Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

TEMPORAL TEXT MINING: FROM FREQUENCIES TO WORD EMBEDDINGS

Relatore:
Chiar.mo Prof.
DANILO MONTESI

Correlatore:
Dott.
STEFANO GIOVANNI RIZZO

Presentata da:
PIERPAOLO ELIO JR DEL COCO

Sessione III
Anno Accademico 2016/2017

A me,
che così poi, in fondo, trovo anche te.

Sommario

Nell'ultimo decennio la quantità di dati testuali disponibili online è cresciuta rapidamente: dalle pagine web ai post nei social network e grazie inoltre alla digitalizzazione di libri e giornali, in ogni istante nuovi dati entrano a far parte degli archivi web. Questo processo continua da anni e oggi le grandi collezioni testuali utilizzate nei task di text mining comprendono documenti che sono stati archiviati per decenni. Da questo punto di vista, gli archivi web sono da considerarsi, oltre che delle sorgenti di dati, dei veri e propri contenitori di testimonianze storiche. Tuttavia, i modelli di word embeddings, nati dai recenti avanzamenti nel campo del machine learning e che rappresentano lo stato dell'arte nella rappresentazione delle parole nel testo, non sono in grado di catturare la dinamicità dei significati espressa dalle parole contenute in queste grandi collezioni testuali. Questo perché, nei modelli di word embeddings, le parole sono rappresentate tramite dei vettori in uno spazio vettoriale, ma ogni parola ha un unico vettore che la rappresenta lungo tutto l'arco temporale del corpus. Sebbene siano stati presentanti molto recentemente dei modelli dinamici di word embeddings (diachronic word embeddings), la letteratura a disposizione è scarsa e lascia aperte molte questioni irrisolte. Inoltre questi lavori presentano dei modelli di evoluzione del linguaggio che sono valutati da una prospettiva fortemente linguistica. In questa tesi, invece, prendendo in considerazione dei corpora in evoluzione continua, come ad esempio i giornali, il problema è affrontato da una diversa prospettiva. In particolare, i word embeddings temporali sono presentati come modelli per catturare la conoscenza accumulata negli anni dagli archivi testuali. Questo rende possibile l'analisi dell'evoluzione semantica, ma anche trovare delle analogie che esistono tra parole usate in contesti temporali differenti, ed infine, effettuare dei task di machine translation temporale. Questa tesi comprende inoltre un'analisi sulle frequenze delle parole utilizzate

nei corpora, corredata da esperimenti che dimostrano una forte correlazione tra la frequenza delle parole e i cambiamenti di significato delle stesse.

Abstract

The last decade has witnessed a tremendous growth in the amount of textual data available from web pages and social media posts, as well as from digitized sources, such as newspapers and books. However, as new data is continuously created to record the events of the moment, old data is archived day by day, for months, years, and decades. From this point of view, web archives play an important role not only as sources of data, but also as testimonials of history. In this respect, state-of-art machine learning models for word representations, namely word embeddings, are not able to capture the dynamic nature of semantics, since they represent a word as a single-state vector which do not consider different time spans of the corpus. Although diachronic word embeddings have started appearing in recent works, the very small literature leaves several open questions that must be addressed. Moreover, these works model language evolution from a strong linguistic perspective. We approach this problem from a slightly different perspective. In particular, we discuss temporal word embeddings models trained on highly evolving corpora, in order to model the knowledge that textual archives have accumulated over the years. This allow to discover semantic evolution of words, but also find temporal analogies and compute temporal translations. Moreover, we conducted experiments on word frequencies. The results of an in-depth temporal analysis of shifts in word semantics, in comparison to word frequencies, show that these two variations are related.

Contents

1

Introduction

The last decade has witnessed a tremendous growth in the amount of textual data, available from web pages and social media posts, as well as from digitized sources, such as newspapers and books. However, as new data is continuously created to record the events of the moment, old data is archived day by day, for months, years, and decades. From this point of view, web archives play an important role not only as sources of data, but also as testimonials of history. Indeed, language within text documents, has been evolving over the course of the years, reflecting the cultural evolution. Semantic meanings and popularities of words constantly change over time, with new concepts developing and old ones disappearing. For example, the word "*smartphone*" has not existed since about 2006, when the word "*palm handheld*" has since started disappearing. Furthermore, the meaning we give to some words may change considerably over time, due to the popularity of their associations. For example, the word "*facebook*", before the explosion in popularity of the social network, was recognized as a book, used in some American universities, which lists names and photographs of students. Similarly, "*apple*" was just a fruit, before it became one of the most important computer companies. On the other hand, named entities changes continuously their associations. For example, the "*white house*" is a word associated to different names of head of state, depending on the time period. The understanding of the temporal evolution of words thus, can be critical

in all those applications in which capturing the true semantics in different time contexts is essential. In this respect, the state of art *machine learning* techniques (e.g., *word2vec*[44][43]), which learn meaningful representations of words, do not dynamically capture semantic changes over time. In fact, these learning algorithms achieve very good results by processing huge amounts of data from text corpora, but the resulting representations of word meanings are aggregated over the whole corpus, and do not differentiate across different time periods. While the proliferation of *unstructured* data available on the Internet has led to significant advances in the development of techniques for *mining* information from natural language data, relatively little attention has been given to understanding the dynamics of temporal evolution of words.

Intuitively, considerable emphasis in *text mining* is given to the handling of natural language. Specifically, much of the focus of text mining falls on the critical role of transforming unstructured raw text into a concept-level data representation. From a text mining perspective, algorithms can extract information from textual documents at basically three different levels of semantics:

- **Words**. Words are the basic level of semantic richness. In general, we define a *word* as a single linguistic token, which is the smallest unit of information in text.

- **Phrases**. An higher level of semantic information can be extracted from phrases. A *phrase* can be defined as a sequence of one or more words carrying a particular meaning. For example, in the following sentence:

  "*The President of the United States lives in the White House*",

  the sequences "*United States*" and "*White House*", as well as "*President*" and "*lives*" are phrases.

- **Concepts**. Finally, we consider another level of semantics. In fact, words and phrases are independent units of information, such that two different units have two different meanings. In this respect, *concepts* enable a measure of semantic similarity between different words and

phrases. For example, "*United States*" and "*United Kingdom*" refer to concepts which are more similar than "*United States*" and "*lives*".

However, as one can easily imagine, it is a non-trivial task to achieve high level of semantics in representations, and research in natural language understanding is continuously evolving. In this respect, the application of novel *machine learning* techniques on text data, have considerably improved the accuracy of natural language understanding. In particular, the state of art techniques, used for modeling natural language, rely on learning algorithms which use vectors for representing words, and *learn* semantic relationships from the raw text. The learned semantics is then directly *embedded* into the word representations themselves. This particular type of semantic representations, namely *word embeddings*, will be widely discussed throughout the thesis.

On the other hand, text mining focus also on the analysis of patterns and trends. This generally implies a *quantification*, in terms of occurrence *frequencies*, of the concepts across textual documents of a given collection. In particular, we are interested in temporal analyses of text, which consider the dates of publication of documents within a collection, so that a comparisons can be made between subsets of documents relating to different periods. Thus, following the definition given by Mei et al.[40], in this thesis, we refer to *temporal text mining* as the discovering of temporal patterns in large collections of documents collected over time.

1.1 Motivation and Problem Statement

A we mentioned, text mining focuses on two crucial aspects: the discovery of patterns and trends in textual corpora, and the identification and extraction of representative features for natural language data[16]. With that in mind, in this thesis, we use text mining techniques to discover changes in popularity and semantics of words over time. These aspects are not totally separated: a semantic shift of a word from one context to another may cause a gain in popularity in the new context, whereas a growing popularity of a word may cause a shift in its meaning.

We found that relatively little attention in the literature has been given to dynamically tracking evolving semantics with word embeddings (i.e., diachronic word embeddings). Word embeddings are vector representations of words, which form a vector space such that similar words are geometrically close. These vector representations have been shown to improve the performance of a variety of text mining and natural language processing tasks. Word embeddings are derived from novel neural network algorithms that capture semantic and syntactic similarities through word neighborhoods over large corpora. However, although corpora may span over a long time period, all the different meanings, across the several time contexts, are collapsed into a single, not-so-accurate, mixed representation for each word over the whole time period. Given the importance and widespread use of word embeddings, which are massively used today in both industrial and academic projects, it is crucial to model word representations to be as accurate and unbiased as possible.

All these issues can be summarized in the following problem statements:

1. Frequencies based methods for mining trends across text corpora does not consider semantic changes over time.

2. Classic word embeddings are static and do not capture dynamics of semantics over time.

3. Whereas trends in popularity may indicate a semantic shift, word frequencies may also influence diachronic word embeddings models of semantics.

We will address these problems by performing two different text mining analyses on a large corpus, spanning a twenty year period. In particular:

- **Word frequencies**. We perform experiments on word frequencies across the corpus, in order to discover trends in popularity of words over time.

- **Temporal word embeddings**. We exploit the expressive power of word embeddings, by modeling temporal word embeddings to capture insights about the cultural evolution across time.

1.2 Thesis Structure

The rest of this thesis is organized into the following chapters:

- **Chapter 2**. Chapter 2 provides a background over two main topics on which this thesis is based: these are word representations and machine learning models for word representations. In particular, in the first section, we will discuss differences between sparse and dense representations, and in the second, we will illustrate basic theory behind machine learning models and artificial neural networks in particular.

- **Chapter 3**. Chapter 3 presents the related work. Specifically, this chapter is divided into two section: the first deals with the n-gram analysis, which concerns frequency based methods for quantifying word occurrences in large corpora; the second deals with word embeddings as approach to model semantics and, in particular, to model temporal changes of semantics.

- **Chapter 4**. Chapter 4 describe a frequency based approach to quantify patterns in word usage over time.

- **Chapter 5**. Chapter 5 focus on learning word embeddings. In particular, this chapter will first discuss how the state-of-art neural network models derive word embeddings from text corpora. Subsequently, the chapter will discuss the core of this thesis: specifically, time-dependent word embeddings (i.e., temporal word embeddings) will be discussed and how to use them in order to mine historical knowledge from text corpora.

- **Chapter 6**. Chapter 6 provides the results obtained by applying methods defined in previous on the New York Times Corpus. In particular, this chapter will present two different analysis: the first is a temporal analysis on frequencies, which is conducted in order to search for words that follow particular frequency-based trends over time; the second is a temporal analysis on word embeddings, which is conducted in order to investigate over changes in word semantics across different time periods in the corpus.

2

Background

To mine text, we first need to process it into a form that natural language processing procedures understand. In particular, we need data structures in order to represent words in textual documents. Word representations are thus the foundation on which a text mining analysis is built. Furthermore, vector representation of words are one of the main topics of this thesis. For this reason, a background is provided, which highlights the differences between the two main classes of word vectors, which are *sparse vectors* and *dense vectors*.

Moreover, as we will see, the state-of-art of word vector representations, namely *word embeddings*, are derived by non-linear models based on particular machine learning algorithms, which are known as *artificial neural network*. Here again, a background is provided, which illustrate the machine learning basics behind neural network models.

In summary, this background chapter is divided into the two following sections:

- **Word representations**. This section will illustrate differences between sparse and dense representations. In particular, it will be explained why dense representations are important in order to achieve a higher level of expressiveness than sparse vectors.

- **Machine learning basics for neural networks**. This section will illustrate the basics of machine learning we will be using later in chapter 5, to generate dense vector representations.

2.1 Word Representations

One of the arguably most important task in *text mining*, and more generally in *natural language processing* (*NLP*), is to generate a semantically accurate representation of text. Indeed, natural language data come in the form of a sequence of discrete symbols, which needs to be converted into numerical inputs for a model.

We define a *word representation* as a mathematical object associated with each word in a source text, which is the input of a natural language understanding task.

Many classic methods in computational linguistics have relied on very strong simplifying assumptions for word representations. In these methods words are treated as atomic units, overlooking the notion of similarity among words; more precisely, a word is represented as an index in a vocabulary list, or as a string of letters. By contrast, in more accurate models of semantics, a word is encoded as a vector in a *vector space* and each dimension of the vector captures some *feature* or attribute of the word. This allow to compute *semantic similarity* between words through vector similarity measures (e.g. cosine similarity).

Vector space models of semantics are very popular in NLP. In fact, we can distinguish between two main different approach to word representations within this class of models:

- **Sparse Representations**. Computational linguistics has a long tradition in word representation which comes from the *Distributional Hypothesis*[21] about language and word meaning. *Distributional representations* capture the intuition in which words that occur in similar contexts tend to have similar meanings. In this respect, the meaning of a word is computed from its distribution in a corpus, in terms of

the *counts* of co-occurrence with other words. However, as we know from Zipf's law[72], the most important characteristic of text data is that they are *sparse* and *high dimensional*. Accordingly, *distributional methods* result in very long vectors which contain mostly zeros, since most words simply never occur in the context of others.

- **Dense Representations**. On the other hand, with progress of machine learning techniques in recent years, it has become possible for more complex models to achieve a more accurate semantic representation of text, so that more meaningful text analysis and mining can be done. In particular, in contrast to *high-dimensional sparse* vectors generated by using the so-called *count-based* methods, more meaningful vectors are generated by using *predict-based* methods[4]. More specifically, artificial neural networks are capable of *learning* word meanings from large corpora, and *embedding* semantics relations into *low-dimensional dense* vectors, namely *word embeddings*, which have shown to improve the generalization of semantics representations[65]. In contrast to *distributional representations*, word embeddings are called *distributed representations* and care should be taken not to confuse one term with another[65].

These two approaches to word representations - in terms of vectors of explicit counts, and in terms of word embeddings - lead to different classes of algorithms as well as to different way of thinking. In section 2.1.1, we will outline some distributional methods, while, in section 2.1.2, we will introduce distributed dense representations. The latter, word embeddings in particular, constitute the basics of the semantics analysis in this thesis, and they will be accurately explained in chapter 5.

2.1.1 Sparse Representations

In traditional count-based NLP methods, words are often mapped as features which take the form of indicators or counts. An *indicator* simply takes a value of 0 or 1, encoding the existence of a condition. For example, an indicator feature for the word *cat* takes the value of 1 if *cat* appears at least once in the document, 0 otherwise. On the other hand, a *count* takes a value depending

on the number of times an event occurs. For example, the number of times the word *cat* appears in the text.

- **One-hot Encoding**. The most simple sparse model for word representation is the **one-hot encoding**. In this type of representation, each word is an $\mathbb{R}^{|V|}$ vector, where V is the *Vocabulary*, that is the set of all words in the corpus. Each word vector then contains a "1" at the index that stands for the word in the sorted vocabulary, and "0" everywhere else.

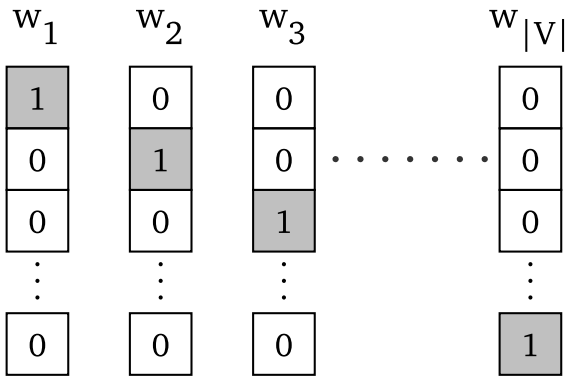


Fig. 2.1.: One Hot Encoding. The figure shows the one hot encoding representations for $|V|$ words.

- **Co-occurrence Matrices**. The one-hot encoding is a simplistic representation which does not give us any notion of similarity among words, since they are represented as independent entities. In order to encode semantic relationships more complex models are needed.

  Contextual information has been shown to provide a good approximation to word meaning[46], since words that are related will often appear in the same context. The idea behind distributional methods is thus to loop over text dataset and collect word co-occurrences counts in form of a matrix. Having the corpus represented as a matrix, makes it possible to perform computational calculations using linear algebra. Common examples of word co-occurrences matrices include word-document matrix and word-word matrix.

– **Word-document matrix**. A *word-document matrix* X describes the frequency of words that occur in a collection of documents. The matrix is built by looping over all the corpus documents and, for each time word i appears in document j, incrementing the count X_{ij}. Each row of the matrix then represents a word in the vocabulary and each column of the matrix represents a document from some collection. This results in a $\mathbb{R}^{|V|\times M}$ matrix, where V is the dictionary and M the numbers of documents. Word-document matrices have been originally defined as structures for the task of document *information retrieval*. Indeed, two documents that are similar will tend to have similar words, and if two documents have similar words their column vectors will tend to be similar.

– **Word-word matrix**. This same principle applies to words: similar words have similar vectors because they tend to occur in similar documents. In this respect, we use a *word-word matrix* X, where the columns are labeled by words rather than documents. This is thus a $\mathbb{R}^{|V|\times|V|}$ matrix, in which each cell X_{ij} records the number of times the *target* word i and the *context* word j co-occur in some context in some corpus. The context could be the document, in which case the cell represents the number of times the two words appear in the same document. It is most common, however, to use smaller contexts, generally a *window* around the word (i.e., n words to the left and n words to the right of the word target). In this case the cell X_{ij} represents the number of times the word j occurs in such a $śn$ word window around the word i. The size of the window used to collect counts can vary based on the goals of the representation, but it generally ranges from 1 to 8 words, on each side of the target word. In general, the shorter the window, the more syntactic the representations, since the information is coming from immediately nearby words; the longer the window, the more semantic the relations [57].

- **Weighting**. The raw frequency is not the best measure of association between words. In fact, there are words like "*the*", "*it*", or "*they*", which occur frequently together with all sorts of words, and they are not

informative about any particular word. In this respect, a measure that shows context words that are particularly informative about the target word should be used.

– **Pointwise mutual information**. The *pointwise mutual information* is the measurement of how much more often than chance two words co-occur[11]. It is defined as follow:

$$\text{PMI}(w_1, w_2) = log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}, \tag{2.1}$$

where $P(w_1, w_2)$ indicates how often the two words, $w_1 and w_2$, are observed together; while $P(w_1)P(w_2)$ indicates how often we would expect the two words to co-occur assuming they each occurred independently.

PMI values range from negative to positive infinity. However, it is more common to use *Positive PMI*(*PPMI*) which replaces all negative PMI values with zero[11]:

$$\text{PPMI}(w_1, w_2) = max(log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}, 0) \tag{2.2}$$

– **Tf-idf**. Other common measure of association is called tf-idf, and it comes from information retrieval. *Tf-idf* is the product of two factors: the first is simply the frequency $\text{TF}(i, j)$ of the word i in the document j; the second factor is the inverse document frequency. The *inverse document frequency* (*IDF*)[61] is used to give a higher weight to words that occur only in a few documents. These words indeed are more descriptive than those which occur frequently across the entire collection. Formally, the IDF is defined as follow:

$$\text{IDF}(i) = log\left(\frac{N}{DF(i)}\right), \tag{2.3}$$

where N is the total number of documents in the collection, and df_i is the number of documents in which term i occurs. The fewer documents in which a term occurs, the higher this weight. The lowest weight of 1 is assigned to terms that occur in all the

documents. Combining term frequency with IDF results in the tf-idf (i.e., TF(i, j)IDF(i)), which prefers words that are frequent in the current document, but rare overall in the collection.

2.1.2 From Sparse to Dense Representations

As seen above, the basic concept behind sparse linear models is that they represent each feature as a single dimension in a one-hot representation. On the contrary, more complex non-linear models encode each core feature into a d-dimensional vector, where d is usually much smaller than the number of features. In practical terms, this means that, for instance, each words in a vocabulary of 50.000 words can be represented as a vector of 100 or 200 features, instead of using a one-hot vector of 50.000 features (i.e., one feature for each word in the vocabulary).

It becomes immediately clear that one advantage of using low-dimensional dense vectors is computational. In fact, this is not the real reason why we prefer dense representations. The main advantage of using dense vectors is their generalization power. Dense vectors indeed capture similarities among words, by using their context. For instance, we consider observing the word "*coffee*" many times in the corpus, but rarely, or even worse, never, it co-occur with the word "*cappuccino*"; if we look at the sparse vector for the word "*coffee*", it would not give us much information about "*cappuccino*"; instead dense vectors for these two word may be similar, because they capture their similar contexts.

There are basically two ways to obtain dense representations:

- **Dense Vectors Via SVD**. The first method is by performing a *Singular Value Decomposition* (*SVD*) on a sparse matrix, and use the rows of the resulting matrix as dense word vectors. This method is beyond the scope of this thesis, but ample information can be found in the work of levy et al[35].

- **Word Embeddings**. A second method for generating dense embeddings draws its inspiration from the neural network models used for language modeling[5]. Neural network language models are given a word and

predict context words. This prediction process can be used to learn embeddings for each target word. The intuition is that words with similar meanings often occur near each other in text corpora. The neural models, therefore, learn a word embedding, by starting with a random vector representation for a word, and then iteratively shifting this vector to be more like the vectors of its neighboring words, and less like the vectors of the other words. Efficient neural models for deriving word embeddings[43][44], will be discussed further in this thesis.

2.2 Machine Learning Basics for Neural Networks

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience.*
>
> **— Tom M. Mitchell**

Machine learning is a very interesting field of artificial intelligence, which is based on the idea of learning from data.

With the proliferation of data, indeed, machine learning provides very useful automated methods for data analysis. In particular, machine learning can be defined as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data[49].

Machine learning is usually divided into two main types:

- **Supervised learning**. In *supervised learning* the goal is to learn a mapping from inputs x to outputs y, given a labeled set of input-output pairs $D = \{(x_i, y_i)\}, i = 1, \ldots, N$, where D is called *the training set*, and N is the number of training examples. Each training input x_i is a vector of numbers, which are called *features*. The output y_i can be categorical

or real-valued: in the first case, the problem is known as *classification problem*; in the second, the problem is known as *regression problem*.

- **Unsupervised learning** In *unsupervised learning*, we are only given inputs, $D = \{x_i\}, i = 1, \ldots, N$, and the objective here is to find descriptive patterns in the data. Unlike supervised learning, in this type of learning problems, we do not have labeled value to which compare the prediction output, and then there is no obvious error metric to use.

In fact, there is a third type of machine learning, known as *reinforcement learning*, in which a model is set to learn how to behave on the basis of reward or punishment signals.

In this thesis, we are interested in artificial neural networks, which are a particular type of machine learning model, and which will be presented in the following section.

2.2.1 Feed-Forward Neural Networks

Artificial neural networks were inspired by the functioning of the brain, and constitute interconnections of computation units, called neurons.

An artificial *neuron* is a model that takes n inputs, $x_1, x_2, \ldots, x_n$, and produces an output y. The neuron uses a simple rule to compute the output: each input is first multiplied by a specific weight $w_1, w_2, \ldots, w_n$; then, the weighted inputs are summed together to produce the *logit* $u = \sum_{i=0}^{n} x_i w_i$ of the neuron. The logit can also include a constant *bias*. Finally, an *activation* function f takes the logit to produces the output $y = f(u)$. The computation can be formulated as:

$$y = f(x \cdot w + b), \tag{2.4}$$

where $x \cdot w$ is the dot product of the input vector $x = x_1, x_2, \ldots, x_n$ and the weight vector $w = w_1, w_2, \ldots, w_n$; while b is the bias.

A single neuron can be used as a *binary classifier*. Specifically, given an input, the neuron fires (i.e., produces an output of 1) only if the data point belongs to the target class. Otherwise, it does not fire (i.e., it produces an output

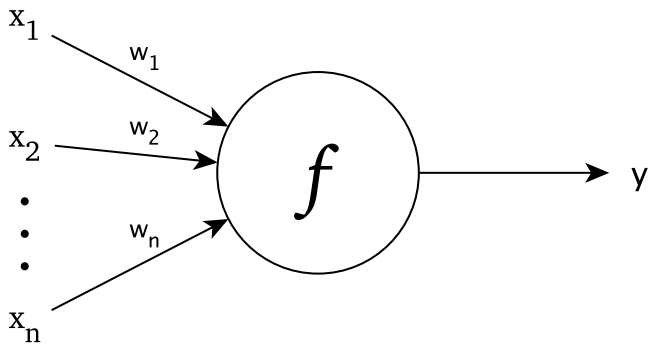


Fig. 2.2.: Artificial neuron. An artificial neuron with n inputs and their respective n weights.

of -1). Without loss of generality, we consider a 2-dimensional decision problem, in which a linear neuron classifier is used to divide the cartesian coordinate plane into two regions. The straight line which separates the two regions of the plane is called *decision boundary*, and it will be perpendicular to the weight vector. Just as in the equation of the straight line, the linear neuron classifier has two inputs, x_1 and x_2 (i.e., the x-coordinate and the y-coordinate), and a constant bias b; it uses the following activation function:

$$f(u) = \begin{cases} 1, & \text{if } u > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

More generally, given a dataset in which each data $x = \begin{bmatrix} x_1 & x_2 & \dots x_n \end{bmatrix}$ is a vector of n *features*, belonging to one of the two classes, then the objective is to learn a model $h(x, \theta)$, with the parameter vector $\theta = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \dots \theta_n \end{bmatrix}$ (i.e., the values for the bias b, and the n input weights w), such that:

$$h(x, \theta) = \begin{cases} 1, & \text{if } x \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix} + \theta_0 > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

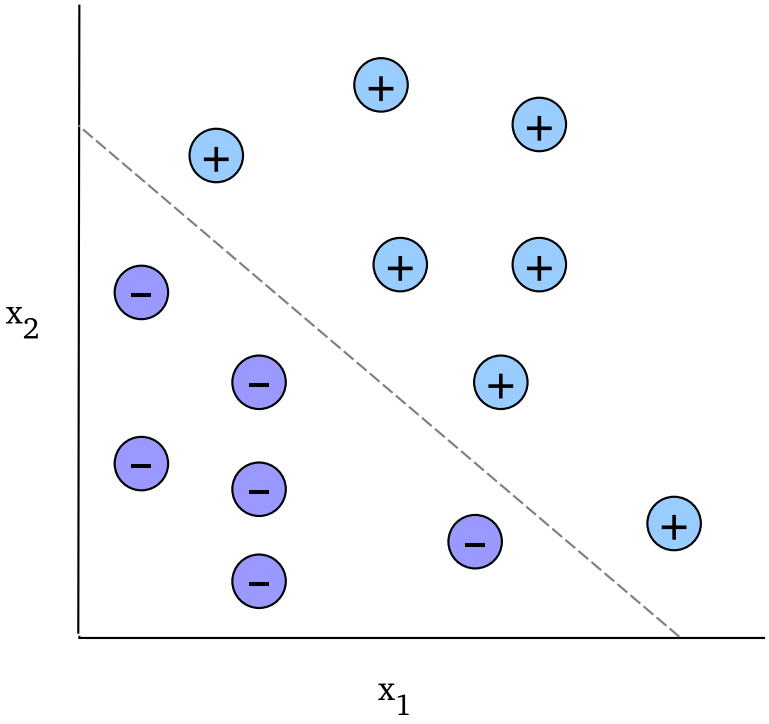


Fig. 2.3.: A linear classifier. The figure shows a concept represantation of a simple linear classification problem. Each data has two features, x_1 and x_2, and a label indicating its class (i.e., "+" or "-"). In this case $\theta = \begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}$ is the parameter vector. The dotted line, which divides points into two groups, is actually the decision boundary learned by the model.

The activation function defined in 2.4, and used by our binary classifier, is a particular type of linear function (i.e., the *unit step function*). The neuron with this activation function is a particular type of neuron called *perceptron*, which is actually not that common. In fact, the neuron types are defined by their activation function. Linear neurons are simple models that can only compute linear functions, and therefore their expressiveness is very limited. Indeed, neural networks which use hidden layers to learn complex relationships, need non-linear neurons.

In this respect, we present the three most common non-linear functions used by neurons:

- **Sigmoid function**. The *Sigmoid* function, which is also known as *logistic function*, is one of the most common activation function. It is defined as:

$$f(u) = \frac{1}{1 + e^{-u}} \tag{2.7}$$

This function works as follow: when the u is very small, then the output of a logistic function is very close to 0. Otherwise, when the u is very large, the output is close to 1.

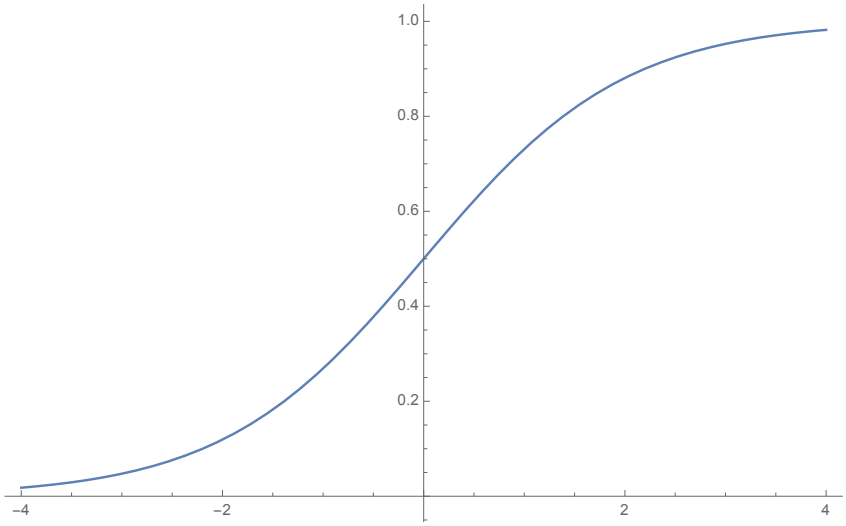


Fig. 2.4.: The Sigmoid function.

- **Tanh function**. The output of the *tanh* function is similar to the output of the logistic function, but it ranges from -1 to 1. It is defined as:

$$f(u) = \tanh(u) \tag{2.8}$$

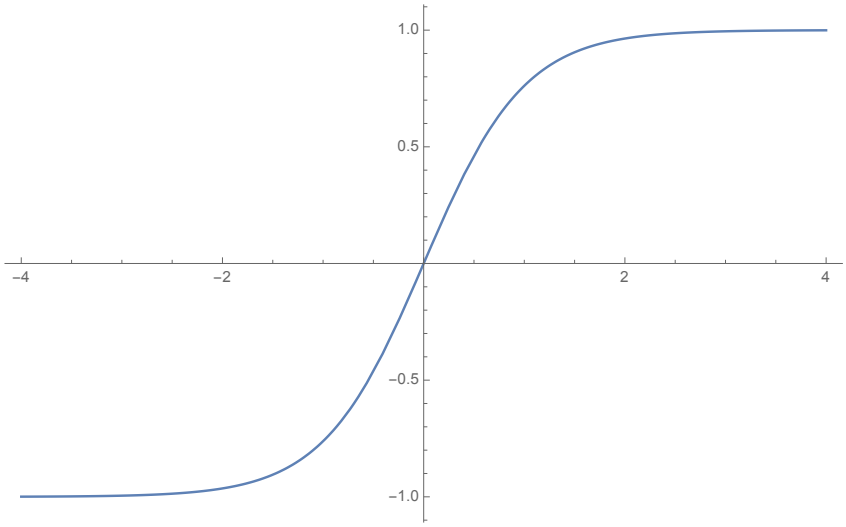


Fig. 2.5.: The Tanh function.

- **ReLU function**. The *restricted linear unit (ReLU)* function has recently become the chosen function for many practical tasks[50]. It is defined as:

$$f(u) = \max(0, u) \tag{2.9}$$

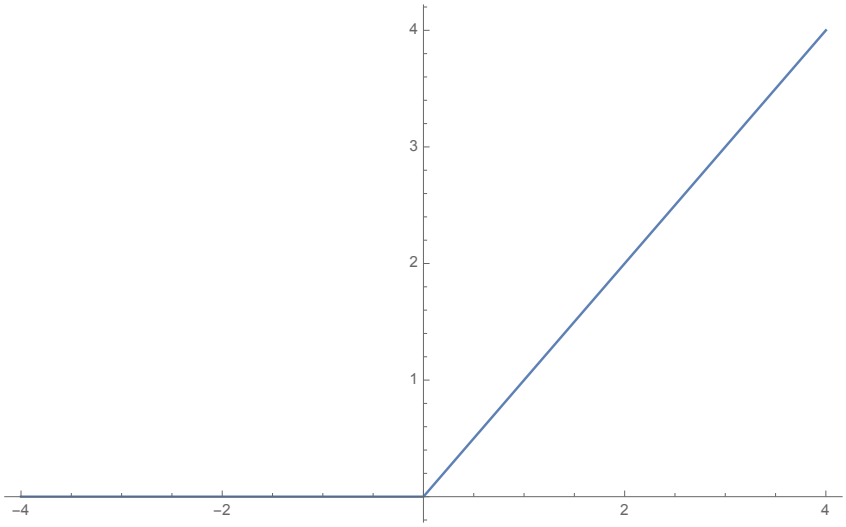


Fig. 2.6.: The Relu function.

Finally, the neurons are connected to each other forming a network, such that, the output of a neuron at layer i feed into the input of all neurons at level $i + 1$. Each connection carries a weight, reflecting its importance. The first and the last layers are respectively the input and the output layers, while the other layers are called *hidden layers*. This type of neural networks is called *feed-forward neural networks*, since data move forward from inputs to the outputs through the network. Artificial neural networks can approximate a very wide range of mathematical functions.

2.2.2 Training Feed-Forward Neural Networks

At this point, we need to discuss how to find the values of the parameters vector. This is accomplished by a process commonly referred to as *training*. During the training, a learning algorithm uses data samples (i.e., the *training set*) to iteratively modify the weights, and builds a model which is able to capture the relationship between features and classes in the data, and then

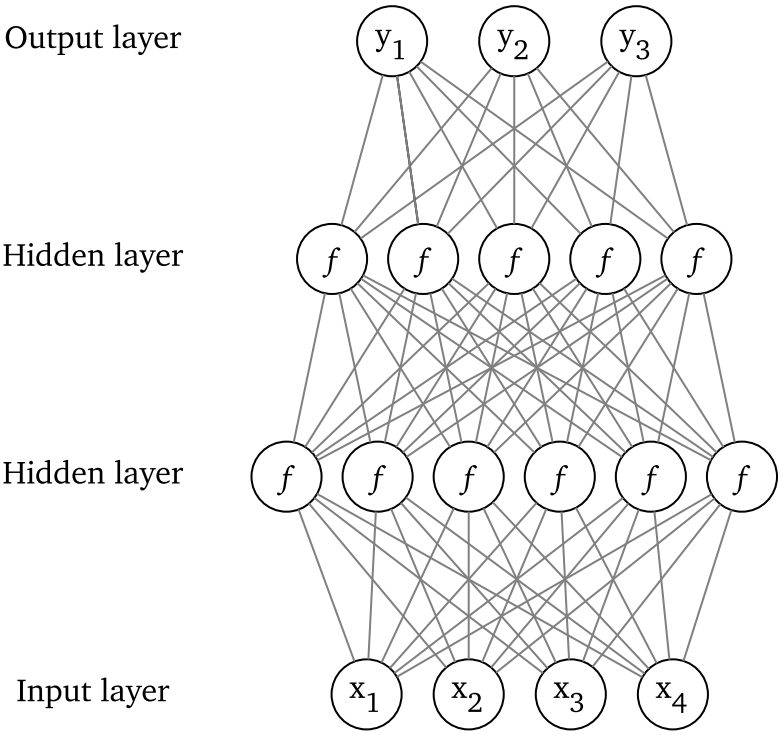


Fig. 2.7.: Artificial neural network. The figure shows an artificial neural network with two hidden layer.

predict classes for future data. This is defined as the problem of finding the optimum combination of values for the parameters vector.

The simplest learning algorithm is the *perceptron algorithm*. As we mentioned, the perceptron produces the output $y = f(u)$, where $f(u)$ is the function defined in 2.4. In this case, weights are modified according to the following update equation:

$$w^{(\text{new})} = w^{(\text{old})} - \eta \cdot (y - t) \cdot x, \tag{2.10}$$

where t is the label of the class, and $\eta, \eta > 0$ is a particular parameter called *learning rate*, which is used to set how quickly the network should change what it has already learned, according to new different data samples.

When the logistic function defined in 2.7 is used, the training objective is defined as the following *cost function*:

$$E = \frac{1}{2}(t - y)^2 \tag{2.11}$$

Since, the output $y = f(u)$ is dependent on u, and u is dependent on input weights w_i, it is possible to use the partial derivative chain rules, in order to minimize the cost function by each parameter. This procedure of trying to find the parameters that minimize error values, by using partial derivatives to differentiate the cost function by each parameter, is called *gradient descent*. Specifically, we compute the derivative of E with regard to w_i by the following formula:

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_i} \\ &= (y - t) \cdot y(1 - y) \cdot x_i \end{aligned} \tag{2.12}$$

Thus, the update equation for the logistic neuron is the following:

$$w^{(\text{new})} = w^{(\text{old})} - \eta \cdot (y - t) \cdot y(1 - y) \cdot x_i \tag{2.13}$$

When we deal with a multi-layer neural network, the concepts is quite similar. As well as before, we need to define the loss function, take the gradient of that function, and update the weights of the neural network in order to minimize the error. In this case, it is very useful to follow a procedure called *back-propagation*: we start to compute the derivative for the last layer, and then move backwards, back-propagating the error to the previous layer.

Without loss of generality, we consider a multi-layer neural network, which is composed by:

- an input layer with K neurons: $x = \{x_1, \ldots, x_K\}$,
- a single hidden layer with N neurons: $h = \{h_1, \ldots, h_N\}$;
- and an output layer with M neurons: $y = \{y_1, \ldots, y_M\}$.

The output for each neuron h_i in the hidden layer is defined as:

$$h_i = \sigma(u_i) = \sigma\Big(\sum_{k=1}^{K} w_{ki} x_k\Big), \tag{2.14}$$

where $W = \{w_{ki}\}$ is the $K \times N$ weights matrix between the input and the hidden layers.

In a similar way, we define the output for each neuron y_j in the output layer:

$$y_j = \sigma(u'_j) = \sigma\Big(\sum_{i=1}^{N} w'_{ij} h_i\Big), \tag{2.15}$$

where $W' = \{w'_{ij}\}$ is the $N \times M$ weights matrix between the hidden and the output layers.

Finally we define the cost function as the following:

$$E\Big(x, t, W, W'\Big) = \frac{1}{2}\sum_{j=1}^{M}(y_j - t_j)^2, \tag{2.16}$$

where $t = \{t_1, \dots t_M\}$ is a M-dimension vector containing the class labels.

For each layer, we perform three step:

1. We first compute the derivative of the error with regard to the output:

$$\frac{\partial E}{\partial y_i} = y_i - t_j. \tag{2.17}$$

2. We then compute the derivative of the error with regard to the logit of the output layer:

$$\frac{\partial E}{\partial u'_j} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial u'_j} = (y_j - t_j) \cdot y_j(1 - y_j) := EI'_j. \tag{2.18}$$

3. Finally, we compute the derivative of the error with regard to the weights between the hidden and the output layers:

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u'_j} \cdot \frac{\partial u'_j}{\partial w'_{ij}} = EI'_j \cdot h_i. \tag{2.19}$$

Thus, the resulting update equation for the weights between the hidden and the output layers is the following:

$$\begin{aligned} w'^{(\text{new})}_{ij} &= w'^{(\text{old})}_{ij} - \eta \cdot \frac{\partial E}{\partial w'_{ij}} \\ &= w'^{(\text{old})}_{ij} - \eta \cdot EI'_j \cdot h_i. \end{aligned} \tag{2.20}$$

At this point, we can repeat the same three steps to compute the update equation for the weights between the input and the hidden layer. Specifically, we use the intermediate result EI'_j in order to back-propagate the factor $\sum_{j=1}^{M} EI'_j \cdot w'_{ij}$ to the hidden layer neuron h_i:

1. We compute the derivative of the error with regard to the output of the hidden layer, which is related to all the neurons in the output layer:

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{M} \frac{\partial E}{\partial u'_j} \frac{\partial u'_j}{\partial h_i} = \sum_{j=1}^{M} EI'_j \cdot w'_{ij}. \tag{2.21}$$

2. Then, we compute the derivative of the error with regard to the logit of the hidden layer:

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial u_i} = \sum_{j=1}^{M} EI'_j \cdot w'_{ij} \cdot h_i(1 - h_i) := EI_i. \tag{2.22}$$

3. Finally, we compute the derivative of the error with regard to the weights between the input and the hidden layers:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial u_i} \cdot \frac{\partial u_i}{\partial w_{ki}} = EI_i \cdot x_k. \tag{2.23}$$

The resulting update equation for the weights between the input and the hidden layers is therefore the following:

$$w_{ki}^{(\text{new})} = w_{ki}^{(\text{old})} - \eta \cdot EI_i \cdot x_k. \tag{2.24}$$

3

Related Work

In this chapter we present the related work. Specifically, we divide this chapter into two section:

- **The n-gram analysis**. We will discuss work related to frequency based methods and quantification of word occurrences over time in particular.

- **Word embeddings**. On the other hand, we will discuss word embeddings as approach to model semantics. In particular, we will present work related to temporal analysis of semantics through word embeddings.

3.1 The N-gram Analysis

N-gram analysis is one of the main computational text mining techniques, which is extremely useful for extracting knowledge from archived historical documents. An *n-gram* is defined as a sequence of words of length *n*. N-gram analysis is a *quantitative* method based on counting the relative volume or frequency that a certain n-gram occurs in a dataset. Frequency based methods are quite popular, and include tools like *Google Trends* and *Google Ngram*

which are largely used in research to analyse and predict trends, especially in economics[10].

3.1.1 Precursors in Quantitative Text Analysis

N-gram analysis is based on *Quantification*, which has a long history in research. Quantification is the result of frequencies obtained through counting the occurrences of content units[17]. In this section, we will discuss some precursors in quantitative methods:

- **Pioneering Works**. Quantitative methods for text analysis have emerged from the corpus linguistics field of research[59] as a solid approach for modeling qualitative linguistic phenomena. The earliest works in quantification of linguistic units in written language were made by Zipf[72] at the beginning of 1930s. He noticed that words over a collection of textual documents occur according to a systematic frequency distribution, such that words have a frequency that scales as a nonlinearly decreasing function of their frequency rank order; formally:

$$f(w) \propto r(w)^{-a}. \tag{3.1}$$

  Subsequently, other linguists such as Bloomfield[68] started thinking that language could be explained in terms of probability distributions through empirical and statistical methods. Moreover, with the emergence of the behavioral and social science, the period between 1930s and 1940s saw an increasing use of survey research and polling. Quantification had thus a key role in helping to handle historical and social problems through systematization, rigor, precision, and exactitude in definitions and measurements, through objectivity and replication of procedures and findings, and in other words, through a scientific approach[17].

  However, quantitative methods for language analysis came up against a brick wall in the 1950s. In fact, during this period there was a tendency in social science towards introspective modeling of cognitive function at the expense of quantitative data-driven methods. In this respect,

the work of linguist Noam Chomsky[33], focused on both a formal methodology and a theory of linguistics, was very influential over the 1960s and 1970s. To make matters worse, textual datasets were at the time too small to make interesting statistical generalizations over large linguistics phenomena. However, as more language data has gradually become available, successful applications of quantitative methods in natural sciences, in humanities, and in linguistics gradually have been increased. Data availability has given rise to the modern age of corpus linguistics, and quantitative techniques can now be meaningfully applied to millions of texts and billions of sentences over the web.

- **Content Analysis of News Media**. *Content analysis* is a research technique for the objective, systematic, and quantitative description of the manifest content of communication[8]. The aim of content analysis can be summarised as being to *turn words into numbers*[18]. Any question asking how much?, how often?, or to what extent? will usually require access to quantitative data. Using content analysis, a researcher is able to extract information from qualitative sources (e.g. newspaper articles) to provide quantitative measures[25].

  Since newspapers are testimonials of history, probably the most common use of content analysis has been to infer the importance news media assign to particular subject-matter form the frequency with which such subject-matter is mentioned. Early examples are analysis of how attention by newspapers to particular news subject has changed over time[67]. With the increase in mass production of newsprint at the beginning of the 20th century, and the resulting interest in public opinion, also emerged demands for ethical standards and for empirical inquiries into the phenomenon of the newspaper. Many studies have been motivated by the feeling that journalistic standards were inadequately applied. For example, concerns for fairness are implied in numerous content analyses that aim to show the inequality of the coverage of the two (or more) sides of a public controversy [29]. *Quantitative newspaper analysis* has provided the needed scientific objectivity for journalistic arguments[70].

Analysis of the content of news media has repeatedly been shown to produce results that are remarkably similar to attitude surveys and opinion polls[6]. In this respect, quantitative methods have largely been used in order to develop quantitative indicators for social and political phenomena.

During World War II, contest analysis of German newspapers has been used in order to describe and predict Nazi communication activity[51]. By applying statistical techniques to the flow of symbols recorded in the "prestige papers" of France, Germany, Great Britain, Russia, and the United States over half a century, Lerner et al.[34] found that conditions of war and totalitarianism tend to restrict the range of symbolism. Furthermore, in the late 1940s, Lasswell et al.[32] found trends in the editorials' language reflecting increasing nationalism and growth of proletarian doctrines, by examining 60 years of editorials in five prestige world newspapers.

More recently, Danielson and Lasorsa[14] carried out a content analysis over 100 years of front page news in The New York Times and The Los Angeles Times, revealing some of the major social and political changes in American society.

Example of quantitative content analysis of newspaper can also be found in several research areas, such as consumer research[27], advertising[38], health and medical[30], psychology and marketing[69], tourism demand modeling[62], etc.

- **Computational Social Sciences and History**. The analysis of massive datasets with traditional human-driven approaches to content analysis is challenging. Leveraging computational power to collect and analyze large datasets has attracted increasing attention in many historical and social studies during the past decades. In this respect, a wide range of computational techniques has been employed in order to reveal patterns of change over time in individual and group behaviors. This has opened up new interdisciplinary field of research, such as computational history[3][2][24][26] and computational social science[13][12].

3.1.2 Temporal N-gram Analysis

During the last decades, the increasing availability of data spanning different epochs, such as newspaper archives, scanned books, and other digital artifacts, has increasingly inspired a new analysis of cultural and social phenomena from a temporal perspective. By looking at the sequence of articles over time, one can discover the birth and the development of trends that marked society and history[63]. In this section, we will discuss major works in this direction.

- **Culturomics**. The term *culturomics* was coined by Michel et al. in their work "*Quantitative analysis of culture using millions of digitized books*"[42]. The study is a massive quantitative text analysis of frequency distribution over time of trillions of n-grams from 15 million books in Google Book (about 4% of all books ever printed). Quantitative exploration of extensive historical text datasets enables identifying and measuring cultural and social patterns of change, which are reflected by the language used.

  The value of the culturomics approach have been largely demonstrated. Among other results, Michel et al.[42] shown how the quantification of term *influenza* resulted in a graph whose peaks correspond with dates of known deadly pandemics throughout history.

  In his study of the Science Hall of Fame, Bohannon[9] suggested culturomics as a method for measuring the fame of scientists over the centuries by using Wikipedia.

  Following its rise, culturomics has been utilized in many disciplines, such as cultural and social science[66], psychology[23], and accounting[1].

- **Learning Technologies**. As we discussed, the goal of temporal n-gram analysis is to identify trends of change over time. In this respect, the field of *learning technologies* undergoes frequent changes as new technologies emerge, experienced, and adopted or abandoned.

Soper and Turel[60] performed a culturomic analysis of the n-grams appeared in the monthly magazine of the *Association for Computing Machinery* (ACM), *Communications* of the ACM, during the years 2000-2010. They demonstrated how temporal quantitative exploration of the articles in *Communications* can be used to better understand the culture, identity, and evolution of computing.

Furthermore, Raban and Gordon[53] used bibliometric analysis methods in order to identify trends of change in the field of learning technologies over a period of five decades.

More recently, Silber-Varod et. al. [58] used a data-driven approach on a corpus of all the papers published in the proceedings volumes of the *Chais Conference for the Study of Innovation and Learning Technologies*, during the years 2006-2014, in order to investigate changes over time in the research of learning technologies.

3.2 Word Embeddings

Context-predicting methods for word representations have shown great results against their *count-based* counterparts[4]. These models, known as Word Embeddings, represent words by encoding their contexts into low-dimensional dense vectors, such that geometric distances between word vectors reflect semantic similarity. We differentiate between two different models of word embeddings:

- **Time-Agnostic Word Embeddings**. We define as *time-agnostic word embeddings* the classic static approach to modeling semantics through word vectors.

- **Time-Dependent Word Embeddings**. On the contrary, we define as *time-dependent word embeddings* the temporal scheme of word embeddings which capture dynamics of semantics over time.

3.2.1 Time-Agnostic Word Embeddings

Actually, the idea of word embeddings has existed at least since 90s, with vectors of co-occurrences[36], through matrix factorization[15] and more recently through neural networks[5]. However, they have become popular since Mikolov et al.[43][44] proposed the skip-gram model with negative sampling (*word2vec*) based on stochastic gradient descent. Since then, word embeddings have largely proved their worth as an efficient and effective way for modeling semantics. In fact, during the last few years, these models have established themselves as state-of-art in almost all nlp tasks, in both academia and industry.

3.2.2 Time-Dependent Word Embeddings

In contrast to traditional approach, relative little literature has been devoted to time-dependent (i.e., diachronic) word embeddings. Specifically, some reference works in this area are Kim et al.[28], Kulkarni et al[31] and Hamilton et. al[20]. These works use large time-span corpora (i.e., Google Books) to model language evolution from a linguistic perspective. However, the problem can be examined under a slightly different light, by applying these temporal models to highly evolving corpora (i.e., newspapers and blogs). In particular, temporal word embeddings allow to capture dynamics in word meanings and named entities associations, by modeling the knowledge that textual archives have accumulated. This can be used to better interpret the textual resources that humans have produced. Only few works have focused on this aspect. In this respect, we mention the very recent works by Yao et al.[71] and Szymanski[64].

4

Temporal Analysis of Frequencies

In this chapter we will describe a frequency based approach to quantify patterns in word usage over time. Frequency based methods are straightforward, since based on simple counting of word occurrences. However, these methods provide an immediate way to identify patterns over time. In particular, we will discuss aspects related to the quantification of words in text corpora, and construct time series of word frequencies. Before examining our quantification methods, we will discuss the data preprocessing pipeline used to prepare initial data for text mining analysis.

4.1 Data Preprocessing

Data preprocessing is a critical stage in text mining that is used to transform the initial raw text into a clean dataset. In this section, we will discuss the major steps involved in data preprocessing, namely:

1. Word tokenization

2. Removing stop-words

3. Stemming and lemmatization

4.1.1 Words and Tokenization

Defining *what a word is* has long been a subject of debate in computational linguistics[19]. A common definition of *word* is *the smallest unit of meaning*[41]. By following this definition, the first step of a text mining and natural language processing task is to segment the input text into linguistic units called tokens. This process is referred to as *tokenization*[39].

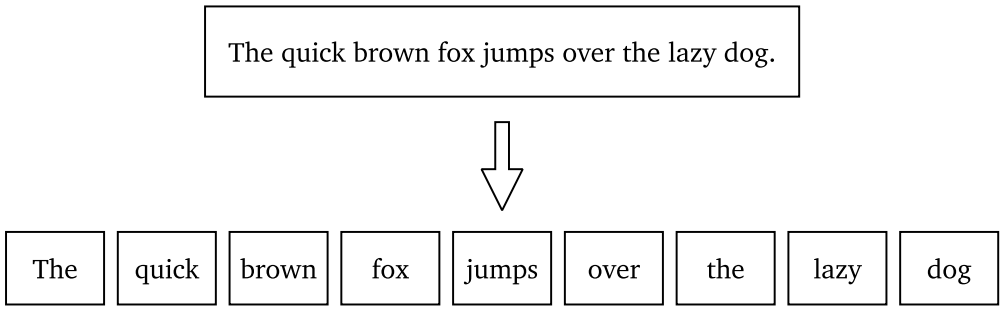


Fig. 4.1.: Tokenization. An example of tokenization based on whitespaces and punctuations.

Identifying token boundaries is a somewhat trivial task. When working in English (as this thesis assumes), tokenizing on whitespaces and punctuations can provide a good approximation of words, although it is important to consider cases such as abbreviations (e.g. U.S.A.) and titles (e.g. Mr.). In other languages, tokenization can be a much more challenging task. For example, in German compound nouns are written as a single word, in Hebrew and Arabic some words attach to the next one without whitespace, and in Chinese there are no whitespaces at all.

After splitting by whitespaces and punctuations, however, there are still some problems to be considered. The main ones are listed below:

- **Contractions**. It is a difficult question to know how to handle English contractions such as *don't* or *I'm*. Thus some tokenizer treat these examples as one word, while others split contractions into two words (e.g. *do not* and *I am*).

- **Hyphenation**. Dealing with hyphens leads to another difficult question: do sequences of letters with a hyphen in between count as one or more words[39]? In fact, there exists different types of hyphens. One is line-breaking hyphens, which are used to improve justification of text. Hyphens are also used for splitting up vowel in words like in *e-mail* or *co-operate*. We can find hyphens in compound words such as *up-to-date* or in names such as *Hewlett-Packard*. Another type of hyphens are used to indicate the correct grouping of words like in *the 26-year-old* or *a final take-it-or-leave-it offer*. Hyphenated words of the first two examples should be treated as one token, and the last should be divided into more tokens, while the middle cases are unclear. Handling hyphens automatically can thus be complex: it can either be done as a classification problem, or more commonly by some heuristic rules.

- **Whitespaces not indicating a word break**. Another common problem happens when we want to keep together a sequence of words separated by whitespace. Words such as *New York* or *San Francisco* should be treated as a single token. As we will see in **??**, this problem is generally addressed at a higher level by using n-gram models.

4.1.2 Removing Stop Words

Stop words are words which provide no information value, from a text mining perspective. The main property of stop-words is that they are extremely frequent words. These words are dependent on natural language and different languages have their own list of stop words. For example, in the English lexicon, words such as "*the*", "*and*", and "*in*" occur very often compared to other rare words. Removing stop words is a necessary step in order to reduce information noise, before more complex NLP tasks can be applied.

We define two types of stop words:

- **Generic stop words**. Generic stop words are non-information carrying words within a specific language. These words can be removed using a *stoplist* of all generic stop words of a specific language. When the document is being parsed, if a word found in the document is present

in the stoplist, then it can be discarded instead of being used for further processing.

- **Misspelling stop words**. Misspelling stop words are not real words but spelling errors within the documents (e.g. "*world*" spelled as "*wrold*"). This terms should be treated as stop words, and removed. A good way to identify misspelling stop words is by using the term frequency profiles of the corpus. In large corpora indeed, terms which occur very infrequently (i.e. only once or twice) are most likely to be misspelling stop words. Unlike the first type of stop words, these terms can thus be removed after statistical calculations on the whole corpus have been applied.

4.1.3 Stemming and Lemmatization

Another important question to consider when processing text data, is whether to keep word forms like *cat* and *cats*, or *fish* and *fishing* separate or to collapse them. The most common way to proceed is to group such forms together and working in terms of *lexemes*. This is usually referred to in the literature as *stemming*, or alternatively *lemmatization*[39].

In fact, the goal of both stemming and lemmatization is to reduce the inflectional forms and derivations of a word to a common base form; however, the two methods differ in the way they operate.

Stemming usually refers to a simple heuristic process which applies a set of rules to an input word in order to remove suffixes and prefixes and obtain its *stem*. The most common algorithm for stemming English is Porter's algorithm[52], (whose details go beyond the scope of this thesis) which has repeatedly been shown to be empirically very effective. As shown in figure 4.3, however, stemming often produces stems that are not valid words.

On the other hand, lemmatization usually refers to a process which is carried out with the use of a vocabulary and through a morphological analysis of words. This process returns the base or dictionary form of a word, which is known as the *lemma*. However, lemmatization is usually computationally more expensive than stemming.

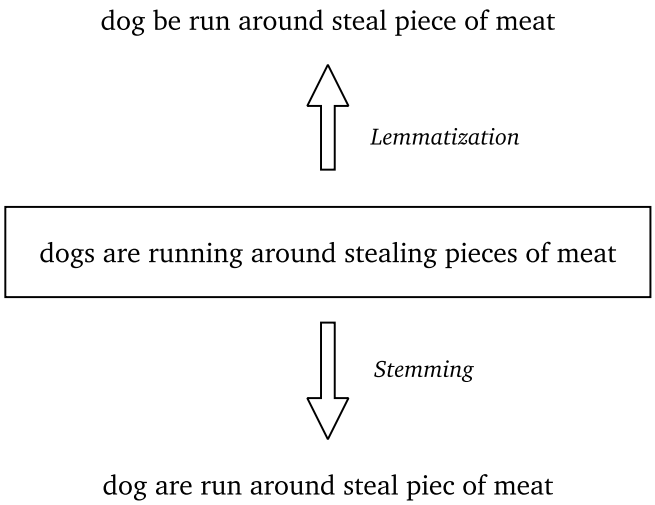


Fig. 4.2.: Stemming and lemmatization. An example showing the comparison between stemming and lemmatization. Note that for the word "pieces" the lemma is "piece", while the stem is "piec". For the word "are" the lemma is "be", while the stem is still "are".

4.2 From Words to Numbers

We aim to quantify significant change in the word usage across time.

Given a temporal corpus $\mathcal{C}$, which is collected over time, we consider n time-frames of length L (e.g., L can be a frame of 1 year); we then divide $\mathcal{C}$ into n corpora $\mathcal{C}_t$, one for each time-frame t.

To model popularity trends in word usage, we follow a two-step procedure:

1. **Words extraction**. We first build the words vocabulary $\mathcal{V}$, by extracting words from corpus $\mathcal{C}$. In particular, in this phase, it is important to consider whether words that make up a sentence, are actually single words or phrases (i.e., sequences of words). In section 4.2.1, we will treat this problem as a problem of *n-gram detection* in a given sentence.

2. **Time series construction**. Subsequently, we create a time series $\mathcal{T}_t(w)$ for each word $w \in \mathcal{V}$, corresponding to the number of occurrences of w, observed in the corpus at the time period t. This process is illustrated

in sections 4.2.2 through 4.2.3, covering the aspects of quantification in general (4.2.2), and temporal quantification in particular (4.2.3).

4.2.1 N-grams Detection

In natural language processing, an *n-gram* is defined as a sequence of words of length *n*, extracted from a larger sequence of words[39]. For example, the phrase *new york city* can be partitioned into three 1-grams (i.e. *new*, *york* and *city*), two 2-grams (i.e. *new york* and *york city*), and one 3-gram (i.e. *new york city*).

As can be easily seen, despite being correct words, some combinations of n-gram have no information value with respect to the source text. Moreover, taken out of their context, some n-grams can be misleading information, too. For example, the 1-gram *york* and the 2-gram *york city* of the phrase *new york city* could refer, respectively, to the city of York in England, and his football club.

In order to perform a more meaningful n-gram analysis, only a few combinations of n-gram, that are *n-grams of interest* (i.e., phrases), should be selected for the quantification. Automatically detecting all and only the n-grams of interest is a non-trivial task. In this respect, it is possible to achieve good results by considering n-grams which consist of frequently co-occurring tokens.

Following the data-driven approach suggested by Mikolov et al.[43], it is possible to score every pair of co-occurring words (i.e. 2-grams) in a corpus. More precisely, the entire corpus is processed as a sequence of unigrams, and a score is assigned to each pair of consecutive words w_i and w_j, by using the following formula:

$$\texttt{score}(w_i, w_j) = \frac{\texttt{count}(w_i w_j) - \delta}{\texttt{count}(w_i) \times \texttt{count}(w_j)}, \tag{4.1}$$

where $\texttt{count}(x)$ is the count of the occurrences of the word x in the corpus, while $\texttt{count}(xy)$ is the number of times the word x co-occurring with the word y; the δ is used as a discounting coefficient and prevents too many

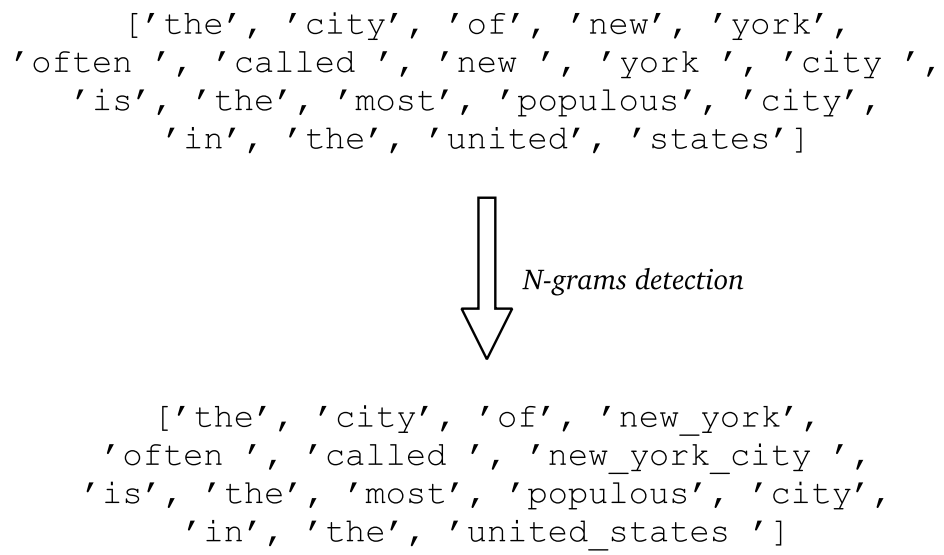


Fig. 4.3.: N-grams Detection. An example showing the output of the function `trigram`(X), where X is the tokenized text: "*The city of New York, often called New York City, is the most populous city in the United States*". The output list consists in unigrams (`the`, `city`, `of`, `often`, `called`, `is`, `most`, `populous` and `in`), bigrams (`new_york` and `united_states`), and trigrams (`new_york_city`). Note that just for the sake of readability, stopwords in this example have not been removed.

phrases consisting of very infrequent words to be formed. The bigrams with a score below a chosen threshold should be discarded.

In this way it is possible to build a function `bigram`(X), which takes as input a list of unigrams X, and returns another list of terms built as follows: for each pair of consecutive unigrams X_i and X_j in X, returns a single term, which is the bigram $X_i\_X_j$ if X_i and X_j form a bigram of interest; the unigram X_i otherwise.

In general, in order to model n-grams with successive values of *n*, it is necessary to reiterate the process. For example, obtaining 3-grams is possible by scoring every pair of consecutive terms occurring in the corpus which is the output of the function `bigram`(X). In this case a function `trigram`(X) is defined as `bigram`(`bigram`(X)).

It should be recalled, however, that due to the sparsity of data, for higher level of aggregation in n-grams (i.e. $n > 3$), a massive amount of data is needed in order to achieve an acceptable precision in scoring. However, meaningful analyses of textual data are possible by considering just unigrams, bigrams and trigrams.

4.2.2 Counting Occurrences

Having defined a function to produce a list of n-grams from a tokenized input text, we can easily build the vocabulary $\mathcal{V}$ of all words (i.e., ngrams of interest) in the corpus.

However, an important question to be considered, before proceeding to the quantification of words, is whether each word forming a n-gram should also be counted as occurrence. For example, having already counted the word *Machine Learning*, one can also consider the single occurrences of words *Machine* and *Learning*, since the notion of learning machine here is preserved by the single unigrams. However, the clearest, and arguably the simplest, approach is to not attempt to quantify single word units of n-grams. Indeed, as mentioned above, often, single parts of n-grams can be misleading information with respect to the original word (e.g. *New York*).

Given the vocabulary $\mathcal{V}$, constructed from the corpus $\mathcal{C}$, we indicate the quantification of the word w, for each $w \in \mathcal{V}$, as follows:

$$\#(w \in \mathcal{C}), \tag{4.2}$$

which is the number of occurrences of the word w in the corpus $\mathcal{C}$.

Having counted the occurrences for each word $w \in \mathcal{V}$, we look for a moment at the distribution of the obtained frequencies. The figure 4.4 shows the typical relationship between the frequency of words, and the number of different words with that frequency. The resulting curve describes the well known Zipf's law of word frequencies in natural language[72]. Without loss of generality, we consider the upper part of the curve: a plethora of words has been used very few times. In general, these words are typing errors, and should be discarded. Specifically, we can establish a threshold of minimum

occurrences, below which a word should be discarded. On the other hand, the lower part of the curve shows that the most common words are used the most. In general, these words do not provide added-value to the text analysis. Here again, a threshold of maximum occurrences should be used, above which a word should be discarded. Finally, the middle portion of the distribution curve contains words with moderate frequency. These are the words we are interested in. It is clearly visible that the overwhelming majority of the terms in the corpus will not be used in our text analysis.

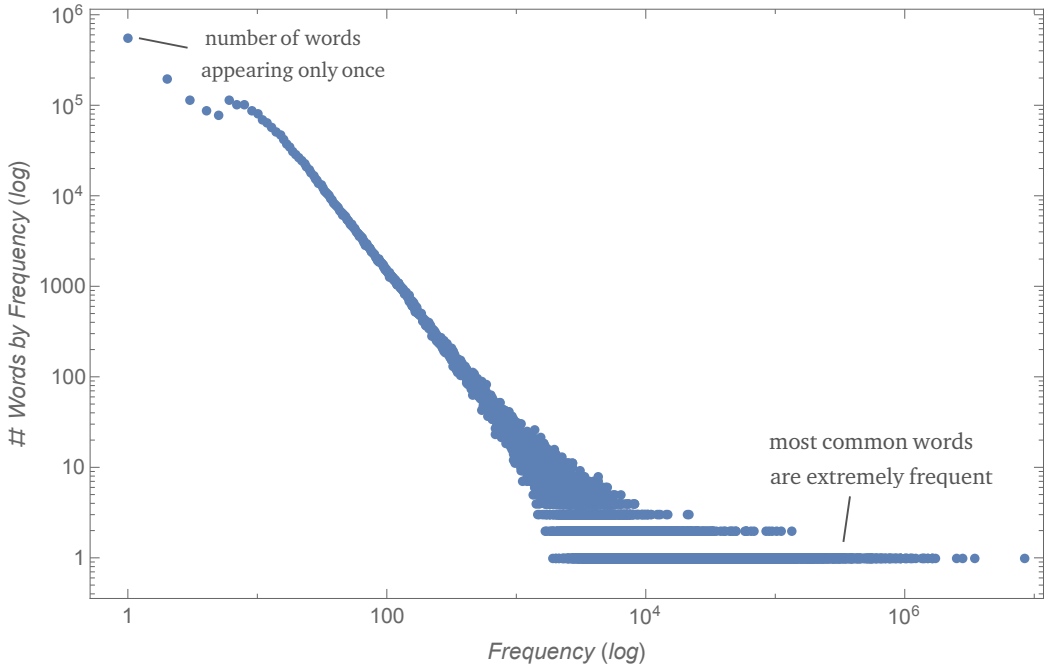


Fig. 4.4.: Frequencies distribution. The log-log plot in figure shows the typical curve obtained by plotting against the word frequencies (x-axis), the number of different words with that frequency (y-axis). Data is taken from the *New York Times Corpus*. The plot shows how common words are also the most frequent; on the other hand, it highlights the number of words which occur only few times (e.g., more than half a million of words occur only once).

4.2.3 Time Series from Word Frequencies

Constructing time series from frequencies is the first step to quantifying patterns of word change. In particular, we are interested in tracking the change in word usage over time. Thus, given the corpus $\mathcal{C}$ collected over a time period T, we divide the corpus into n equally sized time-frames, and indicate by $\mathcal{C}_t$ the corpus slice consisting of all documents in the time-frame

t, where $T = \{T_t\}_{t=1}^{n}$. For each word $w \in \mathcal{V}$, where $\mathcal{V}$ is the vocabulary of the corpus $\mathcal{C}$, we therefore construct a time series $\mathcal{T}(w) = \{\mathcal{T}_t(w)\}_{t=1}^{n}$, where each $\mathcal{T}_t(w)$ is defined as follow:

$$\mathcal{T}_t(w) = \frac{\#(w \in \mathcal{C}_t)}{|\mathcal{C}_t|}, \tag{4.3}$$

that is, the number of times the word w occur in the corpus $\mathcal{C}_t$, normalized by the number of documents in $\mathcal{C}_t$.

Figures 4.5, 4.6 and 4.7 show different patterns of word usage over time, which have been captured by using time series of frequencies. More specifically, we investigate three patterns in word usage:

- **Positive trends**. We define as *positive trend* (fig. 4.5) any word for which we observe a pattern of increasing number of occurrences (i.e. *popularity*) over the whole time period.

- **Negative trends**. Similarly, we define as *negative trend* (fig. 4.6) any word for which we observe a pattern of decreasing number of occurrences over the whole time period.

- **Emerging trends**. Finally, we define as *emerging trend* (fig. 4.7) any word for which there are no occurrences before a time-frame t, with $t < n$, and an increasing number of occurrences is observed over the period starting from t.

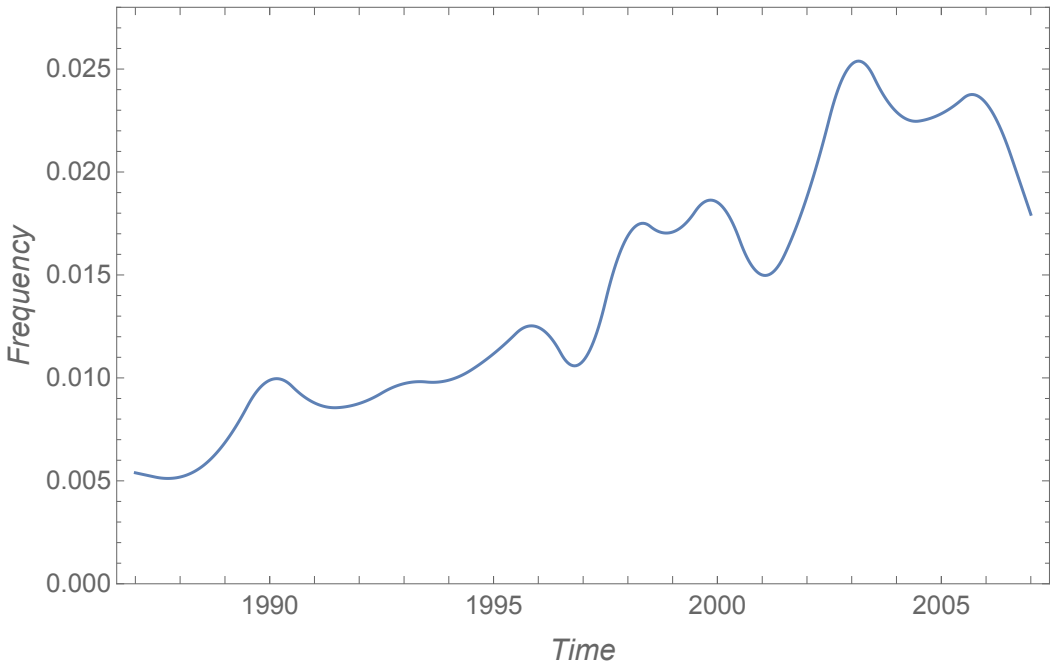


Fig. 4.5.: Positive trend: CD. Figure shows an example of positive trend. The word "*cd*" presents an overall increasing trend over the whole time period of the corpus (i.e., from 1987 to 2007).

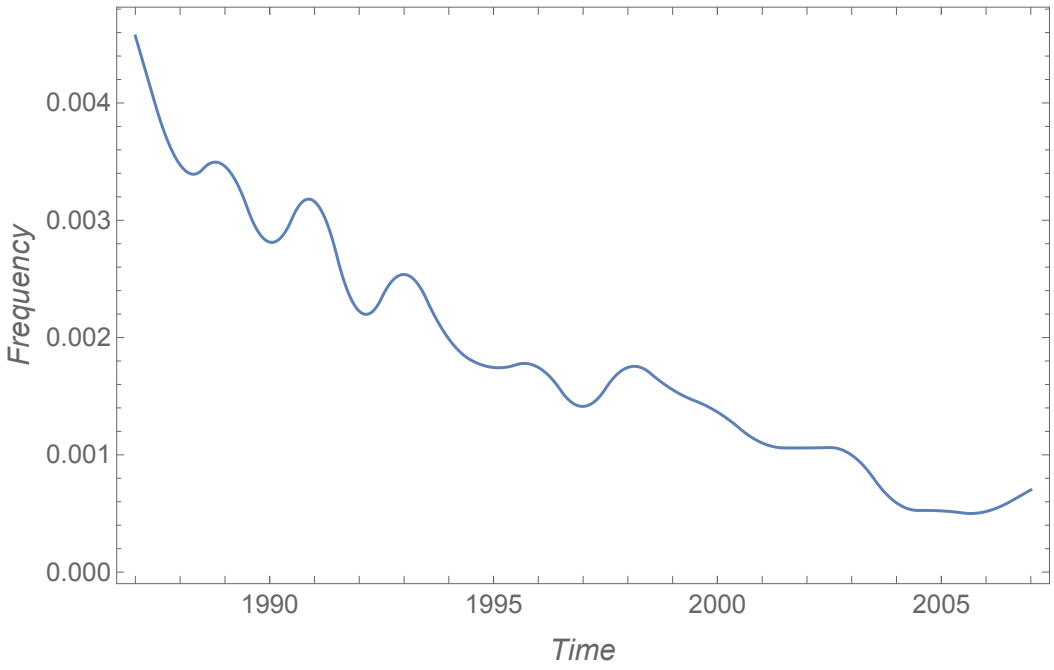


Fig. 4.6.: Negative Trend: Cassette. Figure shows an example of negative trend. The word "*cassette*" presents an overall decreasing trend over the whole time period of the corpus (i.e., from 1987 to 2007).

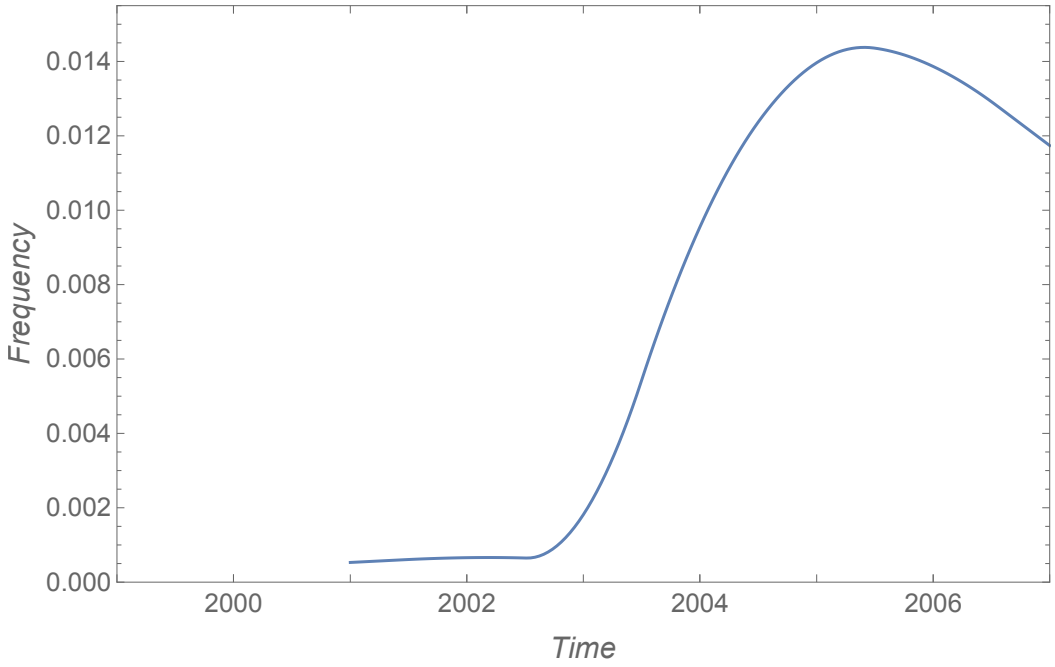


Fig. 4.7.: Emerging Trend: iPod. Figure shows an example of emerging trend. The word "*ipod*" has no occurrences before 2001 (i.e., the release date of the first iPod). After that, it presents an overall increasing trend (i.e., from 2001 to 2007).

5

Temporal Analysis of Semantics

In this chapter, we illustrate our approach to investigate semantic changes over time. In particular, we focus our attention to word embeddings, which are word representations computed using neural networks. Word embeddings are an extremely useful method for semantic analyses, indeed they transform words into vectors which encode their meanings. This enables answering descriptive questions through vector-based mathematical operations. For example, the concept *capital of Italy* can be transformed into something similar to $W['italy'] + W['capital']$, that is summing the vector representations of the words "*Italy*" and "*capital*" respectively, which should results in a vector similar to $W['rome']$, that is the vector representing "*Rome*".

Given the utmost importance of word embeddings for our text mining analysis, the first part of this chapter is devoted to describing how neural networks derive these word representations from text corpora. However, we are interested in temporal analysis of semantics, as we intend to investigate, in particular, the changes in word meanings over time. For this reason, in the second part of the chapter, we introduce temporal word embeddings models. As we will see, unlike traditional models for word embeddings, these models are able to capture shifts in semantics over time. Specifically, this chapter is structured as follow:

- **Learning Word Embeddings**. This section will present the state-of-art neural network model used to produce word embeddings, giving details about the model architecture and functioning. Furthermore, as we will see, learning word embeddings can be a very computationally expansive process. In this respect, the main optimization methods in order to efficiently learn word embeddings will be discussed. Finally, some of the major operations we can perform on the learned word embeddings will be illustrated.

- **Temporal Word Embeddings**. This section is the core of this thesis. We will illustrate how to mine historical knowledge from text corpora collected over time, by defining a temporal implementation scheme for word embeddings models. Finally, we will discuss how to use some features of this temporal model to obtain interesting results.

5.1 Learning Word Embeddings

Learning word embeddings means training a neural network to capture semantics relationships between words or phrases (i.e. n-grams) from the raw text, and encode these relationships into dense, low dimensional, vectors of real numbers.

Very efficient models for word embeddings[43] consist of a simple neural network with a single hidden layer. This neural network is trained in order to perform a language model task, that is, given a set of *target* and *context* word pairs, the goal of the training process is to maximize the conditional probability of these pairs. In fact, setting up this learning problem is not meant for the neural network to do well in producing the output probabilities. Instead, the goal is actually to learn the weights of the hidden layer, which are the vector representations of words that we define as word embeddings, and that actually represent a byproduct of the training process.

Statistical language models define probability distributions over sequences of words[5]. More specifically, such models approximate the probability of observing the *i-th* word given the context history of the preceding $i-1$ words, by the probability of observing it in a shortened context history of the preceding n words. To contrast, neural models such Skip-Gram and CBOW[43] (i.e. the *Word2vec* family of models which is used in this thesis) are trained to perform a slightly different task. To illustrate, instead of the *contexts* being the last n words immediately before the *target* word, consider the *context* to be a randomly chosen word, and the *target* to be another randomly chosen word within a *window* of n words around the *context*. At this point, given the input context word, we want the model to predict what is the randomly chosen target word. This can be defined as a *supervised learning problem* in which sentences, as they appear in the corpus, are used as training samples: for each input word, words within the context window are treated as supervision signals with which to train the neural network.

By sliding a fixed-size context window along the text corpus, it is possible to feed the neural network training samples composed of word pairs (context word and target word). In this way, the model learns the statistics from the

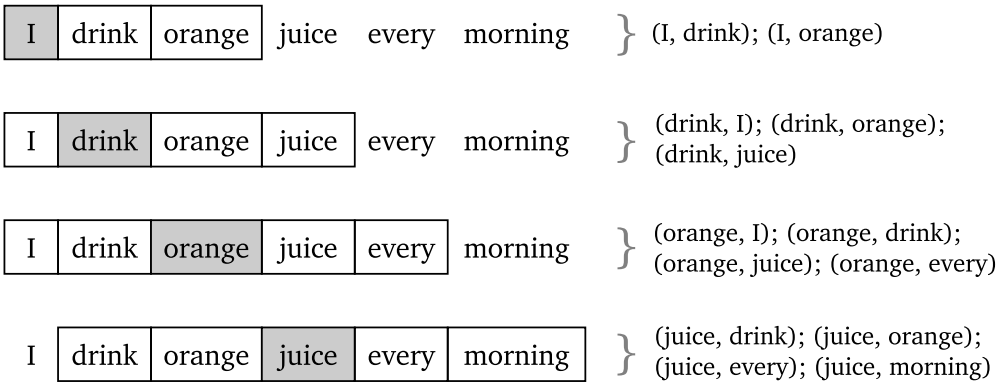


Fig. 5.1.: Training Samples. An example of how training samples are generated from raw text. In the figure, on le left, a context window of size 2 around the target word is slided over the sentece "*I drink orange juice every morning*"; for each step the target-context pairs on the right are produced.

number of times each pairing occurs, capturing the information about how likely it is to find a word in a certain context. Intuitively, there will be more training samples of "*drink*" and "*juice*", or "*play*" and "*football*", than "*drink*" and "*football*". Finally, for each input word, the neural network produces the probability for every word in the vocabulary of being the target.

5.1.1 Neural Network Architecture

The present paragraph provides an overview of the architectural components of the Word2vec models. These models consist of a simple neural network with three layers:

- **Input layer**. A layer containing as many neurons as there are words in the corpus vocabulary.

- **Hidden layer**. The number of neurons contained in the hidden layer defines the dimensionality (i.e., the number of features) of the resulting word embeddings. It is usually chosen to range from 50 to 300.

- **Output layer**. A layer containing the same number of neurons as the input layer (i.e., the number of words in the vocabulary).

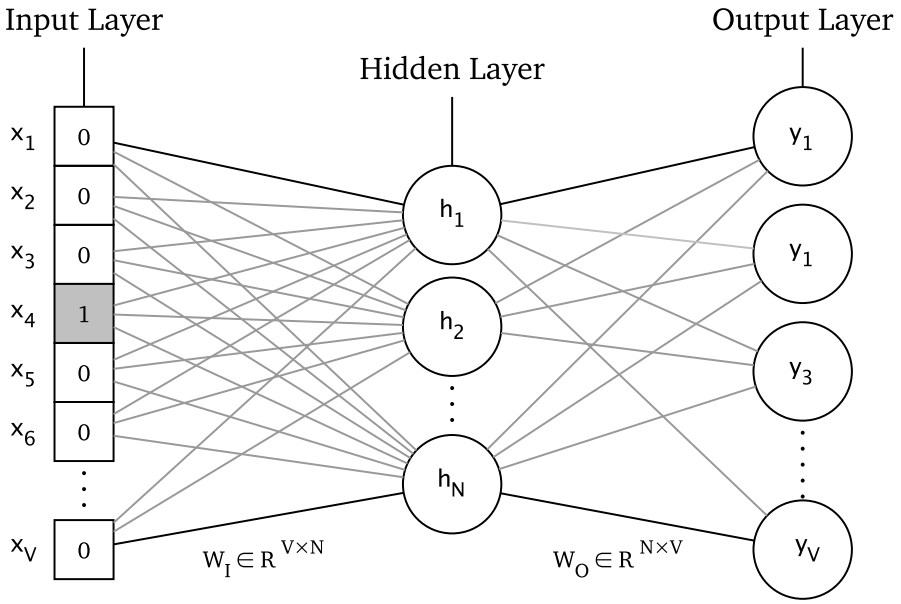


Fig. 5.2.: A Neural Network for Word Embeddings. The figure shows the neural network with its three layers: the input layer that takes as input the one-hot vector, the hidden layer where the vector representation is projected, and the output layer which produces the probability distribution. The connection between the layers are the input and output weight matrices, which are updated during the training process.

During the training process, the input word is presented to the neural network in form of a *one-hot encoding*. This means that, given a vocabulary of size V (i.e., the number of different words in the corpus), the word is encoded as a vector X of size of V, with a *1* at the position corresponding to the word itself, and *zeros* everywhere else. The input to the hidden layer is represented by an input matrix W_I of size $V \times N$, where N is the chosen number of features of the word vectors. We note, moreover, that each row of the matrix W_I is the actual vector representation of word. Yet the hidden layer is connected to the output layer by an output matrix W_O of size $N \times V$. Each column of the W_O matrix represents a word from the given vocabulary.

Before training begins, the matrices W_I and W_O are initialized by using small random values. At this point, the forward pass from the input to hidden layer is a vector by matrix multiplication. We know that by multiplying the one-hot vector X, with *1* at the *i-th* position, by the W_I matrix, the *i-th* row of W_I is effectively selected. The hidden layer acts, therefore, as a lookup table.

A similar calculation is done for the hidden layer and output layer. Specifically, each output neuron has a weight vector, which is a column of the W_O matrix. The weight vector is then multiplied by the word vector from the hidden layer. This produces a score u_j for each word in the vocabulary.

Finally, the model uses *softmax*, which is a log-linear classifier, in order to obtain the posterior distribution of words in the output layer. The softmax function is therefore used for converting values of the output layer into probabilities. More precisely, the output y_j of the *j-th* unit of the output layer is computed by the following equation:

$$p(word_j|word_I) = y_j = \frac{\exp(u_j)}{\sum_{k=1}^{V} \exp(u_k)}. \tag{5.1}$$

Recall that the training objective is to maximize the conditional probability of observing the target word given the input context word, we need to compute the prediction error for the training sample. This is done by subtracting the probability vector from the target vector. Given the prediction error vector, the weight values of W_I and W_O can be adjusted according to that. The technique of propagating errors back in the network and readjusting the weight values of matrices is called *backpropagation*. In this way, the training continues by taking different context-target word pairs from the corpus. This is precisely the core of the learning process, during which the neural network is trained in order to make increasingly better predictions. In particular, in learning word embeddings, if two different words have similar context, the model needs to produce similar results for these two words. This is possible by keeping the two vector representations similar in the weight matrices. Hence, similar words will have similar word vector representations.

5.1.2 Word2vec Models

In fact, the Word2vec family of models presents two alternative ways of learning word embeddings. These are *Continuous Bag of Word* and *Skip-gram* models[43][44]:

- **Continuous Bag of Word (CBOW)**. The Continuous Bag of Words model basically works as just described, except that context is represented by multiple words for a given target word. The architecture of the neural network is slightly different, in particular for the input layer. Instead of directly copying the input vector context word, the CBOW model takes the average of the vectors of the input context words[55]. The update equations remain the same, except that we need to update every input word vector of the context.

- **Skip-gram (SG)**. The Skip-gram is the opposite of the CBOW model. Here, the target word is at the input layer, and the context words at the output. Therefore, the network is trained to predict contexts given a target word. As regards the architecture of the skip-gram, instead of producing one multinomial distribution on the output layer, the model produces C multinomial distributions, where C is the size of the window around the target word. The output is composed by a sequence of C panels, each one containing a multinomial distribution over the vocabulary. Each output panel shares the same weight matrix. Specifically, given the input word $word_I$ and the *c-th* output word $word_{O_c}$ of the C contexts, the output $y_{c,j}$ for the unit j at the panel c is computed by the following equation:

$$p(word_{c,j} = word_{O,c}|word_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{k=1}^{V} \exp(u_k)} \tag{5.2}$$

  The error vectors, from all the panels in the output layer, are then summed up in order to update the weights via backpropagation.

5.1.3 Model Optimization

As we have seen, both CBOW and SG models learn two vector representations for each word: the input vector and the output vector. In particular, in order to update the output vector, for each single training instance, the model needs to produce an output score, a probability prediction, and a prediction error, for every word in the vocabulary. This makes it virtually impossible to compute such calculations for every training instance in real large corpora[55]. To solve this problem, two main different optimization methods have been

proposed in the literature. Both the approaches limit the number of output vectors that must be updated per training instance. These are *hierarchical softmax*[48][47] and *negative sampling*[43]:

- **Hierarchical Softmax**. Hierarchical Softmax uses a binary tree to represent all words in the vocabulary. In particular, each of the V words in the vocabulary is a leaf node of the tree. By definition, there are $V-1$ inner nodes in the tree. Moreover, for each leaf node, there exists a unique path from the root to the node. In this model there is no output vector. Instead, the probability of a word w being the output word is defined as the probability of a random walk starting from the root, and ending at the leaf w. Each inner node has an output vector, which is used to determine the probability of follow the left child and follow the right child in the random walk. By using the hierarchical softmax, the computational complexity per training instance per context word is reduced from $O(V)$ to $O(\log(V))$[55].

- **Negative Sampling**. Negative Sampling was introduced by Mikolov et al.[43] as an efficient method for updating output vectors. The basic idea is instead to update all the output vectors, we only update a sample of them. Specifically, the sample is composed by the output word, which is the *positive* word, and a few other words randomly selected, which are the *negative* words. In particular, a negative word is one for which we want the network to output a 0 for. With negative sampling, only the weights for the words in the sample should be updated. The number of selected samples should range from 5 to 20 for small dataset, and from 2 to 5 for large dataset[43]. The negative samples are selected using a probability distribution related to the frequencies of the words. Essentially, more frequent words are more likely to be selected as negative samples. More precisely, the probability $p(w_i)$ of selecting a word w_i as a negative is given by the follow equation:

$$p(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n}\left(f(w_i)^{3/4}\right)}, \tag{5.3}$$

  where the frequency is raised to the $3/4$ power for the empirical reason that it outperforms other functions[43].

5.1.4 Word Embeddings Properties

Word embeddings lead to a significant improvement of natural language understanding, since very meaningful text analyses can be obtained by performing simple mathematical operations on word vectors.

The main property of word embeddings is that similar words tend to have similar vectors. Thus, defining a similarity function over vectors, allows to compute the similarity between words, and it is at the core of semantic analysis. A common and effective choice for similarity between vectors is the *cosine similarity*, corresponding to the cosine of the angle between the vectors. The cosine similarity between two vector w_1 and w_2 is defined as follow:

$$\text{sim}_{\cos}(w_1, w_2) = \frac{w_1 \cdot w_2}{\|w_1\|_2 \|w_2\|_2} \tag{5.4}$$

It is common to normalize vectors to have a unit length. When using unit vectors w_1 and w_2 (i.e., $\|w_1\|_2 = \|w_2\|_2 = 1$), the cosine similarity reduces to a dot-product, which can be computed very efficiently:

$$\text{sim}_{\cos}(w_1, w_2) = w_1 \cdot w_2 = \sum_i w_1[i] w_2[i] \tag{5.5}$$

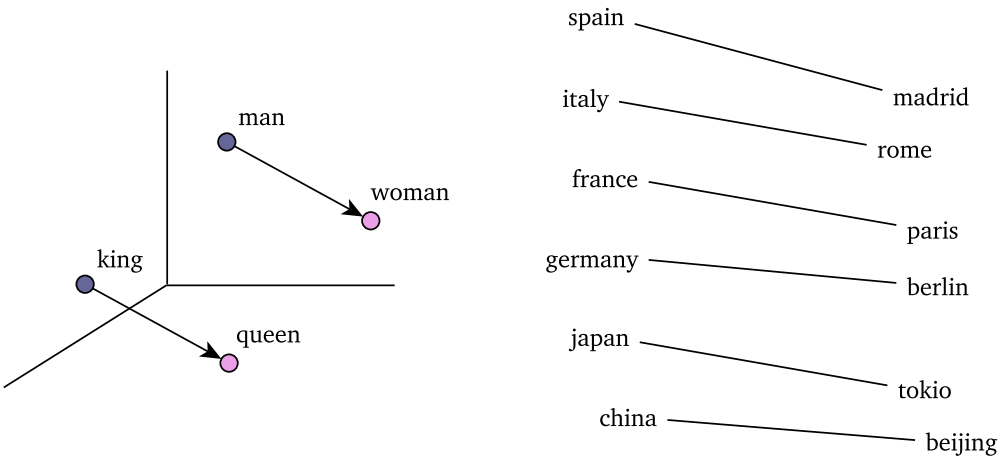
Some of the major operations on word embeddings include clustering words, finding similar words, and computing word analogies:

- **Word Clustering**. Word embeddings can be easily classified into different groups using clustering algorithms such as *KMeans*[22]. In this case, position vectors within the high dimensional space gives a good indication of the semantic class which the words belong to. In order to visualize vectors and clusters into a low dimensional space (i.e., 2D or 3D), dimensionality reduction algorithms such as *t-SNE*[37] can be used.

- **Finding Similar Words**. Often, it is useful to find the *k* most similar words to a given word. For example, we can compute the 5-most similar words to "*iPhone*", and obtain "*iPad*", "*iPod*", "*iOS*", and "*Apple*" as results. Specifically, given the word embedding model W (i.e., the $V \times N$ embeddings matrix of all the *N*-dimension word vectors) and a

word vector w, we define as $W.\texttt{k-similar}(w)$ a method on the model W which returns the *k* most similar words to w. This can be done simply by a matrix-vector multiplication. In particular, given the embedding matrix W and $w = W[i]$ the vector representation of a word i, $s = Ww$ is the vector of similarities between i and all the words represented in the matrix. In this way, the *k* most similar words correspond to the indices of the *k* highest values in s. Such matrix-vector multiplication are highly optimized in modern scientific computing library, and they can be executed very efficiently for embedding matrices with hundred of thousands of vectors.

- **Computing Word Analogies**. A very interesting property of word embeddings, which definitely contributed to the their popularity, is that these representations are surprisingly good at capturing syntactic and semantic regularities in language[45]. In particular, one can perform "algebra" on word vectors and get meaningful results. For example, one can take the vector of "*Germany*", add the vector "*capital*" and get "*Berlin*" as the closest vector to the result. Even more surprisingly, "*King*" - "*Man*" + "*Woman*" results in a vector very close to "*Queen*". This, in particular, allows vector-oriented reasoning based on the offsets between words, such as: $w_{Italy} - w_{Rome} + w_{Germany} \approx w_{Berlin}$.



Male-Female Country-Capital

Fig. 5.3.: Word relationships. The figure is a conceptual representation of the syntactic and semantic regularities captured by word embeddings models.

5.2 Temporal Word Embeddings

The word embeddings models we have seen so far need a huge quantity of text data in order to learn good representations of word meanings. Text data in large document collections and web archives, which are the main resources for text mining models, are often accumulated over time. Documents archived in these datasets cover at least decades, sometimes even centuries. Whilst words keep changing their meanings with time, during the course of human language and cultural evolution, archives play an important role in preserving the cultural heritage[7]. Corpora indeed, especially newspapers and blogs, implicitly encode the change of semantics over time, as new concepts and phenomenons developing and old ones dying. Over the years, brands and technologies change, and politicians who have played important roles at a specific time are replaced.

Word embeddings models, in their classic version, are static and unable to capture this evolution of terminology. Indeed, the "knowledge" learned by these models is *time-agnostic*, and aggregated over the whole time period.

Motivated by the continued growth of research on word embeddings, in this thesis, we are interested in developing *temporal word embeddings*. Specifically, we focus on extending traditional models by adding a time dimension to vector representations. This allow us to perform a temporal analysis of semantics, by studying the evolution of word meanings across time, and exploring word associations in different temporal contexts.

In order to obtain time-aware word embeddings, we adopt the following approach:

1. We divide a corpus into different time frames (e.g., years), and train the model for each one of them. In this way, we obtain different vector spaces (one for each time frame), so that words in different time frames have a different vector representation.

2. We recombine the vector spaces into a single model, which now differentiates word vector representations with respect to the time.

5.2.1 Training Word Vectors across Time

We consider a text corpus that has been collected over a time span T. In particular, we are interested in document collections whose documents include publication time-stamps (e.g., newspapers, social network discussions, etc.). We divide the corpus into n equally sized chronologically ordered time-frames. More precisely, we indicate by $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_n)$ the corpus, where each $\mathcal{C}_t, t = 1, \dots, n$ is the corpus of all documents in the *t-th* time frame. Here, the size of the frames is an important factor to consider: obviously, time-stamp informations determine which time granularity options are available, and they much depend on the type of corpus. For example, newspapers provide published dates with a day granularity, while a social network post may include a time-stamp which is expressed in seconds. In this thesis, however, we are not interested in very fine-grained time-slices (i.e., day granularity, or lower), and there are two main reasons for this:

- **Data sparsity**. As we discussed, models for word embeddings need a large amount of data to produce reliable word representations. By splitting the corpus into very small slices, we could feed the neural network too little training data. This will result in inexpressive word representations.

- **Model complexity**. Another reason is that, the model we propose does not scale well with small granularities. This is due to the difficulty of dividing the corpus into a large number of slices. In this case, further investigations are needed, in order to find solutions which can provide temporal contexts to the model, without the need of dividing corpora into different time frames.

We found that, when dealing with newspaper corpora (as we do in this thesis), a time granularity of years is a good compromise between the quality and complexity of the model.

Next, for each corpus $\mathcal{C}_t$, we update a vocabulary $\mathcal{V}_t = \{\mathcal{V}_t \cup \mathcal{V}_{t-1} \cup \dots \cup \mathcal{V}_1\}$, which is the set of all words in the corpus occurred in the period between the time frames 1 and t. The overall vocabulary $\mathcal{V}$ is therefore defined as the updated vocabulary $\mathcal{V}_t$ at the time $t = n$ (i.e., at the last time frame). Words

which are not found in the vocabulary $\mathcal{V}_{t_i}$, and that exist in the vocabulary $\mathcal{V}_{t_j}$, with $t_j > t_i$, are all the *emerging words* that appear at a certain time t_j. In this respect, our model behaves like a classic word embedding model: they both accumulate "knowledge" over the history; the difference is that our temporal model also keeps track of the intermediate stages of learning. On the other hand, some words in the corpus disappear before the end of the time interval T. Such words are not directly distinguishable with our model. However, a disappearing word can be easily detected by its frequency count.

Thus, given the time-structured corpus $\mathcal{C}$, we seek to learn a dense, low-dimensional, vector representation (i.e, word embedding) $u_w(t) \in \mathbb{R}^d$ for each word $w \in \mathcal{V}_t$ and each time frame $t = 1, \ldots, n$. Specifically, given a time-frame $t = 1, \ldots, n$, our temporal word embeddings model $\text{TWE}_t \mapsto \mathbb{R}^{|\mathcal{V}_t| \times d}$ is defined as the word embeddings matrix whose rows are all the word representations $u_w(t)$ for each word $w \in \mathcal{V}_t$.

In order to learn the temporal embedding matrix TWE_t, we train a neural network to maximize the conditional probability defined in 5.2, for each word pair target-contexts, within a certain window, in the corpus $\mathcal{C}_t$. To this end, an optimization method between negative sampling and hierarchical softmax is required. This training process is carried out for each corpus $\mathcal{C}_t, t = 1, \ldots, n$.

5.2.2 Aligning Word Vectors across Time

Having trained the model for each corpus $\mathcal{C}_t, t = 1, \ldots, n$, we must now take into account the fact that temporal word representations we obtained are actually vectors across independent vector space models, and they are not in one unified coordinate system. This is due to the stochastic nature of the training process, which randomly initialize the weight matrices in the neural network. This, in particular, implies that word embedding models trained on exactly the same data yield vector spaces in which relative distances between vectors are the same across spaces, but positions (i.e., coordinates) are different. Hence, comparing vector representations of the same word in

different independent vector spaces yields inconsistent results. This poses the main challenge to work with temporal word embedding models.

Indeed, in order to use temporal word embeddings matrices in one unified temporal model, embeddings across different vector spaces, must be aligned.

The *alignment problem* can be solved by following three different approaches:

- **Non-random initialization**. A very straightforward method for solving the alignment problem, which is used by Kim et al.[28], is to not randomly initialize the values for the weights matrices in the neural network. Instead, a model is initialized with the values obtained from the training of the previous model. In this thesis, we follow this approach. Specifically, given respectively the end and the start of two consecutive training processes, $[t-1]_{\text{e}}$ and t_{s}, and the input and output weights matrices, $W_I(t)$ and $W_O(t)$, for the training t, we indicate as $W_I(t_{\text{s}}) := W_I([t-1]_{\text{e}})$ and $W_O(t_{\text{s}}) := W_O([t-1]_{\text{s}})$, the initializations for of each training $t, t = 2, \ldots, n$.

- **Piecewise linear regression model**. Another approach, which was used by Kulkarni et al.[31], is to find an alignment that minimize the distance between two vectors of the same given word, in two different spaces, while preserving the local structure of the word. Specifically, this means learning a linear transformation $W_{t' \mapsto t}(w)$ for the vector $u_{t'}(w)$ of the word w, by solving the following optimization problem:

$$W(w)_{t' \mapsto t} = \arg\min_{W} \sum_{w_i \in k\text{-NN}(u_{t'}(w))} \|u_{t'}(w_i)W - u_t(w_i)\|_2^2, \tag{5.6}$$

  where t' and t are the two different vector spaces, k-NN$(u_{t'}(w))$ are the set of k nearest words of w, in the vector space t', which is the local structure to be preserved. Moreover, with this method, a possibly semantic shift can be detected when the alignment model fails to align a word properly. However, the main drawback of this approach is that an alignment (i.e., a piecewise linear regression) must be found separately for each word.

- **Orthogonal Procrustes problem**. Finally, the alignment problem can be treated as an *Orthogonal Procrustes problem*. The Orthogonal Procrustes problem is defined as the problem of finding an orthogonal transformation matrix T, which transform a given matrix A into a given matrix B, by minimizing the sums of squares of the residual matrix $E = AT - B$ [56]. This approach, used by Hamilton et al.[20], is similar to the linear regression method, with the difference that it is applied globally to the entire vector space.

As we mentioned, we choose to follow the non-random initialization approach, and this mainly for its simplicity and easy implementation. Nevertheless, we found relatively little literature addressing this issue. We believe that further investigations, to clarify advantages and disadvantages for all the above mentioned methods, would be a significant contribution to future works on temporal word embedding models.

5.2.3 Temporal Models from Word Embeddings

Having defined how to train our temporal model, it is now useful to summarize its basic features. In particular, a temporal word embeddings model allows comparing same words across different time periods. This is possible by querying the model for different time-frames, in order to obtain specific results for the desired period. Specifically, we consider the temporal model TWE_t which takes as parameter one of the different time-frames t (i.e., $t = 1, \ldots, n$), and provides the following operations:

- **Discovering different contexts across time**. This is similar to finding similar words (i.e., words used in the same context) in static models, with the difference that this time, the search of similar words is restricted to a particular time period. Specifically, given $w = \text{TWE}_t(i)$, the vector of the word i at the time t, we define as $\text{TWE}_t.\texttt{k-similar}(w)$ a method on the temporal model TWE which returns the *k* most similar words to the word vector w, in the time period $t, t = 1, \ldots, n$. This is very useful in order to study semantic evolution of words, and see how these words change their contexts over time. For example, as we know, during the decades, the word "*internet*" has assumed several

meanings. Indeed, by comparing the results provided by our temporal model for different time periods, we can see that, initially, internet was associated with words such as "*computer networks*" and "*electronic mail*", and then it gradually come closer to words such as "*social networking*" and "*mobile phones*".

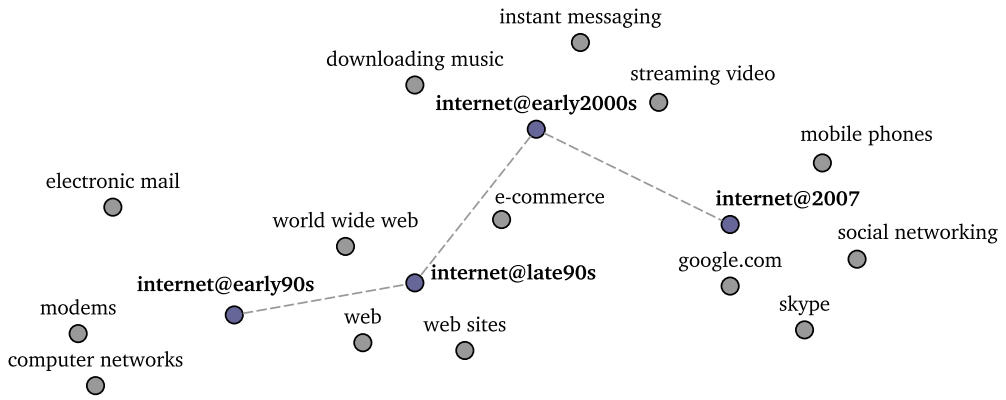


Fig. 5.4.: Semantic evolution over time. The example is a conceptual representation that shows the vector for "*internet*" as it moves through the space at different time periods. This reflects how our use of the Internet has evolved in the years, as we mean different things in different temporal contexts.

- **Discovering temporal word analogies**. Another interesting feature of such temporal models is that they can provide answers to questions such as: What was the counterpart to *iPod* in the nineties? In other words, we are interested in finding terms which are semantically similar with respect to different time contexts. This can be very useful in order to investigate cultural evolution, by identifying concepts which were first indicated by words actively used in the past (e.g., *walkman*), and that now have their current counterparts (e.g., *ipod*). This feature, in particular, highlights the generalization power of word embeddings, which encode analogies between words that do not directly co-occur along the corpus, but still have similar contexts. Furthermore, temporal word embeddings models can tackle the problem of bridging the terminology gap in data archives[7]. Formally, given a source time t_s and a target time t_t, we define the *temporal counterpart* in t_t, of a word i in t_s,

$\text{TC}(i)_{t_s \mapsto t_t}$, as a mapping from the vector space of the source time t_s to the vector space of the target time t_t. Specifically:

$$\text{TC}(i)_{t_s \mapsto t_t} \simeq \text{TWE}_{t_t}.\text{1-similar}(\text{TWE}_{t_s}(i)), \tag{5.7}$$

that is the computation of the most similar vector to $\text{TWE}_{t_s}(i)$, in the model TWE_{t_t}.

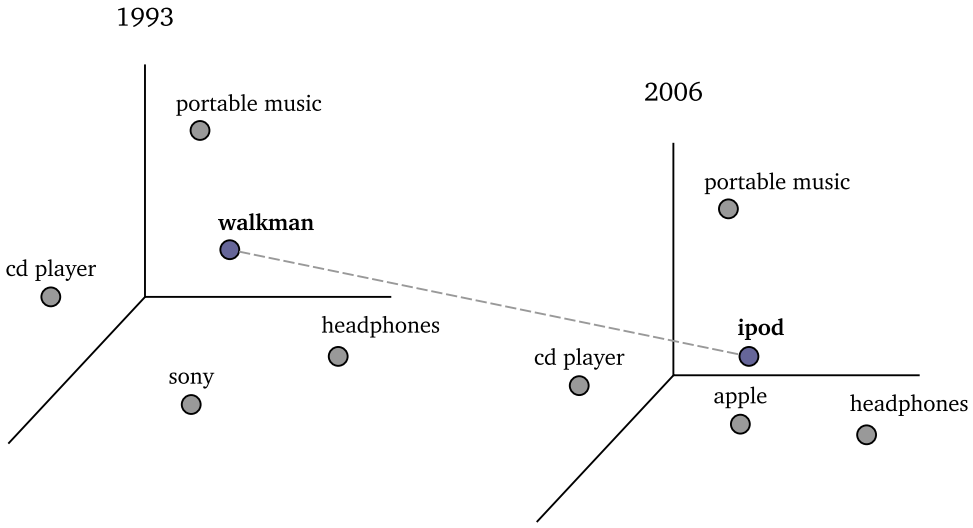


Fig. 5.5.: Temporal Word analogies. The figure shows a conceptual representation of a mapping between two different vector spaces, representing two different time periods. The example illustrates the analogy between the word "*ipod*" and "*walkman*" across the years 2006 and 1993. Indeed, these two words refer to a similar concept, and they still have similar contexts in two totally different time periods.

5.2.4 Time Series from Word Embeddings

To quantify the change in semantics across time, we need to compare vectors of the same word in different vector spaces. In particular, we compute the similarity between the vector of a given word i at a certain time t, and the vector of the same word i at the time $t-1$. We then build a time series $\mathcal{T}(i) = \{\mathcal{T}_t(i)\}$, where $\mathcal{T}_t(i)$ is the measure of the semantic change for the

word i at the time t, with the respect to the previous time period, $t-1$, for each time periods $t, t = 2, \ldots, n$. Specifically, $\mathcal{T}_t(i)$ is defined as follow:

$$\mathcal{T}_t(i) = \frac{\text{TWE}_t(i) \cdot \text{TWE}_{t-1}(i)}{\|\text{TWE}_t(i)\|_2 \|\text{TWE}_{t-1}(i)\|_2}, \tag{5.8}$$

that is the cosine similarity between the two vectors $\text{TWE}_t(i)$ and $\text{TWE}_{t-1}(i)$.

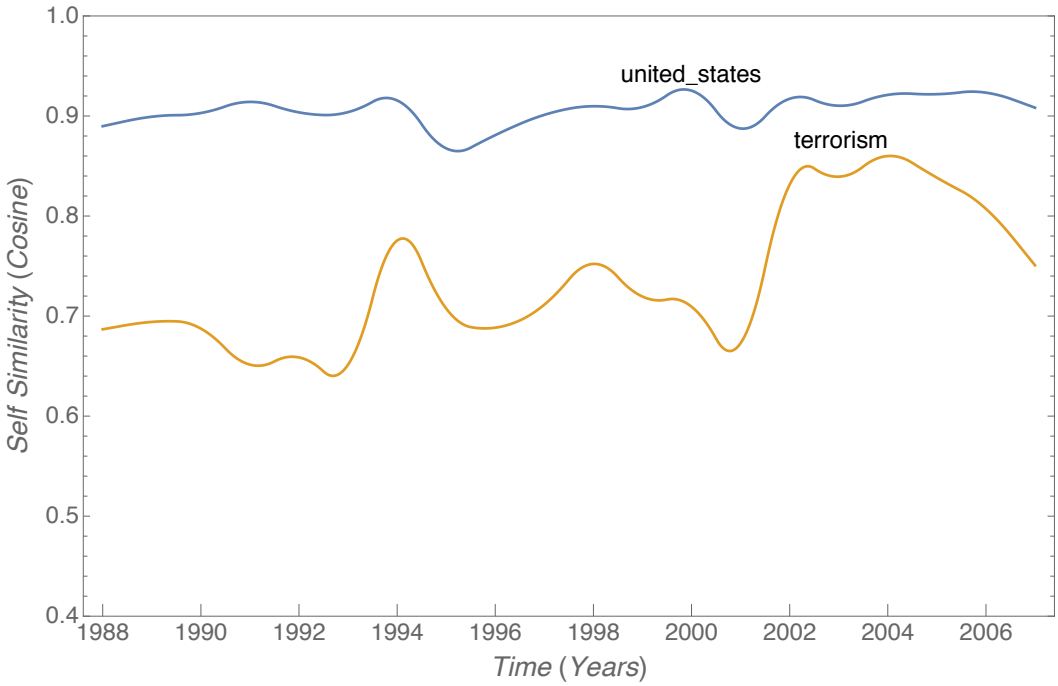


Fig. 5.6.: Self Cosine Similarity. The figure shows the time series for the words "*United States*" and "*terrorism*", resulting by applying the definition in 5.8 for each year from 1988 to 2007. As we can see,"*United States*" refer to a concept that changes little with respect to the concept of "*terrorism*". In particular, we can identify two minimums corresponding to the years of the two major terrorist attacks (i.e., 1993 and 2001). It is evident the strong shift in the meaning of "*terrorism*" experienced in those years.

6

Experimental Results

In this chapter we apply the methods presented in chapters 4 and 5 to the *New York Times Corpus*. In particular, we conducted two different types of experiments:

- **Temporal analysis on frequencies**. First, we conducted an analysis on frequencies, in order to search for words in the corpus that follow particular frequency-based trends over time.

- **Temporal analysis on word embeddings**. Subsequently, we use some of the previous extracted trends as a starting point of a more extensive semantic analysis across time, based on time-dependent word embeddings.

Before discussing our analyses, we introduce the dataset we used in the experiments.

6.1 Dataset Description

The *New York Times Corpus* contains over 1.8 million news articles published by the *New York Times* between January 1, 1987 and June 19, 2007. The New York Times (sometimes abbreviated as The NYT) is an American daily

newspaper based in New York City. It is one of the most widely circulated daily newspaper in the United States, with articles that range across many topics, such as art, business, politics, science, sports, technology and so many others.

News media data in the New York Times Corpus are provided as a collection of *XML* documents. These documents contain, in addition to the text data, metadata information about, among the others, the date of publication of the article. We built an XML Parser to extract text data from title, abstract and body of each document, while used metadata to group documents by their date of publication. In particular, to perform our temporal text mining analyses, we grouped documents by their year of publication. Table 6.1 gives details about the number of documents by year of publication.

Year	# Documents
1987	106104
1988	104541
1989	102918
1990	98812
1991	85135
1992	82685
1993	79200
1994	74925
1995	85392
1996	79077
1997	85625
1998	89391
1999	91310
2000	94497
2001	96282
2002	97258
2003	94235
2004	91362
2005	90004
2006	87052
2007	39953

Tab. 6.1.: The New York Times Corpus: Number of documents by year.

6.2 Temporal Analysis on Frequencies

We started our analysis by searching for trends in word popularity over time. In particular, we quantified the frequencies of word usage for each year of the NYT corpus. We argue that, the search for patterns in word frequencies is a good a starting point for a more comprehensive modeling of semantic evolution over time.

For this purpose, we first preprocessed our text data, by tokenizing and lemmatizing the content of each document in the collection. We proceeded to removing tokens belonging to a stop-words list. It is important to note that tokens can not be quantified as word occurrences. In fact, this first phase is useful in order to create a n-gram detection model. Indeed, n-grams of interest, i.e. phrases, can be detected only by first collecting word co-occurrence statistics over the whole corpus. In this respect, after having transformed the corpus sentences into lists of tokens, we put all of these lists into a single large file, in order to train our model to detect phrases within word sequences. Thus, by using our n-grams detection model, we was able to transform text documents into lists of words (i.e. 1-grams, 2-grams and 3-grams of interest). For each document, we extracted the day of publication from metadata, and stored every occurrence of each word into a table, together with the date of publication. We also included a reference ID to the document. Eventually, the word-date occurrences table could look as the follow:

Word	Date	Document ID
...		
new_york	19870623	00001385
new_york	19870624	00001386
new_york	19870624	00001386
new_york	19870624	00001387
...		
restaurant	19870623	00001385
...		

At this stage, rows were aggregated by word and date, in order to obtain the occurrences count for each word at each day of the corpus. This was performed efficiently, by using a relational DBMS with hashing indices on both word and date fields, and the aggregation function *count()*.

At this point, we used these data to create a time series for each word with a granularity of one day over the whole corpus time-span (i.e., from January 1, 1987 to June 19, 2007). This results in a very large number of time series, i.e., 2.7 million time series. In fact, at this stage we collected every symbol (i.e., 1-grams, 2-grams and 3-grams). In this respect, we decided to remove each symbol occurring less than 5 times, which we indicated as a probable misspelled word.

Subsequently, we searched for three types of trend in our time series:

- **Positive trends**. Words that are observed across the whole corpus, and whose usage has increased over time, can be defined as positive trends. Positive trends can be due to a semantic shift over time.

- **Negative trends**. Words that are observed across the whole corpus, and whose usage has decreased over time, can be defined as negative trends. Negative trends can be due to a word replacement over time.

- **Emerging trends**. Words of which there are no observations within the first half of the corpus, and whose usage has increased over time since their first occurrence, can be defined as emerging trends.

In order to search for these trends in our data, we proceeded to performing a further selection of n-grams based on their overall daily frequency. In particular, we selected n-grams that occurred at least 10% of all days within the whole corpus time-span, in order to search for positive and negative trends. Similarly, we selected n-grams that occurred at least 10% of all days within the second half of the corpus time-span, in order to search for emerging trends.

Trend detection has a long history in literature; however, in this thesis we followed a very straightforward approach: we used the linear regression as a natural way to model a trend and its trajectory. Specifically:

1. We first normalized our time series such that values ranged between 0 and 1. The following min-max scaling normalization was used for this purpose:

$$Y_{norm} = \frac{Y - Y_{min}}{Y_{max} - Y_{min}},$$

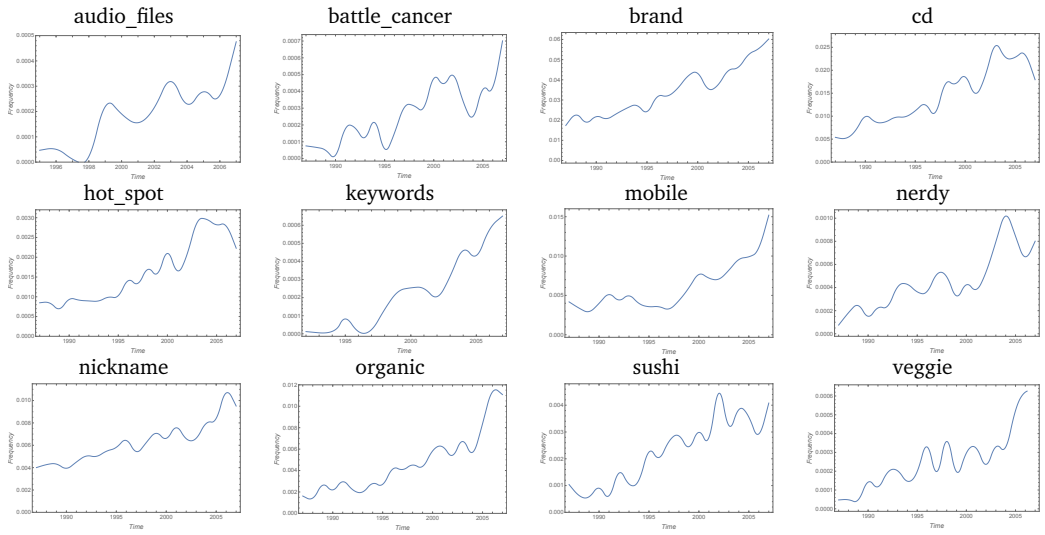
where Y is the actual data point, Y_{min} and Y_{max} are respectively the minimum and the maximum values of Y amongst all the time series data, and Y_{norm} are the normalized value for Y.

2. For each normalized time series, we fitted the data points to the line $y = \alpha + \beta x$ by using the least squares fit method. Thus, we stored the parameters and the cost of each fit.

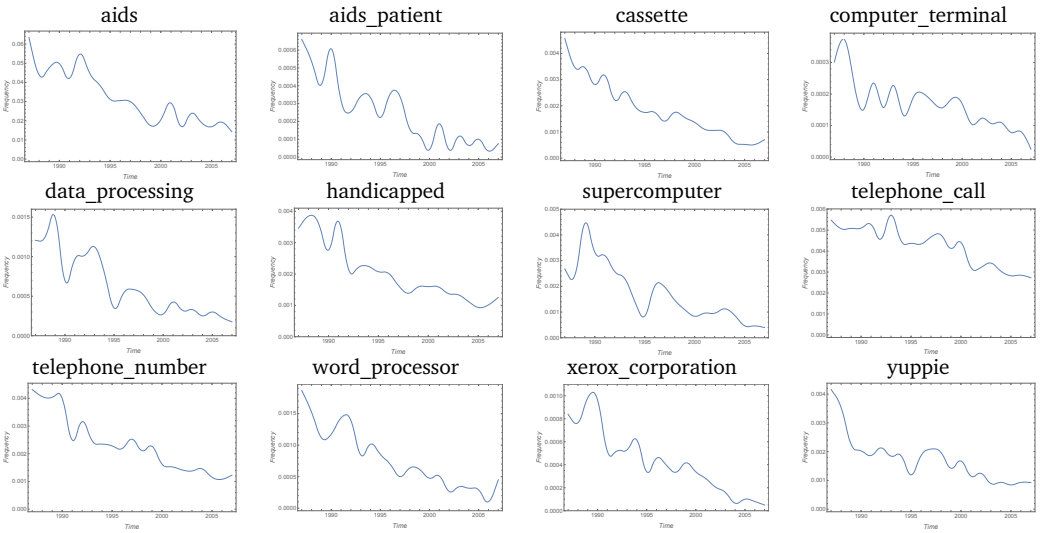
3. Finally, we selected only the time series well-represented by a linear model, that is all the time-series for which the cost of the fit was lower than a certain value. We use the sign of the parameter β (i.e., the slope) to discriminate between positive trends (i.e., $\beta > 0$) and negative trends (i.e., $\beta < 0$).

Some of the most interesting results are reported in the tables 6.2, 6.3 and 6.4. By quantifying trends in word usage we can capture important aspects of the cultural evolution. Usually, trends reflect the evolution of society through some technological development. This is the case for the trends "*audio files*" and "*cd*", which are both positive, in contrast to "*cassette*" which is negative. However, other trends reflect changes in customs and traditions, for example "*organic*", "*sushi*" and "*veggie*" reflect changes in the eating habits; others, again, reflect changes in the health situation: "*aids*" and "*aids patient*" are negative trends, where "*battle (with) cancer*" is positive. Furthermore, there are words with a positive trend in their popularity that is due to a change in their meanings over time. For example, the positive trend for the word "*hot spot*" can be explained by the semantic shift, over the years, from being just a place with a relatively higher temperature, or a place of entertainment, to being a public place where a wireless signal is made available so that the Internet can be accessed. Similarly, the words "*keywords*" and "*mobile*" have both experienced an increasing in their popularity when they have come to be used, respectively, in the terminology of search engines, and for indicating mobile devices. On the other hand, there are negative trends which indicate

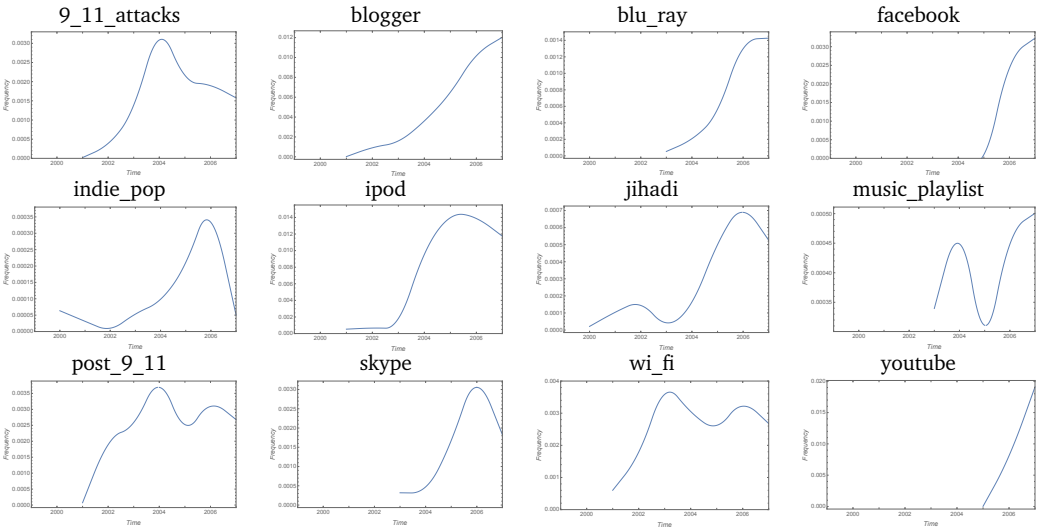
a word replacement over time. This is the case for the word "*handicapped*", that has been used less and less to indicate disabled people, since it carries some negative connotations with it. Other examples are the words "*telephone call*" and "*telephone number*", which have been replaced over the years with the more concise forms "*phone call*" and "*phone number*". Finding the word which has replaced a negative trend over time is not immediate, since often there is no clear evidence of a trend in the opposite direction. Instead, we investigated this aspect by using the temporal analogies feature of our temporal word embeddings model, in order to find future counterparts. The results of this will be illustrated in section 6.3. Finally, we present some of the emerging trends. In particular, we selected positive trends of words that did not exist before the 2000. We found some interesting emerging trends in words such as "*facebook*", "*ipod*", "*skype*" and "*youtube*", that indicate new technology products. Other interesting trends are the word "*blogger*" and "*post 9 11*", which are neologisms related to the technology evolution, and the September 11 attacks, respectively. We further investigate these trends through a temporal analysis of semantics. In particular, we found interesting results by using again our temporal analogies model, this time by searching for past counterparts. To illustrate this particular feature, we have already discussed the example of the "*walkman*" as the "*ipod*" of the nineties. We will discuss further examples in section 6.3.



Tab. 6.2.: Positive trends.



Tab. 6.3.: **Negative trends.**



Tab. 6.4.: **Emerging trends.**

6.3 Temporal Analysis on Word Embeddings

We proceeded with our analysis by training a temporal word embeddings model on the NYT corpus. In order to train our model, we used Gensim[54], which is a python framework providing several tools and models for NLP. In particular, we used the Word2vec implementation of the skip-gram model, with the negative sampling optimization. Specifically:

1. We divided the corpus into 21 slices: each slice with a time-frame of one year (from 1987 to 2007), except for the last one, corresponding to the year 2007, which had a time-frame of 6 months. Furthermore, we used the n-grams detection model to transform corpus documents into lists of n-grams.

2. We trained a word embeddings model for each slice of the corpus, by using the non-random initialization approach. In particular, we used the output embeddings of the iteration i to initialize the embeddings for the iteration $i + 1$. This allowed us to easily impose an alignment on the resulting vector spaces.

3. Finally, for each the iteration, we stored the resulting embeddings matrix. Then, we created our temporal model as a set of embeddings matrices, one for each year of the corpus.

Having trained our temporal model, we obtained time-dependent results for a particular year, by querying the model on the corresponding vector space. This allowed us to perform two different types of analysis:

- **Most similar across time**. We started by investigating how words change their meanings over time. Specifically, given a word, we searched for its nearest neighbors across every year (i.e., vector space) in which the word appeared. This is useful in order to track the semantic evolution of words. Some results are reported in table 6.5. For example, as we mentioned, the word "*internet*" has been constantly changing its meaning, reflecting how the Internet has been used over time. Indeed, the context of "*internet*" changes from the *"communication networks"* in the early 80s, to the internet services such as "*social networking*" and "*google.com*" in the late 2000, as well as the modern access devices such as "*mobile phones*". Similarly, the word "*viral*" changed its meaning, by including words related to brand and marketing in the late 2000. Furthermore, we can find the word "*hot spot*" that changed its context in the early 2000, as we discussed, after becoming a place in which a wireless connection to the Internet is provided. The latter result, in particular, confirm our intuition about why some words experienced an increasing in their popularity. On the other hand, some words, named

entities in particular, changes continuously their associations. For example the words "*white house*" and "*pope*", in table 6.6, are related, respectively, to the presidents who have succeeded to the presidency of the United States over the years, and to the popes.

<table>
<tr><th></th><th colspan="3">Word</th></tr>
<tr><th>Year</th><td>internet</td><td>hot spot</td><td>viral</td></tr>
<tr><td>1987
1988
1989
1990</td><td rowspan="2">communication networks</td><td rowspan="4">panoramic
volcanic
equator
south beach
jazz club
restaurant
brew pub</td><td rowspan="5">microbe
virus
antibodies
disease</td></tr>
<tr><td>1991
1992
1993
1994
1995</td><td>computer networks
modems
world wide web</td></tr>
<tr><td>1996
1997
1998
1999
2000</td><td>web sites
e-commerce
wireless</td></tr>
<tr><td>2001
2002
2003
2004
2005</td><td>downloading music
bittorrent
instant messaging
streaming video</td><td rowspan="2">wireless internet access
wifi</td></tr>
<tr><td>2006</td><td rowspan="2">google.com
skype
mobile phones
social networking</td><td>branded entertainment
social networking</td></tr>
<tr><td>2007</td><td>internet
marketing</td></tr>
</table>

Tab. 6.5.: Semantic evolutions.

- **Temporal Word Analogies**. We proceeded our investigation by discovering temporal word analogies. We found some temporal counterparts by calculating the similarity between word vectors which occupy a similar position into different temporal spaces (i.e., matrices of different years). In particular, given a word vector in the model of the year source, we computed its most similar vector in the model of the year target. Some results are reported in tables 6.8 and 6.7. As starting point, we considered some words which we indicated as negative trends in the frequency analysis. In particular, we can further investigate words that have replaced negative trends over time. For example, we can see how the word "*handicapped*" has been replaced by "*disabled*". Instead,

	Word	
Year	white house	pope
1987 1988	reagan president reagan	john paul ii pope john paul ii
1989 1990 1991 1992	mr bush president bush	
1993 1994 1995 1996 1997 1998 1999 2000	mr clinton president clinton clinton	
2001 2002 2003 2004 2005 2006 2007	mr bush president bush bush bush administration	benedict xvi pope benedict xvi benedict cardinal joseph ratzinger

Tab. 6.6.: Changes in named entities associations.

the word "*computer terminal*" is more ambiguous. In fact, at first, it has been replaced by "*computer screen*", indicating that the terminal was first meant as a screen with the only function to display data. Subsequently, "*computer terminal*" shifted its meaning to indicate the device itself, being replaced over time by the words "*computer laptop*", "*hand held device*" and then "*facsimile machine*". Similarly, the word "*data processing*" was meant in the corpus as the secretarial task of inserting and managing data. This can be seen by the fact that, for a certain period, the word "*data processing*" has been replaced by "*data entry*". Furthermore, we found some relationships between "*hipsters*" and "*yuppies*" across ten years (i.e., 2005-1995).

We considered, moreover, some of the emerging trends in the frequency analysis, and calculated their similar word vectors across the matrices of the past years. The table shows some interesting results. Since emerging trends indicate words that did not exist in the past, these results can be

interpreted as answering the question: what has played a similar role in the past? For example, we found some similarities between "*facebook*" and some of the primordial social network sites such as "*friendster*", as well as the web portal "*aol*" in the late nineties. Furthermore, we found that "*mtv*" was arguably the "*youtube*" of the nineties. This highlights how these two means has been used in two different periods with the same purpose of watching music video clips. Similarly, in the late 2000 "*ipod*" was the new "*walkman*", and in the early nineties, "*floppy disk*" had the same purpose of modern "*flash drive*". Again, we show that, at first "*commodore*", and then"*nintendo*", have played a similar role of the modern game consoles such as "*playstation*" and "*xbox*". In particular, we show this through a complete table of temporal analogies (table 6.10). In this table, we can move back and forth through time, by looking at the resulting word vectors computed by using the year source (vertical) and the year target (horizontal) for the word "*playstation*".

Finally, we found other interesting temporal analogies that we show in table 6.9. In particular, we show how temporal word embeddings can be used to compute some temporal machine translation tasks. For example, "*st petersbourg*" of the year 2000 can be translated to "*leningrad*" of 1990. Similarly, "*soviet union*" was at first recognized as the "*former soviet union*" in the early 90s, and subsequently, in the 2000s, as "*russia*". By using the same principle, we can translate the word "*george w bush*" of the 2001 (i.e., during its presidency) into the "*ronald reagan*" of the 1987 and the "*bill clinton*" of the 1994, who were presidents in those years. Furthermore, we can compute more complex tasks. For example, if we take the vector for the president of the United States in the year 1994, which is "*bill clinton*", and we add up the vector of "*italy*", we obtain the corresponding vector for the prime minister of Italy in that year, which was "*berlusconi*". We can repeat this operation by specifying different years: we obtain "*prodi*" and "*dalema*" for the years 1998 and 1999, respectively; which is correct, since they were prime ministers of Italy in those years. Finally, if we sum the word vectors "*apple*" and "*italy*" in the year 1988, we obtain "*olivetti*", which was a very important Italian computer company in those years.

1989 **computer terminal**	1990 scanning	1991 **computer screen**	1992 video screen
	1993 computer	1994 **laptop computer**	1995 scanner
	1996 computer	1997 facsimile machine	1998 facsimile machine
	1999 facsimile machine	2000 wirelessly	2001 wirelessly
	2002 **hand held device**	2003 **facsimile machine**	2004 wirelessly
	2005 facsimile machine	2006 facsimile machine	2007 facsimile machine

1990 **data processing**	1991 data entry	1992 data entry	1993 data entry
	1994 data entry	1995 data entry	1996 data entry
	1997 data entry	1998 computer	1999 data entry
	2000 **data entry**	2001 computer	2002 intranet
	2003 intranet	2004 intranet	2005 intranet
	2006 intranet	2007 intranet	

1995 **yuppies**	1996 punks	1997 somethings	1998 bohemians
	1999 punks	2000 punks	2001 hippies
	2002 **hippies**	2003 hippies	2004 hippies
	2005 **hipsters**	2006 punks	2007 punks

1992 **handicapped**	1993 disabled	1994 disabled	1995 disabled
	1996 disabled	1997 people disabilities	1998 disabled
	1999 disabled	2000 disabled	2001 disabled
	2002 **disabled**	2003 disabled	2004 hearing impaired
	2005 hearing impaired	2006 hearing impaired	2007 hearing impaired

Tab. 6.7.: Temporal analogies towards the future.

2006	2005	2004	2003
facebook	**friendster**	friendster	friendster
	2002	2001	2000
	ivillage	**geocities**	cnet
	1999	1998	1997
	yahoo	yahoo	**aol**
	1996	1995	1994
	internet	internet	internet
	1993	1992	1991
	taligent	e mail	electronic mail
	1990	1989	1988
	venture capital	interactive	advertisers

2007	2006	2005	2004
youtube	video clips	tivo	kazaa
	2003	2002	2001
	google	google	cnet
	2000	1999	1998
	cnet	cnet	starwave
	1997	1996	1995
	starwave	internet	internet
	1994	1993	1992
	america online	**mtv**	**mtv**
	1991	1990	1989
	electronic mail	video	media

2006	2005	2004	2003
ipod	mp3 player	mp3 player	mp3 player
	2002	2001	2000
	mp3 player	mp3 player	walkman
	1999	1998	1997
	walkman	cd player	cd player
	1996	1995	1994
	portable computer	cd player	cd player
	1993	1992	1991
	walkman	**walkman**	**walkman**
	1990	1989	1988
	cd rom	personal computer	stereo

2006	2005	2004	2003
flash drive	mac pc	hard drive	firewire
	2002	2001	2000
	cd dvd	firewire	pocket pc
	1999	1998	1997
	mb	floppy disk	internet browser
	1996	1995	1994
	hard drive	cd rom drive	cd rom drive
	1993	1992	1991
	cd rom drive	**floppy disk**	**floppy disk**
	1990	1989	1988
	floppy disk	ms dos	disk drive

Tab. 6.8.: Temporal analogies towards the past.

Query			
Key	**Source**	**Target**	**($\simeq$) Most Similar**
'soviet union'	1989	1992	former soviet union
'soviet union'	1989	2000	russia
'kodachrome'	1990	1998	pixels
'st petersburg'	2000	1990	leningrad
'2000'	2000	1995	1995
'2000'	2000	2005	2005
'george w bush'	2001	1994	bill clinton
'george w bush'	2001	1987	ronald reagan
'angela merkel'	2007	1997	mr kohl
'apple' + 'italy'	1988	1988	olivetti
'bill clinton' + 'italy'	1994	1994	berlusconi
'bill clinton' + 'italy'	1998	1998	prodi
'bill clinton' + 'italy'	1999	1999	dalema

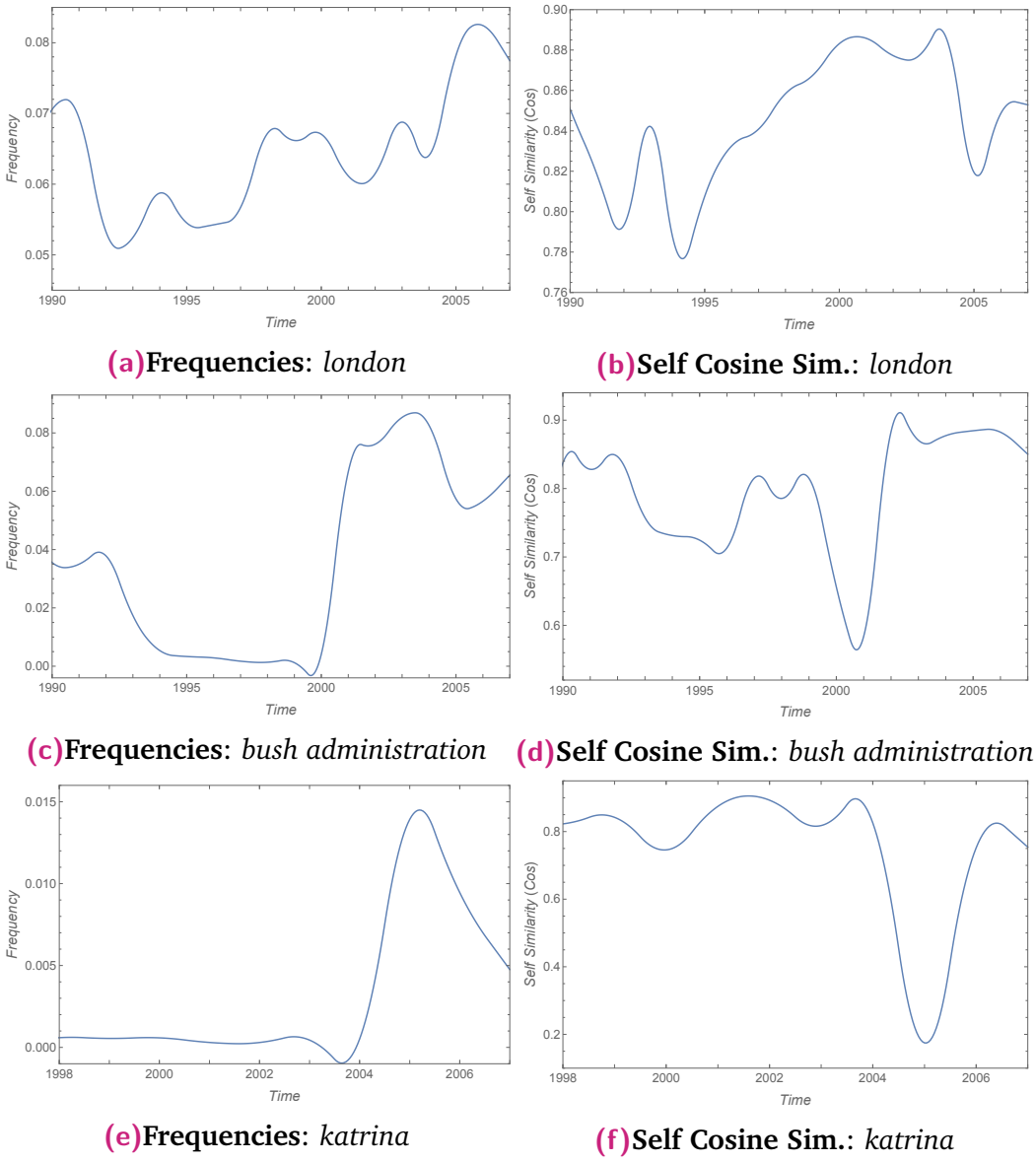
Tab. 6.9.: Temporal translations.

6.4 Comparison and Observation

In order to quantify changes in word meanings, we used time series of self similarities. In particular, we computed cosine similarity between two vectors of the same word for each pair of consecutive years. We noticed that word vectors change significantly in relation with peaks in frequency. We summarize this in figure 6.1. For instance, the word "*katrina*" in 2005 has a vector which is quite different (i.e., cosine similarity is low) compared to the vectors for the other years (fig: fig:katcos). This is due to the fact that the media reported widely the event of Hurricane Katrina, originated on August 2005, which results in a rise of frequency for the word "*katrina*" in 2005 (fig: 6.1e). Similarly, the vector of "*bush administration*" changes in 2001 (fig: 6.1d), simultaneously with the importance given to the presidential election (fig: 6.1c). Again, the vector of "*london*" undergoes a significant semantic shift in 2005 (fig: 6.1b), which was affected by the peak in popularity in the same year due to the terrorist attack (fig: 6.1a).

Word embeddings are useful methods to create expressive models that are able to capture semantic relationships which go beyond the simple co-occurrences. However, as can be seen, these models are highly influenced by word occurrences in the raw text. For this reason, it is important to also

consider frequencies when modeling semantics with word embeddings, in order to create models as unbiased as possible.



(a)Frequencies: *london*

(b)Self Cosine Sim.: *london*

(c)Frequencies: *bush administration*

(d)Self Cosine Sim.: *bush administration*

(e)Frequencies: *katrina*

(f)Self Cosine Sim.: *katrina*

Fig. 6.1.: Frequencies and self cosine similarities comparison. High values of frequency produce semantic shifts.

	1987	1988	1989	1990
1998	camcorder	videogame	**nintendo**	**nintendo**
1999	disk drive	laptop	mips	apple macintosh
2000	microprocessors	personal computer	disk drive	laptop
2001	commodore	video game	laptop	laptop
2002	**amiga**	personal computer	laptop	**nintendo**
2003	**commodore**	personal computer	**apple ii**	**apple ii**
2004	disk drive	personal computer	**nintendo**	**nintendo**
2005	nec	**atari**	**nintendo**	**nintendo**
2006	personal computer	**atari**	intel	**atari**
2007	chip	**atari**	**atari**	**atari**

	1991	1992	1993	1994
1998	**nintendo**	powerbook	**nintendo**	cd rom titles
1999	pen based	powerbook	**nintendo**	**3do**
2000	apple macintosh	powerbook	**sega**	notebook computers
2001	apple macintosh	microsoft windows	**nintendo**	**nintendo**
2002	pen based	apple macintosh	**nintendo**	**sega**
2003	pen based	apple ii	**nintendo**	**sega**
2004	**nintendo**	sx	**nintendo**	**nintendo**
2005	**atari**	intel	**nintendo**	**sega**
2006	advanced micro	video game	**nintendo**	**3do**
2007	**atari**	microsoft windows	**3do**	**3do**

	1995	1996	1997	1999	2000
1998	**nintendo**	mortal kombat	**nintendo**	sony playstation	game console
1999	**nintendo**	**nintendo**	**nintendo**	**dreamcast**	**dreamcast**
2000	**nintendo**	**nintendo**	**nintendo**	**dreamcast**	game console
2001	**nintendo**	**nintendo**	**nintendo**	**dreamcast**	sony playstation
2002	**nintendo**	**nintendo**	**nintendo**	**sega**	**dreamcast**
2003	**nintendo**	**nintendo**	**nintendo**	**nintendo**	**dreamcast**
2004	**nintendo**	**nintendo**	**nintendo**	**sega**	**dreamcast**
2005	**nintendo**	**nintendo**	**nintendo**	**sega**	**dreamcast**
2006	**nintendo**	**nintendo**	**nintendo**	**dreamcast**	**dreamcast**
2007	**nintendo**	**nintendo**	**nintendo**	**sega**	**dreamcast**

	2001	2002	2003	2004
1998	sony playstation	game consoles	game consoles	game consoles
1999	game console	game console	game console	game consoles
2000	game console	**dreamcast**	game console	**dreamcast**
2001	sony playstation	**gamecube**	**gamecube**	**gamecube**
2002	game console	**gamecube**	**gamecube**	**xbox**
2003	**gamecube**	**gamecube**	**gamecube**	**xbox**
2004	**nintendo**	**xbox**	**gamecube**	**xbox**
2005	**dreamcast**	**xbox**	**xbox**	**xbox**
2006	**dreamcast**	**xbox**	**xbox**	**xbox**
2007	game console	**xbox**	game console	**xbox**

	2005	2006	2007
1998	**playstation portable**	**microsoft xbox**	**microsoft xbox**
1999	**playstation portable**	**microsoft xbox**	**microsoft xbox**
2000	**dreamcast**	game console	**dreamcast**
2001	**gamecube**	**gamecube**	**gamecube**
2002	**gamecube**	**gamecube**	**gamecube**
2003	**gamecube**	**gamecube**	**gamecube**
2004	**gamecube**	**gamecube**	**gamecube**
2005	**xbox**	game console	game console
2006	**xbox**	game console	game console
2007	**xbox**	game console	**gamecube**

Tab. 6.10.: Game consoles. Temporal analogies for the word *playstation*. Year source (vertical axis), year target (horizontal axis).

7

Conclusion

In this thesis we proposed text mining methods to capture temporal patterns in large corpora collected over the years. In particular, we presented two different analyses: one using word frequencies to discover trends in word usage, and the second using word embeddings to model changes in semantics. Word frequencies and semantic changes are often inevitably linked to each other. Indeed, words which have acquired a different meaning over time, as well as named entities which have changed their associations, can experience an increase in their popularity. Similarly, spikes in word frequency may cause shifts in semantics. In this respect, we presented frequency based methods, which, although very simple, are immediate ways to quantify patterns over time. More precisely, we investigated words with an increasing popularity (i.e., *positive trends*), words with a decreasing popularity (i.e., *negative trends*), and newly born words (i.e., *emerging trends*).

On the other hand, we considered a word embeddings approach to model changes in semantics over time. Word embeddings are useful methods to meaningfully represent words as vectors, such that words which have similar meanings have geometrically close vectors. Furthermore, these models capture semantic regularities and allow to compute word analogies by using basic mathematical operations over vectors (e.g., the well-known example *king-man+woman* = *queen*).

Word embeddings have recently established themselves as state-of-art word representations in almost all text mining and natural language processing tasks. However, classic models of word embeddings are not able to capture dynamics of semantics, since they represent a word as a single-state vector which do not consider different time spans of the corpus. In this regard, we trained a temporal word embeddings model such that comparison can be done between vectors of the same words in different time periods. We derived temporal word embeddings from highly evolving corpora (i.e., newspapers and blogs), in order to model the knowledge that textual archives have accumulated over the years. By capture dynamics in word meanings and named entities associations, these models can be used to discover semantic evolution of words (e.g. *the Internet* changes from being a "*communication networks*" to the world wide web accessible through *mobile devices*), as well as find *temporal analogies* (e.g., *youtube*@late2000 $\simeq$ *mtv*@early90, since they both have been used for watching music videos across a decade), and finally compute simple *temporal translations* (e.g., *bush*@2001 $=$ *bill clinton*@1994) and more complex ones (e.g., *bill clinton*@1994 $+$ *italy* $=$ *berlusconi*@1994). Furthermore, we found that word embeddings models of semantics are highly influenced by word frequencies in corpora. In this respect, we argue that while using word embeddings to conduct text analyses, and temporal analyses in particular, we should also consider the related word frequencies, in order to obtain less biased models of semantics.

Appendices

A

Time2vec:

a Word Embeddings Time Machine

Time2vec is a web application available online, that we have developed as proof of concepts presented in this thesis. Figure A.1 show screenshots of Time2vec.

The demo use our temporal word embeddings model which is derived from the NYT corpus, and allows users to easily replicate some of the results achieved in this thesis. In particular, by typing into a text input, users can visualize a time series for each typed word, quantifying the changes in semantics across the years. This allows to compare several words by their rate of semantic change.

Furthermore, for each typed word, users are provided with an interactive visual interface for inspecting temporal analogies. More specifically, by moving a slider input, one can set different values for the source year, and visualize the list of matching temporal analogies, one for each target year (i.e., from 1987 to 2007). A measure of similarity, (i.e., cosine similarity) between the typed word and each result analogy, is also displayed .

Time2vec is available at http://smartdata.cs.unibo.it/time2vec/ .

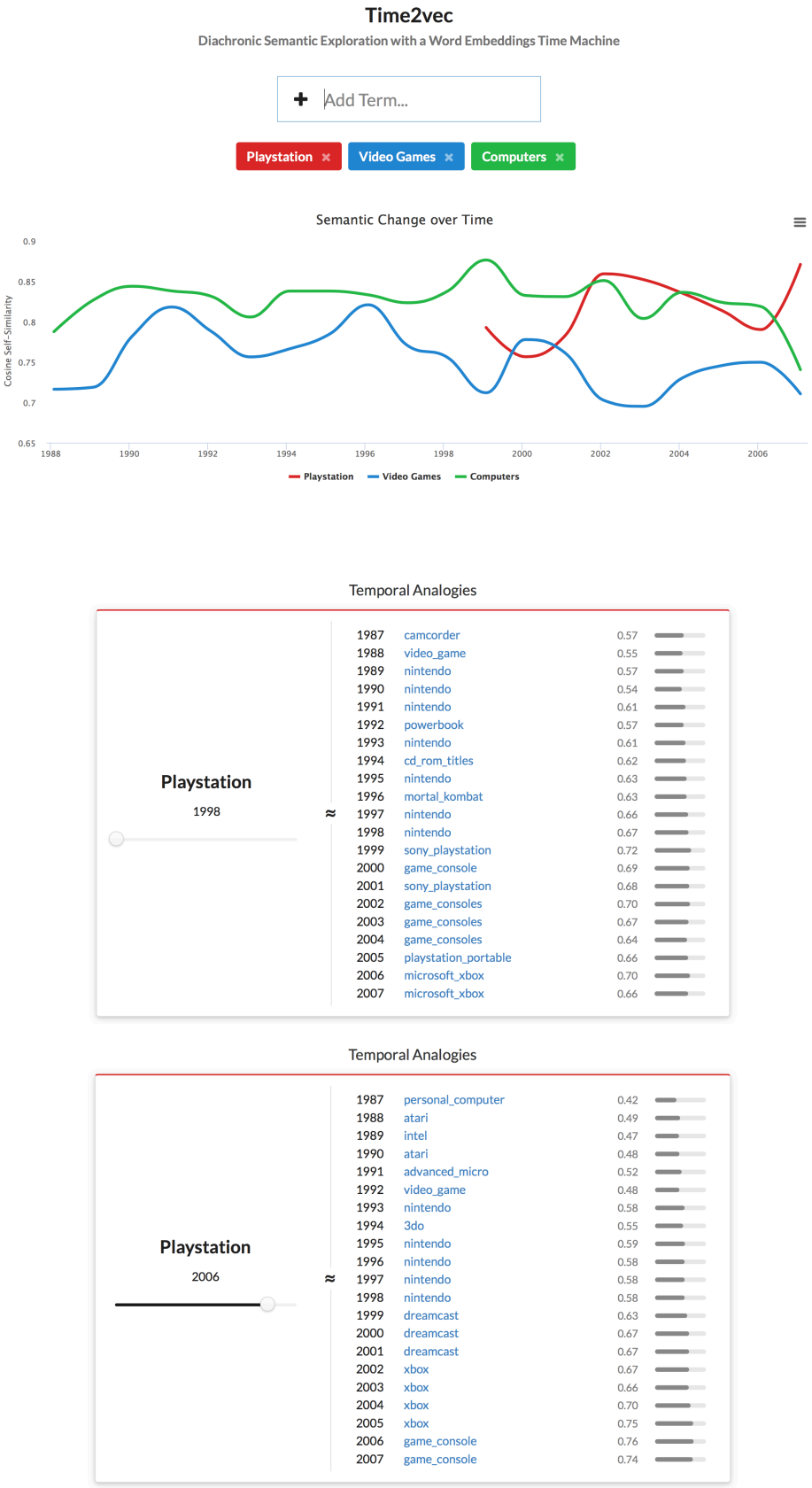


Fig. A.1.: **Time2vec** screenshots

Bibliography

[1] Sunita Ahlawat and Sucheta Ahlawat. „An innovative decade of enduring accounting ideas as seen through the lens of culturomics: 1900-1910“. In: *American Journal of Business Research* 6.1 (2013), p. 63 (cit. on p. 29).

[2] Ashton Anderson, Dan McFarland, and Dan Jurafsky. „Towards a computational history of the acl: 1980-2008“. In: *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*. Association for Computational Linguistics. 2012, pp. 13–21 (cit. on p. 28).

[3] Ching-man Au Yeung and Adam Jatowt. „Studying how the past is remembered: towards computational history through large scale text mining“. In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. 2011, pp. 1231–1240 (cit. on p. 28).

[4] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. „Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.“ In: *ACL (1)*. 2014, pp. 238–247 (cit. on pp. 9, 30).

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. „A neural probabilistic language model“. In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155 (cit. on pp. 13, 31, 47).

[6] David N Bengston, David P Fan, and Doris N Celarier. „A new approach to monitoring the social environment for natural resource management and policy: The case of US national forest benefits and values“. In: *Journal of Environmental Management* 56.3 (1999), pp. 181–193 (cit. on p. 28).

[7] Klaus Berberich, Srikanta J Bedathur, Mauro Sozio, and Gerhard Weikum. „Bridging the Terminology Gap in Web Archive Search.“ In: *WebDB*. 2009 (cit. on pp. 55, 60).

[8] Bernard Berelson. „Content analysis in communications research“. In: (1952) (cit. on p. 27).

[9] John Bohannon. „Google Books, Wikipedia, and the future of culturomics“. In: *Science* 331.6014 (2011), pp. 135–135 (cit. on p. 29).

[10] Hyunyoung Choi and Hal Varian. „Predicting the present with Google Trends“. In: *Economic Record* 88.s1 (2012), pp. 2–9 (cit. on p. 26).

[11] Kenneth Ward Church and Patrick Hanks. „Word association norms, mutual information, and lexicography“. In: *Computational linguistics* 16.1 (1990), pp. 22–29 (cit. on p. 12).

[12] Claudio Cioffi-Revilla. „Introduction to computational social science“. In: *Berlin/New York: Springer* 10 (2014), pp. 978–1 (cit. on p. 28).

[13] Rosaria Conte, Nigel Gilbert, Giulia Bonelli, et al. „Manifesto of computational social science“. In: *European Physical Journal-Special Topics* 214 (2012), p–325 (cit. on p. 28).

[14] Wayne A Danielson and Dominic L Lasorsa. „Perceptions of social change: 100 years of front-page content in The New York Times and The Los Angeles Times“. In: *Text analysis for the social sciences: Methods for drawing statistical inferences from texts and transcripts* (1997), pp. 103–116 (cit. on p. 28).

[15] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. „Indexing by latent semantic analysis“. In: *Journal of the American society for information science* 41.6 (1990), p. 391 (cit. on p. 31).

[16] Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press, 2007 (cit. on p. 3).

[17] Roberto Franzosi. „Content analysis: Objective, systematic, and quantitative description of content“. In: *Content analysis* 1 (2008) (cit. on p. 26).

[18] Roberto Franzosi. *From words to numbers: Narrative, data, and social science*. Vol. 22. Cambridge University Press, 2004 (cit. on p. 27).

[19] Gregory Grefenstette and Pasi Tapanainen. „What is a word, what is a sentence?: problems of Tokenisation“. In: (1994) (cit. on p. 34).

[20] William L Hamilton, Jure Leskovec, and Dan Jurafsky. „Diachronic word embeddings reveal statistical laws of semantic change“. In: *arXiv preprint arXiv:1605.09096* (2016) (cit. on pp. 31, 59).

[21] Zellig S Harris. „Distributional structure“. In: *Word* 10.2-3 (1954), pp. 146–162 (cit. on p. 8).

[22] John A Hartigan and Manchek A Wong. „Algorithm AS 136: A k-means clustering algorithm“. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108 (cit. on p. 53).

[23] Thomas T Hills and James S Adelman. „Recent evolution of learnability in American English from 1800 to 2000“. In: *Cognition* 143 (2015), pp. 87–92 (cit. on p. 29).

[24] Thomas Huet, Joanna Biega, and Fabian M Suchanek. „Mining history with le monde“. In: *Proceedings of the 2013 workshop on Automated knowledge base construction*. ACM. 2013, pp. 49–54 (cit. on p. 28).

[25] Michelle Jackson. „Content analysis“. In: *Research Methods for Health and Social Care* (2008), pp. 78–91 (cit. on p. 27).

[26] Adam Jatowt, Daisuke Kawai, and Katsumi Tanaka. „Digital history meets Wikipedia: Analyzing historical persons in Wikipedia“. In: *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. ACM. 2016, pp. 17–26 (cit. on p. 28).

[27] Harold H Kassarjian. „Content analysis in consumer research“. In: *Journal of consumer research* 4.1 (1977), pp. 8–18 (cit. on p. 28).

[28] Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. „Temporal analysis of language through neural language models“. In: *arXiv preprint arXiv:1405.3515* (2014) (cit. on pp. 31, 58).

[29] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage, 2012 (cit. on p. 27).

[30] Satya P Krishnan, Tracy Durrah, and Karen Winkler. „Coverage of AIDS in popular African American magazines“. In: *Health Communication* 9.3 (1997), pp. 273–288 (cit. on p. 28).

[31] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. „Statistically significant detection of linguistic change“. In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2015, pp. 625–635 (cit. on pp. 31, 58).

[32] Harold D Lasswell, Daniel Lerner, and Ithiel de Sola Pool. „Comparative study of symbols: An introduction.“ In: (1952) (cit. on p. 28).

[33] Robert B Lees and Noam Chomsky. „Syntactic structures“. In: *Language* 33.3 Part 1 (1957), pp. 375–408 (cit. on p. 27).

[34] Daniel Lerner, Ithiel Pool, and Harold D Lasswell. „Comparative analysis of political ideologies: A preliminary statement“. In: *Public Opinion Quarterly* 15.4 (1951), pp. 715–733 (cit. on p. 28).

[35] Omer Levy and Yoav Goldberg. „Neural word embedding as implicit matrix factorization“. In: *Advances in neural information processing systems*. 2014, pp. 2177–2185 (cit. on p. 13).

[36] Kevin Lund and Curt Burgess. „Producing high-dimensional semantic spaces from lexical co-occurrence“. In: *Behavior research methods, instruments, & computers* 28.2 (1996), pp. 203–208 (cit. on p. 31).

[37] Laurens van der Maaten and Geoffrey Hinton. „Visualizing data using t-SNE“. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605 (cit. on p. 53).

[38] Charles S Madden, Marjorie J Caballero, and Shinya Matsukubo. „Analysis of information content in US and Japanese magazine advertising“. In: *Journal of Advertising* 15.3 (1986), pp. 38–45 (cit. on p. 28).

[39] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999 (cit. on pp. 34–36, 38).

[40] Qiaozhu Mei and ChengXiang Zhai. „Discovering evolutionary theme patterns from text: an exploration of temporal text mining“. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 198–207 (cit. on p. 3).

[41] Charles F Meyer. *Introducing English Linguistics*. Vol. 11. Cambridge University Press Cambridge, 2009 (cit. on p. 34).

[42] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, et al. „Quantitative analysis of culture using millions of digitized books“. In: *science* 331.6014 (2011), pp. 176–182 (cit. on p. 29).

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. „Distributed representations of words and phrases and their compositionality“. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119 (cit. on pp. 2, 14, 31, 38, 47, 50, 52).

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. „Efficient estimation of word representations in vector space“. In: *arXiv preprint arXiv:1301.3781* (2013) (cit. on pp. 2, 14, 31, 50).

[45] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. „Linguistic regularities in continuous space word representations“. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013, pp. 746–751 (cit. on p. 54).

[46] George A Miller and Walter G Charles. „Contextual correlates of semantic similarity“. In: *Language and cognitive processes* 6.1 (1991), pp. 1–28 (cit. on p. 10).

[47] Andriy Mnih and Geoffrey E Hinton. „A scalable hierarchical distributed language model“. In: *Advances in neural information processing systems*. 2009, pp. 1081–1088 (cit. on p. 52).

[48] Frederic Morin and Yoshua Bengio. „Hierarchical Probabilistic Neural Network Language Model.“ In: *Aistats*. Vol. 5. Citeseer. 2005, pp. 246–252 (cit. on p. 52).

[49] Kevin P Murphy. „Machine Learning: A Probabilistic Perspective“. In: (2012) (cit. on p. 14).

[50] Vinod Nair and Geoffrey E Hinton. „Rectified linear units improve restricted boltzmann machines“. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. on p. 19).

[51] Kimberly A Neuendorf. *The content analysis guidebook*. Sage, 2016 (cit. on p. 28).

[52] Martin F Porter. „An algorithm for suffix stripping“. In: *Program* 14.3 (1980), pp. 130–137 (cit. on p. 36).

[53] Daphne R Raban and Avishag Gordon. „The effect of technology on learning research trends: a bibliometric analysis over five decades“. In: *Scientometrics* 105.1 (2015), pp. 665–681 (cit. on p. 30).

[54] Radim Řehůřek and Petr Sojka. „Software Framework for Topic Modelling with Large Corpora“. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `http://is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, May 2010, pp. 45–50 (cit. on p. 69).

[55] Xin Rong. „word2vec parameter learning explained“. In: *arXiv preprint arXiv:1411.2738* (2014) (cit. on pp. 51, 52).

[56] Peter H Schönemann. „A generalized solution of the orthogonal procrustes problem“. In: *Psychometrika* 31.1 (1966), pp. 1–10 (cit. on p. 59).

[57] Hinrich Schütze and Jan Pedersen. „A vector model for syntagmatic and paradigmatic relatedness“. In: *Proceedings of the 9th Annual Conference of the UW Centre for the New OED and Text Research*. Oxford. 1993, pp. 104–113 (cit. on p. 11).

[58] Vered Silber-Varod, Yoram Eshet-Alkalai, and Nitza Geri. „Analyzing the Discourse of Chais Conferences for the Study of Innovation and Learning Technologies via a Data-Driven Approach“. In: *Interdisciplinary Journal of e-Skills and Life Long Learning* 12 (2016) (cit. on p. 30).

[59] John Sinclair. *Corpus, concordance, collocation*. Oxford University Press, 1991 (cit. on p. 26).

[60] Daniel S Soper and Ofir Turel. „An n-gram analysis of Communications 2000–2010“. In: *Communications of the ACM* 55.5 (2012), pp. 81–87 (cit. on p. 30).

[61] Karen Sparck Jones. „A statistical interpretation of term specificity and its application in retrieval“. In: *Journal of documentation* 28.1 (1972), pp. 11–21 (cit. on p. 12).

[62] Svetlana Stepchenkova and James S Eales. „Destination image as quantified media messages: The effect of news on tourism demand“. In: *Journal of Travel Research* 50.2 (2011), pp. 198–212 (cit. on p. 28).

[63] Fabian M Suchanek and Nicoleta Preda. „Semantic culturomics“. In: *Proceedings of the VLDB Endowment* 7.12 (2014), pp. 1215–1218 (cit. on p. 29).

[64] Terrence Szymanski. „Temporal Word Analogies: Identifying Lexical Replacement with Diachronic Word Embeddings“. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2. 2017, pp. 448–453 (cit. on p. 31).

[65] Joseph Turian, Lev Ratinov, and Yoshua Bengio. „Word representations: a simple and general method for semi-supervised learning“. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics. 2010, pp. 384–394 (cit. on p. 9).

[66] Klaas Willems. „Culturomics and the representation of the language of the Third Reich in digitized books“. In: *Interdisciplinary Journal for Germanic Linguistics and Semiotic Analysis* (2013) (cit. on p. 29).

[67] Malcolm Macdonald Willey. *The country newspaper: A study of socialization and newspaper content*. University of North Carolina Press, 1926 (cit. on p. 27).

[68] Raymond R Willoughby. „Leonard Bloomfield. Linguistic Aspects of Science.“ In: *Journal of Social Psychology* 11.1 (1940), p. 237 (cit. on p. 26).

[69] Arch G Woodside, Suresh Sood, and Kenneth E Miller. „When consumers and brands talk: Storytelling theory and research in psychology and marketing“. In: *Psychology & Marketing* 25.2 (2008), pp. 97–145 (cit. on p. 28).

[70] Julian L Woodward. „Quantitative newspaper analysis as a technique of opinion research“. In: *Social Forces* 12.4 (1934), pp. 526–537 (cit. on p. 27).

[71] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. „Discovery of Evolving Semantics through Dynamic Word Embedding Learning“. In: *arXiv preprint arXiv:1703.00607* (2017) (cit. on p. 31).

[72] George Kingsley Zipf. „The psycho-biology of language.“ In: (1935) (cit. on pp. 9, 26, 40).

List of Figures

List of Tables

Ringraziamenti

Con questa pagina, per me, non si conclude semplicemente una tesi di laurea, ma un intero capitolo della mia vita. Giunti a questo punto, mi sembra dunque doveroso ringraziare chi mi ha accompagnato in questo percorso:

In primo luogo mi sento di ringraziare il mio relatore, per avermi concesso l'opportunità di approfondire i temi dell'informatica che più mi appassionano.

Ringrazio Stefano, senza il quale questa tesi non sarebbe mai stata scritta, ma anche e soprattutto perché tanto altro nella mia vita, senza di lui, non avrei mai fatto.

Un ringraziamento speciale va a mia madre, che mi ha supportato, e troppo spesso sopportato, specialmente nei momenti più duri.

Ringrazio i miei fratelli, Luigina e Andrea, per la loro stima, ma soprattutto per la loro fiducia.

Ringrazio i miei nipoti, Carlotta e Riccardo, per il loro amore smisurato.

Ringrazio Fabio perché ora ho un nuovo modo per esprimere me stesso.

Ringrazio Francesca per aver profumato la mia vanità.

Ringrazio mio zio Franco per prendersi cura di tutti noi.

Ringrazio Daniele per tutti quei momenti che ci ricordano bambini e quelli che ancora ce lo ricorderanno.

Ringrazio Emiliano per quella sua lucidità che da anni mi guida.

Ringrazio Marta per come è lieve il mondo dopo averle parlato.

Ringrazio Benjamin perché ogni distanza è infinitamente più piccola della gioia che mi dona.

Ringrazio Arianna e Laura per aver accolto la mia iper presenza nella loro dolce casa.

Ringrazio Alessandro per tutti questi anni passati insieme a Bologna.

Ringrazio i miei compagni di avventura Ivan, Claudia e Vincenzo, per il continuo e prezioso scambio di idee.

Ringrazio Fabrizio perché ogni gioia e ogni dolore so come ritrovarli nella luce di un'ora.

In fine ringrazio chi più di tutti ha fatto sì che io fossi esattamente chi voglio essere. Chi se n'è andato e chi dopo tutto è rimasto al mio fianco. Chi in ogni caso è nel mio cuore. Grazie papà. Grazie Elisa.