Word Vector-Space Embeddings of Natural Language Data over Time

Chiraag Lala - s1354622

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2014

Abstract

Words are often mapped to vectors in a vector-space (Euclidean-space). Such mappings, also called embeddings, are used in many Natural Language Processing (NLP) tasks. These word embeddings are, generally, intended to reflect the usage, semantic similarities and relatedness of the words they represent. Simply put, word embeddings reflect the meaning of the words relative to other words. However, word meanings are known to change over time (semantic change). Current publicly available word vector-space embeddings are ‘static’ in nature with no temporal component. Creating ‘dynamic’ word embeddings by adding temporal information opens the possibility of capturing the phenomenon of semantic change. These embeddings (with temporal component) can be used to produce visual animation of semantic change and change in word relations over time. It also has the potential to improve performance of various NLP tasks, particularly those involving time like the task of Diachronic Text Evaluation.

This project achieves the following: (1) Create word embeddings with time component (dynamic embeddings) that captures the meaning/usage/similarities of words across various times ranging between the years 1800 and 2008. (2) Develop a tool/software that animates changes in word relations using the dynamic embeddings. (3) Evaluate the dynamic embeddings created using word similarity measures and Diachronic Text Evaluation task.

Acknowledgements

Thank you Dr Shay Cohen for all the help and guidance. I thoroughly enjoyed working under your supervision and look forward to work with you on more projects in future.

Thank you Dr Mirella Lapata and Dr Bonnie Webber for ideas on evaluation of embeddings

This project is dedicated to all the logophiles out there including my father, Ramesh Lala, and my brother, Rajeev Lala.

A big thanks to my mother, Deepa Lala, for loving me so much!

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Chiraag Lala - s1354622*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Problem

The problem addressed in this thesis is to create word embeddings with temporal component (dynamic embeddings - see 1.1.2). These embeddings should reliably capture the meaning and/or usage of English words over time. An extended problem, also addressed in this thesis, is of visualization of the dynamic embeddings. It is to develop an animation tool that creates visual animations showing how semantic similarities and relations between words have changed over time.

To better understand the problem some essential groundwork needs to be laid. These are included in the following subsections.

1.1.1 Static Word Embeddings

Word Embeddings are mappings (functions) which map words to vectors in a Euclidean vector space (see A.1). Since the projects deals with Euclidean Vectors only, from here onwards 'Euclidean vector space' will simply be called 'vector space'. If V is a vocabulary set of words (and tokens) in English language and $\mathbb{R}^n$ is n-dimensional vector-space for some positive integer n, then a mapping $f : V \to \mathbb{R}^n$ is said to be a word embedding. Given a word w in the vocabulary set V, the vector $f(w)$ is called the word embedding of w (under f). For instance, the vector $f(hello)$ will be called the word embedding of $hello$.

Capturing Semantic Information

Word embeddings are generally intended to reflect the semantic similarity and relatedness of words. Several publicly released word embeddings - Colobert and Weston embeddings (Turian et al., 2010), SENNA embeddings (Collobert et al., 2011), Hierarchical Log Bi-Linear (HLBL) embeddings (Turian et al., 2010) and Huang's embeddings (Huang et al., 2012) - are able to capture surprisingly nuanced semantics even in the absence of sentence structure (Chen et al., 2013). Each feature (component) of a word embedding might have different interpretations - grammatical, semantic, word usage, context, etcetera. Nevertheless, all the popular word embeddings follow the simple untold rule - the closer the word embeddings geometrically, the more similar is their meaning/usage - well almost.

Improving NLP Systems

Word embeddings have proved to be useful at improving the performance of various Natural Language Processing (NLP) systems for tasks like Named Entity Recognition (NER), Chunking, etcetera. This was first demonstrated in Turian et al. (2010) where performance of systems for the tasks NER and Chunking were improved by adding word embeddings - like Collobert and Weston embeddings and HLBL embeddings - as extra word features. Many such studies followed. Low Rank Multi-View Learning (LR-MVL) embeddings achieved state-of-the-art performance on NER and chunking problems (Dhillon et al., 2011). Passos et al. (2014) presents a system that use neural word embeddings to achieve state-of-the-art results on NER in both Computational Natural Language Learning (CoNLL) 2003 training data and Ontonotes NER.

Word embeddings do not include any temporal information. These are fixed in time and hence static. From here onwards, they will be called static word embeddings. The problem at hand is to develop 'dynamic' word embeddings by including temporal information.

1.1.2 Dynamic Word Embeddings

Define T to be a Time Range set. It could be a continuous or a discrete totally ordered set (see A.3). For example, it could be the continuous set of time ranging between years 1800 and 2000, i.e. $T = [1800, 2000]$, or it could be a discrete set of separate years like $T = \{1800, 1805, 1810, ..., 1995, 2000\}$. Each element of T will be called a time slice or a year slice (if it is clear that the time slices represent years). In this project we deal with discrete finite time range sets only, so from here onwards T will be a discrete finite set unless specified otherwise.

Given a time range set $T = \{t_1, t_2, ..., t_m\}$ (where $t_1 < t_2 < ... < t_m$), vocabulary set V and an n-dimensional vector space $\mathbb{R}^n$ (for some positive integer n), a mapping $f : V \times T \rightarrow \mathbb{R}^n$ will be called a dynamic word embedding (or simply dynamic embedding). Given a word w in vocabulary set V and a time slice t in time range set T, the vector $f(w,t)$ will be called dynamic embedding of w at time t. In addition the finite sequence $(f(w,t_1), f(w,t_2), ..., f(w,t_m))$ will be called the trajectory of the word w. For example if $T = \{1800, 1805, 1810, ..., 2000\}$, then $f(hello, 1830)$ will be called the dynamic embedding of $hello$ in the year 1830 and the sequence $(f(hello, 1800), f(hello, 1805), ..., f(hello, 2000))$ will be called the trajectory of $hello$.

One way to look at dynamic embeddings is that it is a stack of static word embeddings, one for each time slice. So to develop 'reliable' dynamic word embeddings means to develop a stack of static word embeddings (one for each t in T) such that each static word embedding reflects the true semantic similarity and relations of words in that time instance.

1.1.3 Visualization of Dynamic Embeddings

Another way to look at dynamic embeddings is from the word perspective. Each word is moving with time, as depicted by the trajectory of the word (see 1.1.2). If the embeddings are reliable then for a collection of words their trajectories show how semantic relations of those words with respect to each other have changed over time. One of the goals of this thesis is to develop an animation tool to visualize the word trajectories. This poses a challenge as dynamic embeddings created are in high dimensional vector spaces (in this project 25, 50, 100 and 200 dimensional dynamic embeddings are created) and not in 2- or 3-dimensional space.

1.2 Motivation

Extending Static Word Embeddings to Dynamic Embeddings is a simple, elegant and yet a challenging idea. This idea is mainly driven by the following reasons.

1.2.1 Semantic Change and Word Formation

The main motivation for creating Dynamic Embeddings comes from the fact that natural language is a dynamic entity. Meaning of words are known to change over time (semantic change). For instance, the English word "abandon" now means "give up completely", like abandoning hope, abandoning a baby or surrendering ourselves to emotion. But in 14th century Middle English it meant "to subjugate or subdue". This example was obtained from Oxford English Dictionary website `http://www.oed.com/` with access provided by University of Edinburgh. The online dictionary has a service called timeline which charts chronologically the story of a word from its birth to the present day. Using it, numerous other examples can be obtained like *mouse* (from a rodent to computer device), *nice* (from foolish to good), *pedant* (from school master to person excessively concerned with minor details), *bully* (from good fellow to a tyrant), etcetera.

In addition to semantic change, new words get created from time to time like *laser*, *laptop*, *twitterati* and the most recent ones like *selfie*, *bestie*, etc. Occasionally many old rarely used words get extinct over time like *abactor* (which meant cattle thief), *roentgen* (which was a unit of x-radiation and gamma radiation), *radix* (for the root of a number in mathematics or the root of a word in linguistics). Although the project does not deal with the theory behind semantic change, word formation, and word extinction, it nevertheless provides sheer empirical data that promises to facilitate the study of these topics from Diachronic Linguistics (Historical Linguistics).

1.2.2 For Machine Learning in NLP Tasks

As mentioned earlier, static word embeddings are already used in many computational systems for various NLP tasks (like NER, chunking, etc.) as extra word feature to improve the performance of the systems (see 1.1.1). With temporal information added to the embeddings, dynamic embeddings could prove to be useful in further improvement of the same systems that use static word embeddings. In fact, dynamic embeddings promise to be particularly useful in the task of Diachronic Text Evaluation (see chapter 5.3). Diachronic Text Evaluation is a machine learning task that addresses the interesting problem to automatically determine the period when a text was written (Popescu and Strapparava, 2013). Here the inherent temporal information in the dynamic embeddings could prove useful. The potential to improve several machine learning and NLP systems is a strong motivation to pursue this project.

1.2.3 This is Pure Fun and Insightful

A final motivation (though one of less academic importance) is that seeing words move around depending on how its meaning and/or usage changes over time is simply a delight, especially for word loving logophiles. Take for example, the words *car*, *machine* and *computer*. Upon entering these words as inputs to the animation/visualization tool developed in the project, you will get to see an animation where initially the words *car* and *machine* move around randomly from the year 1800 onwards and then the two words come close and stay together from the year 1900 onwards. Later in the year 1920s, the word *computer* enters the frame moving randomly. Over the coming years the word *computer* gradually comes closer to the word *machine* and *car* drifts away from the rest. Finally from the year 1970 onwards the words *computer* and *machine* stick together while *car* is away from both. See the animation on `http://homepages.inf.ed.ac.uk/scohen/car-machine-computer.gif`.

The above mentioned animation indicates that before the automobile was invented, *car* and *machine* had little relation. Around the time the automobile was invented, *machine* and *car* had related meanings. At some point, *machine* turned to mean *computer*. Animations like these are good fun to watch and quite insightful into the history of how things and concepts, and not only words, have evolved over time.

1.3 Core Ideas to Solve the Problem

Many problems and challenges, both technical/practical and theoretical, arose during the project. These were most comfortably resolved by elegant, creative and sensible solutions. The solutions involve several interesting ideas. The most important 'core' ideas over which the entire project was built on are as follows:

1.3.1 Learn Meaning of a Word From its Context

"The meaning of a word is its use in the language" (Wittgenstein, 1973)
"You shall know a word by the company it keeps" (Firth, 1957)

People learn meanings of many new words, which they have never seen before, throughout their life. Many times meanings are learnt without referring to a dictionary. This is achieved, perhaps, by guessing the meaning of the new word from its context (neighbouring words with known meanings). Gradually, with repeated encounters with the same word and repeated guesswork that follows, the true meaning of the word is learnt. Several theories on Language Acquisition and Vocabulary Development have been suggested over the years like Noam Chomsky's Language Acquisition Device (LAD) (Chomsky, 1965), Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997), Learning words from sights and sounds (Roy and Pentland, 2002), etc. These theories rely on the concept of word co-occurrence (see A.4). In linguistic sense, word co-occurrence can be interpreted as an indicator of semantic proximity. The project exploits this very idea to learn the dynamic embeddings of words in various time periods using context of those words in that era.

1.3.2 Google Books N-grams

Temporal information for dynamic embeddings needs to be learnt from a reliable source of text data. The data source should be vast spanning across a long time range. It should reliably represent the language of the time period during which a particular text was written. It should, in addition, provide contextual information of the words because after all the idea is to learn word meanings/usage (via embeddings) from the context. Many modern textual data sources, like social network data, have a relatively shorter time range. Language used in twitter, for instance, is very recent because twitter was launched in 2006, just 8 years ago. Also the data should be in a format that is useful to solve the problem with clear indications to the time of use. The open source Google Books N-grams is a smart fit to all our requirements (Michel et al., 2011). It has textual data from the books spanning across a vast time range. The data comes with time markers which marks the time period during which the book containing the data was published. To use this data in generating dynamic embeddings is an important idea over which this project thrived.

1.3.3 Dimension Reduction Using MDS

Animation of word trajectories on a 2-dimensional computer screen calls for dimension reduction of the high-dimensional dynamic embeddings. But dimension reduction could lead to loss of inherent data and similarity information of the data points. This poses a challenge - to preserve the inherent information of the high-dimensional dynamic embeddings after dimension reduction. This problem is addressed mathematically and computationally using Multi-Dimesional Scaling (MDS) (Borg and Groenen, 2005) which forms the core mathematical idea behind the animation/visualization tool developed in this project.

1.4 Testing and Evaluation Techniques

The dynamic embeddings created in the project are evaluated using two techniques - Diachronic Text Evaluation and Word Similarity Measures

1.4.1 Diachronic Text Evaluation

Natural Language changes over time. Texts from different era look and sound different. For example, Shakespeare English is very different from modern day English. Could we have an automated system that determines the period in which a text was written using these differences? This would be a machine learning task in NLP where given a text document as an input, the machine predicts the time/year/period in which the text was written.

Because of the temporal component, an ideal dynamic embedding would carry the true semantic relations of words for each time slice, although latently. Hence dynamic embeddings could, perhaps, contribute to Diachronic Text Evaluation (DTE) task by making use of the stored latent semantic information of words over time. We would like to see whether the use of Dynamic Embeddings as features for the words (instead of static word embeddings) improve the performance of DTE systems or not. In our evaluation tests the performance of a basic DTE system (see 5.3) did improve with dynamic embeddings created. This indicates that the dynamic embeddings created are credible, in the sense that they seem to capture the true semantic relations of words in various times.

1.4.2 Word Similarity

The intuition that similar words have their embeddings close to each other needs to be tested. A small corpus of word pairs rated for similarity by human subjects is obtained. The same word pairs are scored for similarity using the embeddings created. The two scores (human rating and scores from dynamic embeddings) are compared to determine how well word similarities have been represented. The comparisons made in our study show that dynamic embeddings broadly follow the idea - 'the closer the embeddings geometrically, more similar the meanings' - however these are not as accurate as the popular static embeddings - Collobert and Weston embeddings and HLBL embeddings at presenting word similarity. More analysis and conclusions in chapter 5 and 6.

1.5 Outline

The main objective of this project is to create reliable dynamic embeddings and use it for animation. Accordingly the project work was divided into four parts - (1) Literature Review and Background reading to develop the idea firmly, (2) Create the Dynamic Embeddings, (3) Develop Animation/Visualization tool and (4) Evaluate the Dynamic Embeddings. Each of the above four parts of project work are described in detail in dedicated chapters (one for each part). These are chapters 2 to 5. Chapter 6, the final chapter, presents the conclusion and ideas for potential future work arising from the project.

Chapter 2

Background - Literature Review

At the early stages of the project work, time was spent to find and study relevant research from similar domains. This was done mainly to familiarize with all the requirements of the project and to draw ideas from previous works that could then be combined to develop a detailed execution plan. Besides ideas, we also borrowed datasets, evaluation framework and motivation to design algorithms. In addition to early stage readings, a few more publications and research topics were explored at later stages of the project, and even towards to end, to solve technical obstructions encountered and for other on-course corrections that were needed.

2.1 Fields of Research Explored

Computational Linguistics is a wide interdisciplinary subject spanning across different domains in Linguistics and Computer Science. Of many areas of research, three main fields of research of direct relevance to the project were explored in immense detail. These are (1) Word Representation, (2) Machine Learning and (3) Temporal Information in NLP. There is no clear cut boundary separating these fields. In fact most research papers explored come from overlapping of these fields.

2.1.1 Word Representation

The broad field of Word Representations deals with the challenge of computationally representing word meaning and other syntactic and semantic features. Such representations are either symbolic (including information such as the word's morphology, its synonyms, its part of speech and other symbolic information) or as a mathematical object in a vector space or other discrete structure. The focus of this project has been on the latter – word embeddings in a Euclidean space. Several publications on word representations, particularly word embeddings, were studied in detail to get familiarized with the field and to draw ideas for solutions.

2.1.2 Machine Learning

Machine Learning deals with construction and study of systems that can learn from data. It involves using Statistics and Mathematics to perform several tasks like clustering, classification, regression, dimensionality reduction, etcetera. Dimensionality reduction (an integral part of the visualization tool) and Regression (used in Diachronic Text Evaluation task for evaluating the dynamic embeddings created) are machine learning tasks. Many resources and publications on machine learning topics, specially those needed in the project, were studied in great detail.

2.1.3 NLP Dealing with Time

Most current natural language processing (NLP) deals with language as if it were a constant. This however, is not the case as language is continually changing. Few studies in NLP like Dynamic Topic Models (Blei and Lafferty, 2006), Detecting Epoch Changes (Popescu and Strapparava, 2013), Diachronic Text Evaluation, etcetera take temporal information of natural language data into consideration. Research work in NLP addressing temporal information in natural language data can be regarded to form a new research area. Several publications from this area were studied for inspiration and for evaluation framework.

2.2 Key Publications

Just as the practical project work was divided into three parts (1. Create Dynamic Embeddings, 2. Visualization tool and 3. Evaluation) similarly the search for literature was also carried from the same three parts in that order. First publications and books to help develop methods for creating dynamic embeddings were searched and studied. Later papers and resources for developing the animation tool and finally works that could help evaluate the embeddings were explored.

Of all the publications that were read, the important (key) publications, resources and data used in the project work are mentioned in the following subsections.

2.2.1 Literature and Data to Create Dynamic Embeddings

Computational Word Representations have been well studied in Turian et al. (2010). The paper describes how performance of existing NLP systems improve when word representations are taken as extra word features. The study uses publicly available Brown Clusters (Brown et al., 1992), Collobert and Weston embeddings and HLBL embeddings (Turian et al., 2010). The latter two are static word embeddings. These open source static word embeddings were downloaded from `http://metaoptimize.com/projects/wordreprs/` website provided in Turian et al. (2010). These embeddings form the starting points from which our dynamic embeddings were created.

Dynamic embeddings are intended to reflect the meaning and/or semantic similarity and/or relatedness of words. To learn word meaning, we borrowed the popular idea - distributional hypothesis (Harris, 1954) - that the context captures the meaning of a word. Distributional hypothesis has been documented and used in various works like the publication - Landauer and Dumais (1997) - on Latent Semantic Analysis. The paradigm of acquiring word representation based on distributional hypothesis using n-grams (see A.5) has been widely explored. Words are clustered into groups based on their context in Brown Clustering using n-grams (Brown et al., 1992). Similar clustering of words using n-grams was demonstrated in Uszkoreit and Brants (2008). N-grams are also used in creation of static word embeddings (Sahlgren, 2006; Turney et al., 2010). This inspired us to develop a method, based on n-grams, to create our dynamic embeddings.

Temporal component is what makes dynamic embeddings different from other existing embeddings. This calls for ways to deal with time. For ideas on dealing data with time information, we looked at past works in NLP that addressed temporal information. These works include Dynamic Topic Models (Blei and Lafferty, 2006; Wang et al., 2012), Detection of Language Change over Time in Large Corpora (Popescu and Strapparava, 2013, 2014), and Quantitative Analysis of Culture Over Time (Michel et al., 2011). Dynamic Topic Models analyze the time evolution of topics in large document collections. Blei and Lafferty (2006) analyzed the OCRed archives of the journal

Science from 1880 through 2000 for time evolution of topics in a discrete sense - i.e. where time range set was discrete. Wang et al. (2012) analysed two news corpora for time evolution of topics in a continuous sense - i.e. where time range set was continuous. The works on culture change over time (Michel et al., 2011) and language change over time (Popescu and Strapparava, 2014) introduced and demonstrated use of Google Books N-grams. The Google Books N-grams data (that has time component) was found suitable to our needs (see 1.3.2). Hence it was partially downloaded (`http://storage.googleapis.com/books/ngrams/books/datasetsv2.html`) and used in creating our dynamic embeddings. The Google N-gram data was downloaded 'partly' due disk-space constraints.

2.2.2 Literature and Resources for Visualization

The greatest challenge for the visualization task was Dimension Reduction. Various Machine learning techniques for dimension reduction, like Principal Componenet Analysis (PCA) and Singular Value Decomposition (SVD) and Multi-Dimensional Scaling (MDS), were studied in great detail from popular Machine Learning Text Books - Murphy (2012) and Borg and Groenen (2005) - and research paper - Shaw and Jebara (2009). Later MDS was used for dimension reduction as it was found easier to apply to dynamic embeddings created (see section 4.1). All programs were planned to be developed in Python programming language due to its suitability to NLP tasks. To maintain continuity of programming language, a Python based Machine Learning toolkit - Scikit Learn (Pedregosa et al., 2011) - was used for MDS. Just for animation, a Perl Script was developed. The script takes 2-dimensional dynamic embeddings as input and produce an animation of trajectories using Bresenham line drawing algorithm (Bresenham, 1965) in animated GIF.

2.2.3 Literature and Data for Embedding Evaluation

Dynamic embeddings are evaluated from two perspectives - (1) How well do they represent similarity and relatedness of words and (2) How well do they represent the language change over time.

Evaluation of Similarity and Relatedness

We looked at ways in which existing static embeddings are evaluated for similarity and relatedness. We came across two methods - (1) Evaluation using classifiers (Chen et al., 2013) and (2) Evaluation using word similarity measures (Agirre et al., 2009).

Consider a training set of word pairs, each pair labelled as synonyms or antonyms. Similarly consider a test set (different from training set) containing synonym pairs and antonym pairs. Train a 2-class classifier on the embeddings of training set and test the classifier on the embeddings of test set. The idea is that if the classifier is successful at detecting synonym and antonym pairs then it indicates that the classifier has learnt synonym/antonym relation between words from the embeddings and thus the embeddings are reliable at storing the synonym and antonym relations of words. This evaluation framework using classifiers has been introduced and demonstrated in Chen et al. (2013).

Evaluation using similarity measures stems from the idea that geometric distance between embeddings measures the extent of similarity and relatedness of the words. Consider a set of word pairs where each pair is rated for similarity and relatedness by human subjects. On other hand measure the similarity and relatedness of same set of word pairs using distance between the corresponding word embeddings. Compare the two scores using correlation to test how well the embeddings match to the ratings by human subjects. This evaluation framework has been inspired from Agirre et al. (2009). WordSim353 dataset (Agirre et al., 2009) of human ratings of similarity and relatedness of word pairs was used in evaluation of our dynamic embeddings. The dataset was downloaded from `http://alfonseca.org/eng/research/wordsim353.html`.

Evaluation of Language Change Over Time

Popescu and Strapparava (2014) used Google N-gram data to explore diachronic phenomena (i.e. language changing over time). The study used statistical approaches to epoch detection (or time period detection) using language differences and in the processes it introduced the Diachronic Text Evaluation (DTE) task. We believe, our dynamic embeddings also reflect the diachronic phenomena and hence we identified DTE useful for evaluation of our embeddings as well (see chapter 5.3). The DTE task involves building a system to automatically detect the period in which a text was written. A novel supervised machine learning system that uses dynamic embeddings was developed for the purpose. This machine learning system uses linear ridge regression from scikit-learn Machine Learning tool (Pedregosa et al., 2011). Ridge regression and other regressions were studied in detail from Machine Leaerning text book - Murphy

(2012). DTE is task 7 of the upcoming SemEval 2015 (SemEval is an ongoing series of evaluations of computational semantic analysis systems). Trial data for training and testing was downloaded from `http://alt.qcri.org/semeval2015/task7/index.php?id=data-and-tools` (SemEval task 7 website).

2.3 Summary

From all the gathered research several ideas, datasets and resources were collected and combined to develop a detailed execution plan. This plan was executed as follows.

Collobert and Weston embeddings and HLBL embeddings (Turian et al., 2010) were combined with Google Books N-gram data (Michel et al., 2011) to develop a novel way of creating dynamic embeddings. This novel way of combining static word embeddings and n-gram data with temporal information was inspired by several works from fields of 'Word Representations' and 'NLP to deal with time'(see section 2.1). A visualization tool was developed using MDS, Bresenham algorithm and animated GIF. MDS was chosen for visualization tool, after analysing several dimension reduction techniques, for its applicability. The idea of evaluation frameworks using word similarity and relatedness was borrowed from Agirre et al. (2009). DTE task (introduced in Popescu and Strapparava (2014)) was used to evaluate dynamic embedding as we believed that our embeddings reflect the diachronic phenomena. For the purpose, a novel supervised machine learning system (involving ridge regression) that uses dynamic embeddings was developed for the DTE task.

Chapter 3

Creating Dynamic Embeddings

Dynamic embeddings are mappings (functions) f of the form $f : V \times T \to \mathbb{R}^n$, where V is vocabulary set, T is a time range set and $\mathbb{R}^n$ is an n-dimensional Euclidean vector space for some positive integer n (see section 1.1.2). Static word embeddings are much simpler mappings g of the form $g : V \to \mathbb{R}^n$ (V, and $\mathbb{R}^n$ are same as before) (see section 1.1.1). An n-gram with time marker refers to a pair of the form $((a_1, a_2, ..., a_n), t)$, where $(a_1, a_2, ..., a_n)$ is an n-gram of words (and tokens) (see A.5) and t is a time marker.

This chapter presents a general recipe of creating dynamic embeddings by combining a static embedding and a list of n-grams with time markers. The recipe was followed in creation of four sets of dynamic embeddings in the project using Collobert and Weston embeddings and HLBL embeddings, and Google Books N-grams data. The entire process of creating the four sets of dynamic embeddings has been described in detail.

3.1 General Recipe

3.1.1 Mathematical Formulation

Let V be a vocabulary set of words (and tokens) and T be a time range set. In addition, we are given a d-dimensional static word embeddings $g : V \to \mathbb{R}^d$ and a finite list L of n-grams with time markers such that all words of the n-grams are in vocabulary set V and all the time markers are in the time range set T. Let the list be

$$L = (((a_1^1, a_2^1, ..., a_n^1), t^1), ((a_1^2, a_2^2, ..., a_n^2), t^2), ..., ((a_1^m, a_2^m, ..., a_n^m), t^m)$$

such that $a^i_j \in V$ and $t^i \in T$, for all $i \in \{1,2,...,m\}$ and for all $j \in \{1,2,...,n\}$.

We choose an integer p from the set $\{1,2,...,n\}$ and call it ‘target position’. In addition we also choose a function $h : \mathbb{R}^{d\times n} \rightarrow \mathbb{R}^k$ and call it the ‘vector operation’ where k is some positive integer. Note that for practical purpose, vector operation h should be computable which is to say that for any given input x in $\mathbb{R}^{d\times n}$ a computer should be able to generate an output $h(x)$ in finite time (faster the better). For any word w_o in V, define I_{w_o} to be the indicator function - i.e. $I_{w_o}(w_o) = 1$ and for $w \neq w_o$ we have $I_{w_o}(w) = 0$. Also let $\odot$ denote the concatenation - i.e. $(a,b,c) \odot (d,e,f) = (a,b,c,d,e,f)$

We are now in a position to formulate the dynamic embeddings. Define dynamic embeddings $f : V \times T \rightarrow \mathbb{R}^k$ as

$$f(w,t) = \frac{\sum_{i=1}^m I_w(a^i_p) \times I_t(t^i) \times h(g(a^i_1) \odot g(a^i_2) \odot ... \odot g(a^i_n))}{\sum_{i=1}^m I_w(a^i_p) \times I_t(t^i)} \tag{3.1}$$

NOTE: It is possible that sometimes for a certain word w' and certain time slice t', the dynamic embeddings $f(w',t') = 0/0$ i.e. *not defined*. Such cases occur when there is no n-gram with time marker t' and with word w' at target position p. In such cases, we mark $f(w',t') = Unk$, short for Unknown which is to say that dynamic embedding of w' at time t' does not exist. Hence for mathematical consistency our dynamic embeddings should be of the form $f : V \times T \rightarrow \mathbb{R}^k \cup \{Unk\}$. So although Unk is not mentioned, the definition of dynamic embeddings is understood to have it in the co-domain.

3.1.2 Description

The formula for dynamic embeddings goes to say that for a given word w and time slice t, we look at only those n-grams that have time marker t and have w at target position. If no such n-grams exist in L then dynamic embeddings of w at t is simply labelled as Unk. On the other hand, if such n-grams do exist then these represent the context of the word w at target position p including itself. Let all the n-grams with w at target position and t time marker form a list $L_{w,t}$. The context depends on length of n-gram , i.e. n, and target position p. If words that occur only to the right of target word are to be considered as context then choose $p = 1$. If only words occurring to left are to be considered as context then choose $p =$n. For a larger context choose n to be large.

The idea of distributional hypothesis is that the context represents the meaning of the target word. So to create dynamic embedding of w, its context words should be used to generate the embedding. One direct way is to start with static embeddings of the context words and then manipulate and combine them to obtain the dynamic embedding of w at t. The required manipulation and combination of static embeddings of context words is achieved using the vector operation h. This way for each different n-gram in $L_{w,t}$ a new vector is obtained. It makes sense to average these new vectors because the more frequent contexts of w must have greater influence on its meaning. Averaging these new vectors amounts to summing the new vectors and then dividing by the total number of these vectors (which is same as number of n-grams in $L_{w,t}$). This is exactly what is depicted in formula 3.1.

Many different types of dynamic embeddings can be created for same set of static embedding and N-gram (with time marker) data. This can be simply achieved by twisting the vector operation h. A different h can lead to a different dynamic embedding. Equation 3.1 can be better understood using examples. A couple of examples are: (1) Simple concatenation of static embeddings of the context words excluding the target word (see 3.1.2.1), and (2) Simple averaging of static embeddings of the context words excluding target word (see 3.1.2.2). Before proceeding to examples here are some simplifications. Let L be list of five-grams with time markers

$$L = (((l_1^1, l_2^1, w^1, r_1^1, r_2^1), t^1), ((l_1^2, l_2^2, w^2, r_1^2, r_2^2), t^2), ..., ((l_1^m, l_2^m, w^m, r_1^m, r_2^m), t^m))$$

We would like to look at context as both words on the right and words on the left of a target word. Hence it makes sense to choose the middle word of five-grams, target position $p = 3$, as target words. For the q^{th} five-gram in the list L, (l_1^q, l_2^q) form the left context and (r_1^q, r_2^q) for the right context of target word w^q.

3.1.2.1 Example 1 - $f_{concatenation}$

Define the vector operation h to concatenate the static embeddings of the context words excluding target word.

$$h(g(l_1) \odot g(l_2) \odot g(w) \odot g(r_1) \odot g(r_2)) = g(l_1) \odot g(l_2) \odot g(r_1) \odot g(r_2)$$

This vector operation was implemented in the project work. From here onwards in the thesis, dynamic embeddings created using the above vector operation h will be called

$f_{concatenation}$ (or simply f_c).

$$f_c(w,t) = \frac{\sum_{i=1}^{m} I_w(w^i) \times I_t(t^i) \times (g(l_1^i) \odot g(l_2^i) \odot g(r_1^1) \odot g(r_2^i))}{\sum_{i=1}^{m} I_w(w^i) \times I_t(t^i)} \tag{3.2}$$

NOTE: If the static word embedding is in $d-$dimensional space ($g : V \rightarrow \mathbb{R}^d$), then we get dynamic embedding in higher $4d$-dimensional space as $f_c : V \times T \rightarrow \mathbb{R}^{d \times 4}$

3.1.2.2 Example 2 - $f_{average}$

Define the vector operation h to average the static embeddings of the context words excluding target word.

$$h(g(l_1) \odot g(l_2) \odot g(w) \odot g(r_1) \odot g(r_2)) = (g(l_1) + g(l_2) + g(r_1) + g(r_2))/4$$

This vector operation was implemented in the project work. From here onwards in the thesis, dynamic embeddings created using the above vector operation h will be called $f_{average}$ (or simply f_a).

$$f_a(w,t) = \frac{\sum_{i=1}^{m} I_w(w^i) \times I_t(t^i) \times (g(l_1^i) + g(l_2^i) + g(r_1^1) + g(r_2^i))/4}{\sum_{i=1}^{m} I_w(w^i) \times I_t(t^i)} \tag{3.3}$$

NOTE: In this case static word embedding and dynamic word embedding are in same $d-$dimensional space - i.e. if $g : V \rightarrow \mathbb{R}^d$, then $f_a : V \times T \rightarrow \mathbb{R}^d$

The main difference between these examples is that in f_c each position in the context is treated separately while in f_a context positions are irrelevant. For instance, consider the following five-grams.

- *the court instructed the jury*
- *the jury instructed the court*

Under $f_{concatenation}$ the contexts will have different contribution to the dynamic embedding of 'instruction'. Under $f_{average}$ the two contexts will have the same contribution.

Both $f_{average}$ and $f_{concatenation}$ have been implemented in the project work. The implementation has been described in detail in the following section.

3.2 The Implementation

Implementation of the recipe (equation 3.1) is achieved first by procuring the required ingredients (datasets) and processing them to be in suitable formats. This is followed by designing an efficient and scalable algorithm to compute the dynamic embeddings and finally running the programming to generate them.

3.2.1 Procuring Data

The two ingredients of the general recipe (see equation 3.1) are (1) Static Word Embedding and (2) N-gram data with time markers.

3.2.1.1 Static Embeddings

Several publicly available static word embeddings were explored. Some of these are

Collobert and Weston Embeddings	`http://wordvectors.org/demo.php`
HLBL Embeddings	`http://wordvectors.org/demo.php`
SENNA Embeddings	`http://ronan.collobert.com/senna/`
RNN Embeddings	`http://rnnlm.org/`

Of the many static word embeddings, Collobert and Weston embeddings and HLBL embeddings have been demonstrated to improve performance of existing systems of NLP tasks - NER and Chunking - simply by adding them as additional word feature (Turian et al., 2010). It indicates that, perhaps, these static word embeddings are of good 'quality' (because these were found useful in Turian et al. (2010)) and can add to the quality of dynamic embedding created. Hence it was decided to develop our embeddings from these two static embeddings. Both these embeddings come in different dimension and for different vocabulary sizes. They are also trained using different methods.

Collobert and Weston Embeddings data provides static word embeddings of 268810 lexical terms (includes words) in 25, 50, 100 and 200 dimensional Euclidean spaces. These are trained based on Neural Language Model (NLM) (For more details on NLM refer Bengio et al. (2006)). HLBL embeddings data provides static word embeddings of 246122 lexical terms in 50 and 100 dimensional Euclidean spaces. These are trained on Hierarchical log bi-linear (HLBL) language model (for more details on HLBL language model refer to Mnih and Hinton (2009)). Both the embeddings are induced as described in the paper - Turian et al. (2010) - on the Reuters Corpus Volume 1 (RCV1) corpus, cleaned as described in the paper (roughly 37M words of News text). Also, these embeddings are scaled and come in a common format as a Text (.txt) file where each line looks as follows:

```
word    vector
```

An example line of a word and its 5 dimensional embedding,

```
artificial   1.135467   2.937684   8.443371   6.378421   0.323859
```

For the project, 25-dimensional Collobert and Weston embeddings and 50-dimensional HLBL embeddings were downloaded.

3.2.1.2 N-gram With Time Stamps

Ideally to capture more context, n-grams with larger n should be selected. But then the data would be enormous and will take lot of computing time for processing. A right balance of context and ease of computing was to be struck. The downloaded HLBL embeddings were trained using five-grams (Turian et al., 2010). Smaller n-grams were expected to have mostly functional words (‘the’, ‘a’, ‘an’, ‘of’, ‘in’, etcetera) around nouns and verbs - like `the ball of`. This might not be fruitful. Larger than 5, the n-grams data would be massive and difficult to process. In the project we settled for five-grams with context of two terms to the left and two to the right of the target term.

Google Books five-grams were found suitable to our purpose. The five grams came with time stamps as years in which the books containing the five-grams were published. These are books from official sources documented by Google and hence we believe that Google Books five grams reliably capture the language of various epoch. The time span covered by Google Books N-grams is vast ranging in reverse chronological order from the year 2008 to years as far back as 1505 (or even more). Several different 5-grams dating back to 16th and 17th century, like the five gram `aire of Odcombe in the` from 1618 were found. Notice the use of the word ‘aire’ which is an archaic word no longer used and not found in n-grams of later years. Back then ‘aire’ meant ’an altar’ (Online Oxford English Dictionary’s timeline service, `http://www.oed.com/search?q=aire&scope=ENTRY&timeline=true&type=dictionarysearch`, with access provided by University of Edinburgh was used to find the history of the word ‘aire’). The five-grams data from google is the only source of n-gram data of such kind. Due to its reliability and time range, we decided to use it for our purpose.

These datasets come as a text file where each line is in the following format.

```
5gram     year     match-count     volume-count
```

For our purpose, only the 5gram, year and match count data is needed. Match count is the number of times the specific 5gram appears in the specified year. More that 1TB of such data is available. We looked at only a small fraction of the n-gram data of around 176GB. The 5gram in the dataset included Part-Of-Speech tags appended to the lexical terms. POS tags were of no use and hence were removed using ‘regular expressions’ from re module in python (for more on regular expressions in python refer to Wikibooks module on the topic - `http://en.wikibooks.org/wiki/Python_Programming/Regular_Expression`). After stripping the POS tags, the datasets text file had each line that look like,

```
analysis is often described as     1991     2     1
```

Here the 5gram (`analysis is often described as`) appears 2 times in the books that were published in the year 1991. The data was further segregated into text files for each year - for example all five-grams with same time marker, say 1998, were stored in one text file for that year (see **??**).

3.2.2 Developing Algorithm

Algorithms to compute dynamic embedding by combining static embedding and N-gram data were developed. Two algorithms, one for $f_{average}$ (see 3.1.2.2) and one for $f_{concatenation}$ (see 3.1.2.1) were developed. These two algorithms are exactly similar, except for the vector operation h used - in one case vectors are added and in other case the vectors are concatenated. Due to the sheer size of the 5-gram data, importance was given to make the algorithm efficient and scalable.

Ideas For Efficiency

For the task of computing dynamic embeddings, inspired by the formula 3.1, a brute-force method would be to compute $f(w,t)$ by iterating over the entire range of n-grams repeatedly for each word w and each time t. Such a brute-force algorithm would be highly inefficient. The processing time can be cut down drastically by smart usage of the data structure ‘dictionaries’ in python (for more on dictionaries in python refer to Wikibooks module on the topic - `http://en.wikibooks.org/wiki/Python_Programming/Dictionaries`). Secondly, instead of iterating over the entire or even a small list of n-grams, the algorithms were designed to iterate just once over the n-gram list.

Algorithm

A pseudocode of the algorithm is presented below.

```
Step 1: Store static word embedding $g$ as a dictionary $W$ with words $w$ from vocabulary set $V$ as its keys and vectors $g(w)$ as the corresponding values.
Step 2: Define two other dictionaries $H$ and $C$ with all word-time pairs $(w,t)$ from $V \times T$ as its keys and initialize the corresponding values in $C$ to 0 and corresponding values in $H$ to zero vector $(0,0,...,0)$. The dimension of vectors in $H$ is same as the expected dimension of dynamic embeddings.
Step 3: Read five-gram data, one line at a time. Each five-gram represented as $((l_1,l_2,w,r_1,r_2),t,c)$ where $c$ is the count of how many times the five-gram was repeated in that year $t$.
    Sub-step 3.1: Search the static embeddings of the context words in $W$ i.e $g(l_1),g(l_2),g(r_1),g(r_2)$
    Sub-step 3.2: Compute $h(g(l_1) \odot g(l_2) \odot g(w) \odot g(r_1) \odot g(r_2))$. Vector operation $h$ could be any - like averaging or concatenation excluding target word $w$.
    Sub-step 3.3: Add the vector computed in step 3.2 to value of the key $(w,t)$ in dictionary $H$. Add the count $c$ to the value of the key $(w,t)$ in dictionary $C$.
Step 4: For all $(w,t)$ pairs, divide the value of the key $(w,t)$ in $H$ by the value of the same key in $C$. In case, the count value is 0 then label the value of the key $(w,t)$ in $H$ to be $Unk$.
Step 5: $H$ obtained is the dynamic embedding.
```

3.2.3 Generating Dynamic Embeddings

Two sets of python programs on the basis of algorithm in 3.2.2 were developed - one for $f_{concatenation}$ and other for $f_{average}$. These were made to run on two static word embeddings - 25-dimensional Collobert and Weston embeddings (also called NLM embeddings) and 50-dimensional HLBL embeddings. Hence four different dynamic embeddings were created.

- $f_{average}$ using NLM embeddings (25-dimensional dynamic embedding)
- $f_{concatenation}$ using NLM embeddings (100-dimensional dynamic embedding)
- $f_{average}$ using HLBL embeddings (50-dimensional dynamic embeddings)
- $f_{concatenation}$ using HLBL embeddings (200-dimensional dynamic embedding)

Vocabulary set V was same as all the words and terms in the static word embeddings. Time range set T was chosen to be

$$T = \{1800, 1805, 1810, ..., 2000, 2005, 2008\}.$$

Around 7.5 GB of dynamic embeddings were generated in almost 60 days starting from first week of June 2014 to first week of August 2014. These are stored in text (.txt) files, one file for each time slice. The files are in the same format as the static embeddings (see 3.2.1.1). For example, head and tail excerpts of dynamic embedding created using $f_{average}$ and NLM embeddings for the year 1975 are presented in Appendix **??**. Such files for each year slice in time range T and for each of the four dynamic embeddings were created.

3.3 Summary

The figure 3.1 presents a succinct outline. The general recipe (see 3.1) of dynamic embeddings takes static word embeddings and combines it with n-gram data (with time markers). The choice of static word embedding g data, n-gram data, target position p, vector operation h, vocabulary set V and time range set T uniquely determine the dynamic embedding to be created. In the project Collobert and Weston, and HLBL static word embeddings were considered (see 3.2.1.1). A small fragment of Google Books five-grams was used (see 3.2.1.2). The middle word of the five-grams was the chosen target position $p = 3$. Averaging and Concatenation excluding target word were

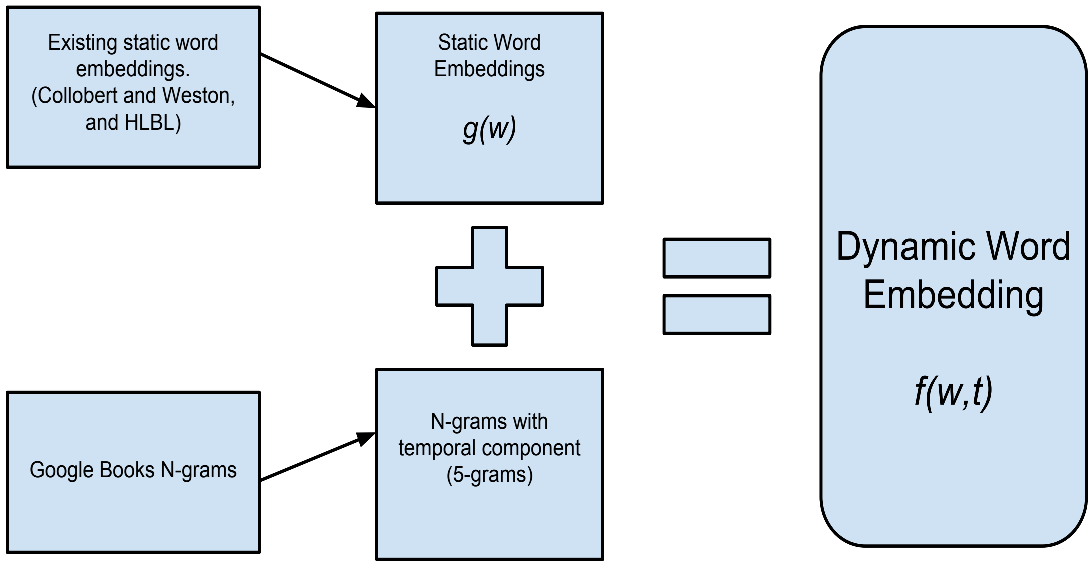

Existing static word embeddings. (Collobert and Weston, and HLBL)
Static Word Embeddings
g(w)
Google Books N-grams
N-grams with temporal component (5-grams)
Dynamic Word Embedding
f(w,t)

Figure 3.1: *Creating Dynamic Embeddings: An Outline.*

the vector operations used (see 3.1.2.2 and 3.1.2.1). Vocabulary set was same as the words in static embeddings. Time range set were taken at steps of 5 years starting from 1800 to 2005 in addition to the year 2008. All these components were combined using an efficient algorithm designed for this purpose (see 3.2.2). Four different sets of dynamic word embeddings were generated and stored as text files (see 3.2.3).

Chapter 4

Visualization Tool

Dynamic embeddings $f(w,t)$ can be viewed from two perspectives - (1) time perspective and (2) word perspective. The time perspective looks at dynamic embeddings as a stack of static word embeddings, one for each time slice. Given a fixed time t, the set of embeddings $\{f(w,t)|w \in V\}$ looks like any other static word embedding. The word perspective looks at trajectories of words (see 1.1.2). Given a word w, its trajectory - $(f(w,1800), f(w,1805), ..., f(w,2005), f(w,2008))$ - in high dimensional space depicts the motion of the word. Trajectories of three or four (or more) words taken together are expected to depict the changing relations between those words over time.

A visualization tool that animates the trajectories of set of words given to it as input was developed. This chapter describes that tool.

4.1 The Challenge

Dynamic embeddings generated are in high dimensional space (25-, 50-, 100- and 200-dimensional dynamic embeddings were created. See 3.2.3). Dimension reduction to 2-dimensional space is required to visualize the trajectories of the input set of words.

The main idea is to project the trajectories of words to a 2-dimensional plane within the high dimensional space. The choice should be such that there is minimum loss of the word relationship information. For instance, imagine two word having trajectories in 3-dimensional Euclidean space. Let the trajectory of first word be
$((1,0,0),(0,1,0),(-1,0,0))$ and the trajectory of second word be
$((-1,0,0),(0,-1,0),(-1,0,0))$. Notice that both words are in the X-Y plane in the

3-dimensional space. If now the trajectories are projected onto the X-Z plane then there is loss of information. After projection to X-Z plane the first word trajectory is $((1,0),(0,0),(-1,0))$ and for second word it is $((-1,0),(0,0),(-1,0))$. It is intuitively clear that there is loss of information because the two trajectories meet at $(0,0)$ after projection to the X-Z plane. The correct projection, in this case, would have been to the X-Y plane where relation between trajectories would be preserved.

Choice of the plane poses a challenge. Such problems are addressed in dimensionality reduction techniques like Singular Value Decomposition (SVD) and Multidimensional Scaling (MDS). Both the techniques are based on Principal Component Analysis (PCA) and make use of eigenvectors and eigenvalues from linear algebra. For a detailed mathematical treatment on SVD, refer to chapter 5, 'Singular value decomposition and principal component analysis', of the book 'A practical approach to microarray data analysis' (Wall et al., 2003). For MDS refer to book 'Modern Multidimensional Scaling: Theory and Applications' (Borg and Groenen, 2005).

4.2 Solution

We believe that Euclidean distance (see A.2) between the word embeddings represent their semantic similarity - Smaller the distance between the embeddings of two words, more similar is the meaning/usage of two words. Hence the objective is to preserve the distance between the trajectories of the high dimensional space as much as possible after dimension reduction to 2-dimensional space. MDS does exactly that and hence it was chosen for the purpose.

4.2.1 Formulating MDS

Let finite number of objects be $\{Ob_1, Ob_2, ..., Ob_n\}$. Suppose the distance between each pair of objects is known. Denote $\delta_{i,j}$ as the distance between Ob_i and Ob_j. Dissimilarity matrix (or distance matrix) Δ is a $n \times n$ square matrix defined as

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & ... & \delta_{1,n} \\ \delta_{2,1} & \delta_{2,2} & ... & \delta_{2,n} \\ \vdots & \vdots & & \vdots \\ \delta_{n,1} & \delta_{n,2} & ... & \delta_{n,n} \end{pmatrix}$$

Given Δ and a positive integer m, MDS finds n vectors $x_1, x_2, ..., x_n$ in m-dimensional Euclidean space $\mathbb{R}^m$ satisfying the following condition.

Given any n vectors $y_1, y_2, ..., y_n$ in $\mathbb{R}^m$, then

$$\sum_{i<j} (\|x_i - x_j\| - \delta_{i,j})^2 \leq \sum_{i<j} (\|y_i - y_j\| - \delta_{i,j})^2 \tag{4.1}$$

In other words, MDS finds n vectors $x_1, x_2, ..., x_n$ (representing the n objects $\{Ob_1, Ob_2, ..., Ob_n\}$) in m-dimensional Euclidean space $\mathbb{R}^m$ in such a way that the Euclidean distance (see A.2) $\|x_i - x_j\| \approx \delta_{i,j}$ as much as possible. Simply put, MDS finds vectors in m-dimensional space representing the objects and preserves the distance relation between the objects.

4.2.2 Applying MDS

Scikit-Learn is a python based machine learning module which has an in-built function for applying MDS called manifold.MDS. This function uses numerical methods and PCA to solve the optimization problem 4.1. Starting with initial n vectors $v_1, v_2, ..., v_n$ it reaches to a solution $x_1, x_2, ..., x_n$ after many iterations improving its solution in each iteration. For visualization purpose, manifold.MDS was used to find objects in 2-dimensional Euclidean space. The procedure involves taking a distance matrix as a list of list in python and providing it as input to manifold.MDS. It produces 2-dimensional solution vectors $x_1, x_2, ..., x_n$ as a list of list. Other parameters of the manifold.MDS function like maximum number of iterations, number decimal places of accuracy, dimension to be reduced to (in our case 2), etcetera are also required to be set before applying the function.

4.3 The Tool

A visualization tool based on MDS and animation was developed. This tool allows users to compare the trajectories of any set of words (from the dynamic embeddings). An outline of how the application works is presented in the flowchart 4.1. A web interface (see 4.2) for the application has also been created. Please check the web application at `http://kinloch.inf.ed.ac.uk/words/index.php`.

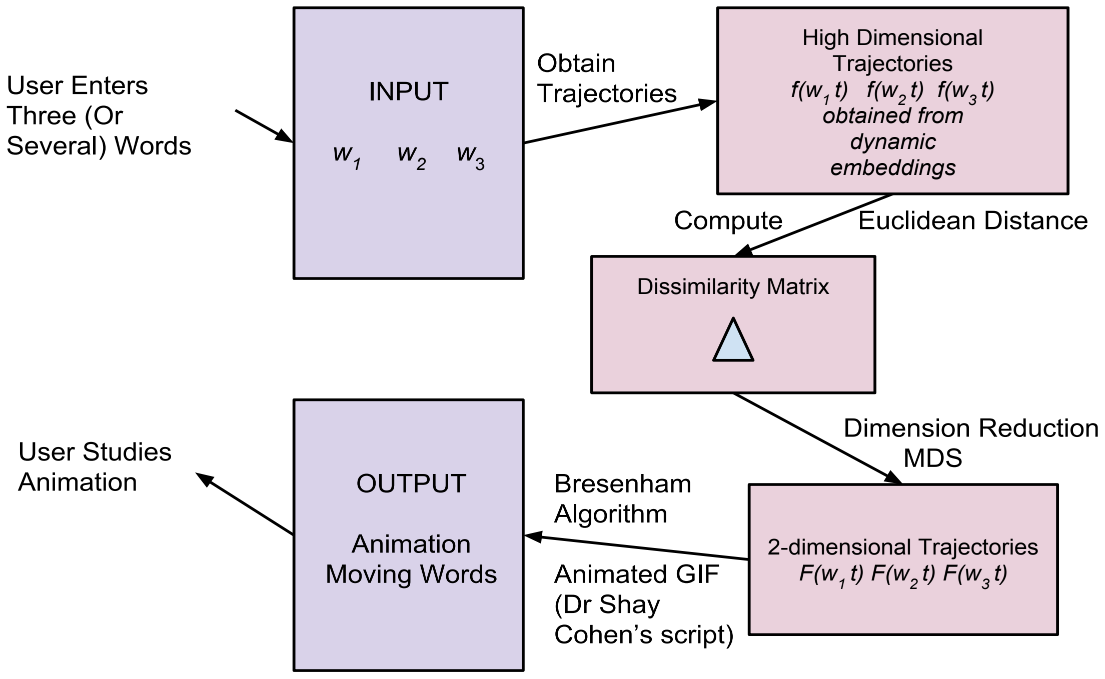

User Enters Three (Or Several) Words
INPUT
w_1 w_2 w_3
Obtain Trajectories
High Dimensional Trajectories
$f(w_1 t)$ $f(w_2 t)$ $f(w_3 t)$
obtained from dynamic embeddings
Compute Euclidean Distance
Dissimilarity Matrix
Dimension Reduction MDS
2-dimensional Trajectories
$F(w_1 t)$ $F(w_2 t)$ $F(w_3 t)$
Bresenham Algorithm
Animated GIF (Dr Shay Cohen's script)
OUTPUT
Animation Moving Words
User Studies Animation

Figure 4.1: *Visualization Tool: An Outline*

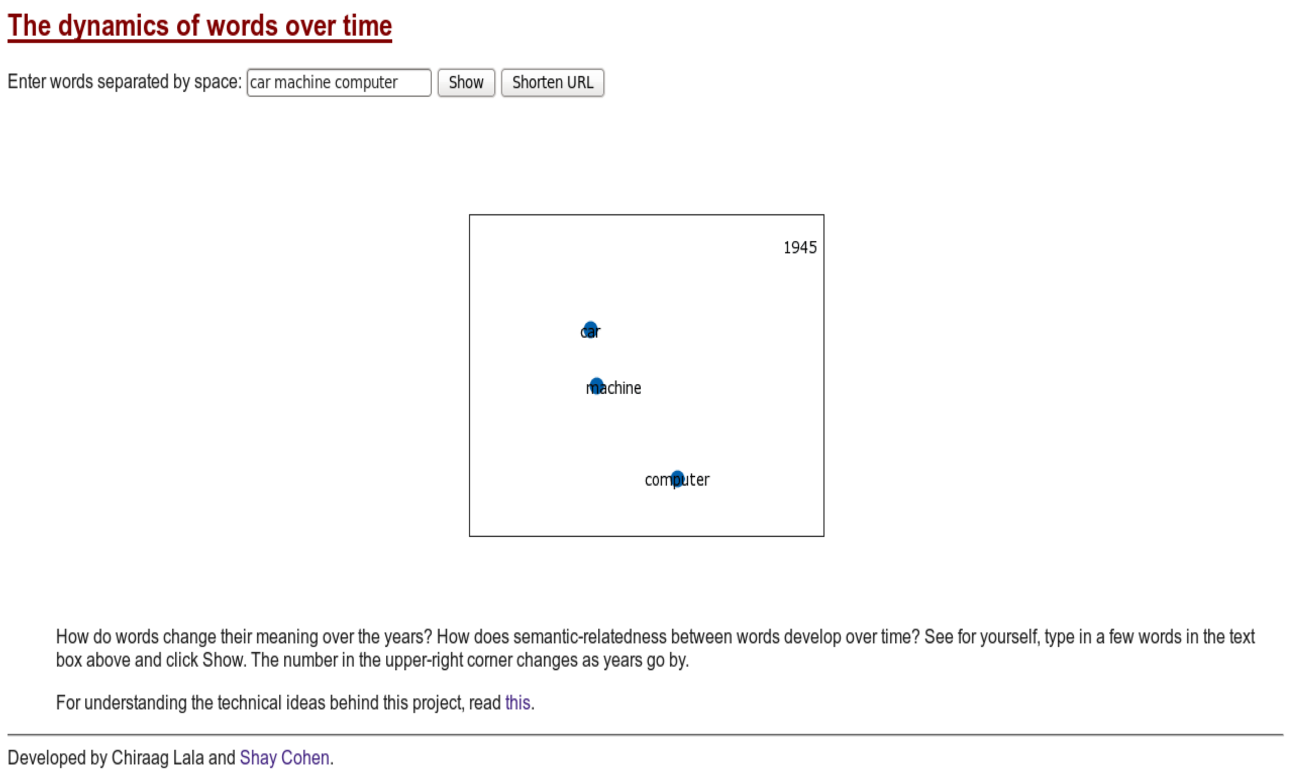
User gets to enter any number of words in a dialogue box on the interface. The dynamic embeddings (see 3) of the entered words are searched to obtain their trajectories. Assume a user enters 3 words w_1, w_2 and w_3. For dynamic embedding $f : V \times T \to \mathbb{R}^d$ where $T = \{1800, 1805, 1810, ..., 2005, 2008\}$ and $d = 25$ (or 50 or 100 or 200 depending on the type of dynamic embeddings chosen), the trajectories

$$f_{\forall t}(w_1, t) = (f(w_1, 1800), f(w_1, 1805), f(w_1, 1810), ..., f(w_1, 2005), f(w_1, 2008))$$

$$f_{\forall t}(w_2, t) = (f(w_2, 1800), f(w_2, 1805), f(w_2, 1810), ..., f(w_2, 2005), f(w_2, 2008))$$

$$f_{\forall t}(w_3, t) = (f(w_3, 1800), f(w_3, 1805), f(w_3, 1810), ..., f(w_3, 2005), f(w_3, 2008))$$

are obtained. The trajectories could have Unk terms. All the vectors in each trajectory (except the Unk terms) are collected as objects. Euclidean distances (see A.2) between each pair of vectors/objects is computed and stored in a dissimilarity matrix (see 4.2.1). Scikit-Learn's manifold.MDS function is applied to the dissimilarity matrix. It generates 2-dimensional vectors for the high dimensional objects preserving the distance relation between the objects. The 2-dimensional vectors obtained repre-


The dynamics of words over time
Enter words separated by space: car machine computer
Show
Shorten URL
1945
car
machine
computer

How do words change their meaning over the years? How does semantic-relatedness between words develop over time? See for yourself, type in a few words in the text box above and click Show. The number in the upper-right corner changes as years go by.

For understanding the technical ideas behind this project, read this.

Developed by Chiraag Lala and Shay Cohen.

Figure 4.2: *Visualization Application: Web Based User Interface*

sent 2-dimensional trajectories for words w_1, w_2 and w_3. Denoting the 2-dimensional trajectories using $F : \{w_1, w_2, w_3\} \times T \rightarrow \mathbb{R}^2$, we get

$$\begin{aligned}
F_{\forall t}(w_1,t) &= (F(w_1,1800), F(w_1,1805), F(w_1,1810), ..., F(w_1,2005), F(w_1,2008)) \\
F_{\forall t}(w_2,t) &= (F(w_2,1800), F(w_2,1805), F(w_2,1810), ..., F(w_2,2005), F(w_2,2008)) \\
F_{\forall t}(w_3,t) &= (F(w_3,1800), F(w_3,1805), F(w_3,1810), ..., F(w_3,2005), F(w_3,2008))
\end{aligned} \tag{4.2}$$

These 2-dimensional trajectories are given as inputs to an animation script. The script produces an animation of the words (entered by user) moving in space over time using Bresenham algorithm (Bresenham (1965)) and animated GIF. The script scales the input 2-dimensional vectors to fit to screen when producing animation. A time indicator

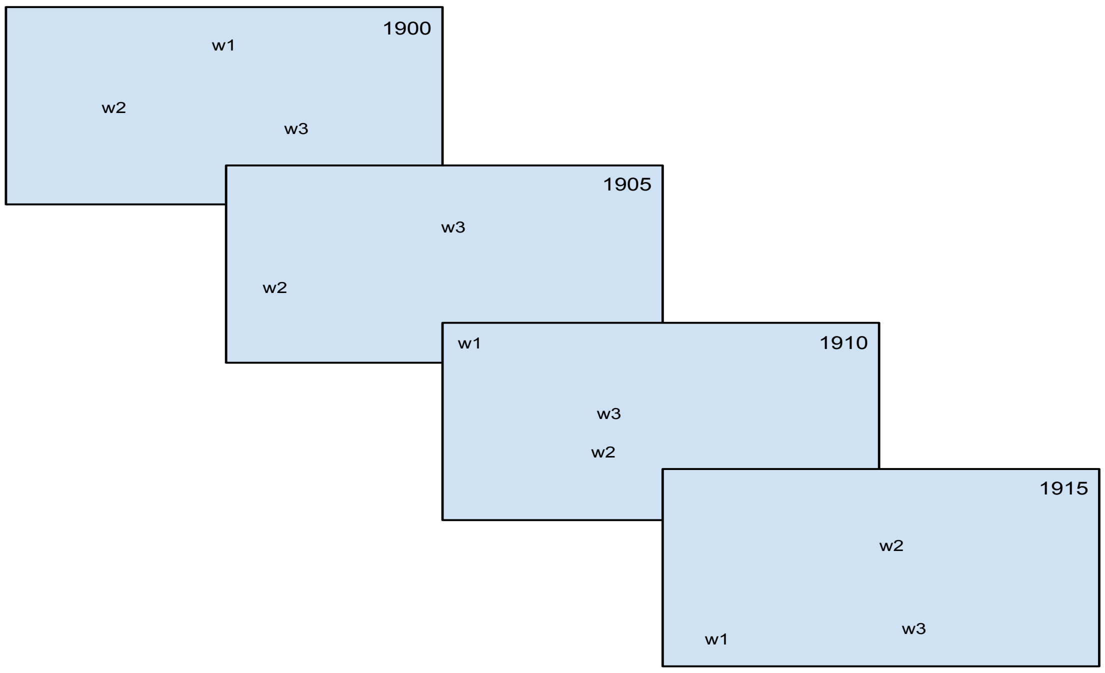


Figure 4.3: *Trajectories As A Stack Of 2-dimensional Static Word Embeddings*

is presented on the top right of the animation screen 4.2 to indicate the changing years in the animation.

NOTE: The 2-dimensional trajectories (see 4.2) are discrete points on a plane. From the time perspective the trajectories can be viewed as a stack of 2-dimensional static word embeddings (see 4.3). Animation we create is based on the assumption that word relations do not change abruptly and hence the real trajectory of the words is continuous. For a word w with 2-dimensional trajectory points $(F(w,1800),F(w,1805),...)$, the above assumption would imply that the true trajectory of the word w is continuous over time passing through the discrete points $(F(w,1800),F(w,1805),...)$. For the purpose of animation this continuity is shown using a straight line. For example, the continuous trajectory of w is made to move along a straight line from $F(w,1800)$ to $F(w,1805)$ at a uniform speed and then along another straight line from $F(w,1805)$ to $F(w,1810)$ and so on (see 4.4). This straight line motion of words is achieved using the Bresenham Algorithm. Many slides of GIF images of the words at varying positions

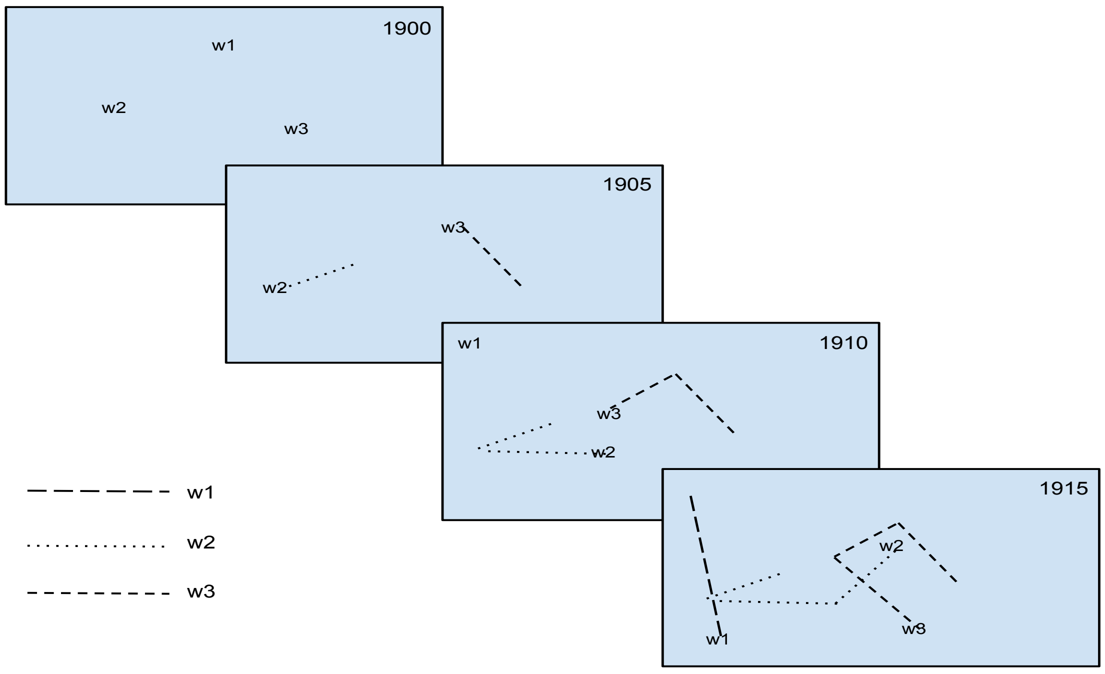


Figure 4.4: *Animation: Words Moving Along A Straight Line*

are generated. These GIF images slides are then made to animate.

Chapter 5

Evaluation

This chapter presents a qualitative and a couple of quantitative analysis of the created dynamic embeddings. Qualitative analysis is achieved using some preliminary observations using the visualization tool. Quantitative analysis involves addressing the following two questions - (1) Do the created dynamic embeddings represent the true semantic similarity and relatedness of words? - (2) Do the created dynamic embeddings represent language change over time? To answer these questions, two evaluation tasks were set in place (one for each question) and experiments were run on those tasks. The set up of the evaluation tasks and our findings have been presented in this chapter.

5.1 Preliminary Qualitative Observations

The qualitative evaluation involves simple visual checking of whether the animations generated using visualization tool reflects known relations of words over time or not? Recall the example of words *car*, *machine* and *computer* mentioned in section 1.2.3. This example shows the changing relations between the three words. In addition to changing relations, we see many animations where there is no significant movement. These correspond to concepts and words that have not changed meaning and semantic-relations over time. Take, for example, the words *red*, *green*, *blue* and *yellow* (colours) and *one*, *two*, *three* and *four*(numbers). These words have not changed their meaning/relations with other words over a long time. The animation of trajectories of these words show that the words *red*, *green*, *blue* and *yellow* stay glued together over time and so do *one*, *two*, *three* and *four*. The two sets - colours and numbers stay separated throughout the animation from year 1800 to 2008 - which is what we expect. Consider a similar type of example of words *wall*, *door*, *window*, *pen*, *paper* and *ink*. The six words are clustered again into two sets over time - *wall*, *door* and *room* in one set and *pen*, *paper* and *ink* in the other. Such interesting animations reflect that the dynamic embeddings created are behaving in the way we expect them to behave, at least for certain cases. It is however difficult to make sense of many animations where words seem to move randomly. Hence qualitative observations are not enough. This made us shift our focus to quantitative evaluations.

5.2 Word Similarity Measures

Do the dynamic embeddings created represent the true semantic similarity and relatedness of words?

To answer this question, first we need to understand what does 'true' semantic similarity and relatedness mean. 'True' here represents the collective human sense, so 'true semantic similarity of a pair of words' represents the extent of similarity/relatedness of the two words as perceived by people in general. For instance, consider the word *money* and two of its synonyms *cash* and *wealth*. In many cases each individual has a perception of one of the synonyms being semantically more similar to the given word. Similarly, in the case of words *money*, *property* and *bank* where words are not synonyms, one of the words could be perceived as conceptually more related to *money* than the other. An averaged perception of all people could indicate the true extent of semantic similarity (or relatedness) between word pairs. In an experiment described in Agirre et al. (2009), the word *cash* was found semantically more similar to *money* than *wealth* and *bank* was found conceptually more related to *money* than *property*. This indicates that there are, perhaps, different extents of semantic similarity (and/or relatedness) of words as perceived by people. An evaluation task, that involves comparing the semantic and relatedness information captured in the dynamic embeddings to the human perceived extents of similarity, had been designed and executed. The details of the evaluation task and results are presented in the following subsections. For results, also see appendix B.2.

5.2.1 Goldstandard Human Ratings

In a survey experiment described in Agirre et al. (2009), a set of 153 word pairs were rated on a scale from 0 to 10 for similarity and relatedness by thirteen subjects. In another survey experiment (Agirre et al., 2009), a set of 200 word pairs (different pairs from previous 153) were rated on the same scale of 0 to 10 for similarity and relatedness by sixteen subjects. The individual ratings by the subjects were averaged for each word pair. This way, each of the 353 (153+200) word pairs received a score representing human perception of their extent of similarity and relatedness. See the first three columns of table in Appendix B.4 - it shows a small excerpt of the 353 word pairs and human scores (third column). The first 153 word pairs with mean human scores from 13 subjects will be called 'Goldstandard 153'. The next 200 word pairs with mean human scores from 16 subects will be called 'Goldstandard 200'. The whole dataset of 353 word pairs with human (mean) scores shall be called 'Goldstandard mixed'

Later, the 353 word pairs with human mean score for each were segregated into two parts - (1) word pairs with mean score more than 5 (250 of these exist) and (2) word pairs with mean score less than or equal to 5 (103 of these exist). Of the 250 word pairs with mean score greater than 5, except for one word pair the rest 249 were further split into two sets. One set included all synonym pairs and antonym pairs (100 of these exist) and other set included the rest 149 pairs (these are word pairs that are conceptually related and not by meaning). Agirre et al. (2009) used WordNet (Miller, 1995) to identify the synonym antonym pairs. The set with 100 synonym and antonym pairs along with 103 word pairs with mean score ≤ 5 comprised a new set for testing similarity in Agirre et al. (2009). We shall call this new set to be 'Goldstandard for similarity'. The set with 149 remaining pairs along with the 103 word pairs with mean score ≤ 5 comprised another new set for testing relatedness in Agirre et al. (2009). This will be called 'Goldstandard for relatedness'.

Simply put, there are five gold-standard data sets:
Goldstandard 153
Goldstandard 200
Goldstandard mixed
Goldstandard for similarity
Goldstandard for relatedness

5.2.2 Evaluation Framework

In static embeddings, we believe, that the distance between word embeddings represents the extent of similarity and/or relatedness of the corresponding words. The untold intuitive rule is - the closer the embeddings, the more similar/related the words (meanings).

The evaluation framework compares this distance between word pairs to the human ratings of the same word pairs. Correlation of distance data with human ratings for word pairs is computed and analysed.

5.2.2.1 Mathematical Formulation

Let there be n word pairs $(u_1, v_1), (u_2, v_2), ..., (u_n, v_n)$ with mean human score of similarity (and/or relatedness) $h_1, h_2, ..., h_n$ respectively. Let $g : V \rightarrow \mathbb{R}^k$ be a static embedding we wish to test (for similarity and relatedness). Compute euclidean distances $d_1, d_2, ..., d_n$ such that $d_i = \|g(u_i) - g(v_i)\|$.. Find the correlation between the two scores $(h_1, h_2, ..., h_n)$ and $(d_1, d_2, ..., d_n)$ using the following procedure. Compute the expected values (means)

$$E_{distance} = (d_1 + d_2 + ... + d_n)/n$$

$$E_{human-score} = (h_1 + h_2 + ... + h_n)/n.$$

Compute the Standard deviations,

$$StdDev_{distance} = \sqrt{((d_1^2 + d_2^2 + ... + d_n^2)/n) - (E_{distance}^2)}$$

$$StdDev_{human-score} = \sqrt{((h_1^2 + h_2^2 + ... + h_n^2)/n) - (E_{human-score}^2)}.$$

Finally, the correlation ρ between distances and human judgement scores is computed using

$$\rho = (\sum_{i=1}^{n} (\frac{d_i - E_{distance}}{StdDev_{distance}} \times \frac{h_i - E_{human-score}}{StdDev_{human-score}}))/(n-1) \tag{5.1}$$

In the ideal hypothetical case, if both the human scores of similarity/relatedness of word pairs and distances between the corresponding static embeddings represent to 'true' relations between the words then a correlation between the two data sets should come out to be $\rho = -1$. Note that it would be -1 because as distance between word embeddings decrease then they represent that the words are more similar and hence the human rating is higher. In practice, for good static embeddings, we look for correlations $\rho \longrightarrow -1$. For results to reflect all the embeddings, it is preferred to have larger values of n (i.e more word pairs scored) with scoring by more larger sample of human subjects.

5.2.2.2 Extending the Framework to Evaluate Dynamic Embeddings

The evaluation framework described in above section is meant for static word embeddings. For the dynamic word embeddings, an extension of the same procedure is required. Two possible extensions are proposed here.

The main idea used in extending the framework is that from time's perspective, dynamic embeddings are a stack of static word embedding, one for each time slice t. Hence, for each time t, we can compare the stack of dynamic embedding at t with the human scores on the same set of word pairs. This way, correlation coefficients ρ_t is obtained for each time slice t in time range T. The $\rho_t's$ could be evaluated further by averaging them or plotting them over time to see the changes.

Another way to evaluate would be to compute an average static embedding from the dynamic embeddings for each word. The averaging is done over time. Let $f : V \times T \longrightarrow \mathbb{R}^k$ be a dynamic embeddings where T has say q number of time slices. For a word w in V, its average static embedding denoted as $a_f(w)$ is computed as,

$$a_f(w) = \frac{\sum_{t \in T} f(w,t)}{q}.$$

The averaged $a_f : V \longrightarrow \mathbb{R}^k$ is a static embedding and can be evaluated using the framework 5.2.2.1. This evaluation would represent the average quality of the dynamic embedding over time and will be represented as $\rho_{averaged}$. Note here, as language is changing and it is very different in long past from the present, we expect word relations in the past very different from their current relations. Hence we expect the dynamic embeddings of the past to bring down the quality of the averaged dynamic embeddings a_f. Also note that $\rho_{averaged}$ and average of ρ_t's ($= \frac{\sum_{t \in T} \rho_t}{q}$) are two different concepts giving rise to two different values.

5.2.2.3 Baselines

To know where do dynamic embeddings stand in terms of quality with respect to established static word embeddings, we need to have baselines computed using the latter. Note that quality of dynamic embeddings at representing similarity and relatedness is measured using (1) $\rho'_t s$, (2) average of $\rho'_t s$ and (3) $\rho_{averaged}$ (see above section 5.2.2.2). It is these measures that are compared with the baselines. The baselines are computed in the same way as in equation 5.1

Since dynamic embeddings f are created using a 'static word embedding' g and N-grams with time markers, it makes sense to have baselines for f obtained from g (g being at the root of f). In the project, four dynamic embeddings were created - two from Collobert and Weston (NLM) embeddings and two from HLBL embeddings. So two correlation coefficients - ρ_{nlm} from NLM embeddings and ρ_{hlbl} from HLBL embeddings could be computed using the same evaluation framework 5.2.2.1. These would form the baselines.

5.2.3 Execution

For word similarity and relatedness evaluation of the four dynamic embeddings created, the following procedure was followed.

(1) Five goldstandards (see section 5.2.1) were obtained from `http://alfonseca.org/eng/research/wordsim353.html` (Agirre et al., 2009).

(2) For each time slice t in $t = \{1800, 1805, ..., 2005, 2008\}$, for each dynamic embedding f that was created and for each word pair (u, v) in the WordSim353 dataset (Agirre et al., 2009), euclidean distances $\|f(u,t) - f(v,t)\|$ were computed. For verification, the cosine angle measures - i.e. $\frac{f(u,t) \bullet f(v,t)}{\|f(u,t)\| \times \|f(v,t)\|}$ were also computed.

(3) For each static embedding g (out of NLM and HLBL embeddings), euclidean distances $\|g(u) - g(v)\|$ (and $\frac{g(u) \bullet g(v)}{\|g(u)\| \times \|g(v)\|}$) were computed.

(4) Using computed distances (and cosines) in above steps 2 and 3, $\rho'_t s$, average of $\rho'_t s$, $\rho_{averaged}$ and baselines ρ_{nlm}, ρ_{hlbl}, and same correlation coefficients using cosine angle measures were computed for each of the four dynamic embeddings

Table 5.1: Goldstandard 153: Correlation Scores

	$\rho_{t-average}$	$\rho_{averaged}$	Baseline ρ_{nlm}	Baseline ρ_{hlbl}
$f_{average-nlm}$	-0.1621063594	-0.165640657	-0.2886195278	
$f_{concat-nlm}$	-0.176488843	-0.1870198196	-0.2886195278	
$f_{average-hlbl}$	-0.2040145741	-0.2042482948		-0.3542788386
$f_{concat-hlbl}$	-0.2161318757	-0.2271269078		-0.3542788386

and each of the goldstandards of step 1. Equation 5.1 was used for the purpose of computing correlations.

(5) Results obtained in step 4 were analysed.

5.2.4 Results

Average of $\rho_t's$ (To be denoted as $\rho_{t-average}$), $\rho_{averaged}$ and baselines ρ_{nlm} and ρ_{hlbl} were computed for each of the five goldstandard dataset 5.2.1. For instance, we get the following table for 'goldstandard 153'.

Similar such tables for other goldstandard datasets are presented in appendix B.2.2. In addition, correlations of individual years ρ_ts were plotted against time t as a histogram to identify patterns (if any). For 'goldstandard 153', see the chart 5.1.

5.2.5 Analysis

From tables B.4 and in appendix B.2.2, values of different types of correlations ρ are negative in almost all cases. However, these values are still quite far off from -1. This reflects that the dynamic embeddings do appear to follow the intuitive rule - the closer the embeddings, the more similar the meanings/relatedness/similarity - however, they are not very accurate at representing the true word similarity and relatedness (assuming the goldstandard are true representations of similarity and relatedness.

The correlations obtained for Goldstandard 153 (see table B.4) are very different to those obtained for Goldstandard 200 (see table B.2 in appendix). In fact, for Goldstandard 200 the dynamic embeddings seem to be almost unrelated with the human scores

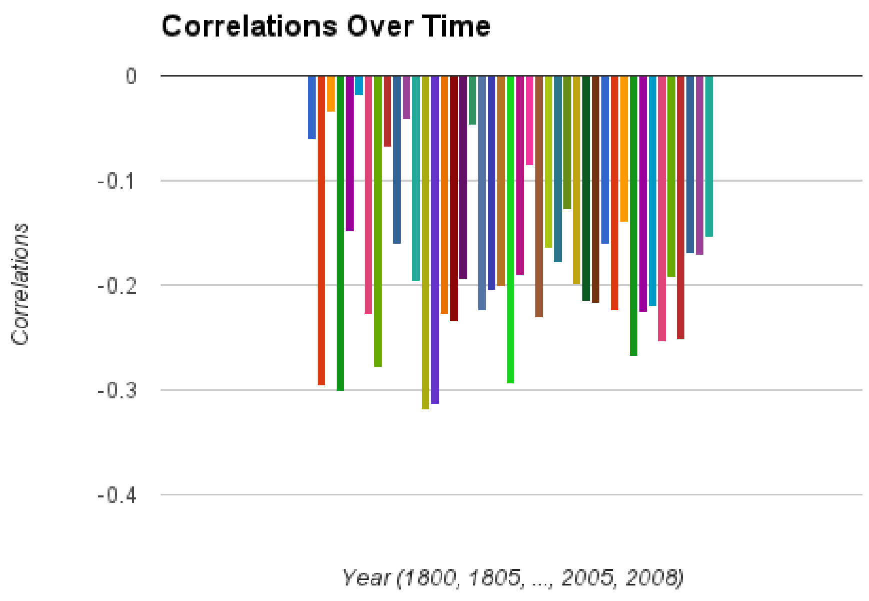


Figure 5.1: *Goldstandard 153: Dynamic Embedding $f_{average-nlm}$ - Correlation of Each Stack (Static Word Embedding within $f_{average-nlm}$) Over Time - i.e. ρ_t vs t - Time Ascends from Left to Right*

with $\rho \approx 0$, while for Goldstandard 153 $\rho \approx 0.19$. The two goldstandard datasets were created by two separate groups of subjects. It indicates, perhaps, the WordSim353 datasets are not very reliable (score by same dynamic embedding varying drastically from one group of subjects to another). More reliable measures of human scoring of word similarity and relatedness could help.

Baselines from static word embedding are always found to be closer to -1 than correlations from dynamic word embeddings (see tables B.4 and B.2.2). Created dynamic embeddings seem to have lost quality compared to the static embedding from which it was created. This calls for better ways of developing dynamic embeddings.

HLBL embeddings have better correlations ρ than NLM embeddings and the other dynamic embeddings. The Dynamic embeddings created using HLBL embeddings have better correlations (closer to -1) compared to those created using NLM embeddings. This gives hints that, dynamic embeddings created from better quality static word embeddings are likely to be better. Future developments of dynamic embeddings should focus on developing them from better quality static embeddings.

Dynamic embeddings based on the concatenation vector operation have better correlation (closer to -1) than those created using average vector operation (notice $f_{concat-g}$ has better correlation scores than $f_{average-g}$, for whatever static embedding g might be (see tables B.4 and B.2.2). Concatenation operation better than average operation at preserving similarity and relatedness of words.

Correlations over time (see plot 5.1) do not show the trend we expected. The expected trend would be that correlations improve (get closer to -1) as time increases and finally nearing the present time it is the best representing current word relations as perceived by people in present era. However the plot 5.1 seems to vary a lot. A faint observation though is that for years 1800 to 1900 the correlation scores ρ_t vary a lot (from -0.3 for year 1860 to almost 0 for year 1825), while for years closer to 2000, the correlation scores become consistent and remain more or less constant (varying little between -1.2 to -2.6). This is, perhaps due to sparseness of Google five-gram data for earlier years. The future developments of dynamic embeddings must bring such variations for older times under control by training on a much larger corpus of N-grams.

Simply put, the dynamic embeddings created are certainly not the best but they do seem to be on the right track with scope for further improvements.

5.3 Diachronic Text Evaluation

Do dynamic embeddings created capture the diachronic phenomena (language change)?

In an attempt to answer this question, a Machine Learning task called Diachronic Text Evaluation (DTE) was explored. DTE task is: given a document of text, predict the time when it was likely written based on words, terms and style of language used. The idea is that if the created dynamic embeddings do capture language change over time then it should, in theory, be possible to use the dynamic embeddings to detect when a document of text was written. The current attempts to address this task, discussed in Popescu and Strapparava (2013), Popescu and Strapparava (2014) and Mihalcea and Nastase (2012) are based on computing word occurrence in a particular epoch. For instance, in one approach to DTE a probability distribution over various time epochs is computed for each word. This distribution assigns to a word-time pair (w,t), the probability of using the word w in the time epoch t. For the entire document, a new probability distribution over time epochs is computed using the word probability distributions of the words in the document. This probability distribution of the document is then used to make a prediction of the year when a text was written. Diachronic Text Evaluation is task 7 of the upcoming SemEval 2015.

Current works on Diachronic Text evaluation are still in an infant stage. Further improvements to current DTE systems are being explore based on syntax, semantics and other features as described in Popescu and Strapparava (2014). This thesis proposes the use of dynamic word embeddings as an extra word feature in the current systems for DTE to improve performance. Our proposal is based on the belief that dynamic word embeddings capture the 'true' semantic relations between words over time ('True' refers to the actual semantic relations of the time period). These relations stored in the dynamic embeddings could be exploited for better epoch detections in the DTE task.

For the purpose of this project, a novel and simple machine learning (ML) system for DTE task was developed that uses dynamic embeddings. For comparison, the same machine learning system that uses static word embedding (instead of dynamic) was also developed. The two systems were trained and tested on the same dataset. The performance of the two systems was then compared and analysed.

5.3.1 A Simple ML System for DTE Task

A supervised ML system, based on linear regression, for DTE task was developed. It required data for training the ML system.

5.3.1.1 Trial Data

Diachronic Text Evaluation, being task 7 of SemEval 2015, has a website `http://alt.qcri.org/semeval2015/task7/` dedicated to task. The website provides information, resources and data to develop systems for DTE. At the time this thesis was printed, a complete training data was still being developed and only a trial data was available on the DTE task website. It was decided to use the Trial data for both training and testing our supervised ML system. The trial data comprises of 84 short documents with time epoch markers like in the example in appendix B.1. Three time epoch markers of varying size are provided. In the example B.1, the time markers are “1946-1952”, “1943-1955” and “1939-1959”. We consider the average of the smallest time range to be a time stamp of the document. In this case, the time stamp of the document “Sears Famous ... soapthat’s all.” is 1949. The trial data set of 84 examples is denoted as a list of document and time-stamp pair

$$Trial = ((d_1,t_1),(d_2,t_2),...,(d_{84},t_{84}))$$

where d_i are the documents and t_i are corresponding time stamps.

5.3.1.2 The ML System

Let $((d_1,t_1),(d_2,t_2),...,(d_n,t_n))$ be n number of document-timestamp pairs used for training ML system. Assume there exists a mapping $v:\{d_1,d_2,...,d_n\}\to\mathbb{R}^k$ that maps the n documents to vectors in some k-dimensional Euclidean space. Call $v(d_i)$ as the vector of document d_i. Denote each component of $v(d_i)$ as $v(d_i)_j\in\mathbb{R}$. In other words the vector

$$v(d_i) = (v(d_i)_1, v(d_i)_2, ..., v(d_i)_k)^{Transpose}.$$

The simple ML system developed in the project is based on linear regression.

Consider the linear equation,

$$y = (c_0) + (c_1 \times x_1) + (c_2 \times x_2) + ... + (c_k \times x_k) \tag{5.2}$$

The ML system learns the values of the coefficient vector $C = (c_0, c_1, c_2, ..., c_k)$ by fitting the vectors of documents from the training data to $(x_1, x_2, ..., x_k)$ and timestamps of the training data to y in equation 5.2. Mathematically, if

$$V(d_i) = (1, v(d_i)) = (1, v(d_i)_1, v(d_i)_2, ..., v(d_i)_k)$$

then the ML system learns the linear function 5.2, by minimizing the following

$$\min_{c_0, c_1, ..., c_k} (t_1 - (C \bullet V(d_1)))^2 + (t_2 - (C \bullet V(d_2))^2 + ... + (t_k - (C \bullet V(d_k))^2 \tag{5.3}$$

where $C \bullet V(d_i) = c_0 + (c_1 \times v(d_i)_1) + (c_2 \times v(d_i)_2) + ... + (c_k \times v(d_i)_k)$
The ML system developed in the project goes a step further, it performs a ridge regression, i.e it learns the coefficients of the following equation from the training data

$$y = (c_0) + (c_1 \times x_1) + (c_2 \times x_2) + ... + (c_k \times x_k) + (\alpha \times (c_0^2 + c_1^2 + ... + c_k^2)) \tag{5.4}$$

NOTE: In the project, $\alpha = 0.5$ was chosen after a few cross validation experiments with trial data (see Machine learning: a probabilistic perspective Murphy (2012) for more details on cross validation for selection of α parameter in ridge regression).

5.3.1.3 Document Vector Using Dynamic Embedding

The ML system developed was intended to use dynamic embeddings created for DTE task. A simple and novel way of incorporating dynamic embeddings is to use it to compute the vector of the documents. This is achieved by the following process.

Let $f : V \times T \to \mathbb{R}^p$ be dynamic embedding. Let document d consist the sequence of words $(w_1, w_2, ..., w_{l(d)})$ where l(d) is the length of document d. For each time slice t, compute vector of the document d at time t as

$$v_t(d) = \frac{f(w_1, t) + f(w_2, t) + ... + f(w_{l(d)}, t)}{l(d)} \tag{5.5}$$

NOTE: If a particular word does not have a dynamic embedding at time t then that word is simply ignored from the document in computing the document vector at time t. Finally we define the document vector as

$$v(d) = v_{1800}(d) \odot v_{1805}(d) \odot ... \odot v_{2008}(d) \tag{5.6}$$

where $\odot$ represents concatenation.

This way, for the 84 examples of documents $(d_1, d_2, ..., d_{84})$, the document vectors $(v(d_1), v(d_2), ..., v(d_{84}))$ are computed.

5.3.1.4 Baseline Document Vectors

If dynamic embedding f is created from static word embedding g, then it makes sense to compare f to g in the DTE task. This would help answer the question : does dynamic embedding f (instead of static word embedding g) improve the performance of ML system for DTE. Hence baseline document vectors using static word embeddings g are computed as follows.

If document d consists the sequence of words $(w_1, w_2, ..., w_{l(d)})$ where l(d) is the length of document d, then baseline document vector b is defined as

$$b(d) = \frac{b(w_1) + b(w_2) + ... + b(w_{l(d)})}{l(d)} \tag{5.7}$$

NOTE: Again, if a particular word w does not have a static embedding $(g(w) = Unk)$ then simply exclude the word from the document.

This way, for the 84 examples of documents $(d_1, d_2, ..., d_{84})$ in trial data, the document vectors $(b(d_1), b(d_2), ..., b(d_{84}))$ are computed.

Another baseline b_2 of document vectors is obtained from equation 5.6. In that equation, replace concatenation with vector addition. The vector addition will take the time factor away from the dynamic embedding for document vector. Mathematically, baseline 2 document vector b_2 is defined as

$$b_2(d) = v_{1800}(d) + v_{1805}(d) + ... + v_{2008}(d) \tag{5.8}$$

where $v_t(d)$ is as defined in equation 5.5.

This way, for the 84 examples of documents $(d_1, d_2, ..., d_{84})$ in trial data, the document vectors $(b_2(d_1), b_2(d_2), ..., b_2(d_{84}))$ are computed.

5.3.1.5 One Leave Out Cross Validation

As training data was not available (at the time this thesis went for printing), the 84 document–time-stamp pairs in trial data were used to do both - training and testing of the ML system. This was achieved using ‘one leave out cross validation’.

Let $D = ((d_1,t_1),(d_2,t_2),...,(d_{84},t_{84}))$. Define

$$D_i = D \setminus \{(d_i,t_i)\}$$

where the D_i includes every example other than (d_i,t_i)

One leave out cross validation is to train the ML system on D_i and test it on (d_i,t_i), iteratively for i in list $(1,2,...,84)$. For instance, when ML system trains on D_i, it learns the coefficients and parameters of the equation 5.4 from all the data set other than (d_i,t_i). Call this function learnt as eq_i. When eq_i is tested on d_i, then it predicts a time of when the document d_i might have been written - call the predicted time T_i. The difference between predicted time T_i and actual time stamp t_i will be called the error e_i. Let E denote the mean square error which is,

$$E = \frac{e_1^2 + e_2^2 + ... + e_{84}^2}{84}.$$

The smaller is the mean square error E, the better is the ML system at the task of DTE.

5.3.2 Execution

Three Machine Learning systems, all three exactly similar differing only in how the vectors representing the documents are constructed, were developed. An in-built ridge regression function in scikit - learn module Pedregosa et al. (2011) is used to learn the regression from data. Let the three ML systems be called M_v, M_b and M_{b_2}. M_v system uses document vector as defined in section 5.3.1.3 using dynamic embeddings. M_b system uses baseline b to compute the document vectors (see 5.7) using static word embeddings and M_{b_2} uses baseline b_2 as defined in equation 5.8. Let E_v, E_b and E_{b_2} be mean square errors of ML systems M_v, M_b and M_{b_2} respectively. For each of the four dynamic embeddings created, the three systems are made to compute the mean square errors through one-leave-out cross validation on the trial dataset (see 5.3.1.5). Twelve mean square error values of three ML systems and four dynamic embeddings are computed and results are analysed.

Table 5.2: Mean Square Errors after One Leave Out Cross Validation

	E_v	Baseline E_b	Baseline E_{b_2}
$f_{average-nlm}$	3758.264319	3315.528777	5180.170875
$f_{concat-nlm}$	3321.795009	3315.528777	3619.567825
$f_{average-hlbl}$	3839.25929	4557.360831	5300.940471
$f_{concat-hlbl}$	3301.842722	4557.360831	3946.231923

5.3.3 Results

Three ML systems made to run on four different dynamic embeddings generate twelve mean square errors, which are tabulated below.

5.3.4 Analysis

Mean square errors tabulated in section 5.2 are used for analysis. An important point to add here is that the ML systems were trained and tested on a very small trial dataset so results cannot be considered very significant. However these could be considered to form a preliminary study, of dynamic embeddings at DTE task, before training data is available on SemEval task 7 website `http://alt.qcri.org/semeval2015/task7/`

The NLM embeddings with lower mean square errors seem better at DTE task than HLBL embeddings. In analysis of Word Similarity (see section 5.2.5) HLBL embeddings were inferred to be better at representing similarity and relatedness. So it indicates that static embeddings could be good at some tasks and not good at other. Perhaps, the way the two embeddings - NLM and HLBL - were developed might hold key to answer why NLM is better at DTE while HLBL is better at similarity and relatedness measures.

ML systems using dynamic embeddings $f_{concat-g}$ are always better than $f_{average-g}$ where g could be any - NLM or HLBL embeddings. This means dynamic embeddings developed using concatenation seem better at DTE task. This implies that, perhaps, dynamic embeddings developed using the concatenation vector operation capture language change much better than vector addition.

ML system M_b is transformed into M_v if instead of HLBL embeddings, dynamic embeddings created using HLBL $f_{\star-hlbl}$ are used. This replacement improves performance of the system drastically and hence supports the view that dynamic embeddings could prove useful in DTE task. Same cannot be said for NLM though. Hence more studies are required.

E_v is always less than corresponding baseline E_{b_2}. Both E_v and E_{b_2} were computed from the dynamic embeddings with one difference - E_v is computed for M_v that concatenates the document vectors over time (see equation 5.6) while E_{b_2} is for M_{b_2} that performs vector addition of the documents vectors over time (see equation 5.8). Concatenation ensures that differences over time are not lost, in the sense that each component of the document vector $v(d)$ refers to some time slice. On the other hand, none of the components of $b_2(d)$ can be identified to any time slice due to vector addition, an indicator to loss of differences over time. E_v better than E_{b_2} in all cases indicates that differences between $v_{1800}(d), v_{1805}(d), ..., v_{2008}(d)$ could be useful at DTE task. It justifies our belief that dynamic embeddings do seem to capture language change. Hence future developments should try and preserve the differences over time.

Chapter 6

Conclusions

6.1 Achievements

The project began with the aim of creating word vector space embeddings over time (dynamic embeddings). In attempts to create these, a general recipe of generating new dynamic embeddings from (1) static word embeddings and (2) n-gram data with time markers was mathematically formulated. Using Google n-gram corpus, NLM embeddings and HLBL embeddings creation of four sets of dynamic embeddings of varying dimensions was demonstrated.

Dynamic embeddings can be viewed from two perspectives - (1) From word perspective as trajectories of words moving in space with changing times and (2) From time perspective as a stack of static word embeddings (one for each time slice). The word perspective inspired us to create an application of the dynamic embedding: the visualization tool (see chapter 4). This tool helps visualize semantic change and other diachronic phenomena. This tool posed a challenge of reducing the dimension of trajectories in high dimensional space to 2-dimensional space. Dimension reduction was achieved using MDS. Animation was achieved using Bresenham algorithm and animated GIF. A web application using the developed visualization tool was also created - `http://kinloch.inf.ed.ac.uk/words/index.php`.

A supervised Machine learning system that uses dynamic embedding for the Diachronic Text Evaluation task was developed. It demonstrates a potential of using dynamic embeddings in improving systems that detect the period when a text was written (see chapter 5).

6.2 Quality of Dynamic Embeddings

Do dynamic embeddings represent true semantic similarity and relatedness of words?

Using Word Similarity evaluation framework (see chapter 5), the dynamic embeddings were identified to follow the 'intuitive rule': the closer the embeddings geometrically, the more related and similar the words are. This is because, correlation between distances of embeddings and similarity measures of corresponding word pairs was almost always found negative (see appendix B.2). However, because these correlation values were close to 0 in spite of being negative, it was concluded that the distance metric was not very accurate at representing the 'true' word similarity and relatedness. An average $\rho_{average} \approx -0.102$ of the dynamic embeddings (averaged over all the four dynamic embeddings) was computed. This was compared to the average baseline $\rho_{baseline} \approx -0.304$. The result shows that there is loss in quality when dynamic embedding is generated. Compared to baseline static embeddings, our dynamic embeddings had $\rho_{average}$ values of nearly one-third of the baseline $\rho_{baselines}$ values as also seen by the correlations above. Hence it was concluded that dynamic embeddings had lesser quality than the quality of the static embedding from which they are being created and the loss was nearly 2/3rd of the ρ score. Quality here is referred in terms of correlation coefficients ρ as computed in equation 5.1. In addition, it was noted that to create good quality dynamic embeddings, start with a better quality static embeddings. This is because dynamic embeddings from good quality static embeddings were found to have more negative ρ values (see tables in B.2. At the same time it was noted that the human ratings of word similarity itself varies from one group of subjects to another (see analysis 5.2.5). Hence nothing too conclusive could be found to answer the above question except for the 'intuitive rule'.

Do dynamic embeddings capture diachronic phenomena (language change)?

Use of dynamic embeddings instead of HLBL embeddings were shown to improve the performance of ML system for Diachronic Text Evaluation (DTE) (See the mean value scores in table in section 5.2). In addition taking the average of the mean square errors, we get average $E_v \approx 3524$ and average of $E_b \approx 3936$ and average of $E_{b_2} \approx 4517$. This surely is a good sign, in the sense that even in after averaging over four dynamic embeddings created, there is a near $10 percent$ lesser mean square errors (between E_v

and E_B). ML system using dynamic embeddings were also found to perform better than baseline 2 system at all times. It is the way dynamic embeddings are used that contributes to how the ML system gets improved. As described in analysis of DTE task and results above, our belief that dynamic embeddings do capture diachronic phenomena has been strengthened.

6.3 Other Conclusions

HLBL embeddings was found better at representing similarity and relatedness of words while NLM embeddings were found better in DTE task of detecting time when a document is written. This implies that embeddings vary in quality from task to task. Concatenation based dynamic embeddings were found better than average based ones. Concatenation operation used in the DTE task as well was found to improve performance of the systems. It could thus be concluded that concatenation is a good operation (compared to averaging) at preserving both the semantic similarity information and language change information when used in DTE task.

6.4 Future Avenues

6.4.1 Improvement of Dynamic Embeddings

The general recipe of creating dynamic embeddings can be explored further to make new kinds of embeddings. Dynamic embeddings created have one shortcoming which is that it cannot capture words at start or end of a sentence, because of our requirement to have context on both sides of target word. This could also be explored. Including other syntactic and semantic cues, especially Part Of Speech tags to create dynamic embeddings can be looked at. Training on the entire Google Books N-gram corpus (including n-grams of other sizes) can be explored, functional words could be dealt with separately. Combining different types of static embeddings and creating new kinds of dynamic embeddings like the one in which time range is continuous can be thought through. Words from other languages like French, German, etcetera could also be incorporated.

6.4.2 Improvements of Visualization Tool

Distance metric was found to be not a very accurate measure of semantic similarity and relatedness of words. So new kinds of distance metrics could be explored for better representation of word similarity and relatedness. Visualization and animation of word trajectories in 3-dimensional space, a bit challenging but not impossible, would definitely be next step forward. Other improvements to user experience can also be explored.

6.4.3 Diachronic Text Evaluation

Dynamic embeddings do show signs of great applicability, specially for DTE task, due to its temporal features and indications of capturing the diachronic phenomena in the study presented in this thesis. A novel Machine Learning system for DTE that includes dynamic embeddings as word features could be developed. Incorporating dynamic embeddings of words as extra word features to existing Machine Learning systems for DTE (to improve performance of systems) could be looked at. One promising future prospect would be to build on the ideas of this thesis to participate and contribute in SemEval 2015 task 7 of Diachronic Text Evaluation.

Bibliography

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics. (on pp. 13, 14, 33, 34, and 37)

Anton, H. and Rorres, C. (2010). *Elementary Linear Algebra: Applications Version*. John Wiley & Sons. (on p. 56)

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. (2006). Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer. (on p. 19)

Blei, D. M. and Lafferty, J. D. (2006). Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM. (on pp. 10 and 11)

Borg, I. and Groenen, P. (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer. (on pp. 7, 12, and 26)

Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30. (on pp. 12 and 29)

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479. (on p. 11)

Chen, Y., Perozzi, B., Al-Rfou', R., and Skiena, S. (2013). The expressive power of word embeddings. *CoRR*, abs/1301.3226. (on pp. 2, 12, and 13)

Chomsky, N. (1965). *Aspects of the Theory of Syntax.* Massachusetts Institute of Technology. M.I.T. Press. (on p. 6)

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537. (on p. 2)

Dhillon, P., Foster, D. P., and Ungar, L. H. (2011). Multi-view learning of word embeddings via cca. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 199–207. Curran Associates, Inc. (on p. 2)

Firth, J. (1957). *Papers in linguistics, 1934-1951.* Oxford University Press. (on p. 6)

Harris, Z. S. (1954). Distributional structure. *Word.* (on p. 11)

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics. (on p. 2)

Kroeger, P. R. (2005). *Analyzing grammar: An introduction.* Cambridge University Press. (on p. 57)

Landauer, T. K. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211. (on pp. 6 and 11)

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., et al. (2011). Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182. (on pp. 6, 11, 12, and 14)

Mihalcea, R. and Nastase, V. (2012). Word epoch disambiguation: Finding how words change over time. In *ACL (2)*, pages 259–263. (on p. 41)

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41. (on p. 34)

Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088. (on p. 20)

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. (on pp. 12, 13, and 43)

Passos, A., Kumar, V., and McCallum, A. (2014). Lexicon infused phrase embeddings for named entity resolution. *CoRR*, abs/1404.5367. (on p. 2)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. (on pp. 12, 13, and 45)

Popescu, O. and Strapparava, C. (2013). Behind the times: Detecting epoch changes using large corpora. (on pp. 5, 10, 11, and 41)

Popescu, O. and Strapparava, C. (2014). Time corpora: Epochs, opinions and changes. *Knowledge-Based Systems*. (on pp. 11, 12, 13, 14, and 41)

Roy, D. K. and Pentland, A. P. (2002). Learning words from sights and sounds: A computational model. *Cognitive science*, 26(1):113–146. (on p. 6)

Sahlgren, M. (2006). The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. (on p. 11)

Shaw, B. and Jebara, T. (2009). Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 937–944. ACM. (on p. 12)

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics. (on pp. 2, 11, 14, 19, and 20)

Turney, P. D., Pantel, P., et al. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188. (on p. 11)

Uszkoreit, J. and Brants, T. (2008). Distributed word clustering for large scale class-based language modeling in machine translation. Citeseer. (on p. 11)

Wall, M. E., Rechtsteiner, A., and Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer. (on p. 26)

Wang, C., Blei, D., and Heckerman, D. (2012). Continuous time dynamic topic models. *arXiv preprint arXiv:1206.3298*. (on pp. 11 and 12)

Wittgenstein, L. (1973). *Philosophical investigations:*. Philosophical Investigations: The English Text of the Third Edition. Macmillan. (on p. 6)

Appendix A

Definitions

A.1 Vectors And Euclidean Vector Spaces

In mathematics, $\mathbb{R}$ is the set of all real numbers geometrically represented using an infinite line. The n-dimensional Euclidean vector space $\mathbb{R}^n$, for a positive integer n, is the set $\{(r_1, r_2, ..., r_n) | r_1, r_2, ..., r_n \in \mathbb{R}\}$. $\mathbb{R}^2$ and $\mathbb{R}^3$ represents our intuition of a geometric plane and space respectively. Any element $(a_1, a_2, ..., a_n)$ in $\mathbb{R}^n$ is called a vector in the n-dimensional Euclidean vector space. Intuitively, the vectors represent the points in the vector space.

More about vectors, vector spaces and operations on such spaces can be found in conventional linear algebra text books. Chapter 3 in 'Elementary Linear Algebra: Applications Version' (Anton and Rorres, 2010) presents a nice, thorough and sufficient introduction to basic concepts of vectors and Euclidean vector spaces.

A.2 Euclidean Distance

Let $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n)$ be two vectors in Euclidean space $\mathbb{R}^n$ (see A.1) for some positive integer n. Then Euclidean distance between a and b, denoted as $||a - b||$ is

$$||a-b|| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2}.$$

Euclidean distance represents the common intuition of distance in physical world.

A.3 Totally Ordered Set

In mathematics, a set S is called a **totally ordered set** if it has a relation called total order, denoted as $\leq$, such that the following are satisfied,

- $a \leq a$, for all elements a in S.
- $a \leq b$ and $b \leq a$ implies $a = b$.
- $a \leq b$ and $b \leq c$ implies $a \leq c$.
- For any two elements a and b in S, either $a \leq b$ or $b \leq a$.

Example: Time periods are totally ordered sets where the relation $a \leq b$ means that either time instance a comes before b or the two time instances are the same.

A.4 Word Co-occurrence

In linguistics, word co-occurence can either mean concurrence / coincidence of words (terms) or, in a more specific sense, the frequency of occurrence of two terms from a text corpus alongside each other in a certain order (Kroeger, 2005).

A.5 N-grams

In computational linguistics, a contiguous sequence of n items from a given text is called an N-gram. The items can be letters or words or tokens. An N-gram of length 1 is called a“unigram”; length 2 is a “bigram”; length 3 is a “trigram”. Larger sizes are referred to by the value of n like “four-gram”, “five-gram”, and so on. This project deals with five-grams of words and tokens.

Appendix B

Datasets

B.1 Trial Data for Diachronic Text Evaluation

Example of a document in trial data for DTE task is given below:

```
<text id="365cp1113232119">
<textF no="1904-1910" no="1911-1917" no="1918-1924" no="1925-1931" no="1932-193
no="1939-1945" yes="1946-1952" no="1953-1959" no="1960-1966" no="1967-1973"
no="1974-1980" no="1981-1987" no="1988-1994" no="1995-2001" no="2002-2008"
no="2009-2015">
<textM no="1696-1708" no="1709-1721" no="1722-1734" no="1735-1747" no="1748-176
no="1761-1773" no="1774-1786" no="1787-1799" no="1800-1812" no="1813-1825"
no="1826-1838" no="1839-1851" no="1852-1864" no="1865-1877" no="1878-1890"
no="1891-1903" no="1904-1916" no="1917-1929" no="1930-1942" yes="1943-1955"
no="1956-1968" no="1969-1981" no="1982-1994" no="1995-2007" no="2008-2020">
<textC no="1708-1728" no="1729-1749" no="1750-1770" no="1771-1791" no="1792-181
no="1813-1833" no="1834-1854" no="1855-1875" no="1876-1896" no="1897-1917"
no="1918-1938" yes="1939-1959" no="1960-1980" no="1981-2001" no="2002-2022">
Sears Famous Kenmore Completely Automatic Washer.It’s like magicfood-It,
set it and forget it.  Washes all kinds of clothes amazingly clean, automatical
Rinses ail clothes 7 times, automatically.  Spins all clothes damp dry,
automatically.  Two little dials do all the workno watching, no waiting
just load the clothes, set the dial,add soapthat’s all.
</text>
```

Table B.1: Excerpt of Word-Pair Similarity and Relatedness

WORD 1	WORD 2	HUMAN (MEAN)		YEARS 1800	1805 ...
money	dollar	8.42		Unk	Unk
money	cash	9.08		0.4003086431	0.3403557557
money	wealth	8.27		0.2614127117	0.2039819144
money	property	7.57		0.3134598117	0.2542957324
money	bank	8.5		0.7593744204	0.3769771715
space	world	6.53		0.3914747233	0.4322863634
preservation	world	6.19		0.6500469855	0.4416165133
direction	combination	2.25		0.5949834979	0.5269165972

Table B.2: Goldstandard 200: Correlation Scores

	$\rho_{t-average}$	$\rho_{averaged}$	Baseline ρ_{nlm}	Baseline ρ_{hlbl}
$f_{average-nlm}$	-0.03747611492	-0.03626861278	-0.2722516676	
$f_{concat-nlm}$	-0.03444177037	-0.02855099298	-0.2722516676	
$f_{average-hlbl}$	-0.02567159053	0.03181987821		-0.2961077762
$f_{concat-hlbl}$	-0.02365395643	0.02758360941		-0.2961077762

B.2 Results For Similarity And Relatedness

B.2.1 Table B.1

Refer to table B.1. Mean of human ratings of similarity in third column. Euclidean distance between word pairs in dynamic embeddings for various years in fifth column onwards.

Table B.3: Goldstandard Mixed: Correlation Scores

	$\rho_{t-average}$	$\rho_{averaged}$	Baseline ρ_{nlm}	Baseline ρ_{hlbl}
$f_{average-nlm}$	-0.08748873505	-0.08715957471	-0.290286389	
$f_{concat-nlm}$	-0.09636033926	-0.09879840571	-0.290286389	
$f_{average-hlbl}$	-0.1040119688	-0.07856670688		-0.321061014
$f_{concat-hlbl}$	-0.110360536	-0.09512138281		-0.321061014

Table B.4: Goldstandard for Similarity: Correlation Scores

	$\rho_{t-average}$	$\rho_{averaged}$	Baseline ρ_{nlm}	Baseline ρ_{hlbl}
$f_{average-nlm}$	-0.1307137174	-0.1234126906	-0.4059548027	
$f_{concat-nlm}$	-0.1332167122	-0.1209145616	-0.4059548027	
$f_{average-hlbl}$	-0.1669964071	-0.1386733081		-0.3922579879
$f_{concat-hlbl}$	-0.1679772554	-0.1486743488		-0.3922579879

B.2.2 Correlation Tables

B.2.2.1 Goldstandard 200: Table B.2

B.2.2.2 Goldstandard Mixed: Table B.3

B.2.2.3 Goldstandard for Similarity: Table B.4

B.2.2.4 Goldstandard for Relatedness: Table B.5

Table B.5: Goldstandard for Relatedness: Correlation Scores

	$\rho_{t-average}$	$\rho_{averaged}$	Baseline ρ_{nlm}	Baseline ρ_{hlbl}
$f_{average-nlm}$	-0.06619116297	-0.0715297706	-0.1796253312	
$f_{concat-nlm}$	-0.07390115563	-0.0947179799	-0.1796253312	
$f_{average-hlbl}$	-0.08110358021	-0.05536940697		-0.2622672459
$f_{concat-hlbl}$	-0.07390115563	-0.0947179799		-0.2622672459