University of York

Final year project

Word Embeddings

Joel Strouts

supervised by

Dr. Ben Powell

Abstract

In this report we attempt to provide a complete introduction to the core theories of word embeddings (numerical vector representations of words as they appear in text) sufficient to enable further reading of more advanced topics. This introduction covers briefly the linguistic context, theoretical development, most impactful implementations, and practical considerations of application of word embedding methods. In the first section we discuss the nature of the words, in the second we describe the cross disciplinary events which lead to development the key mathematical techniques for producing embeddings and explain their operation, and finally we discuss practical considerations of these algorithm's applications. As the primary relevance of these methods within research is their practical application we take a broader contextual approach opposed to a purely mathematical one and focus on the developments and methods most influential in the sphere of application. We assume familiarity with core mathematics at an undergraduate level as prerequisite for our descriptions of the selected implementation methods but otherwise assume no prior knowledge of the subject.

June 28, 2021

TABLE OF CONTENTS

INTRODUCTION

What are word embeddings?

Word embeddings are numerical representations of words which are derived from the patterns of word usage observed in a body of example text. These representations take the form of position vectors within a *Word Space*. Schütze (1993) summarises the key properties of the resulting *Word Space Model* as follows:

> “ [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 (Illustrative Word Space). The word space in Figure 1 illustrates both components of this 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 if considered in isolation from one another (there is no trait like ‘size’ which the x or y value for a word is a description of).

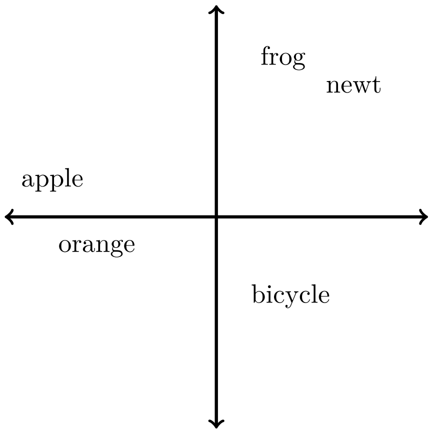

Figure 1: A two dimensional word space encoding semantic information about the five words; *apple, orange, frog, newt & bicycle*

word	features	
	x	y
apple	-1.55	0.35
orange	-1.13	-0.3
bicycle	0.7	-0.75
frog	0.63	1.5
newt	1.3	1.25

Table 1: Feature vectors representing the words in Figure 1

Unlike those in example 1 however, word embedding vectors are typically *high-dimensional*; it is common for the dimension of a word space to be in the order of tens or hundreds. In an exemplary pre-trained word-embedding model[1], the word orange (represented by the vector $\langle -1.55, 0.35\rangle$ in the above example) is represented by the 50-dimensional vector:

orange ↦

```
⟨ -0.42783 ,  0.43089 , -0.50351 ,  0.5776   ,  0.097786,  0.2608   , -0.68767 , -0.31936 , -0.25337 , -0.37255 ,
  -0.045907, -0.53688 ,  0.97511 , -0.44595 , -0.50414 , -0.086751, -1.0645   ,  0.36625 , -0.52428 , -1.3413   ,
  -0.2391   , -0.58808 ,  0.56378 , -0.062501, -1.7429   , -0.88077 , -0.27933 ,  1.4705   ,  0.50436 , -0.69174 ,
   2.0018   ,  0.26663 , -0.85679 , -0.18893 , -0.021125, -0.055118, -0.50337 , -0.67157 ,  0.55502 , -0.8009   ,
   0.10695 ,  0.1459   , -0.55588 , -0.64971 ,  0.22046 ,  0.67415 , -0.45119 , -1.1462   ,  0.16348 , -0.62946 ⟩
```

These high-dimensional vectors produced by typical word embedding algorithms cannot be plotten on an x, y graph like in Figure 1.

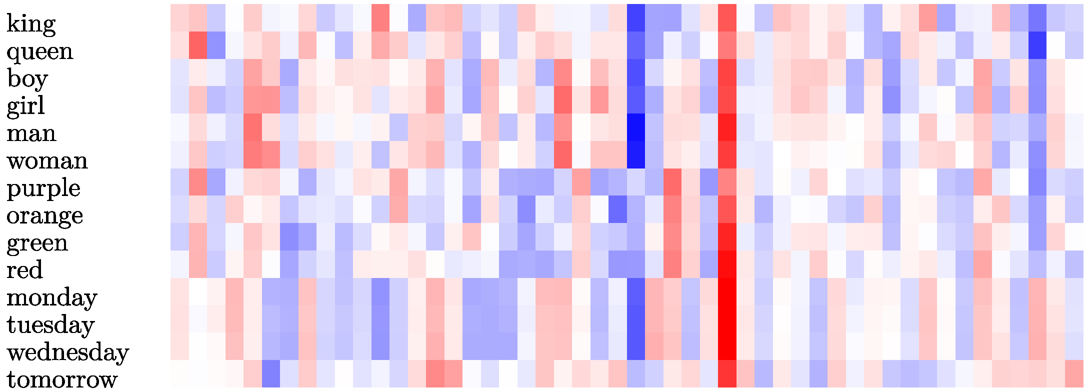

Figure 2: A visualisation of the embeddings of 14 words from a pre-trained 50-dimensional *GloVe* model (trained on wikipedia articles and newswire text)

The focus of this report is the description of various *word embedding algorithms*: processes which take a collection of language usage examples (like the collection of all english-language wikipedia articles) and *automatically* i.e., without any human supervision, produce word embeddings like those in Figure . The key idea leveraged (with differing exact formulations) by these algorithms to produce word embeddings is that of *the distributional hypthesis*, summarised by Firth (1957) as

[1] Provided by Pennington, Socher, and Manning (2014). Model available for download at (GloVe project website)

"You will know a word by the company it keeps", that words which appear in more similar contexts have more similar meanings.

What are word embeddings used for?

Report Content & Structure

In chapter 1 we discuss the key linguistic concepts which inform embedding-algorithm design: how we define 'word', what is meant by 'semantic relatedness' & core concepts from distributional semantics. In chapter 2 we discuss the particulars of generating embeddings from a training corpus -first looking at 'count models' then 'predict models'. Finally, in chapter 3 we address practial considerations for the application of these methods: text preprocessing, model tuning & consideration of bias.

CHAPTER 1
WORDS

In some situations, a theoretical understanding of linguistic concepts is not necessarily required, however, we consider a grounding in these contepts illuminating context more reasoned application.

1.1 What are words?

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

Halliday and Teubert (2004)

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*[1]. The difficulty of the question is summarised in the book’s conclusion as follows:

> “ 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

Consider *so called* filler-words like “uh” or “um”, or compound forms like “rock ’n’ roll” and “New York”. These are examples of language elements that are sometimes treated as words and sometimes not; it is not clear where the boundary should be drawn. Even agreed upon boundaries change with time; the word “tomorrow”, listed in Johnson’s 1755 dictionary of the English language as two distinct components “to”, and “morrow” yet these parts have merged and are considered a single indivisible word. Halliday and Teubert (2004) asks “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?”.

[1] Dixon 2002.

The understanding of ‘word’ formed within the English language cannot simply 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 written without spaces, like Chinese, Japanese, and Thai present one particular example of the challenges in applying the ‘word’ concept beyond this scope: a single sentence may be divided up into different word-like units depending on the approach taken,Chen et al. (2017) gives the following example:

Example 1.1.1. 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 segmentations methods; ’Chinese Treebank’ segmentation, or ’Peking University’ segmentation, respectively:

```
姚明      进入      总决赛
YaoMing  reaches  finals
```

```
姚   明   进入     总        决赛
Yao  Ming  reaches  overall  finals
```

Different languages present different difficulties -In the case of Arabic, expressions formed by adding consecutive suffixes to a single root word (like establish → establishment → establishmentarianism) are a more essential part of the language, to the extent that complete sentences can be expressed as single word-units. As a result, ‘words’ in the sense of ‘things occurring between spaces’ describe a unit of meaning larger and more complex than the term describes in English, meaning representations of Arabic words defined this way describe something quite different to the English analogue.

The term **lexical item** describes ‘units’ of language more broadly; in Chinese they may be individual characters or groups with a conventional meaning. in Arabic they may be the set of roots and suffixes In English they may be the words, inflections, and multi-word phrases used as single parts.

1.2 Features of Written Language

While the linguistic & philosophical discussion of ‘word’ informs the theory of computational linguistics, in our context the question of “what is a word?” is realised as a question of text-preprocessing. It is assumed that the input to the embedding process is a written text and it is the task of text preprocessing to extract ‘words’ from that text. The implementation used is more motivated by the application requirements & task performance than linguistic theory.

Embedding algorithms are not discriminatory about what you classify as ‘a word’ in their input (in fact they can be applied to forms of structured data other than just language, as we will discuss in the final section of this report) and the decision about how to make these divisions will depend on the task at hand. If the intended application of the embeddings is within the field of chemistry, then a compound name like `(NH4)2SO4` should absolutely be treated as a word, while in other contexts it is possible that such an irregular form may be discounted without negative consequences. The input material impacts these decisions too; text in newspapers has a narrower scope of common word-like forms than a medium like tweets, and text preprocessing decisions should reflect this.

The process of extracting word-units from a text is called **tokenization** and the resulting units are called **word-tokens**. Word tokens with the same form (like *red* and *red*) are said to be instances of same word-class (or word-type). For example, the sentence;

> *The limits of my language means the limits of my world* - Wittgenstein, 1922

Contains 11 word-tokens, and 7 word-classes (assuming the sentence is tokenized by identifying word boundaries at both ends and each blank space).

A detailed discussion of tokenization & text preprocessing in general is beyond the scope of this report but some practical considerations can be found in **??**

1.3 Objects of Representation

Lexicology is the study of words, and **semantics** is the study of meaning. The ‘embedding’ in ‘word embeddings’ derives from the mathematical sense ‘to embed in a space’ (for example, to embed a graph in a coordinate space). Word embedding algorithms are methods for embedding *lexicological objects* in *semantic space*, i.e., words a space where position relates to meaning somehow). This is achieved by statistical analysis of large quantities of written language (by extraction of word-tokens).

The conventions of written language, called **orthography**, make word-tokens an imperfect representation of ‘words’ in the lexicological sense. In particular, one problem is that distinct words do not always have distinct realisations in written language. Orthographic forms with this property are called **polysemus** (poly as in *many* and semus as in *meaning*, from the same root as semantics). Alternatively, the distinct words sharing this same form are referred to as **homonyms** (having the same name). The words *bank* as in money holding organisation and *bank* as in riverside are often used to illustrate this property. Indeed, some words, like *frequent* as in often, and *frequent* as in attend are examples of words distinct in spoken language despite being identical in written language; such words are said to be **homographs** (same orthography, different names).

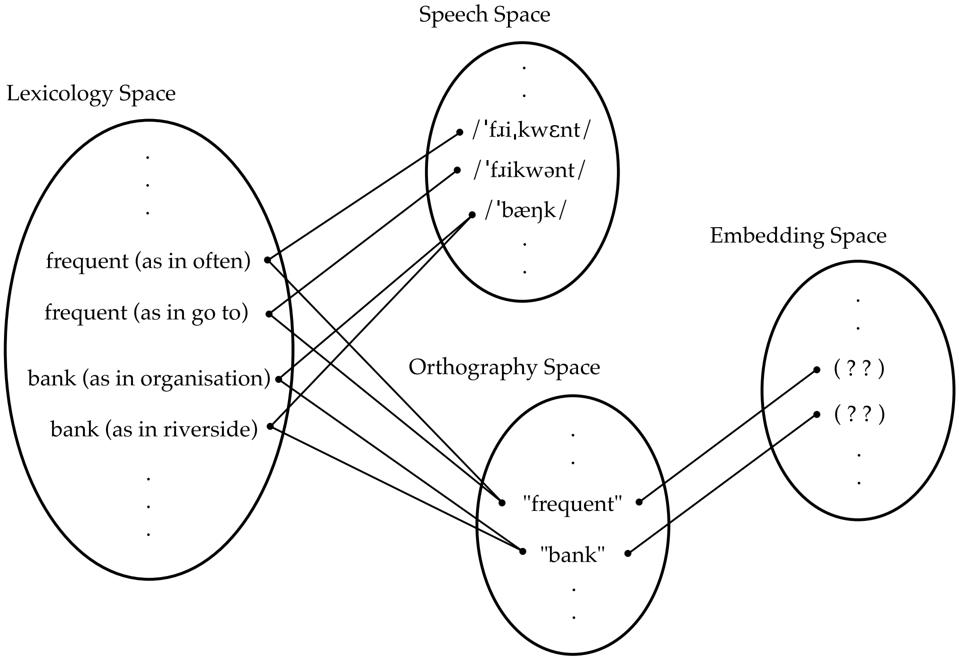

Figure 1.1: The *problem of polysemy* illustrated

Embedding algorithms which generate their representations without taking this *problem of polysemy* into account produce representations which conflate the multiple meanings of a word-form into a single point in the embedding space.

CHAPTER 2

GENERATING EMBEDDINGS

2.1 Preliminaries

We begin by defining terms already encountered in mathematical terms.

Definition 2.1.1 – *Orthography*

An orthography is a finite set of symbols.

Example 2.1.1. The sets:

$$\mathscr{O}_L = \{c \mid c \text{ is a symbol that can be copied and pasted the pdf of this report}\} \tag{2.1}$$

$$\mathscr{O}_P = \{c \mid c \text{ is the inscription of a regular polygon with eight sides or less}\} \tag{2.2}$$

$$\mathscr{O}_K = \{c \mid c \text{ is a typable character marked on Figure 2.1}\} \tag{2.3}$$

are all examples of orthographies.

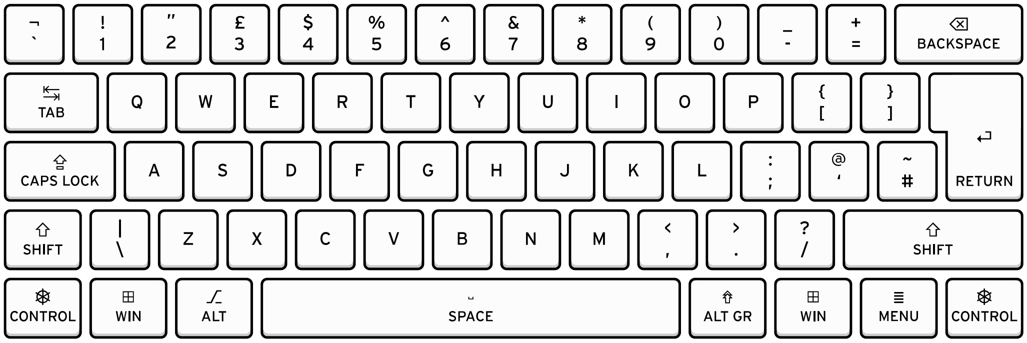

Figure 2.1: A standard uk keyboard layout

Note, the term ‘orthography’ in linguistics refers to the conventions of written language in general and is distinct from the use of ‘an orthography’ we use here.

Definition 2.1.2 – *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 2.1.2 (This Report). The sequence of symbols obtained by copy pasting this entire report into a text editor (ordered by starting with the cursor at the top of the file and moving the cursor with the arrow keys one position to the right at a time -including newlines) is a document using the orthography 2.1.

Example 2.1.3 (Shapes Document). The sequence of shapes: △△□△△⬠ is a document using the orthograpy 2.2.

Example 2.1.4 (Plato Documents). Any of the works of Plato available at the Project Gutenberg website, copied and pasted from their plain text formats and using the same procedure described above, are documents using the orthography 2.3.

Definition 2.1.3 – *Word-Token*

A word token w has the same attributes as a document; it is a finite sequence $c_0, c_1, \ldots, c_n$ $(c \in \mathscr{O})$, where $\mathscr{O}$ is an orthography. We only refer to sequences of symbols as word tokens when they are the output of a tokenizer, however.

Definition 2.1.4 – *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 refered 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.[1].

Example 2.1.5 (This Report Tokenized). The document from example 2.1.2 if tokenized by treating everything occuring between two spaces (or between a space and the start or end of a line) yields a sequence of word tokens beginning *"THE" "UNIVERSITY" "OF"*.

Example 2.1.6 (Shapes Document Tokenized). The document from example 2.1.3 if tokenized using the function:

$$T(d):\ \texttt{Group repeated symbols in the sequence } d \texttt{ into word-tokens}$$
$$\texttt{(in the order of occurrence).}$$

produces the output: *"△ △", "□", "△ △", "⬠"*

Example 2.1.7 (Simple Text Tokenizer). Given a document d using the orthography $\mathscr{O}_K$ from example 2.1.1 apply the following algorithm:

Definition 2.1.5 – ***Corpus***

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

Example 2.1.8 (Plato Corpus). All the of Plato works referred to in example 2.1.4, if converted to documents using the same procedure described there, considered together form a corpus of documents. We will refer to this corpus as C_p.

Definition 2.1.6 – ***Vocabulary***

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

Example 2.1.9 (Shapes Document Vocabulary). Using the same tokenization as before, the document from example 2.1.3 has the vocabulary:

$$V(\text{“△ △”}, \text{“□”}, \text{“△ △”}, \text{“⬠”})$$

$$= \{\text{“△ △”}, \text{“□”}, \text{“⬠”}\}$$

Definition 2.1.7 – # *Operator*

The # symbol is used to denote the number of elements of the object it precedes. Before sequences it counts the number of elements in the sequence: $\#W_d = n$ (for $W_d = w_1, \ldots, w_n$) [2]. Before sets it is a shorthand for the size of that set: $\#V(d) = |V(d)| =$ the number of words in the vocabulary $V(d)$. Before a corpus it counts the number of documents in that corpus.

Example 2.1.10 (Shapes Document Sizes). The word count of the tokenized shapes document

$$W_d = \text{“△ △”}, \text{“□”}, \text{“△ △”}, \text{“⬠”}$$

is $\#W_d = 4$ and the vocabulary size is $\#V(W_d) = 3$.

Example 2.1.11 (Plato Document Sizes). Word token counts and vocabulary sizes for a selection Plato documents from 2.1.4, tokenized using the simple method: [a] have

[a] the code used to create these examples and others examples on the same corpus is available in the appendix

Example 2.1.12 (Plato Corpus Size). The complete Plato corpus (2.1.8) 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 $\#V(C_P) = 19,432$.

2.2 Count Models

The embedding algorithms discussed in this report represent the confluence of two key ideas: first that the meaning of a word corresponds with the contexts in which it occurs, and second that the meaning of a word can be represented by a vector of numerical values.

This first idea is called the **distributional hypothesis**, was "widespread in linguistic theory in the 1950s" Jurafsky (2021), with key contributions to its conception contributed by Joos (1950) *Description of Language Design*, Harris (1954) *Distributional Structure*, and Firth (1957) *A synopsis of linguistic theory 1930-1955*.

> To illustrate the distributional hypothesis, consider for example the possible meanings of "bank" discussed above in relation to polysemy: *bank* as in the organisation, and *bank* as in riverside. The first sense of the word is identified with neighboring word contexts like: account, manager, desk, etc. whereas the second sense is distinguished by context words like: river, grassy, marsh.

The way these distributional relationships are inferred by analysis of a **corpus** (a collection of reference texts) and synthesised in vector form differ according to the embedding algorithm used. The 'count models' developed in the 1960s and refined thereafter were the first to apply the distributional hypothesis to create word vectors, and remained the state of the art until a series of papers published by Milokov et al (2013) presented efficient methods for generating word vectors using neural networks, along with a standard software implementation `word2vec` which then became ubiquitous.

We will discuss count models first as they lay the contextual foundation for the innovations of the later predict models.

2.2.1 History

The adoption of distributional perspectives within linguistic spheres in the 1950s coincided with an emerging interest in mechanical and computational methods for document indexing and information retrieval (IR); this purpose was one of the first applications considered for modern computers. The question of quantifying document 'aboutness' and relatedness suggested, as viewed through the right mathematical lens, a notion of a 'document space' populated by 'document vectors' where relatedness could be quantified by a distance function on the space. From this the notion word vectors naturally followed. We summarise some key events in this lineage below.

In 1948 two chemists and leading advocates of punched cards, James W. Perry and G. Malcolm Dyson, petitioned IBM head, Thomas Watson, Sr to invest in research & development on "mechanical solutions for scientists' specific information problems", who selected Hans Peter Luhn for the task [cite: Rieder, B 2020].

This appointment resulted in "a remarkable series of papers" [cite: salton 1987] published by Luhn between 1957 & 1959, in particular: *A Statistical Approach to Mechanized Encoding and Searching of Literary Information*, Luhn (1957). The concepts discussed in this paper were prescient of future developments in the mechanical processing of documents but details of computation are omitted.

Maron & Kuhns (1960) provided details of a potential approach to the computation of such a system in the influential *On relevance, Probabilistic Indexing and Information Retrieval.* published around the same time. Crucially, the paper suggests the use of conditional probabilities to compute 'closeness of index terms'. This was an application of distributional thinking to quantify word similarity, but it did not specify a vector-space conception of the problem.

This final piece was supplied by Paul Switzer in *Vector Images in Document Retrieval* (1964), which references Maron & Kuhn's paper and elaborates by suggesting the construction of a 'term-term association matrix' based on co-occurence frequencies producing an 'index term vector space' in which documents are then located. It is the first paper to suggest that document similarity could be determined by projecting index terms and documents into a vector space.

Details of the resulting model and the most widely adopted refinements are the focus of this section.

2.2.2 Term-Document & Term-Term Matricies

Definition 2.2.1 – *Document Word Occurences*

The number of times a word-token w occurs in a tokenized document W_d is given by $\mathrm{count}(w, W_d)$.

> **Definition 2.2.2 –** ***Term Document Matrix***
>
> A term-document matrix X, (for corpus C and tokenizer T) has columns corresponding to each corpus document, and rows corresponding to each word in the corpus vocabulary (hence having dimensions $\#V(C) \times \#C$). The matrix entry

2.2.3 Term Weighting

2.2.4 Smoothing

2.2.5 Singular Value Decomposition

2.3 Predict Models

2.3.1 Early Developments

2.3.2 word2vec: Skip-Gram with Negative Sampling

2.3.3 word2vec: Continuous Bag of Words

2.4 Measuring Similarity

2.5 Comparison

2.6 Other Approaches

CHAPTER 3

PRACTICAL CONSIDERATIONS

3.1 Preprocessing

3.2 Hyper-parameters

3.3 Bias

CONCLUSION

FURTHER READING

BIBLIOGRAPHY

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

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

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

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

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.

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.