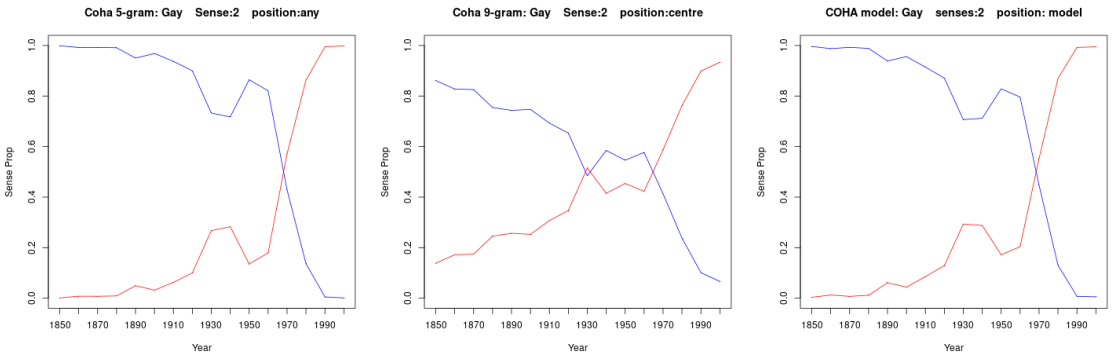


Figure 21 - COHA Gay Advanced Models

Position any

SENSE 0: lesbian, right, man, <p>, marriage, community, bar, us, activist, n't, (, ?, straight,), do, people, you, couple, openly, ", murder, i, liberation, male, about, know, who, student, because, against

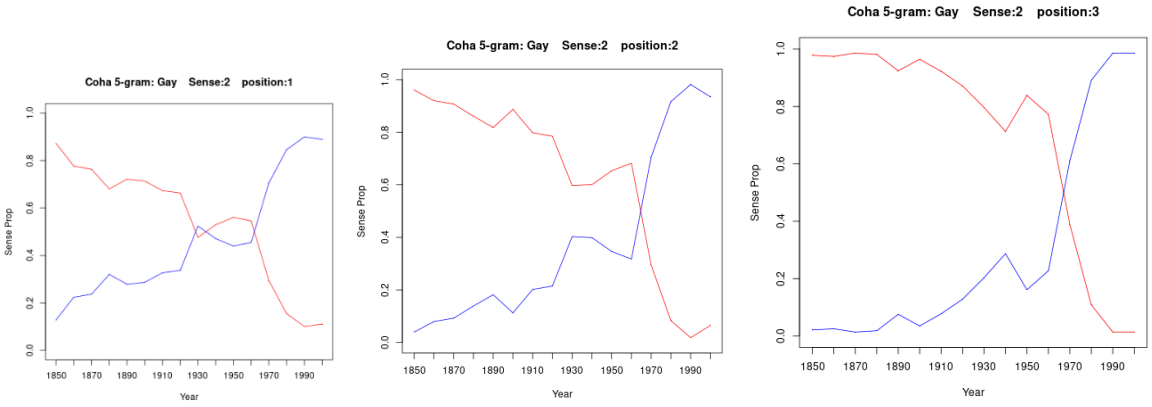
SENSE 1: little, her, laugh, color, his, voice, ;, flower, with, bright, party, smile, very, spirit, happy, girl, company, light, grave, laughter, world, young, their, its, gallant, scene, bird, all, dress, eye

Position 9 gram centre

SENSE 0: you, i, lesbian, n't, <p>, man, ", (, do, ?, right,), not, will, get, marriage, community, know, say, us, bar, straight, ', feel, if, no, too, can, we, !

SENSE 1: with, flower, her, his, little, color, bright, their, smile, voice, dress, laughter, which, scene, its, of, ;, laugh, eye, red, song, bird, crowd, brilliant, the, gallant, music, light, world, girl

Position Fixed



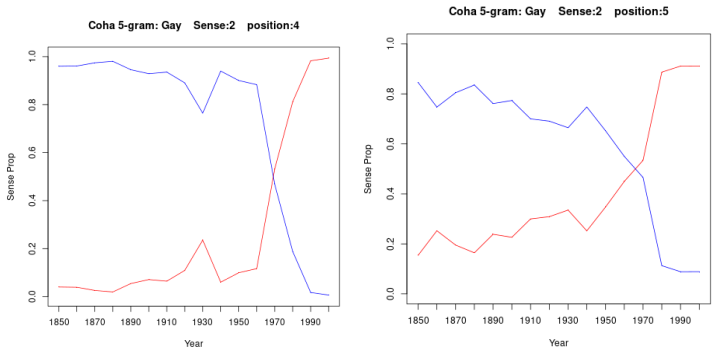


Figure 22 - COHA Gay Fixed position

Position 1

SENSE 0: her, of, flower, with, little, bright, his, which, party, from, most, brilliant, light, young, color, gallant, bird, scene, ribbon, girl, crowd, red, on, beautiful, full, more, song, the, attire, festive

SENSE 1: ", ?, i, you, lesbian, not, right, do, <p>, say, ', man, marriage,), community, , n't, will, (, can, he, ., it, bar, be, there, no, activist, what, people

Position 2

SENSE 0: little, laugh, her, color, party, flower, voice, his, happy, girl, world, smile, with, company, spirit, laughter, its, their, throng, as, light, ;, crowd, bright, bird, song, of, scene, young, attire

SENSE 1: lesbian, man, right, ?, <p>, marriage, ", us, community,), bar, i, activist, not, (, n't, do, you, people, straight, get, no, couple, feel, ., be, or, ', military, liberation

Position 3

SENSE 0: little, her, laugh, color, his, very, party, voice, happy, bright, with, spirit, flower, light, smile, ;, grave, its, girl, gallant, company, all, their, laughter, time, dress, world, bird, !, so

SENSE 1: lesbian, man, right, marriage, us, community, <p>, bar, activist, straight, ?, ", who, people, you, (, do, openly, make, couple,), liberation, male, about, movement, 's, ban, pride, n't, woman

Position 4

SENSE 0: man, right, marriage, community, bar, lesbian, (, <p>, activist, against, people,), openly, couple, about, know, n't, ?, liberation, male, who, ban, straight, for, you, i, pride, woman, many, issue

SENSE 1: little, very, his, laugh, color, voice, her, all, so, with, ;, grave, smile, bright, dress, light, world, company, their, young, party, flower, scene, make, always, girl, !, spirit, she, laughter

Position 5

SENSE 0: , n't, i, do, you, <p>, lesbian, know, 's, about, if, get, that, group, to, against, will, ?, think, try, because, no, (, openly, straight, by, abortion, mean, '', many

SENSE 1: bright, her, and, grave, all, ;, light, with, its, their, little, his, still, so, smile, happy, very, almost, then, dress, laugh, which, eye, murder, heart, these, girl, voice, young, day

Analysis

Of the advanced models the any is significantly more accurate than the 9-gram. Possibly lending credence to the benefit of this overcounting.

For the fixed positions again the more central positions are more effective at capturing meaning.

COHA Jet

Total occurrences are 6872

Advanced Models

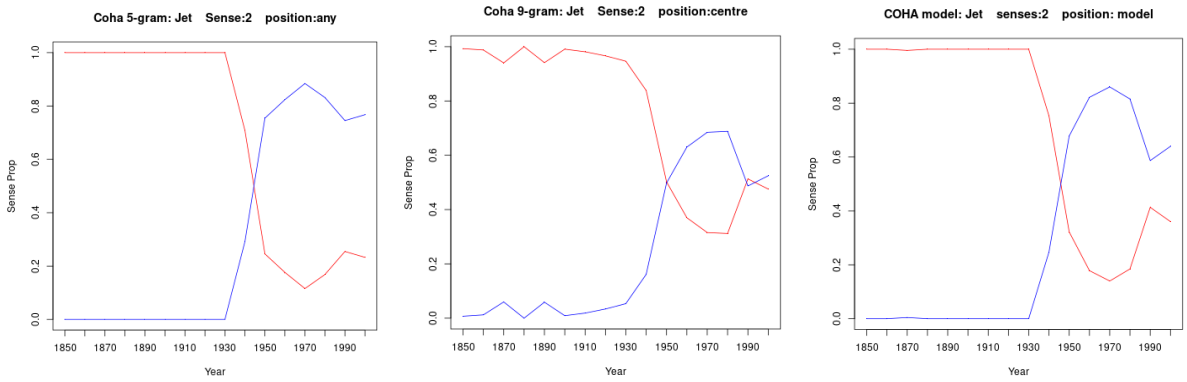


Figure 23- COHA jet Advanced models

Position any

SENSE 0: color, ever, most, cat, dimly, transparent, towards, black, see, smoke, these, encircle, handsome, ring, graceful, hempen, lofty, shred, silvery, very, point, once, sparkle, we, garb, visible, negro, enliven, aspire, moonlight

SENSE 1: ever, most, color, these, we, test, ring, see, moonlight, towards, very, black, follow, fast-flying, experimentation, los, commercially, once, visible, alter, transcontinental, six, verdant, wrench, stress, half, mine, smoke, graceful, money

Position 9gram

SENSE 0: gas, black, water, her, hair, flame, through, stream, she, out, smoke, turn, steam, eye, little, '', light, you, up, so, it, spray, long, as, under, like, !, tell, of, he

SENSE 1: new, york, laboratory, (, american, fly,), u.s.-, :, airline, fighter, private, ', propulsion, transport, phantom, force, <p>, boeing, -, ,first, at, airport, aircraft, after, air, flight, israeli, last

Position Model

Position Fixed

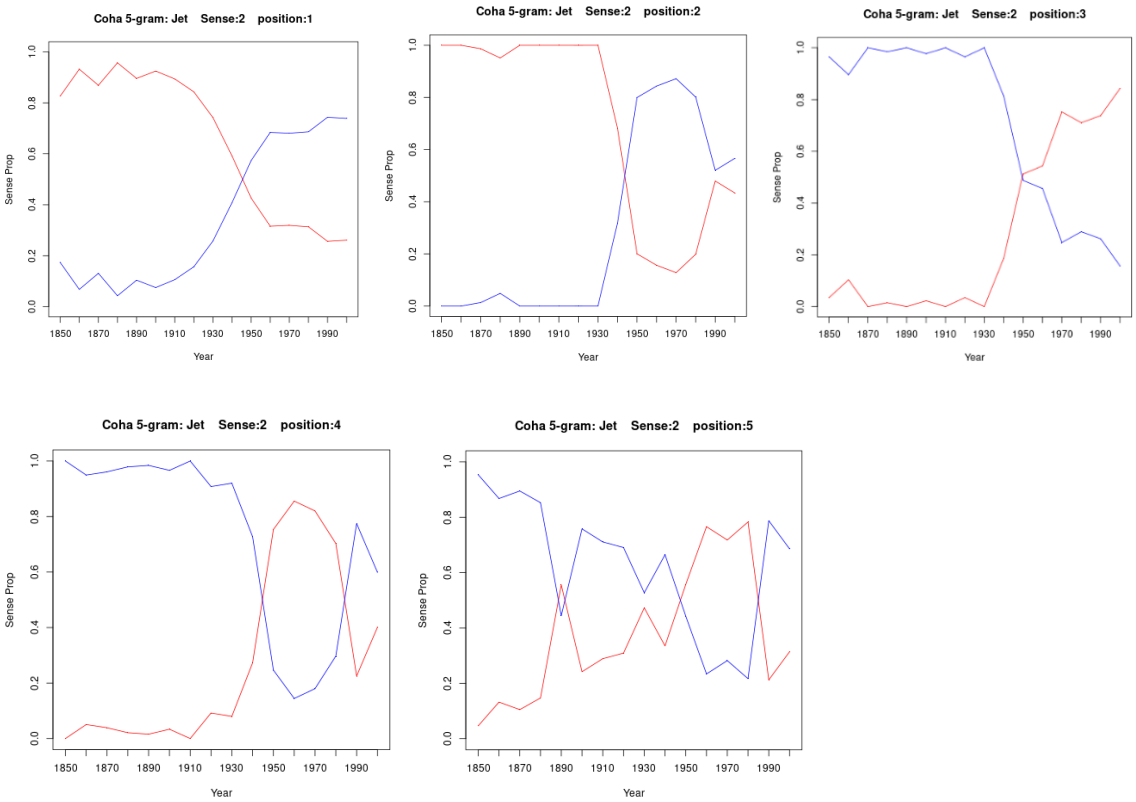


Figure 24- COHA Jet Fixed Positions

Position 1

SENSE 0: of, black, water, out, into, flame, hair, gas, down, shoot, smoke, through, steam, light, air, bead, spray, force, during, eye, ;, across, strike, spurt, up, white, over, fire, sky, american

SENSE 1: propulsion, laboratory, <p>, to,), transport, , (, take, say, have, do, land, lag, fuel, n't, can, ', he, :, pilot, by, time, last, who, i, aircraft, off, helicopter, fly

Position 2

SENSE 0: gas, black, water, her, hair, flame, of, steam, tell, smoke, little, ", light, out, bead, fire, lag,
, spray, tiny, she, eye, spurt, through, play, blood, white, ski, room, liquid

SENSE 1: fighter, plane, fly, laboratory, bomber, aircraft, engine, (, transport, age, pilot, propulsion, commercial, airliner, to, phantom, force, u.s.-, have, flight, liner, crash, passenger, airline, Israeli, for, american, down, by, supersonic

Position 3

SENSE 0: new, jumbo, york, private, laboratory, propulsion, 's, lag, (, tell, <p>, commercial, ", fighter, say, ', to, corporate, airline, ski, passenger, :, american, time, land, could, sabre, at, dog-ass, lear

SENSE 1: gas, black, water, air, hair, flame, steam, bomber, of, force, it, through, smoke, eye, light, little, bead, out, tiny, single, like, white, ;, liner, line, as, small, motor, exhaust, burn

Position 4

SENSE 0: plane, fighter, air, aircraft, age, transport, bomber, force, american, commercial, phantom, fly, u.s.-, big, an, flight, jumbo, airliner, airline, passenger, israeli, use, take, first, boeing, supersonic, aboard,liner, will, chartered

SENSE 1: gas, black, lag, tell, ", her, york, eye, hair, propulsion, up, ski, send, bead, long, out, he, this, she, boat, !, It, light, off,
, tiny, nasa, my, little, smoke

Position 5

SENSE 0: air, force, an, american, airline, small, big, phantom, three, ,from, of, aboard, more, supersonic, israeli, united, sabre, boeing, passenger, flight, 707, state, those, chartered, control, roar, sound, 747, jumbo

SENSE 1: ", it, <p>, tell,), black, over, there, up, at, send, she, long, hair, n't, my, you, would, out, he, this, last, her, ?,
, bill, we, many, ;, who

Analysis

The “any” case is more successful than the 9gram case. The model combining them proves most successful of all.

Again, the central cases are more reliable than those on the edge.

5.1.3 Explanation

Evaluating the results, it was seen from the fixed positions that the 5-grams with the target word in central positions were consistently more accurate than those on the edge. Looking at most significant words it was found that the edge positions frequently hadn’t captured words that were significant to the more successful centre positions.

The any case was typically successful if these central positions were. An example of an unsuccessful case was shown with COHA mouse. Likely unsuccessful due to a lack of data. There were only around 6000 occurrences. Neither the Any nor the 9-gram case was successful. Following this none of the position fixed cases were successful.

Comparing the effects of the Any case against the 9-gram case, the any case was found to be more effective. In both Gay and Jet, it performed significantly better, including instances where the 9-gram was unsuccessful. This would lend some credence to the idea that the over counting of these words nearer the target word is of benefit.

The new model was typically slightly more successful. This can be seen in Jet, where the bumps seen in the “any” case are removed with the new model. The model clearly indicates a start date around the 1930s, with very little noise.

Ultimately, it was observed that there was a difference in accuracy for position fixed cases with the central positions being more effective.

When comparing the Any case to the 9-gram, the any case was more effective, possibly indicating the benefit of this overcounting.

The new model proved even more effective than the any.

5.2 Pseudo-Neologisms

This section seeks to better analyse the relationship between position and sense detection using pseudo-neologisms.

5.2.1 COHA

From the previous results section, it appeared that position did have an effect. With this came a benefit to the overcounting involved in the any case. As well, the new model performed better again.

Using pseudo neologisms, this section hopes to give a clearer picture of this and use more edge cases to test the limits of this, by locating the frequencies at which the algorithm begins to struggle.

COHA Pray/Movie

The words Pray and Movie are combined into a pseudo word. Each words total frequency is around the 20,000 mark.

Neologism and Steady Word Year Counts

Pray	
Total Count: 21374	
Year	Count
1850	1962
1860	1863
1870	1695
1880	1733
1890	1854
1900	1228
1910	1070
1920	1227
1930	869
1940	973
1950	1076
1960	969
1970	1018
1980	1134
1990	1252
2000	1451

Movie	
Total Count: 20705	
Year	Count
1850	0
1860	0
1870	0
1880	0
1890	1
1900	4
1910	94
1920	660
1930	735
1940	1157
1950	1840
1960	1971
1970	2566
1980	3117
1990	3979
2000	4581

Figure 25 - Pray and Movie Year Counts

Calculated Error

Pray/Movie			
	9gram	Any	Model
5 year gap	0.054013	0.060243	0.060183
3 year gap	0.057828	0.063403	0.062763

Figure 26- Pray/Movie errors

Advanced Positions

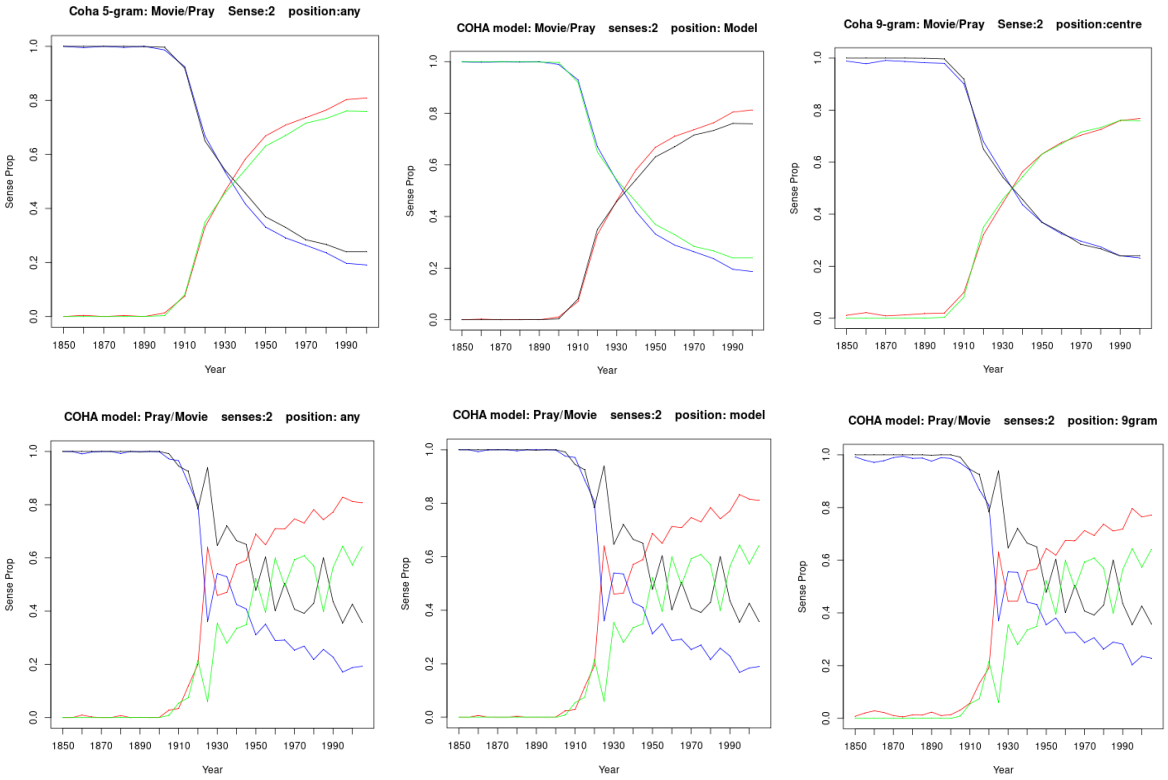


Figure 27 - COHA Pray/Movie Advanced Models

Position Fixed

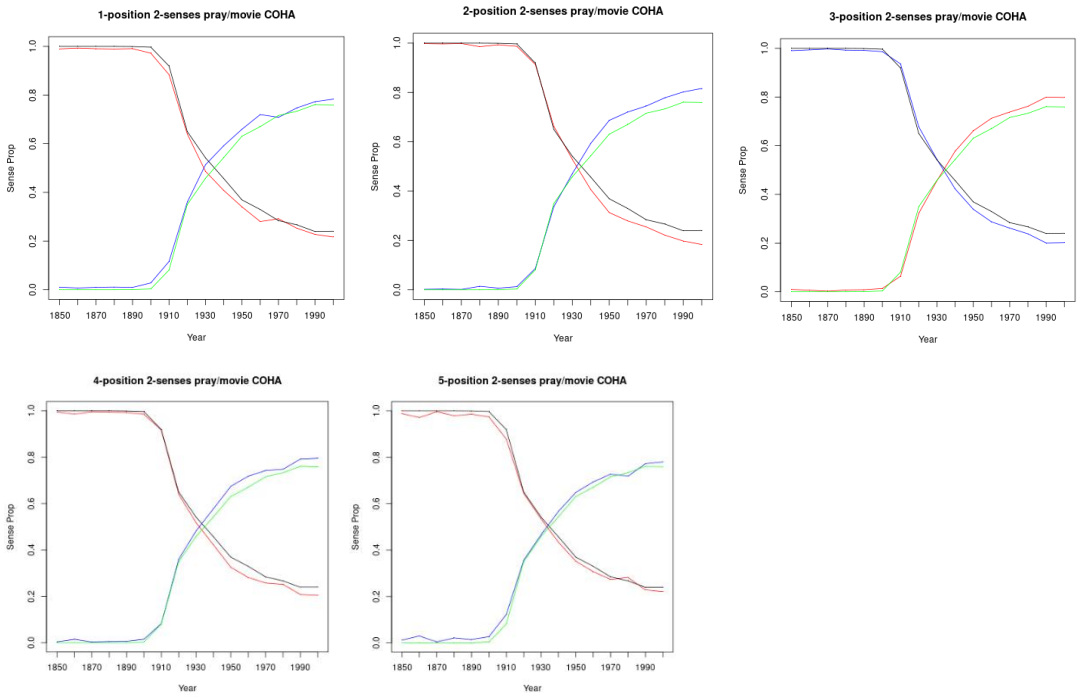


Figure 28 - COHA Pray/Movie Position Fixed

Analysis

Here we see a reasonably successful case across all models. There is very little to sperate any of the cases. Shorter year gaps are tested for the advanced models to see if they models began to separate but they remained consistent.

From the calculated errors, it is the 9-gram which performs best. However, this is very marginal. The new model does perform slightly better than the any case.

COHA Mirror/Computer

This pseudo word combines the words Mirror and computer. Each word has a total frequency around the 15000 mark.

Neologism and Steady Word Year Counts

Mirror	
Total Count: 16775	
Year	Count
1850	379
1860	307
1870	488
1880	427
1890	427
1900	645
1910	622
1920	907
1930	932
1940	1005
1950	1339
1960	1217
1970	1388
1980	1577
1990	2537
2000	2578

Computer	
Total Count: 14817	
Year	Count
1850	0
1860	0
1870	3
1880	0
1890	2
1900	2
1910	1
1920	1
1930	6
1940	9
1950	131
1960	684
1970	1438
1980	3769
1990	4946
2000	3825

Figure 29 – Mirror and Computer Year Counts

Calculated Error

Mirror/Computer			
	any	9gram	model
10steps	0.00729	0.018855	0.005899
5steps	0.043876	0.053887	0.371298

Figure 30 - Mirror/Computer errors

Advanced Models

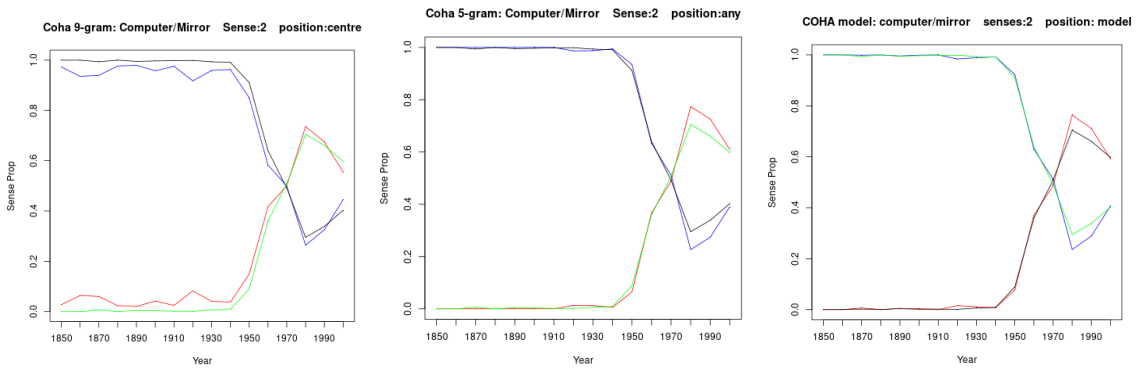


Figure 31 - Computer/Mirror advanced models

Position Fixed

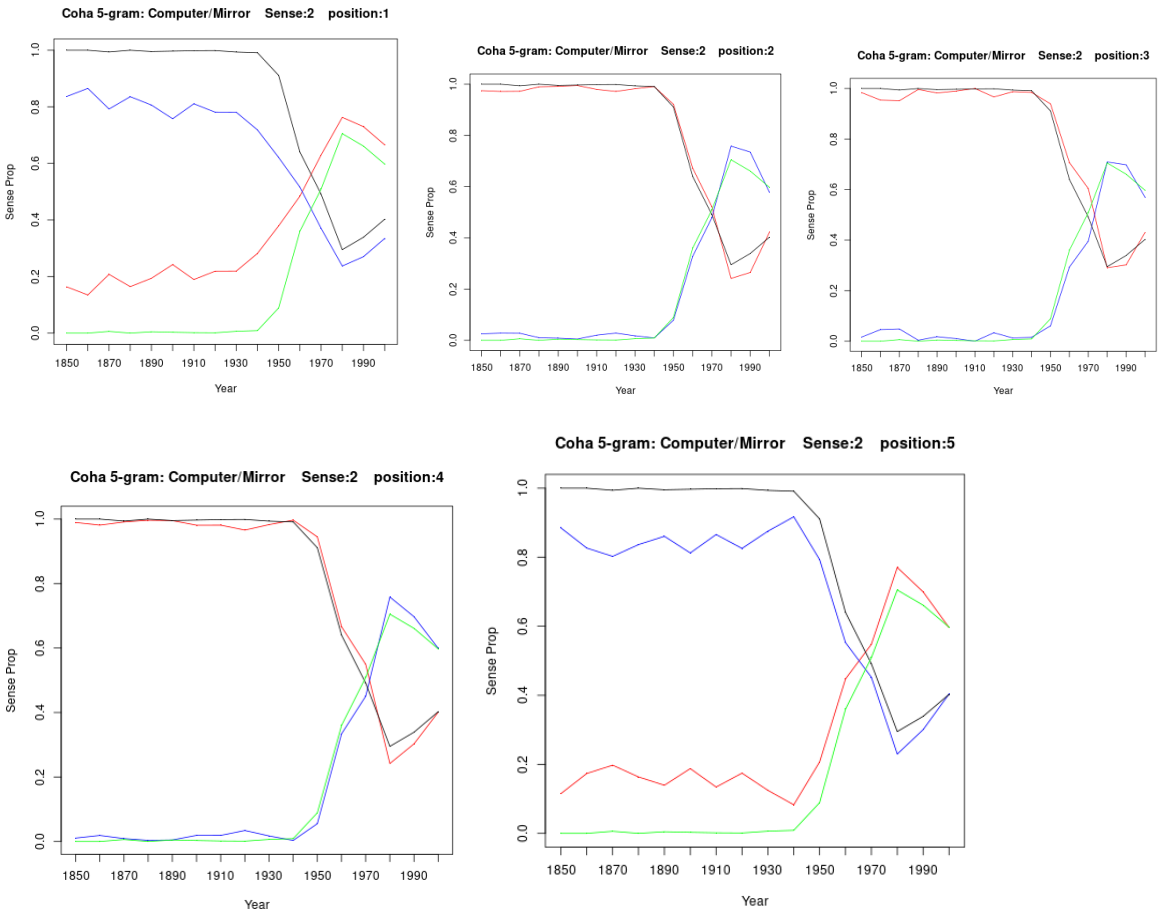


Figure 32 - COHA Computer/Mirror Fixed Position

Analysis

Here it is clearly seen that the “any” case and new model do a much better job of finding the true values than the 9gram case. Looking at the errors, the new model does a slightly better job than the “any” case

From the fixed positions it can be seen that the central positions do a better job than the edge positions.

COHA Garage/Insect

A pseudo word made of Garage and insect. The total frequency for each word is around the 7000 mark.

Neologism and Steady Word Year Counts

Garage	
Total Count: 5946	
Year	Count
1850	0
1860	0
1870	0
1880	0
1890	26
1900	37
1910	164
1920	456
1930	614
1940	540
1950	545
1960	505
1970	535
1980	582
1990	895
2000	1047

Hammer	
Total Count: 8558	
Year	Count
1850	232
1860	226
1870	247
1880	298
1890	325
1900	445
1910	364
1920	523
1930	616
1940	1278
1950	668
1960	651
1970	622
1980	614
1990	710
2000	739

Figure 33- Garage and Hammer Year Counts

Calculated Error

Garage/insect			
	9gram	any	Model
10year	0.027185	0.033539	0.028633

Figure 34- Garage/insect errors

Advanced positions

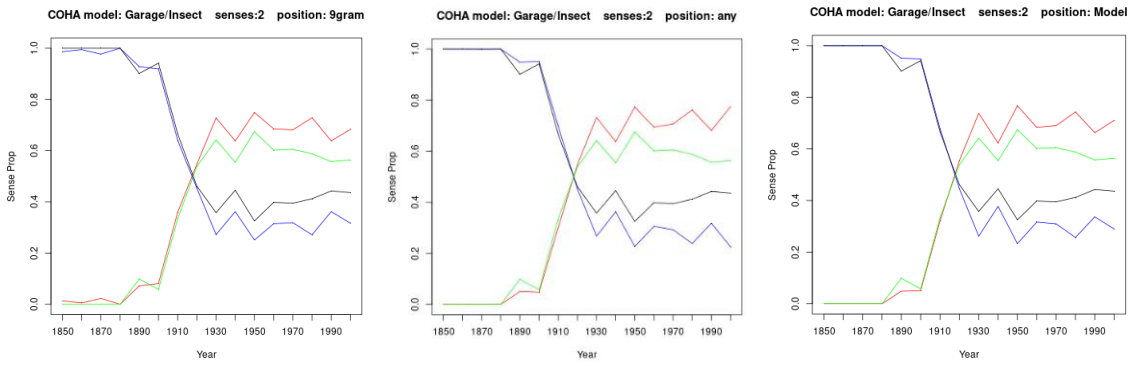


Figure 35- COHA Garage/Insect Advance Positions

Analysis

This test uses much lower numbers. In this case the 9-gram does better than the “any” case. ~The new model does slightly worse than the 9-gram, although this is very marginal.

COHA Magpie/Genocide

This pseudo word uses a very low number of occurrences. Each around the 400 mark.

Neologism and Steady Word Year Counts

Magpie	
Total Count: 392	
Year	Count
1850	4
1860	7
1870	20
1880	20
1890	27
1900	30
1910	103
1920	22
1930	19
1940	24
1950	25
1960	16
1970	19
1980	7
1990	34
2000	15

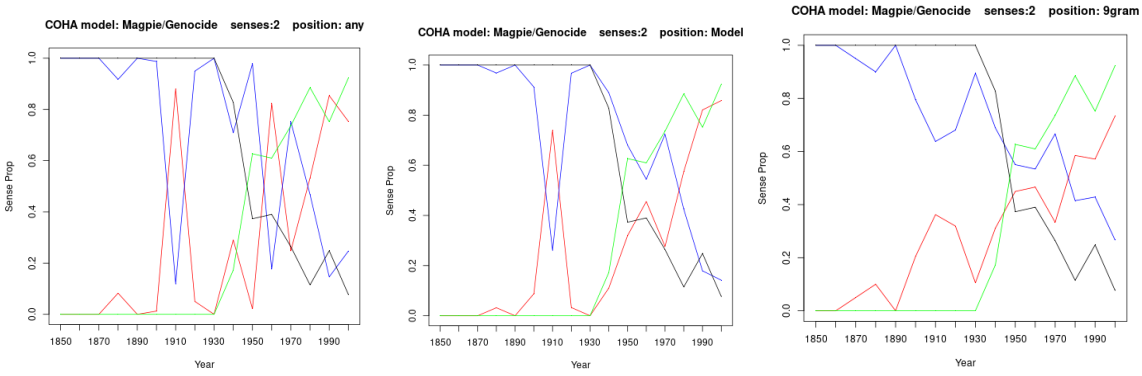
Genocide	
Total Count: 462	
Year	Count
1850	0
1860	0
1870	0
1880	0
1890	0
1900	0
1910	0
1920	0
1930	0
1940	5
1950	42
1960	25
1970	53
1980	54
1990	103
2000	180

Figure 36- Magpie and Genocide year Counts

Calculated Error

Magpie/Genocide			
	9gram	Any	Model
10year	0.083554	0.09614	0.072474

Advance Positions



Analysis

At these low numbers, the model begins to struggle. Each model failed to accurately find the date of sense emergence. Still, it was found that the new model had the lowest error.

COHA BANANA/Genocide

This pseudo word compares a case where one sense is far less common than the other. In this case genocide only has 462 occurrences while Banana has 2845.

Neologism and Steady Word Year Counts

Genocide	
Total Count: 462	
Year	Count
1850	0
1860	0
1870	0
1880	0
1890	0
1900	0
1910	0
1920	0
1930	0
1940	5
1950	42
1960	25
1970	53
1980	54
1990	103
2000	180

Banana	
Total Count: 2845	
Year	Count
1850	13
1860	44
1870	21
1880	75
1890	37
1900	80
1910	168
1920	241
1930	164
1940	178
1950	248
1960	193
1970	397
1980	256
1990	333
2000	397

Figure 37- Banana and Genocide Year Counts

Calculated Error

Banana/Genocide			
	model	any	9-gram
10-senses	0.207972	0.205695	0.210209

Figure 38 - Banana/Genocide error

Advanced Models

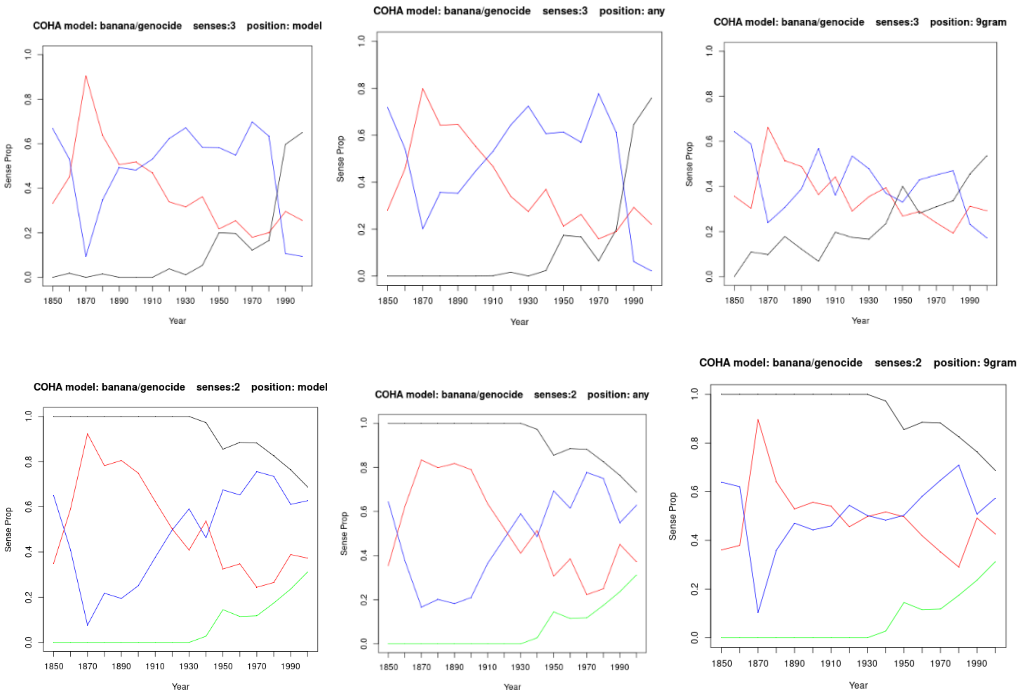


Figure 39 - COHA banana/genocide Advanced models

Analysis

Using only a two-sense model each model failed to detect the sense emergence. However, in this case it was found that still the new model produced the lowest error.

Tests were run with a three sense (i.e. with the assumption that this pseudo word contained three senses). In this case it was found that the "any" and new model were able to detect the sense emergence, while the 9-gram was not.

5.2.2 Explanation

These pseudo-neologisms found that the algorithm can begin to struggle around the low thousand mark. Again it was seen from fixing position that central words typically prove more useful.

By comparing error, it was generally found that the any case and new model proved more effective. Occasionally the 9-gram proved more effective marginally, however this was in cases when every model succeeded in capturing emergence and predicting the sense probabilities with great accuracy.

The new model was the most effective across the board. A crude way of measuring this can be to take the average of all errors.

In cases where one sense was far less likely, adding a third sense proved effective for the any and new model.

Ultimately, the central weighting of the any and new model proved more effective than the 9-gram.

If words which matter are more likely to be near, than boosting these may help them rise above the noise more often.

6. Conclusions

This section seeks to summarise the conclusions of this project. For sake of clarity, its best to break up the project into its sub-goals.

6.1 Position Fixing

By fixing position it was seen that there was indeed a difference in the results. The central positions were consistently more effective at capturing the sense of the given target word.

By looking at the results of the inspector programme, it was seen that by biasing one side important words were frequently missed. As such the EM algorithm had a harder time identifying senses for positions 1 and 5.

Looking at the real words, this was found to be the case for all three of the google 5-gram words, and all the successful COHA words. Mouse in the COHA corpus proved to be unsuccessful. This test is still relevant in showing the limits of the programme in determining sense. Mouse had the least number of occurrences.

Using pseudo-neologisms this central bias was shown again.

6.2 Comparing Models 9gram vs Any

For the real words the central weighting of the Any case proved more effective than the 9-gram. In cases like Jet, it was the case that the 9-gram failed while the “any” case succeeded.

This central weighting likely helped more important words rise above the noise. This can be seen in the inspector programme.

Looking at the pseudo-neologisms, it was found that the Any case typically performed better. In cases where the 9-gram achieved lower error, it was generally that every model had performed particularly well.

It was found that the model became significantly less accurate around the low thousands mark. This is roughly where the unsuccessful mouse case lies as well.

6.3 New model

The new model which sought to combine the 9-gram with the any, in effective giving the effective weighting combined with the zoomed out 9-gram proved very effective.

Error was typically lower and proved a much more reliable method than either the 9-gram or the any model. The benefit of this model was less so in reduced error, but rather improved generality. Although the Any proved more effective in most cases, in cases where the 9-gram performed better the new model would perform better than the any.

As a crude method of analysis, it was found that it had the lowest average error of the 3 methods.

While rather straightforward attempt, this model shows that by manipulating the creation of n-grams weighting can be applied to areas around the N-gram. This in turn can improve results.

7. Future Work

From this project it was found that target word position was a factor of success in N-gram analysis. The benefit of a centrally weighted model, which favoured words nearer the target word was shown to be effective.

Further exploration could be done into whether there is a difference for different parts of speech (i.e., adjectives, nouns, verbs, etc…).

A possible avenue of further study would be to implement more complex models. These would possibly give a weight to the left or right accordingly for different parts of speech if it was determined that such a bias existed for different parts of speech.

Bibliography

[1] J. R. Firth, “A synopsis of linguistic theory, 1930-1955,” *Stud. Linguist. Anal.*, 1957.

[2] R. Navigli, “Word sense disambiguation: A survey,” *ACM Comput. Surv.*, vol. 41, no. 2, 2009, doi: 10.1145/1459352.1459355.

[3] S. Schwab, “The Internet Isn’t Available in Most Languages,” *The Atlantic*, 2015.

[4] W. Weaver, “Translation,” in *Machine Translation of Comput Languages: Fourteen Essays*, 1949.

[5] M. Nasiruddin, “A State of the Art of Word Sense Induction : A Way Towards Word Sense Disambiguation for Under-Resourced Languages Word Sense Induction (WSI),” no. February, 2013.

[6] H. Schütze, “Dimensions of meaning,” in *Proceedings of the International Conference on Supercomputing*, 1992, vol. Part F129723, doi: 10.1109/superc.1992.236684.

[7] H. Schuetze, “Automatic Word Sense Discrimination,” *Comput. Linguist.*, vol. 24, no. 1, pp. 97–123, 1998.

[8] M. EMMS, “An unsupervised EM method to infer time variation in sense probabilities,” *Icon*, vol. 2015, p. 12th, 2015.

[9] M. Emms and A. Jayapal, “Dynamic generative model for diachronic sense emergence detection,” *COLING 2016 - 26th Int. Conf. Comput. Linguist. Proc. COLING 2016 Tech. Pap.*, pp. 1362–1373, 2016.

[10] A. Jayapal, “Finding diachronic sense changes by unsupervised methods.,” Trinity College, University of Dublin.

[11] G. Wiedemann, S. Remus, A. Chawla, and C. Biemann, “Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings,” Sep. 2019, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/1909.10430.