University of York

Final year project

Word Embeddings

Joel Strouts

supervised by

Dr. Ben Powell

Abstract

In this report we attempt to provide a comprehensive primer on key topics relating to word embeddings, sufficient to enable further reading of more advanced material. The reader does not need prior experience in the domain of natural language processing (NLP), but familiarity with core mathematics at an undergraduate level is assumed. We take a more cross-disciplinary approach than purely mathematical one to reflect the primarily applied nature the subject matter. In the first chapter we summarise the key features of word embeddings: What they are, how they are created, what they are for, and other uses the theory of word embeddings outside of NLP. In the second chapter we discuss the difficulties of defining word meaning and present a framework to help distinguish between different senses of 'word'. In the third chapter we describe key concepts and definitions generally assumed by the different methods for generating word embeddings. Finally in the fourth chapter we illustrate these ideas by describing the operation of particular class of embedding methods: count models.

July 14, 2021

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 What are word embeddings?

Word embeddings are numerical representations of words which are derived from the observed patterns of word usage in a body of example text. These representations take the form of position vectors within a *Word Space*. The resulting *Word Space Model* was first described by Schütze (1993) in the terms:

> “ [The Word Space Model] uses feature vectors to represent words, but the features cannot be interpreted on their own. Vector similarity is the only information present in Word Space: semantically related words are close, unrelated words are distant. ”

Example 1.1.1 *(Two Dimensional Word Space)***.** The word space in Figure 1.1 illustrates both components of the above definition: first that related words are closer in space than less related words (the words *apple* and *orange* are closer in space than *apple* and *bicycle*), and second that the individual axis of variation are not meaningful considered in isolation from one another (there is no trait like ‘size’ which the x or y value for a word is descriptive of).

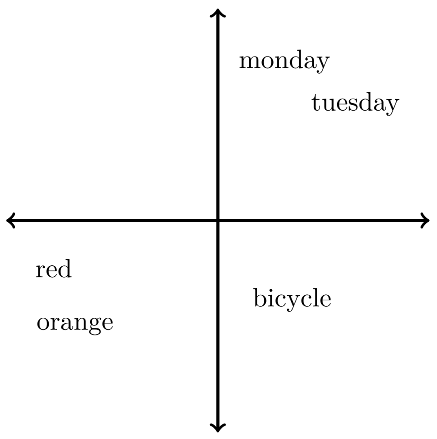


Figure 1.1: A two dimensional word space encoding semantic information about the five words; *red, orange, monday, tuesday & bicycle*

word	features	
	x	y
red	-1.55	-0.45
orange	-1.35	-1.0
monday	0.63	1.5
tuesday	1.3	1.1
bicycle	0.7	-0.75

Table 1.1: Feature vectors representing the words in Figure 1.1

More recent research has suggested that word spaces can in fact arrange word-representations with more order than Schütze's early description of these spaces accounted for. Notably, (Mikolov, Yih, and Zweig 2013) demonstrated their potential for geometrically encoding semantic relationships between words (the *direction* in which one vector is distant from another can capture semantic information, not only the distance). Figure 1.2 illustrates the idea of geometrically encoded semantic information. Indeed, modern methodologies for assessing the quality of a word space consider the extent to which *multifaceted* linguistic information about the vocabulary (not just semantic, but; phonetic, morphological, and syntactic information about the encoded words) is recoverable by the embeddings (Yaghoobzadeh and Schütze 2016; B. Wang et al. 2019).

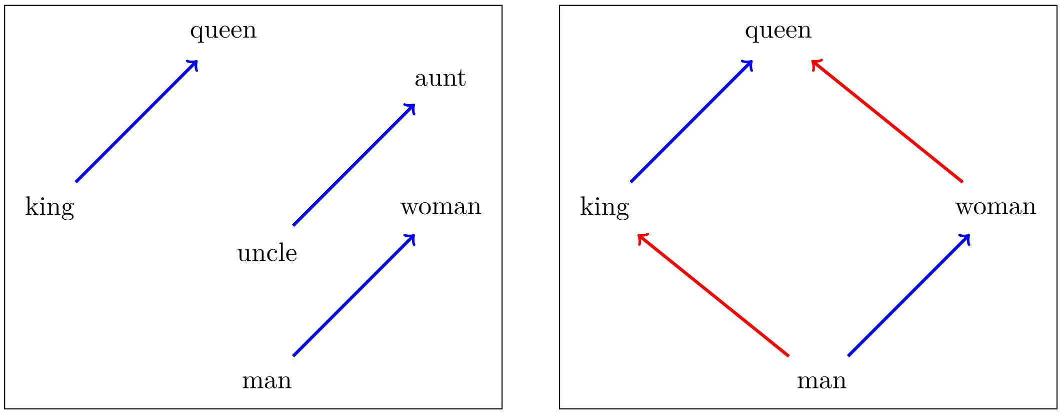


Figure 1.2: An idealised illustration of how semantic & syntactic information could be geometrically encoded by embeddings in a word space[1].

Typical word spaces, unlike the simplified form illustrated in example 1.1.1, are *high-dimensional*: it is common for the dimension of a word space to be in the order of tens, hundreds, or thousands. In a relatively low-dimensional pre-trained word embedding model[2] the word *orange* (which is represented in the above example by the vector orange $\mapsto \langle -1.55, 0.35 \rangle$) is embedded as the 50-dimensional vector:

1. We use the term *idealised* here because although this figure is a reproduction of that which accompanies the famous $\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} = \overrightarrow{\text{queen}}$ example from (Mikolov, Yih, and Zweig 2013), more recent investigations (Linzen 2016; Rogers, Drozd, and Li 2017) have concluded that these analogical relationships are not the *robust* features of word space geometry suggested by these figures. Nevertheless, investigations into the geometry of word space have found some notable structure (Liu et al. 2018).

2. Model provided by Pennington, Socher, and Manning (2014). Available for download at (GloVe project website)

$$\text{orange} \mapsto$$

$$\begin{array}{l}
\langle\ \texttt{-0.42783},\ \texttt{0.43089},\ \texttt{-0.50351},\ \texttt{0.5776},\ \texttt{0.097786},\ \texttt{0.2608},\ \texttt{-0.68767},\ \texttt{-0.31936},\ \texttt{-0.25337},\ \texttt{-0.37255},\\
\texttt{-0.045907},\ \texttt{-0.53688},\ \texttt{0.97511},\ \texttt{-0.44595},\ \texttt{-0.50414},\ \texttt{-0.086751},\ \texttt{-1.0645},\ \texttt{0.36625},\ \texttt{-0.52428},\ \texttt{-1.3413},\\
\texttt{-0.2391},\ \texttt{-0.58808},\ \texttt{0.56378},\ \texttt{-0.062501},\ \texttt{-1.7429},\ \texttt{-0.88077},\ \texttt{-0.27933},\ \texttt{1.4705},\ \texttt{0.50436},\ \texttt{-0.69174},\\
\texttt{2.0018},\ \texttt{0.26663},\ \texttt{-0.85679},\ \texttt{-0.18893},\ \texttt{-0.021125},\ \texttt{-0.055118},\ \texttt{-0.50337},\ \texttt{-0.67157},\ \texttt{0.55502},\ \texttt{-0.8009},\\
\texttt{0.10695},\ \texttt{0.1459},\ \texttt{-0.55588},\ \texttt{-0.64971},\ \texttt{0.22046},\ \texttt{0.67415},\ \texttt{-0.45119},\ \texttt{-1.1462},\ \texttt{0.16348},\ \texttt{-0.62946}\ \rangle
\end{array} \tag{1.0}$$

High-dimensional vectors like (1.0) cannot be plotted on an x, y plane without loss of information, so we can't visualise points in a typical word space with the same approach as we used in example 1.1.1. An alternate method for visualising points in word space which still allows us to judge the similarity of vectors by their appearance, and is applicable higher dimensional vectors, is to map the magnitude of vector components onto a smooth colour gradient and then to compare the resulting colour-vectors, rather than the original numerical forms.

Example 1.1.2 *(50 dimensional Word Space)***.** Figure 1.3 applies the principle described above to visualise word vector outputs from a real word embedding model. Each row of coloured cells represents the word vector for the word to its left. More negative entries are more blue, and more positive entries are more red. You can see, for example, that the vectors for *monday* and *tuesday* are very similar to one another, whereas the vectors for *tuesday* and *bicycle* are very different.

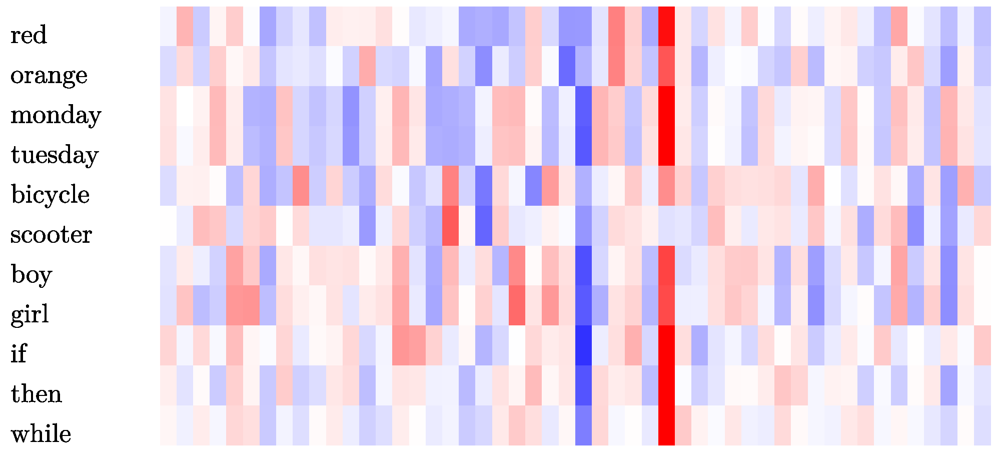


Figure 1.3: A selection of embeddings from the same pre-trained 50-dimensional *GloVe* word space model from which the *orange* vector in (1.0) was taken.

1.2 How are they created?

Word embeddings are the output of *word embedding algorithms*: Processes which take as their input large body of example language-usage and *automatically produce* (i.e., without need for further instruction from the practitioner) representations of the observed words *which encode a means of quantifying the semantic-relatedness of those words.*

> "What makes the word-space model unique in comparison with other geometrical models of meaning is that the space is constructed with no human intervention, and with no a priori knowledge or constraints about meaning similarities. In the word-space model, the similarities between words are automatically extracted from language data by looking at empirical evidence of real language use. "
>
> Sahlgren (2006)

How exactly is this information automatically extracted from a corpora of "real language usage" though? Sahlgren continues:

> "As data, the word-space model uses statistics about the distributional properties of words. The idea is to put words with similar distributional properties in similar regions of the word space, so that proximity reflects distributional similarity. The fundamental idea behind the use of distributional information is the so-called distributional hypothesis. "

Definition 1.2.1 – *The Distributional Hypothesis*

There are many variations of the statement of the distributional hypothesis; Sahlgren opts for the more general phrasing *"words with similar distributional properties have similar meanings"*, but with reference to its application in creating word spaces cites the more concrete *"words with similar meanings will occur with similar neighbours if enough text material is available"* of Schutze and Pedersen (1995) and the concise *"words which are similar in meaning occur in similar contexts"* of Rubenstein and Goodenough (1965). Firth (1957) notably captured the essence of this hypothesis with the much quoted statement *"You will know a word by the company it keeps"*.

Example 1.2.1 *(The distributional hypothesis)***.** In the figure from example 1.1.2, the vector representations of the words "Monday" and "Tuesday" are almost identical to one another. These representations were generated by analysing the contexts each word occurred in, so implied by this result is that the words "Monday" and "Tuesday" occur in very similar contexts.

Consider these partial quotations[3]:

"I'd be quite happy if I spent from Saturday night until _______ morning..." (1.1)

"It was a _______ and they walked on a tightrope to the sun." (1.2)

"I will love you at 4:15 pm next _______." (1.3)

"...the kind that blindside you at 4 pm on some idle _______." (1.4)

In each of these contexts one can infer that the *type* of word which is missing is likely to be a day of the week, but it is difficult to guess with confidence any more specifically; the words "Monday" and "Tuesday" seem to be almost equally probable guesses -*their similarity of meaning is reflected in the similarity of their contexts.*

This *distributional* linguistic perspective was widespread in the 1950s (Jurafsky and Martin 2021), with key contributions to the theory made by Joos (1950), Harris (1954), and Firth (1957), and is related Wittgenstein's (1953) ideas about 'family resemblance' from the same period (P. D. Turney and Pantel 2010).

The question of how word embeddings are created can then instead be phrased *How exactly is the distributional hypothesis employed by these (word embedding) algorithms?*. Below we paraphrase the descriptions of seven notable classes of word embedding methods, due to B. Wang et al. (2019):

Summary: Word Embedding Method Categorisation

***Co-occurrence Matrix** These methods work by building a matrix which counts the number of time each word co-occurs within a particular context. Different versions define different meanings of "context" (P. D. Turney and Pantel (2010) and Sahlgren (2006) provide a good overview of approaches).

Continuous-Bag-of-Words and skip-gram Two *iteration-based* neural network methods proposed in the `word2vec` paper (Mikolov, K. Chen, et al. 2013) and widely used thereafter. The first model (CBOW) predicts the center word from its surrounding context and the second (SG) predicts the surrounding context words given a center word.

3. These quotations were found by searching for quotes containing the word "Monday" (in the case of (1.1) and (1.2)) or "Tuesday" (in the case of (**??**) and (1.4)) on the goodreads website

Neural Network Language Model The NNLM (Bengio et al. 2003) *jointly learns a word vector representation and a statistical language model*, and is "very computationally complex".

FastText This method exists to address issues with other approaches which produce poor estimations for rarely used words. "FastText uses subword information explicitly so embedding for rare words can still be represented well". It is based on the skip-gram model, where each word is represented as a bag of character

***N*-gram Model** *n*-grams are multi-word elements with *n* parts (for example "a man" is a 2-gram). *n*-gram models are methods which operate on *n*-grams in addition to, or in place of words. Zhao et al. (2017) incorporates the n-gram model in various baseline embedding models such as `word2vec`, GloVe, PPMI, and SVD.

Dictionary Model Dictionary models learn word representations from dictionary entries as well as large unlabelled corpus, both sources of information are incorporated into one representation. (Tissier, Gravier, and Habrard 2017) implements this method as `dict2vec`.

Deep Contextualised Model (Peters et al. 2018) introduced a model for "deep contextualised word representations which represent complex characteristics of words and word usage across different linguistic contexts". An *Embeddings from Language Models* (ElMo) representation is generated with by a function which operates on language at the sentence level.

In this report we describe algorithms for generating embeddings using the class of methods described here as the *co-occurrence matrix* category (which we refer to as "count models").

1.3 What are they for?

"The task of representing words and documents is part and parcel of most, if not all, Natural Language Processing (NLP) tasks. In general, it has been found to be useful to represent them as vectors, which have an appealing, intuitive interpretation, can be the subject of useful operations (e.g. addition, subtraction, distance measures, etc) and lend themselves well to be used in many Machine Learning (ML) algorithms and strategies."

Almeida and Xexéo (2019)

Although there are a variety of different word embedding algorithms, each being suited to a different niche of application, they are all united by the following traits:

Summary: Shared Characteristics of Word Embedding Algorithms

Semantic Comprehension They provide a means of quantitatively assessing the semantic-relatedness of words;

Convenient Form The semantic information is encoded by *vector* representations of vocabulary words;

Automatic Generation Process The semantic information is "*automatically* extracted from language data by looking at empirical evidence of real language use".

The *semantic comprehension* quality makes them appropriate for many tasks in natural language processing, and their *convenient form* (like Almeida and Xexéo notes above) in particular makes them work well as inputs to machine learning pipelines -Schütze (1993) notes: *"Neural networks perform best when similarity of targets corresponds to similarity of inputs; traditional symbolic representations do not have this property"*[4]. Finally, their *automatic generation process* makes them preferable to alternative means of assessing word-relatedness which depend on extensive manual knowledge-input, like building the structured lexical database: `wordnet` (Miller, Beckwith, et al. 1990).

P. D. Turney and Pantel (2010) include a thorough discussion of applications of *Vector Space Models* (the class of embedding algorithms referred to as *count models* in this report), listing the following examples of uses: word clustering, word classification, automatic thesaurus generation, word sense disambiguation, context sensitive spelling correction, semantic role labelling, query expansion, textual advertising & information extraction. Figure 1.4 illustrates the use of the embeddings from Figure 1.3 to perform the first task from this list of examples: automatic word clustering.

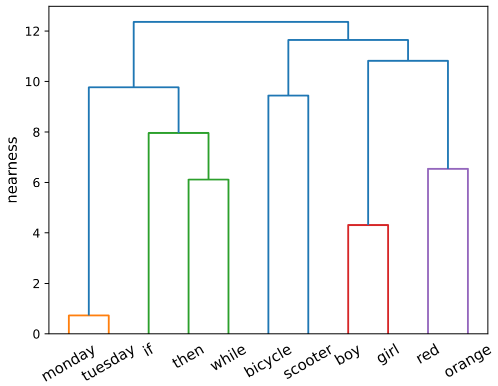


Figure 1.4: Word clusters identified by analysis of word vectors featured in Figure 1.3

Word embeddings do not have a predefined list of applications of course, they can be used as

4. The term "traditional inputs" here refers to "one-hot" (OH) representations which encode a word with a vector as long as the vocabulary, with all entries zero except for the index corresponding to the place of that word in the vocabulary, which has a value of 1 instead. These vectors contained no information about what words meant, and additionally they were often inconveniently long (as long as the vocabulary).

a tool any way that can be imagined for them. Novel applications we have come across in our research include: detecting signs of dementia (Mirheidari et al. 2018), learning representations of medical concepts (Choi, Chiu, and Sontag 2016), to investigate the nature of human language acquisition (Landauer and Dutnais 1997), & to map the changing biases encoded by language over the span of a century (Garg et al. 2018).

Example 1.3.1 *(Word Clustering Using Embeddings)*. Figure 1.4 shows the result of using a point-clustering algorithm on the embeddings from Figure 1.3 to identify groups of related words. The resulting clusters, identified entirely computationally, correspond with the groupings an English speaker would naturally select given the same word choices.

1.4 Beyond word embeddings

Word embedding algorithms extract semantic information about language without 'knowing' what language is. To these algorithms, language is just lists of discrete elements with underlying patterns to be identified. Due to this generality, word embedding algorithms can be used to encode the semantics of other types of ordered data. Many novel applications of the embedding process to capture semantic associations outside of the written language domain have been explored. Below we list some notable embedding applications we have come across in researching for this report:

Summary: Novel Applications of the Embedding Process

Graph Embeddings Grohe (2020) presents an overview of techniques which generalise the word embedding process to embed structured data -graphs (possibly labelled or weighted), or more generally relational structures, nodes of a large graph, or tuples appearing in a relational structure.

Recommendation Systems Grbovic and Cheng (2018) demonstrate the use of *user* and *listing* embeddings to improve real-time personalisation and "similar listing" recommendations at Airbnb, and J. Wang et al. (2018) employ item-graph embeddings generated from users' behaviour history to improve recommendation systems at the global e-commerce platform Alibaba.

Music Embeddings Chuan, Agres, and Herremans (2018) model functional chord relationships and harmonic associations by embedding slices of music from a large corpus music across eight genres.

Customer Lifetime Value Prediction Chamberlain et al. (2017) propose a method to generate embeddings of customers which provide daily estimates of the future value of every customer, employed by the ASOS fashion retailer.

While the focus of this document is on the use of embeddings to process language, it is our intention to present the theory of word embeddings in such a way that the underlying techniques *not specific to language processing* can be understood in their own right. We intend that novel applications like those mentioned above, falling outside of the field of NLP, can be approached with proper context following this primer.

1.5 Report Content & Structure

In chapter 2 we discuss words themselves: what are they and what does it mean to represent them? In chapter 2 we present key concepts and definitions necessary to make sense of the details of the different word embedding processes. In chapter 4 we describe two describe two methods for creating word spaces, one based on documents and one based on context-windows. Once we have presented these count-based embedding methods we compare their qualities to those of other possible approaches. Finally we provide references for Further Reading and make closing remarks on the content & concepts presented in this report in the Conclusion.

CHAPTER 2
WORDS

2.1 What are words?

“ To many people the most obvious feature of a language is that it consists of words ”

Halliday and Teubert (2004)

“ There have been many definitions of the word, and if any had been successful I would have given it long ago, instead of dodging the issue until now ”

Matthews (1991)

The ‘Word’ workshop, held at the international research centre for linguistic typology 12-16 August 2000 consisted of 16 presentations discussing the answer to this question. Ten of those were published in the book *Word : a cross-linguistic typology.* The difficulty of the question is summarised in the book’s conclusion:

“ The problem of the word has worried general linguists for the best part of a century. In investigating any language, one can hardly fail to make divisions between units that are word-like by at least some of the relevant criteria. But these units may be simple or both long and complex, and other criteria may establish other units. It is therefore natural to ask if `words' are universal, or what properties might define them as such. ”

P.H. Matthews, (Dixon 2002)

Some definitions of ‘word’ (like Bloomfield’s (1926) *‘A minimum free form is a word’*) are sufficient but not necessary, whereas others (like Lyons’s (1968) *‘a word may be defined as the unit of a particular meaning with a particular complex of sounds capable of a particular grammatical employment’*) is a potentially necessary, but certainly not sufficient (Dixon 2002). Is it possible to conceive a watertight definition?

Our intuition is often inconsistent, making it difficult to codify -consider *so called* filler-words like “uh” or “um”, or compound forms like “rock ‘n’ roll” and “New York” which are always used as whole unbroken units, what makes these *not* words? Consider that the words we agree on now are subject to ever changing convention; the word “tomorrow” is listed in Johnson’s 1755 dictionary of the English language as two distinct components “to”, and “morrow” despite being indisputably considered a word *to-day.* Halliday and Teubert (2004) ask “How do we decide about sequences like lunchtime (lunch-time, lunch time), dinner-time, breakfast time? How many words in isn't, pick-me-up, CD?".

An understanding of 'word' formed within the English language cannot blindly be applied beyond that scope either, Dixon (2002) notes "The idea of 'word' as a unit of language was developed for the familiar languages of Europe". Languages which are written without spaces, (like Chinese, Japanese, and Thai) present one particular example of the challenges in applying a local conception of 'word' beyond this scope: ambiguous sentence segmentation.

Example 2.1.1 *(Chinese Sentence Segmentation)***.** X. Chen et al. (2017) gives the following example of the difficulty in agreeing on 'word' boundaries in Chinese: The sentence "Yao Ming reaches the finals" (姚明进入总决赛) May be divided up to yield either three words or five using the two segmentation methods; 'Chinese Treebank' segmentation, or 'Peking University' segmentation, respectively:

姚明	进入	总决赛			
`YaoMing`	`reaches`	`finals`			(Chinese Treebank)

姚	明	进入	总	决赛	
`Yao`	`Ming`	`reaches`	`overall`	`finals`	(Peking University)

Different languages present different difficulties -In the case of Arabic, expressions formed by adding consecutive suffixes to a single root word (like the sequence *establish* → *establishment* → *establishmentarianism* in English) are a more essential part of the language, such that complete sentences can be expressed as unbroken 'word' units. As a result a 'word' unit which is identified by unbroken character sequences occurring between spaces will, in Arabic, represent larger more complex chunks of meaning than same framework would identify for comparable sentences in English. What is the 'correct' amount of information that our word-units should encode?

2.2 A 'word' framework

To clarify what is meant by 'word' beyond just a description of convention requires a new vocabulary: we need terms which allow us to describe the various commonalities of form and function which unify of these elements of everyday language. We present a framework due to Dixon (2002) to assist us toward this end:

Summary: Distinguishing between uses of the term *word* (Dixon 2002)

The word 'word' is used in many ways in everyday speech, and in much linguistic discourse. It is important to make certain fundamental distinctions:

1. Between a lexeme and its varying forms;
2. Between an orthographic word (something written between two spaces) and other types of word;
3. Between a unit primarily defined on grammatical criteria and one primarily defined on phonological criteria.

Let us briefly clarify what is meant by these terms before discussing their relevance to word embeddings:

Definition 2.2.1 – *Lexemes and inflected forms*

Lexemes are distinct *root forms* which make up the *lexicon* of a language (they are the elements of which dictionaries generally constitute), and serve to group so-called *inflected forms* which have a meaning related to their root. Lexemes are the object being referred to as 'word' in statements like "look, looks, looked and looking are forms of the same word". The same expression could also be phrased: "the lexeme *look* is realised as the word-forms: *look, looks, looked & looking*".

Example 2.2.1 *(Lexemes and inflected forms)***.** The example given above of contrasting the lexeme "look" to its inflected forms is presented here in a table:

root or underlying form	inflected forms	grammatical function
look	look	present, non-3sg subject
	looks	present, 3sg subject
	looked	past
	looking	participle

Table 2.1: An comparison between the lexeme for the English verb *look* and its inflected word-form realisations

Definition 2.2.2 – *Orthographic words*

Orthographic words are distinct elements of *orthography* (written language), most often defined as *something written between two spaces*.

Example 2.2.2 *(Orthographic words)***.** Orthographic words entirely reflect conventions of writing, so the form "cannot" is one orthographic word whereas the form "must not" (despite having the same apparent rules of usage) is two orthographic words.

Definition 2.2.3 – *Grammatical and phonological words*

This term describes the language-parts which result from applying a grammatical framework to recognise regular word-like elements in observed language. A grammatical framework is one concerned with the rules governing the arrangement of *morphemes* (the smallest distinct units of meaning) Dixon (2002) provides the following criteria for identifying grammatical words: A **grammatical word** consists of a number of grammatical elements which;

(a) always occur together, rather than scattered through the clause (the criterion of cohesiveness);

(b) occur in a fixed order

(c) have a conventionalised coherence and meaning

Dixon also provide a definition of *phonological word* in the same vein which we omit here since phonological words are less directly related to the embedding problem[1].

1. Whereas grammatical words are concerned with elements of a grammatical structure with a regular form, phonological words are concerned with elements of a phonological structure (consisting of phonemes: the smallest

Example 2.2.3 *(Grammatical words)***.** Grammatical words conform less to our conventional notion of ‘word’ but are more well behaved as they derive from a static criteria; The inflected forms of the lexeme *look* in example 2.2.1 (*look, looks, looked, looking*) are all examples of grammatical words, but so are the forms “New York”, “Rock ‘n’ Roll”, “dinner-time”, “breakfast time”, “isn’t”, “pick-me-up”, and “CD” (which we wondered about the proper place of at the start of this chapter). Since the criteria for identifying grammatical words takes meaning into account, different words with the same word-form (like *bank* (as in river) and *bank* (as in financial institution)) are also considered distinct grammatical words.

We highlight these distinctions because without this terminology it is easy to overlook an implicit *word=orthographic word* assumption made when working with word embeddings. This assumption is not uncommon, as Dixon notes:

> “ In many language communities a word is thought of as having (semantic, grammatical and phonological) unity and, in writing, words are conventionally separated by spaces ”

but it is worth discussing explicitly since our assumptions will be encoded by our embedding algorithms. The issues relating most to the generation of word embeddings regard *semantic* unity -if a word does not have semantic unity (i.e., orthographic word and semantic meaning are not one-to-one) then it is said to be *polysemous* (*poly* [many] *semous*→*semantic* [meanings]) and the resulting problem w.r.t. word embeddings is called *the problem of polysemy*. A related problem is that of representing grammatical words which consist of multiple orthographic-word parts -these are considered as separate parts and not as a whole. Ideally word

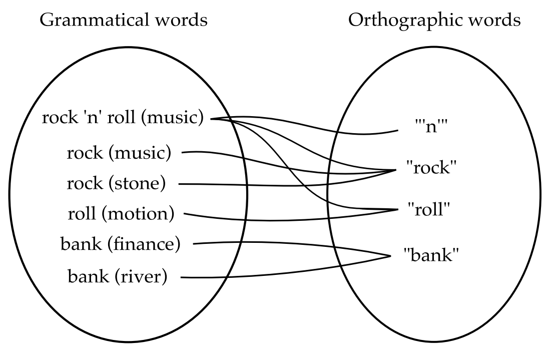


Figure 2.1: The *problem of polysemy* illustrated: Which sense of the orthographic words “bank” or “rock” should their embeddings encode? and what about the embedding of the grammatical word “rock ‘n’ roll”, not present in the orthographic words?

distinct units of vocalised language) with a regular form. Since word embeddings deal with written language the vocal realisation of language is of less significance.

embeddings algorithms would describe a bijective map between *grammatical words* and points in word space. Since in actual fact word embedding algorithms operate on *orthographic words*, the idiosyncrasies of the map (called writing) between grammatical words and orthographic words (illustrated in Figure 2.1) is inherited by the resulting representations. The surjectivity of this map creates the problem of polysemy and the one-to-many nature of the map means multi-part grammatical words are misrepresented too. Halliday and Teubert discuss some of the pitfalls of seeking *fixed, representative* meanings of words in natural language in the following excerpt:

> “ As users of language we know that someone's mention of a recent television programme about big cats in Africa implies a different meaning of cat from a reference to the number of stray cats in the city of New York. And if someone talks about 'letting the cat out of the bag' or 'setting the cat among the pigeons', we know that the meaning has to be taken from the whole expression, not from a word-by-word reading of Felis catus jumping out of a bag or chasing Columbidae. Any good dictionary recognises this by such strategies as listing different senses of a word, giving examples of usage, and treating certain combinations of words (such as idioms) as lexical units. But it is important to recognise that this contextualisation of meaning is in the very nature of language and not some unfortunate deviation from an ideal situation in which every word of the language always makes exactly the same semantic contribution to any utterance or discourse. **For reasons such as these, we should be cautious about the view that words have a basic or core meaning, surrounded by peripheral or subsidiary meaning(s)** ”

Clearly these qualities of language, particularly in its written form, present some difficulties for embedding algorithms. Despite this however, even the earliest embedding methods, which did not take measures to address these concerns, reported that they *did* capture semantic information that was useful and interesting (Deerwester et al. 1990).

The embedding methods we present in this report (elementary count models based on document-term or word-context matrices) do not address the problem of polysemy or the many-to-one problem of multi-part grammatical words, but with the information presented here the reader should: ① know the compromises made by models not accounting for these factors, and ② have the context and prerequisites upon finishing this report to understand methods which *do* address the issues. In particular the Deep Contextualised Model addresses polysemy concerns, and embeddings operating on n-grams (N-gram Model) address the many-to-one problem.

CHAPTER 3
EMBEDDING METHODS: PRELIMINARIES

In this chapter we outline briefly terms and distinctions to be used by the embedding algorithms discussed in later chapters.

3.1 Types and Tokens

To process language, it is useful to make the following distinction between two ways of talking about at word-instances, identified by the different ways they are counted:

Definition 3.1.1 – *Types and tokens*

When we count the number of unique vocabulary items in a document we call the things counted *word-types*, whereas when we add up the number of words in a "word count" repetitions are permitted and we call the things counted *word tokens*.

Consider the following example:

Example 3.1.1 *(Types and tokens)*.

"The limits of my language means the limits of my world"

Wittgenstein (1922)

This sentence contains 11 word-tokens, but only 7 word-types because the four words "*the limits of my*" are each repeated once.

This distinction is very useful when processing language, however, as Kaplan (1990) notes, this type/token distinction is very much an *orthographic* construction -this is most straightforwardly apparent in the case of identical tokens which represent separate "grammatical words" (like the different grammatical words 'rock (music)' and 'rock (stone)' both represented by the same orthographic word 'rock' in Figure 2.1). Kaplan discusses the shortcomings of this type/token model as a tool for *understanding* language in depth, asserting "The criterion of word identity is not resemblance" -however, whether this distinction has *explanatory power* regarding language or otherwise, it inarguably has *utility* when processing language.

3.2 Distributional Similarity & Semantic relatedness

Up until this point we have used the terms "semantic relatedness" and "word similarity" interchangeably without clarifying what we are referring to in either case. We have said that word embedding algorithms use observed *distributional similarity* to infer underlying *semantic relatedness/word similarity*, but not discussed the specifics of this inference. In this section we first define a variety of ways in which words can be semantically related, then we discuss types of distributional similarity which can reveal semantic relations, and finally we discuss the limits of this methodology.

We reference P. D. Turney and Pantel's (2010) framework for distinguishing between types of semantic relatedness. First, they describe the general concept of semantic relatedness as follows:

> "The term semantic relatedness in computational linguistics (Budanitsky and Hirst 2001) corresponds to attributional similarity in cognitive science (Gentner 1983). Two words are semantically related if they have any kind of semantic relation (Budanitsky and Hirst 2001); they are semantically related to the degree that they share attributes (Peter D. Turney 2006). Examples are synonyms (bank and trust company), meronyms (car and wheel), antonyms (hot and cold), and words that are functionally related or frequently associated (pencil and paper). We might not usually think that antonyms are similar, but antonyms have a high degree of attributional similarity (hot and cold are kinds of temperature, black and white are kinds of colour, loud and quiet are kinds of sound)."

Then they present two particular semantic relations ("taxonomic similarity" and "semantic association") which can be characteristically linked with distributional circumstances (termed "paradigmatic parallels", or "syntagmatic associates" Schütze (1993)). We paraphrase their definitions here:

Summary: Types of semantic relation (P. D. Turney and Pantel 2010), & related manners of distribution (Resnik 1995)

The following ways that words can be semantically related:

Semantic association Words which tend to co-occur frequently (e.g. *bee* and *honey*)

Taxonomic similarity Words which share a hypernym (*car* and *bicycle* are taxonomically similar, because they share the hypernym *vehicle*).

Are associated with these two following manners in which words can be *distributionally* related:

Syntagmatic associates Words which tend to be each other's neighbours.

Paradigmatic parallels Words which tend to share the same neighbours.

This link between semantic relation and distributional is justified by the observation that words which are each other's neighbours (syntagmatic associates like *bee* and *honey*) "are often different parts of speech", whereas words which share similar neighbours (taxonomically similar words like *doctor* and *nurse*) "are usually the same part of speech".

The inference we can make based on distributional conditions then, is this:

$$\boxed{\textit{syntagmatic associates}} \Longrightarrow \boxed{\textit{semantic associates}} \tag{3.1}$$

$$\boxed{\textit{paradigmatic parallels}} \Longrightarrow \boxed{\textit{taxonomically similar}} \tag{3.2}$$

We will discuss in the following chapters how co-occurrence matrix based embedding methods algorithms can be used to generate word spaces reflecting more *syntagmatic association* or *paradigmatic parallel* type distributional relationships. As you will see, however, the ability to target specific semantic relationships with these algorithms is quite coarse-grained -even if these two categories of semantic relation (semantic association and paradigmatic parallelism) could be captured with some precision, what about the more specific types of relation mentioned by P. D. Turney and Pantel (synonymy, meronymy & antonymy) at the start of this section, or other relations?

> "A commonly raised criticism for both types of semantic space models (i.e., word-based and syntax-based)[1]concerns the notion of semantic similarity. Proximity between two words in the semantic space cannot indicate the nature of the lexical relations between them. Distributionally similar words can be antonyms, synonyms, hyponyms or in some cases semantically unrelated. This limits the application of semantic space models for NLP tasks which require distinguishing between lexical relations"
>
> Padó and Lapata (2003)

This limit of word space models (referred to as *semantic space models* by Padó and Lapata above) is in fact essential to their construction. The semantic relatedness they model is of a more general nature. This is not necessarily a shortcoming of the models however. Sahlgren (2006) writes in his thesis:

1. The term "syntax based" used in contrast to "word based" refers to models which make use of an a-priori encoded conception of word-order relationships like SUBJECT-OBJECT.

> “ This criticism is arguably valid from a prescriptive perspective where these relations are a priorily given as part of the linguistic ontology. From a descriptive perspective, however, these relations are not axiomatic, and the broad notion of semantic similarity seems perfectly plausible. There are studies that demonstrate the psychological reality of the concept of semantic similarity. For example, Miller and Charles (1991) [...]; people appear to instinctively understand what semantic similarity is [...]. Several researchers report high inter-subject agreement when asking a number of test subjects to provide semantic similarity ratings for a given number of word pairs (Rubenstein and Goodenough 1965; Henley 1969; Miller and Charles 1991) ”

Indeed, it is not clear how these more particular semantic relationships *could* be captured by the arrangements of points in a word space -Sahlgren continues:

> “ the nature of the similarities in the word-space model is a consequence of using the distributional methodology as discovery procedure, and the geometric metaphor of meaning as representational basis. [...] If we want to claim that we extract and represent some particular type of semantic relation in the word-space model, we need to modify either the distributional hypothesis or the geometric metaphor. ”

3.3 Pointwise Mutual information

The next concept is from the field of information theory and measures the amount of information revealed about a system when its outcome is partly decided. It is useful in embedding processes for quantifying how novel it is for two words to have co-occurred together (very frequent words like “the” are less novel partners to co-occur with than more domain-specific terms).

Definition 3.3.1 – *Pointwise Mutual Information*

Given two discrete random variables with a joint distribution $P : X \times Y \to [0,1]$, such that the probability of x and y co-occurring is given by $P(x,y)$, the *pointwise mutual information* of x and y is given by:

$$I(x,y) = \log \frac{P(x,y)}{P(x)P(y)} \tag{3.3}$$

Pointwise mutual information is a measure of how much information about one random variable's value is gained, given that the value taken by the other is already known. For this reason it is natural to describe it in terms of conditional probabilities:

$$I(x, y) = \log \frac{P(x \mid y)}{P(x)} \tag{3.4}$$

This value happens to be symmetric with respect to x and y:

$$I(x, y) = \log \frac{P(x \mid y)P(y)}{P(x)P(Y)} \;=\; \log \frac{P(y \mid x)P(x)}{P(y)P(x)} = \log \frac{P(x, y)}{P(x)P(y)} \tag{3.5}$$

which is why it is called the pointwise *mutual* information.

If all outcomes in a joint distribution are equally likely, then learning the value of one of the variables does not give you any new information about the probability that the remaining variable takes any particular value. If the two are highly dependent on one another however, the outcome of one can be very informative about the probable outcomes for the other. This is the intuition which pointwise mutual information formalises.

Example 3.3.1 *(Pointwise mutual information)*. Given the definition we expect that for the two joint distributions (P_1, P_2, on X and Y shown in Table 3.1 and Table 3.2), in the first case the pointwise mutual information for particular values (we choose x_3 and y_3) would be none, and in the second case would be positive. We calculate both below, referring to the first value I_1 and the second I_2.

	z_1	x_2	x_3	x_4	
y_1	1/16	1/16	1/16	1/16	0.25
y_2	1/16	1/16	1/16	1/16	0.25
y_3	1/16	1/16	1/16	1/16	0.25
y_4	1/16	1/16	1/16	1/16	0.25
	0.25	0.25	0.25	0.25	1

Table 3.1: Case 1: X and Y with constant joint distribution P_1.

	x_1	x_2	x_3	x_4	
y_1	0.08	0.09	0.03	0.06	0.26
y_2	0.05	0.08	0.04	0.13	0.30
y_3	0.11	0.03	0.03	0.06	0.23
y_4	0.05	0.10	0.02	0.04	0.21
	0.29	0.30	0.12	0.29	1

Table 3.2: Case 2: X and Y with varied joint distribution P_2.

$$\begin{aligned} I_1(x_3, y_3) = \log \frac{P_1(x_3, y_3)}{P_1(x_3)P_1(y_3)} &= \log \frac{1/16}{0.25 \cdot 0.25} \\ &= \log 1 \ (= 0) && \text{(first case)} \end{aligned}$$

$$\begin{aligned} I_1(x_3, y_3) = \log \frac{P_2(x_3, y_3)}{P_2(x_3)P_2(y_3)} &= \log \frac{0.03}{0.23 \cdot 0.12} \\ &= \log \frac{25}{23} \ (> 0) && \text{(second case)} \end{aligned}$$

You can see that the calculated PMI values conform with the intuition we described above.

3.4 Tokenizing

In this section we present a formal description of *tokens*, the core unit upon which embedding algorithms operate on, in mathematical language. We did not find a formal description of tokens as mathematical elements like this in the literature -it is often assumed that a tokenized corpus is already prepared and understood- so we briefly present our own.

Since one of our aims in this report is to highlight the fact that many key ideas used to generate word spaces do not *assume* that the objects being operated on are actually words (discussed in Beyond word embeddings), we thought a formal description of type of input generally expected by these algorithms would be a useful detail to specify.

Definition 3.4.1 – *Orthography*

An orthography is a finite set of symbols.

Example 3.4.1. The sets:

$\mathscr{O}_P = \{$🟢, 🔵, 🔴$\}$ (3.6)

$$\mathscr{O}_L = \{c \mid c \text{ is a letter, number or special character marked on Figure 3.1}\} \tag{3.7}$$

are both examples of orthographies.

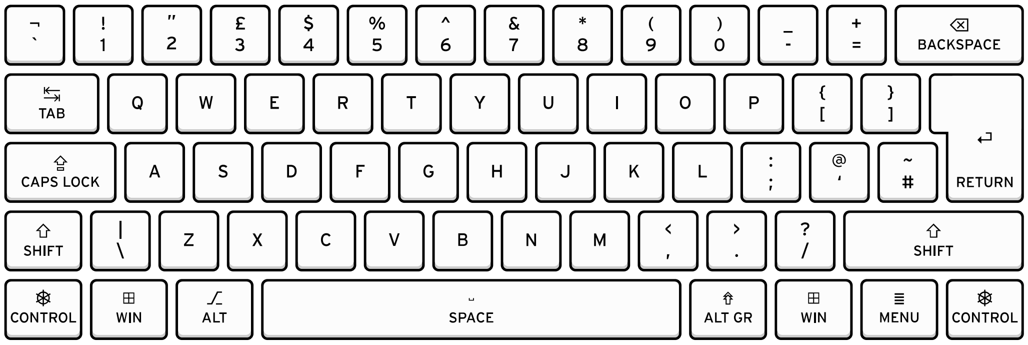
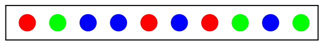

¬ ` ! 1 " 2 £ 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = BACKSPACE
TAB Q W E R T Y U I O P { [}]
CAPS LOCK A S D F G H J K L : ; @ ' ~ # RETURN
SHIFT | \ Z X C V B N M < , > . ? / SHIFT
CONTROL WIN ALT SPACE ALT GR WIN MENU CONTROL

Figure 3.1: A standard uk keyboard layout

Definition 3.4.2 – *Document*

A document d is a finite sequence $c_0, c_1, \ldots, c_n$ $(c \in \mathscr{O})$, where $\mathscr{O}$ is an orthography.

Example 3.4.2 *(Dots Documents).* The sequence of symbols:

$d_1 =$ 

is a document using orthography (3.6).

Example 3.4.3 *(Plato Documents).* Plain text versions of the complete works of Plato (English translations) are available at the Project Gutenberg website. These are examples of documents using keyboard orthography 3.7 referenced above.

Definition 3.4.3 – *Word-Token*

A word token w is like a document -it is a sequence of symbols $c_0, c_1, \ldots, c_n$ $(c \in \mathscr{O}$ from an orthography $\mathscr{O}$. The difference between a word token and a document is context: we call a sequence of symbols from an orthography a *word token* when it is the output of a tokenizer (defined next).

Definition 3.4.4 – *Tokenizer*

A tokenizer T is a function that maps a document to a sequence of word tokens; $T(d) = w_1, w_2, \ldots, w_n$. The output is referred to as the 'tokenized' form of the document. We use the shorthand W_d to refer to the tokenized document $T(d)$ when it is clear which tokenization has been used.[2].

Example 3.4.4 *(Dots Document Tokenized)***.** Using the tokenizing procedure: *treat blue dots as token-deliminators*, the tokenization of document (3.4.2) yields:

$$
\begin{aligned}
T(d_1) &= \boxed{\text{🔴 🟢}}, \boxed{\text{🔴}}, \boxed{\text{🔴 🟢}}, \boxed{\text{🟢}} \qquad (3.8)\\
&= W_{d_1}
\end{aligned}
$$

Example 3.4.5 *(Simple Text Tokenizer)***.** In algorithm 1 we present a simple algorithm for tokenizing text which ignores a predefined list of special symbols, and identifies token-boundaries using predefined deliminator symbols.

Setting ignore= (”!”,”.”,”,”,”;”,” : ”,”′”), and deliminators= (” ”), the following document:

$$d_2 = \boxed{\text{Predicting is difficult -especially about the future.}}$$

using the orthography (3.7) has the tokenized form:

$$
\begin{aligned}
T(d_2) &= \boxed{\text{Predicting}}, \boxed{\text{is}}, \boxed{\text{difficult}}, \boxed{\text{-especially}}, \boxed{\text{about}}, \boxed{\text{the}}, \boxed{\text{future}} \qquad (3.9)\\
&= W_{d_2}
\end{aligned}
$$

Clearly the token $\boxed{\text{-especially}}$ is not in the form an English speaker would transcribe if tasked with tokenizing the sentence by hand -but we use this example to highlight the difficulty of tokenizing: with this approach hyphenated expressions like “brother-in-law” will be correctly tokenized as single units, but in cases like (3.9) mistakes are made. It is not possible to handle both cases correctly with our simple algorithm.

2. Here we consider tokenization in an abstract sense essentially saying “A tokenizer is a function that extracts tokens”. For a practical look at pre-processing see Further Reading.

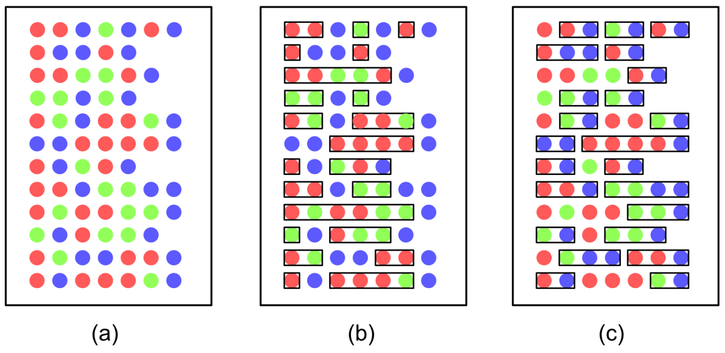


Figure 3.2: Tokens identified by two different tokenizations, (b) & (c), performed on the same document (a) (using the orthography (3.6)). The tokenizer used in the case of (b) identifies tokens as sequences of red and green dots delimited by blue dots, and for (c) the tokenizer identified tokens as sequences of consecutive repeated symbols which are terminated by any number of blue dots.

```
input  : A list of symbols: document, a list of ignored "special" symbols: ignore, and a list of
         deliminator symbols: deliminators
output: A list of word tokens: tokens
tokens ← list()
next_token ← list()
/* Go through the document one symbol at a time, skipping special characters, and
   adding next_token to tokens when a deliminator is reached */
for symbol in document do
    if symbol is in deliminators then
        /* Add next_token to tokens if next_token only if it contains symbols */
        if next_token is not empty then
            tokens ← append(next_token, tokens)
            next_token ← list()
        else
            /* Keep parsing symbols to add to next_token until the next deliminator is
               reached */
        end
    else if symbol is in ignore then
        /* Ignore this symbol */
    else
        /* Add this symbol to the next_token variable */
        next_token ← append(symbol, next_token)
    end
end
return(tokens)
```

Algorithm 1: Simple Text Tokenizer

We have described tokenization for orthographies of letter-based and dot-based documents but of course there are many other possibilities, for example tokenizing musical notation by delimiting tokens at bar-boundaries.

Definition 3.4.5 – *Corpus*

A corpus C is a finite set of documents $\{d_1, d_2, \ldots, d_n\}$, $n \in \mathbb{N}$.

Example 3.4.6 *(Plato Corpus).* The complete works of Plato in plain-text form, referenced in example 3.4.4 is an example of a corpus.

Definition 3.4.6 – *Vocabulary*

The vocabulary of a tokenized document W_d is the set of unique word tokens in the tokenization of that document; $V(W_d) = \{w \mid w \in$ the sequence $W_d\}$. The vocabulary of a corpus is the union of the vocabularies of each document in that corpus; $V(C) = V(W_{d_1}) \cup V(W_{d_2}) \cup \cdots \cup V(W_{d_n})$.

Definition 3.4.7 – $\#$ *Operator*

The $\#$ symbol, used preceding a set or a sequence, denotes "the number of elements in". $\#W_d$ denotes the number of tokens in the tokenized form of document d then, and $\#V(W_d)$ denotes the vocabulary size of document d.

Example 3.4.7 *(Dots Document Properties).* The vocabulary of the tokenized document (3.8) is:

$$\begin{aligned} V(W_{d_1}) &= V(\boxed{\text{🔴 🟢}}, \boxed{\text{🔴}}, \boxed{\text{🔴 🟢}}, \boxed{\text{🟢}}) \\ &= \{\boxed{\text{🔴 🟢}}, \boxed{\text{🔴}}, \boxed{\text{🟢}}\} \end{aligned}$$

The number of tokens (the word count) for this document is: $\#W_{d_1} = 4$ and the vocabulary size is $\#V(W_{d_1}) = 3$

Example 3.4.8 *(Plato Document Sizes).* Word token counts and vocabulary sizes for a selection Plato documents from 3.4.4, tokenized using a variation of the simple tokenizer[3]described by algorithm 1, are listed here:

Attribute	Document (d)					
	`laws`	`republic`	`cratylus`	`meno`	`apology`	`crito`
$\#W_d$	140,406	118,283	23,883	12,716	11,392	5,332
$\#V(W_d)$	8,100	8,105	2,963	1,493	1,789	1,030

Table 3.3: Number of Word Tokens & Vocabulary Sizes of a selection of Documents in the Plato Corpus (example 3.4.6)

Example 3.4.9 *(Plato Corpus Size)***.** The complete Plato corpus (3.4.6) consists of 25 documents and, each tokenized the same way as before, has the word count $\sum_{d \in C_P} \#W_d = 670,936$ and the vocabulary size $\#V(C_P) = 19,432$.

These are the essential mathematical objects which are used to construct co-occurrence matrices as we describe in the following chapter.

3.5 Similarity Measure

There are many possible measures of similarity for comparing two vectors, but in the word embedding domain by far the most prevalent is the cosine similarity. We present only this method, but suggest reading Bullinaria and J. P. Levy (2007) for a discussion and comparison of other distance measuring approaches.

Definition 3.5.1 – *Cosine Similarity*

Give two non-zero vectors $\mathbf{x}$ and $\mathbf{y}$, the cosine similarity between these two vectors is given by:

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||||\mathbf{y}||} \tag{3.10}$$

where $\mathbf{x} \cdot \mathbf{y}$ is the dot product of $\mathbf{x}$ and $\mathbf{y}$ and $||\mathbf{v}||$ is the norm of $\mathbf{x}$. This formula is derived from the dot product formula: $\mathbf{x} \cdot \mathbf{y} = ||\mathbf{x}||||\mathbf{y}|| \cos \theta$.

3. The code used to produce this table, and other examples using this data set, is available on Github.

CHAPTER 4
CASE STUDY: COUNT MODELS

So-called *count models* were the first methods investigated for creating word embeddings. They emerged from work in the field of information retrieval (IR). In particular, influential contributions (Dubin 2004) were made by Salton, Wong, and Yang's (1975) work on the SMART information retrieval system which "pioneered many of the concepts that are used in modern search engines" (P. D. Turney and Pantel 2010). Initially word-similarity was not the focus of the developed models, but instead *document similarity* was (for matching search queries to relevant/related material), however Deerwester et al. (1990) observed that by looking at row vectors rather than column vectors in a document-term matrix the objects described were instead *word vectors.*

Co-occurrence matrices, the heart of *count models* do what they say: they count instances of co-occurrence. They can be thought of like big tally tables.

We use the framework described by P. D. Turney and Pantel (2010) to group "count models" according to the structure (choice of rows and columns) of the co-occurrence matrix they create. In the case of document-term matrices which we discuss first, the rows represent documents and the columns represent terms in the vocabulary of the document corpus. In the case of word-context matrices the rows represent target words (occurring at the centre of a context-window), and the columns represent context words (occurring in a target word's context-window).

In the following sections we define these terms concretely.

4.1 Document-term Matrices

Definition 4.1.1 – *Document-term co-occurrence count*

In the context of document-term matrices, the function $f(w, W_d)$ indicates the number of times the token w appears in a tokenized document W_d.

$$f(w, W_d) = \#\{t \in W_d \mid t = w\} \tag{4.1}$$

Definition 4.1.2 – *Document-term matrices*

Given a corpus of documents $C = d_1, d_2, \ldots, d_n$, and a tokenizer T, a document-term matrix $\boldsymbol{X}$

is given by:

$$\boldsymbol{X} = \begin{bmatrix} f(w_1, W_{d_1}) & f(w_1, W_{d_2}) & \dots & f(w_1, W_{d_n}) \\ f(w_2, W_{d_1}) & f(w_2, W_{d_2}) & \dots & f(w_2, W_{d_n}) \\ \vdots & \vdots & \ddots & \vdots \\ f(w_m, W_{d_1}) & f(w_m, W_{d_2}) & \dots & f(w_m, W_{d_n}) \end{bmatrix} \tag{4.2}$$

where the w_i terms are from the vocabulary of C, $V(C) = w_1, w_2, \dots, w_m$. The number of times the i^{th} word appears in the j^{th} document then, is given by $\boldsymbol{X}_{ij}$.

Example 4.1.1 *(Plato Document Vectors)***.** A document-term co-occurrence matrix has document-vectors for columns and word-vectors for rows. Here we present a section of a document-term matrix for the tokenized Plato corpus, and show in Figure 4.1 how these counts can be used to position document-vectors in a term-space.

Term	Document (d)					
	`laws`	`republic`	`cratylus`	`meno`	`apology`	`crito`
law	436	54	2	0	9	5
virtue	127	81	6	142	10	4
the	9,296	7,048	1498	455	493	239
earth	24	29	14	0	5	0
god	119	59	22	3	25	1
socrates	0	68	65	57	18	28

Table 4.1: A document-term co-occurrence matrix for selected documents from the Plato corpus and selected terms.

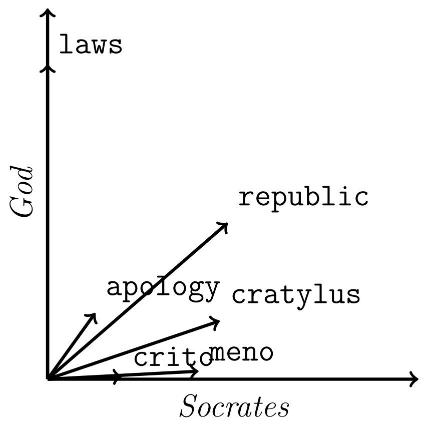


Figure 4.1: Works of Plato arranged in a two dimensional term-space according to the number of times the terms “God” and “Socrates” occurred in their tokenized forms.

The original purpose of these document-term matrices was to help relating the content of a document to a document-query. So, using our table above, if a query about Plato’s writings contained the word “God” it would be more readily linked to `laws` and `apology`, whereas if the query contained “Socrates” the link would be with `crito` and `meno` instead.

Deerwester et al. (1990) observed that this same procedure could be used to measure word-similarity, however “a document is not necessarily the optimal length of text for measuring word similarity”. So the concept of a *word-context* matrix is conceived.

Definition 4.1.3 – *Context window*

given a token w_i from a tokenized document W_d, the function $\text{context}(w, b)$ returns a sequence containing the following and preceding b tokens from W_d not including w_i itself. The variable b is called the window size.

$$\text{context}(w_i, b) = (w_{i-b},\ \dots,\ w_{i-1},\ w_{i+1},\ \dots,\ w_{i+b}) \tag{4.3}$$

Example 4.1.2 *(Context window).* A context window around the word “next” in the quote (4.4) with a window size of 3,

$$\boxed{\texttt{We}}\ \boxed{\texttt{know}}\ \underset{-3}{\boxed{\texttt{not}}}\ \underset{-2}{\boxed{\texttt{what}}}\ \underset{-1}{\boxed{\texttt{comes}}}\ \underset{0}{\boxed{\texttt{next}}},\ \underset{+1}{\boxed{\texttt{or}}}\ \underset{+2}{\boxed{\texttt{what}}}\ \underset{+3}{\boxed{\texttt{follows}}}\ \boxed{\texttt{after}}.\quad \text{— Virginia Woolf} \tag{4.4}$$

is given by the sequence:

$$\text{context}\left(\boxed{\texttt{next}}, 3\right) = \left(\boxed{\texttt{not}}, \boxed{\texttt{what}}, \boxed{\texttt{comes}}, \boxed{\texttt{or}}, \boxed{\texttt{what}}, \boxed{\texttt{follows}}\right)$$

Definition 4.1.4 – *Word-context co-occurrence count*

Given a tokenized document W_d, we denote the number of times a token c appears in the context of another token w: $\#_c$ context(w). Then, we denote the total number of times c appears in the context of any instance of the w token throughout the document W_d by

$$f(w, c, W_d) = \sum_{\{t \in W_d \mid t = w\}} \#_c \text{ context}(t) \tag{4.5}$$

and, given a corpus of documents C we denote the sum of all these co-occurrence counts over all the documents in the corpus:

$$F(w, c) = \sum_{d \in C} f(w, c, d) \tag{4.6}$$

From here it is straightforward to define the word-context co-occurrence matrix:

Definition 4.1.5 – *Word-context co-occurrence matrix*

Given a corpus of documents $C = d_1, d_2, \dots, d_n$, and a tokenizer T, a term document matrix $\boldsymbol{X}$ is given by:

$$\boldsymbol{X} = \begin{bmatrix} F(w_1, c_1) & F(w_1, c_2) & \dots & F(w_1, c_m) \\ F(w_2, c_1) & F(w_2, c_2) & \dots & F(w_2, c_m) \\ \vdots & \vdots & \ddots & \vdots \\ F(w_m, c_1) & F(w_m, c_2) & \dots & F(w_m, c_m) \end{bmatrix} \tag{4.7}$$

where the w_i and c_i terms are both from the vocabulary of C, $V(C) = w_1, w_2, \dots, w_m = c_1, c_2, \dots, m$. The number of times the i^{th} word appears in a context window with the j^{th} context word is then given by $\boldsymbol{X}_{ij}$.

Example 4.1.3 *(Meno's answer)*. A word-context co-occurrence matrix generated using this procedure with a window size of 10, using the works of Plato as its corpus identified these words as the ten *most similar* (using the cosine measure of similarity) to "virtue":

word	virtue	wisdom	a	justice	this	an	men	courage	knowledge	all
similarity	0.999	0.981	0.980	0.978	0.977	0.976	0.972	0.972	0.972	0.972

Table 4.2: The most similar words to "virtue" according to a word-context matrix trained on the works of Plato.

clearly stopwords like "a" and "this" have been included because of their high frequency in all contexts -this problem is discussed in section 4.2, however we see that some element of Plato's conception of virtue *has* been captured by the proximity of words *wisdom, justice, courage, and knowledge.*

The consequence of using a particular one these two main co-occurrence matrix types (with documents as the basis elements or with context words as basis elements) corresponds the difference in semantic relatedness discussed in section 3.2. Words that notably co-occur in the context of a document are likely to be *paradigmatic parallels*, whereas words which co-occur in the context of a window around a target word are likely to be *semantically similar.* Sahlgren (2006) explores this connection in more depth in his thesis.

4.2 Weighting

The problem with raw co-occurrence counts is, like we see in Table 4.2, that "stop words" like *the*, which tell us very little about the meaning of a given passage, are the highest weighted elements whereas domain specific terms which are the most informative only occur infrequently so contribute less to the overall position of a count-based vector.

We remedy this by applying a weighting function to the elements of $\boldsymbol{X}$. *"The idea of weighting is to give more weight to surprising events and less weight to expected events"* (P. D. Turney and Pantel 2010).

For document-term matrices the most commonly used weighting method is the "term frequency $\times$ inverse document frequency" approach. The idea is that a term's weight should be high if it was frequent in that document but rare in other documents. This formulation has two parts: the term frequency and the inverse document frequency. Both of these parts have a variety of forms, often justified heuristically (S. Robertson (2004) includes a good discussion of justifications for the method, concluding that the theory of relevance weights (S. E. Robertson and Jones 1976) makes the strongest theoretical foundation) but nevertheless these methods "have proved extraordinarily robust and difficult to beat, even by much more carefully worked out models and theories" (S. Robertson 2004).

Definition 4.2.1 – *Term frequency*

The term frequency should be higher if the term is very frequent in the document and lower otherwise. Given a corpus C, a tokenizer T, and a co-occurrence count function f such that $f(w, W_d)$ equals the number of times that a token w appears in a tokenized document W_d, the term frequency $\times$ inverse document frequency weighting for w & d is given by:

$$\mathrm{tf}(w, W_d) = \frac{f(w, W_d)}{\sum_{t \in W_d} f(t, W_d)} \tag{4.8}$$

Definition 4.2.2 – *Inverse document frequency*

The inverse document frequency captures the idea of a 'rare' word: it should be higher if the term appears only in very few documents, and lower if the term appears in the majority of documents. First we define the document frequency:

Given a corpus C, a tokenizer T, and a token t, the document frequency df of the term t is given by:

$$\mathrm{df}_{t,d} = \frac{\#\{d \in C \mid t \in W_d\}}{\#C} \tag{4.9}$$

The "inverse document frequency" idf is then given by $\frac{1}{\mathrm{df}_t}$, however, this raw frequency is commonly scaled with a $\log$ function. The heuristic justification for this is because the significance of a term does not increase in proportion to its frequency -we want to emphasise the less frequent and "squash" the weights of the more frequent. Finally this gives us:

$$\mathrm{idf}_t = \log \frac{1}{\mathrm{df}} \tag{4.10}$$

Definition 4.2.3 – *TF-IDF weighting*

The final weight is then given by combining these terms:

$$\mathrm{tfidf}_{t,d} = \mathrm{tf}_{t,d}\,\mathrm{idf}_t \tag{4.11}$$

For word-context matrices the most common approach is to use a pointwise-mutual-information weighting (PMI).

The concept of pointwise mutual information (PMI) is defined in terms two jointly distributed random variables X and Y. In the case of word-context matrices we are interested in the joint distribution of two tokens throughout the text corpus, ww, and c. We apply the concept of pointwise mutual information by approximating the joint distribution probabilities of w and c.

Definition 4.2.4 – *PMI weighting*

Recall that the formula for PMI is:

$$I(w,c) = \log \frac{P(w,c)}{P(w)P(c)}$$

Given a corpus C, a tokenizer T, and a word-context co-occurrence count function F such that $F(w,c)$ equals the number of times w appeared with c in its context, approximate the value of the terms $P(w,c), P(w), P(c)$ as with the terms $p_{w,c}, p_w, p_c$ defined as follows:

$$p_{w,c} = \frac{F(w,c)}{\sum_{s\in V(C)} \sum_{t\in V(C)} F(t,s)} \tag{4.12}$$

$$p_w = \frac{\sum_{t\in V(C)} F(w,t)}{\sum_{s\in V(C)} \sum_{t\in V(C)} F(t,s)} \tag{4.13}$$

$$p_c = \frac{\sum_{t\in V(T)} F(c,t)}{\sum_{s\in V(C)} \sum_{t\in V(C)} F(t,s)} \tag{4.14}$$

Then the PMI weight is defined:

$$\mathrm{pmi}(w,c) = \log\left(1 + \frac{p_{w,c}}{p_w p_c}\right) \tag{4.15}$$

The $+1$ within the logarithm is to avoid evaluating the undefined $\log 0$ since the numerator $(p_{w,c})$ is often equal to zero because the tokens w and c never co-occurred.

4.3 Dimensionality Reduction

One issue with the word spaces produced by count models is they are inconveniently high dimensional (as many dimensions as there are word-types in the vocabulary), and that they are *sparse*: a large number of the entries are zero.

One common approach to tackle this problem is to factor the sparse matrix into smaller dense matrices using a technique called *singular value decomposition* (SVD), a method first employed by Dumais et al. (1988). Another approach is *feature selection* where only those terms considered informative beyond a certain threshold are included in the co-occurrence matrix and the rest are discarded.

We describe the basic operation of SVD here:

Definition 4.3.1 – *Singular value decomposition (Meyer 2000)*

For each $\boldsymbol{A} \in \Re^{m\times n}$ of rank r there are orthogonal matrices $U_{m\times n}, V_{n\times n}$ and a diagonal matrix $D_{r\times r} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ such that:

$$\boldsymbol{A} = \boldsymbol{U} \begin{pmatrix} \boldsymbol{D} & 0 \\ 0 & 0 \end{pmatrix}_{m\times n} \boldsymbol{V^T} \quad \text{with} \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \tag{4.16}$$

The σ_is are called the nonzero *singular values* of $\boldsymbol{A}$.

The product $\boldsymbol{A} = \boldsymbol{U\Sigma V^T}$ can be written:

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{4.17}$$

The key property that makes the SVD useful is that the product of *truncated* matrices U_k, Σ_k, V_k^T (only the first k columns of U, only the top left $k \times k$ of Σ_k and only the top k rows of V^T) yields lower dimensional approximation of A: A_k:

$$\boldsymbol{A} \approx A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T \tag{4.18}$$

Figure 4.2 illustrates this principle.

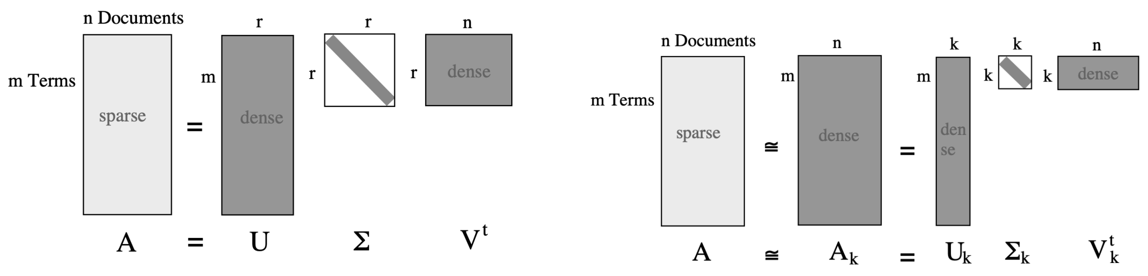


Figure 4.2: The creation of a more dense, lower-dimensional approximation $\boldsymbol{A}_k$ of a matrix $\boldsymbol{A}$ using singular value decomposition (Albright 2004)

4.4 Comparison with other models

Count models were the dominant approach to word embedding until Mikolov, K. Chen, et al. (2013) described a method for *efficiently* generating embeddings using techniques from machine learning. Embeddings methods using machine learning techniques, called *predict* models, initially developed somewhat independently. The first occurrence of this approach was in (Bengio et al. 2003), which described a machine-learning language model which produced word embeddings as a by-product of its primary function. Other predict models were concieved, improving upon this initial concept (Morin and Bengio 2005; Mnih and Hinton 2007; Collobert and Weston 2008) but this embedding paradigm remained less well known until the success of Mikolov, K. Chen, et al.'s (2013) `word2vec` models encouraged direct comparisons and cross-paradigm research (Baroni, Dinu, and Kruszewski 2014; O. Levy and Goldberg 2014).

Although the methods seem quite different on the surface, O. Levy and Goldberg (2014) found that it is possible to view the operation of these predict models as *implicitly* factoring a co-occurrence matrix -so a similarly underlying principle is present in both.

The methods we have presented here (document-term matrices and context-word matrices) are the foundation of the theory of count models but particular models can vary greatly. Most notably, Pennington, Socher, and Manning's (2014) GloVe model reports better results than other count-based models and prediction based models of the `word2vec` software package in the tasks of word-analogy and named entity recognition.

The cutting edge of embedding methods, at the time of writing is the "deep contextualised model" of Peters et al. (2018) which incorporates sentence-level information and leverages a language model to capture context-dependent aspects of word meaning. This is one particularly successful approach to tackling the difficulties endemic to word-meaning inference discussed in Words and Distributional Similarity & Semantic relatedness.

FURTHER READING

In this section we briefly identify key resources used in the writing of this report and suggest other material for further reading, in particular regarding the practical application of embedding algorithms.

Summary: Selected topics and references

Key references These resources have been invaluable in writing this report:

1. On words: Lexicology and Corpus Linguistics*;* Word : a cross-linguistic typology (Halliday and Teubert 2004; Dixon 2002),
2. On Count models: *"From Frequency To Meaning: Vector Space Models of Semantics";* The word-space model : using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces (P. D. Turney and Pantel 2010; Sahlgren 2006),,
3. An overview of embedding methods: *"Evaluating word embedding models: methods and experimental results"; "Word Embeddings: a Survey"* (B. Wang et al. 2019; Almeida and Xexéo 2019)

Pre-processing Although we discussed tokenization in this report we did not discuss the more general problem of text pre-processing. Purposes of pre-processing can be lemmatization (replacing words with their "lemma" form i.e., looks→look and running→run), lowercasing, stop-word removal, multiword grouping (identifying multi-part grammatical words as single tokens). Suggested reading:

1. *"The impact of preprocessing on text classification"* (Uysal and Gunal 2014),
2. *"On the Role of Text Preprocessing in Neural Network Architectures: an Evaluation Study on Text Categorization and Sentiment Analysis"* (Camacho-Collados and Pilehvar 2017).

Bias Since embedding algorithms do not understand language, they just identify statistical regularities in corpora, if that training data contains bias then that bias will be represented (or even amplified) in the embedding output. Suggested reading:

1. *"Semantics derived automatically from language corpora contain human-like biases"* (Caliskan, Bryson, and Narayanan 2017),
2. *"Man Is To Computer Programmer As Woman Is To Homemaker? Debiasing Word Embeddings"* (Bolukbasi et al. 2016).

CONCLUSION

Word embeddings are an essential tool in the NLP toolbox -they are “a standard component of most state-of-the-art NLP architectures” (Peters et al. 2018). They enable many systems which have to interact with language in some capacity do so more ‘intelligently’. In this report we have tried to describe the problems which make the creation of *truly* intelligent word embeddings more difficult. Engineering better methods for representing language requires understanding what written language *is* and how it performs its function. Cutting edge methods like that of Devlin et al. (2019), which incorporate an element of language modelling, will likely become the standard for this reason. Despite this, it is remarkable how much semantic information is encoded by even the most simple models, applicable to any type of sequenced data because of their *lack* of assumptions. The core idea behind word embeddings, *the distributional hypothesis*, is a powerful one. Finally, for practitioners of these methods the Bias aspect is essential to consider: embedded representations say nothing ‘objective’ about the concepts they represent, they are only a reflection of their training data. This is not a shortcoming of the methods used, it should just inform their manner of application -a good use of this bias-capturing nature of embeddings is for studying bias itself, see (Garg et al. 2018) for such an investigation.

BIBLIOGRAPHY

Albright, Russ (2004). “Taming Text with the SVD”. In: *SAS Institute Inc.*

Almeida, Felipe and Geraldo Xexéo (2019). “Word Embeddings: a Survey”. In: *CoRR.*

Baroni, Marco, Georgiana Dinu, and Germán Kruszewski (June 2014). “Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Baltimore, Maryland: Association for Computational Linguistics, pp. 238–247. DOI: `10.3115/v1/P14-1023`.

Bengio, Yoshua et al. (Mar. 2003). “A Neural Probabilistic Language Model”. In: *J. Mach. Learn. Res.* 3.null, pp. 1137–1155. ISSN: 1532-4435.

Bloomfield, Leonard (1926). “A Set of Postulates for the Science of Language”. In: *Language* 2.3, pp. 153–164. ISSN: 00978507, 15350665.

Bolukbasi, Tolga et al. (2016). “Man Is To Computer Programmer As Woman Is To Homemaker? Debiasing Word Embeddings”. In: *CoRR.*

Budanitsky, Alexander and Graeme Hirst (May 2001). “Semantic Distance in Wordnet: An Experimental, Application-Oriented Evaluation of Five Measures”. In: *Workshop on WordNet and Other Lexical Resources.*

Bullinaria, John A. and Joseph P. Levy (Aug. 2007). “Extracting semantic representations from word co-occurrence statistics: A computational study”. In: *Behav. Res. Methods* 39.3, pp. 510–526. ISSN: 1554-3528. DOI: `10.3758/BF03193020`.

Caliskan, Aylin, J. Bryson, and A. Narayanan (2017). “Semantics derived automatically from language corpora contain human-like biases”. In: *Science* 356, pp. 183–186.

Camacho-Collados, Jose and Mohammad Taher Pilehvar (2017). “On the Role of Text Preprocessing in Neural Network Architectures: an Evaluation Study on Text Categorization and Sentiment Analysis”. In: *CoRR.*

Chamberlain, Benjamin Paul et al. (Aug. 2017). “Customer Lifetime Value Prediction Using Embeddings”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, nil. DOI: `10.1145/3097983.3098123`.

Chen, Xinchi et al. (Apr. 2017). “Adversarial Multi-Criteria Learning for Chinese Word Segmentation”. In: *arXiv.*

Choi, Youngduck, Yi-I Chiu, and David Sontag (2016). “Learning Low-Dimensional Representations of Medical Concepts”. In: *Proceedings of the AMIA Summit on Clinical Research Informatics (CRI).*

Chuan, Ching-Hua, Kat Agres, and Dorien Herremans (Dec. 2018). “From context to concept: exploring semantic relationships in music with word2vec”. In: *Neural Computing and Applications* 32.4, pp. 1023–1036. ISSN: 1433-3058. DOI: `10.1007/s00521-018-3923-1`.

Collobert, Ronan and Jason Weston (2008). “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”. In: *Proceedings of the 25th International Conference on Machine Learning.* ICML ’08. Helsinki, Finland: Association for Computing Machinery, pp. 160–167. ISBN: 9781605582054. DOI: `10.1145/1390156.1390177`.

Deerwester, Scott et al. (1990). “Indexing by latent semantic analysis”. In: *Journal of the American Society for Information Science* 41.6, pp. 391–407. DOI: `https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9`.

Devlin, Jacob et al. (June 2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`.

Dixon, Robert (2002). *Word : a cross-linguistic typology.* Cambridge: Cambridge University Press. ISBN: 052104605X.

Dubin, David (Mar. 2004). “The Most Influential Paper Gerard Salton Never Wrote”. In: *Library Trends* 52.

Dumais, S. T. et al. (1988). “Using Latent Semantic Analysis to Improve Access to Textual Information”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’88. Washington, D.C., USA: Association for Computing Machinery, pp. 281–285. ISBN: 0201142376. DOI: 10.1145/57167.57214.

Firth, J. (1957). “A Synopsis of Linguistic Theory, 1930-1955”. In: Studies in Linguistic Analysis, Philological. Longman.

Garg, Nikhil et al. (2018). “Word embeddings quantify 100 years of gender and ethnic stereotypes”. In: *Proceedings of the National Academy of Sciences* 115.16, E3635–E3644. ISSN: 0027-8424. DOI: 10.1073/pnas.1720347115.

Gentner, Dedre (1983). “Structure-mapping: A theoretical framework for analogy”. In: *Cognitive Science* 7.2, pp. 155–170. ISSN: 0364-0213. DOI: https://doi.org/10.1016/S0364-0213(83)80009-3.

Grbovic, Mihajlo and Haibin Cheng (2018). “Real-Time Personalization Using Embeddings for Search Ranking at Airbnb”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’18. London, United Kingdom: Association for Computing Machinery, pp. 311–320. ISBN: 9781450355520. DOI: 10.1145/3219819.3219885.

Grohe, Martin (2020). “Word2vec, Node2vec, Graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data”. In: *CoRR*.

Halliday, M.A.K. and W. Teubert (2004). *Lexicology and Corpus Linguistics*. Open Linguistics. Bloomsbury Academic. ISBN: 9780826448620.

Harris, Zellig S. (Aug. 1954). “Distributional Structure”. In: *WORD* 10.2-3, pp. 146–162. ISSN: 2373-5112. DOI: 10.1080/00437956.1954.11659520.

Henley, Nancy M. (1969). “A psychological study of the semantics of animal terms”. In: *Journal of Verbal Learning and Verbal Behavior* 8.2, pp. 176–184. ISSN: 0022-5371. DOI: https://doi.org/10.1016/S0022-5371(69)80058-7.

Joos, Martin (Nov. 1950). “Description of Language Design”. In: *The Journal of the Acoustical Society of America* 22.6, pp. 701–707. ISSN: 0001-4966. DOI: 10.1121/1.1906674.

Jurafsky, Dan and James H. Martin (2021). “Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition”. Draft of third edition of the book.

Kaplan, David (1990). “Words”. In: *Proceedings of the Aristotelian Society, Supplementary Volumes* 64, pp. 93–119. ISSN: 03097013, 14678349.

Landauer, Thomas K and Susan T. Dutnais (1997). “A solution to Plato’ s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge”. In: *PSYCHOLOGICAL REVIEW* 104.2, pp. 211–240.

Levy, Omer and Yoav Goldberg (2014). “Neural Word Embedding as Implicit Matrix Factorization”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’14. Montreal, Canada: MIT Press, pp. 2177–2185.

Linzen, Tal (Aug. 2016). “Issues in evaluating semantic spaces using word analogies”. In: *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. Berlin, Germany: Association for Computational Linguistics, pp. 13–18. DOI: 10.18653/v1/W16-2503.

Liu, Shusen et al. (Jan. 2018). “Visual Exploration of Semantic Relationships in Neural Word Embeddings”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1, pp. 553–562. ISSN: 2160-9306. DOI: 10.1109/tvcg.2017.2745141.

Lyons, J. (1968). *Introduction to Theoretical Linguistics*. Introduction to Theoretical Linguistics. Cambridge University Press. ISBN: 9780521095105.

Matthews, P. H (1991). *Morphology*. Cambridge England New York: Cambridge University Press. ISBN: 9780521410434.

Meyer, C. D (2000). *Matrix analysis and applied linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics. ISBN: 0898714540.

Mikolov, Tomas, Kai Chen, et al. (2013). “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR*.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (June 2013). “Linguistic Regularities in Continuous Space Word Representations”. In: *Proceedings of the 2013 Con-*

ference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Atlanta, Georgia: Association for Computational Linguistics, pp. 746–751.

Miller, George A., Richard Beckwith, et al. (1990). “Introduction to WordNet: An On-line Lexical Database*”. In: *International Journal of Lexicography* 3.4, pp. 235–244. ISSN: 1477-4577. DOI: 10.1093/ijl/3.4.235.

Miller, George A. and Walter G. Charles (1991). “Contextual correlates of semantic similarity”. In: *Language and Cognitive Processes* 6.1, pp. 1–28. DOI: 10.1080/01690969108406936.

Mirheidari, Bahman et al. (Sept. 2018). “Detecting Signs of Dementia Using Word Vector Representations”. In: *Interspeech 2018.* DOI: 10.21437/interspeech.2018-1764.

Mnih, Andriy and Geoffrey Hinton (2007). “Three New Graphical Models for Statistical Language Modelling”. In: *Proceedings of the 24th International Conference on Machine Learning.* ICML ’07. Corvalis, Oregon, USA: Association for Computing Machinery, pp. 641–648. ISBN: 9781595937933. DOI: 10.1145/1273496.1273577.

Morin, Frederic and Yoshua Bengio (2005). “Hierarchical probabilistic neural network language model”. In: *AISTATS’ 05*, pp. 246–252.

Padó, Sebastian and Mirella Lapata (July 2003). “Constructing Semantic Space Models from Parsed Corpora”. In: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics.* Sapporo, Japan: Association for Computational Linguistics, pp. 128–135. DOI: 10.3115/1075096.1075113.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Peters, Matthew E. et al. (2018). “Deep Contextualized Word Representations”. In: *CoRR.*

Resnik, Philip (Nov. 1995). “Using Information Content to Evaluate Semantic Similarity in a Taxonomy”. In: *arXiv.*

Robertson, S. E. and K. Sparck Jones (May 1976). “Relevance weighting of search terms”. In: *Journal of the American Society for Information Science* 27.3, pp. 129–146. ISSN: 1097-4571. DOI: 10.1002/asi.4630270302.

Robertson, Stephen (Oct. 2004). “Understanding inverse document frequency: on theoretical arguments for IDF”. In: *Journal of Documentation.* DOI: 10.1108/00220410410560582.

Rogers, Anna, Aleksandr Drozd, and Bofang Li (Aug. 2017). “The (too Many) Problems of Analogical Reasoning with Word Vectors”. In: *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017).* Vancouver, Canada: Association for Computational Linguistics, pp. 135–148. DOI: 10.18653/v1/S17-1017.

Rubenstein, Herbert and John B. Goodenough (Oct. 1965). “Contextual Correlates of Synonymy”. In: *Commun. ACM* 8.10, pp. 627–633. ISSN: 0001-0782. DOI: 10.1145/365628.365657.

Sahlgren, Magnus (2006). *The word-space model : using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* Stockholm: Dep. of Linguistics, Stockholm Univ. ISBN: 9171552812.

Salton, G., A. Wong, and C. S. Yang (Nov. 1975). “A Vector Space Model for Automatic Indexing”. In: *Commun. ACM* 18.11, pp. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220.

Schutze, Hinrich and Jan O. Pedersen (1995). “Information Retrieval Based on Word Senses”. In: *In Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pp. 161–175.

Schütze, Hinrich (1993). “Word Space”. In: *Advances in Neural Information Processing Systems.* Ed. by S. Hanson, J. Cowan, and C. Giles. Vol. 5. Morgan-Kaufmann.

Tissier, Julien, Christophe Gravier, and Amaury Habrard (Sept. 2017). “Dict2vec : Learning Word Embeddings using Lexical Dictionaries”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* Copenhagen, Denmark: Association for Computational Linguistics, pp. 254–263. DOI: 10.18653/v1/D17-1024.

Turney, P. D. and P. Pantel (2010). “From Frequency To Meaning: Vector Space Models of Semantics”. In: *Journal of Artificial Intelli-*

gence Research 37.nil, pp. 141–188. DOI: 10.1613/jair.2934.

Turney, Peter D. (2006). "Similarity of Semantic Relations". In: *Computational Linguistics* 32.3, pp. 379–416. DOI: 10.1162/coli.2006.32.3.379.

Uysal, Alper Kursat and Serkan Gunal (Jan. 2014). "The impact of preprocessing on text classification". In: *Information Processing & Management* 50.1, pp. 104–112. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2013.08.006.

Wang, Bin et al. (2019). "Evaluating word embedding models: methods and experimental results". In: *APSIPA Transactions on Signal and Information Processing* 8, e19. DOI: 10.1017/ATSIP.2019.12.

Wang, Jizhe et al. (2018). "Billion-Scale Commodity Embedding for E-Commerce Recommendation in Alibaba". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &; Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery, pp. 839–848. ISBN: 9781450355520. DOI: 10.1145/3219819.3219869.

Wittgenstein, Ludwig (1922). *Tractatus logico-philosophicus*. London : Routledge & Kegan Paul.

— (1953). *Philosophical investigations*. Oxford: B. Blackwell. ISBN: 0631146709.

Yaghoobzadeh, Yadollah and Hinrich Schütze (Aug. 2016). "Intrinsic Subspace Evaluation of Word Embedding Representations". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 236–246. DOI: 10.18653/v1/P16-1023.

Zhao, Zhe et al. (Sept. 2017). "Ngram2vec: Learning Improved Word Representations from Ngram Co-occurrence Statistics". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 244–253. DOI: 10.18653/v1/D17-1023.