Master Seminar Word Embedding Spaces: Static Word Embeddings

Lukas Mielczarek
lukas.mielczarek@hhu.de

Linbo Zhang
linbo.zhang@hhu.de

Abstract

In this extended abstract we explain the motivation for representing natural language words with multi-dimensional vectors, explore a linguistic intuition of word meaning and present several methods for generating meaningful word representations. These *static word embeddings* map a word to a fixed vector that can be used in downstream applications. The methods we explore include traditional frequency-based approaches (term-document matrices, co-occurrence matrices, TF-IDF, PMI, LSA) as well as newer neural network-based methods (word2vec, FastText, GloVe). Finally, we discuss how to evaluate the quality of word embeddings generated with these methods.

1 Introduction

The starting point of our journey is the following question: How to represent a natural language word for use in machine learning algorithms? Of course, as one often does for discrete categorical features, one can assign each word a unique number and use a *one-hot-encoding*. An encoding for the word with ID 4 would then be a vector of length $|V|$ for a vocabulary V with a one at position 4 and all other entries filled with zeros.

This, however, comes with two problems: (1) the resulting vectors are very large for large vocabulary sizes and (2) they do not provide us with any useful information about the words. A word is as close to another word as it is to any other word in the vocabulary [21].

One-hot embeddings	
Strengths	**Weaknesses**
Easy to generate	Sparse
	High-dimensional
	No relationships between vectors

1.1 Word Meaning

Instead, when we humans think about words, we associate them with their meaning. Thus, we would like to quantify this meaning somehow to generate *word embeddings*. The *distributional hypothesis* tells us that words that occur in similar contexts tend to have similar meanings [9]. Let us take a look at the following example [12]:

1. *Ongchoi* is delicious sauteed with garlic.
2. *Ongchoi* is superb over rice.
3. ...*ongchoi* leaves with salty sauces...

Even if you do not know what *ongchoi* is, since words like *spinach, chard* and *collard greens* also occur with *rice, garlic, delicious* and *salty* in speech and text, you can infer that *ongchoi* is a leafy green similar to other leafy greens.[1]

We can leverage this intuition to represent a word by observing word distributions in a text corpus and constructing a *static word embedding* for each word in our vocabulary. We want these word embedding to fulfil the following criteria:

1. Synonyms should have very similar embeddings.
2. Similar words should have similar embeddings.
3. Words that are not similar should have very different embeddings.

Note, however, that there are other types of relatedness besides similarity. For instance, the words *coffee* and *cup* are not similar but still related [12]. The words *man* and *woman* are similar to a large extent (both are humans) but they differ in one feature: natural gender. They are so-called *antonyms*

[1] Indeed, ongchoi is the Cantonese term for *Ipomoea aquatica*, commonly known as water spinach [12].

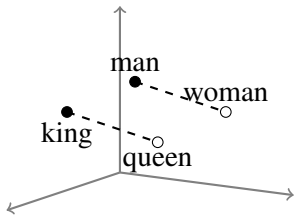

Figure 1: The difference between the embeddings for *man* and *woman* is expected to be the same as the difference between *king* and *queen*.

[12]. In our representation space a type of relation should be expressed similarly (i.e. by a similar movement in space) across word pairs, as is visualised in Figure 1.

2 Sparse Frequency-based Methods

In the following, we will present several frequency-based methods that can be used to create word-embeddings. All of them have in common that they involve observing word occurrences in texts.

Sparse Frequency-based Methods	
Strengths	**Weaknesses**
Easy to implement	Sparse
Easy to understand	High-dimensional

2.1 Term-Document Matrix

For a selection of words V and a selection of documents D one can record how often each word appears within each document. This gives rise to a $|V| \times |D|$ matrix where the rows can be used as static vectors to characterise a word.

2.2 Co-occurrence Counts

Alternatively, one can count how often two words appear together within some pre-defined distance in a document, producing a $|V| \times |V|$ co-occurrence matrix. The entries of the matrix reflect word association strengths. Again, a row can be used to represent the corresponding word.

2.3 Term Frequency-Inverse Document Frequency

The *Term Frequency-Inverse Document Frequency* (TF-IDF) method was developed in the context of information retrieval [19]. It is a statistical measure used to evaluate the importance of a word in the context of a document within a collection of documents.

Let $TF(t, d)$ be the *term frequency* measuring how often a term t appears in a document d:

$$TF(t, d) = \frac{\text{Number of times } t \text{ appears in } d}{\text{Total number of terms in } d} \quad (1)$$

Let $IDF(t, D)$ be the *inverse document frequency* which measures the commonness of a term t across all documents in a corpus D:

$$IDF(t, D) = \log(\frac{\text{Number of documents in } D}{\text{Number of documents with } t}) \quad (2)$$

Finally, TF-IDF is defined as follows:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \cdot IDF(t, D) \quad (3)$$

Now, for a collection of terms and a collection of documents, one can construct a matrix of TF-IDF values. For each word, one can retrieve a sparse vector (row) that characterises it.

2.4 Pointwise (Positive) Mutual Information

Pointwise Mutual information (PMI) measures word associations beyond simple co-occurrence. PMI for a target word w and a context word c is defined as follows [5]:

$$PMI(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)} \quad (4)$$

Generally, one adjusts PMI to ignore negative values, focusing only on positive associations [5]:

$$PPMI(w, c) = \max(PMI(w, c), 0) \quad (5)$$

One can construct a PMI matrix that contains the PMI values for all pairs of word and context word. Again, a row of this matrix can be used to represent a word.

3 Introducing Dense Embeddings

The methods described so far produce sparse and high-dimensional embeddings, i.e. vectors with a lot of zeros as entries. This is not optimal for many applications and relates to the phenomenon known as the *curse of dimensionality* [1]. Instead, one can embed words in a lower dimensional space of 50-1000 dimensions instead of $|V|$ for a vocabulary V [11]. *Dense* embeddings work better in many NLP tasks than sparse vectors for several reasons: (1) one needs to learn fewer weights and (2) they prevent overfitting [11].

3.1 Latent Semantic Analysis

Latent semantic analysis (LSA) analyses the relationship between a set of documents and the terms they contain [6]. *Singular value decomposition* (SVD) [8] is used to factorise a term-document matrix into several components that allow to estimate the latent semantic structure in the data and to remove the noise:

$$A = U\Sigma V^T \tag{6}$$

with

- A: term-document matrix
- U: *term-concept matrix*
- Σ: diagonal matrix containing singular values which indicate the importance of each concept
- V: *document-concept matrix* transpose

The matrix U is generally used for word embeddings. This step produces dense representations out of sparse term-document matrices and helps to focus on the most significant underlying patterns in the data [6].

3.2 Cosine Similarity

Dense word embeddings make it possible to define a meaningful similarity measure between vectors $\mathbf{a}$ and $\mathbf{b}$ in the range $[0, 1]$ where θ is the angle between the vectors:

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| \cdot |\mathbf{b}|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{7}$$

4 Word2vec

The next three sections will deal with neural network-based methods for generating dense word embeddings. *Word2vec* is the first such method, developed in 2013 by Mikolov et al. [17]. It uses a shallow neural network to learn from a large text corpus.

Instead of counting word co-occurrences, the idea is to learn word vectors as parameters of a prediction task. The word2vec library offers two different context representations: *skip-gram* and *continuous bag of words* (CBOW) as well as two different optimisation objectives: *negative sampling* and *hierarchical softmax*. In the following, we will explain both skip-gram and CBOW combined with the negative sampling objective.

Word2vec	
Strengths	**Weaknesses**
Simple task: binary classification	Fixed vocabulary size
Simple architecture	Ignores internal structure of words
Fast computation	

4.1 Skip-gram with Negative Sampling (SGNS)

The idea of skip-gram is to train a classifier on the binary prediction class 'Is word c likely to show up near the word w?' in a self-supervised way, i.e. using running text as the gold/correct answer. A logistic regression classifier is trained to distinguish positive and negative cases. Random words are sampled from the lexicon and treated as negative cases (negative sampling) to make the optimisation computationally tractable. Finally, the classifier weights are used as word embeddings [11].

Context words are chosen from a fixed context window of size L around the focus word. Given a tuple (w, c) of focus word w and context word c the classifier will return the probability $P(+|w, c)$ that c occurs in the context window. The negative case is defined as $P(-|w, c) = 1 - P(+|w, c)$.

Skip-gram relies on the assumption that similarity predicts the likelihood of co-occurrence. Here, similarity is defined as the unnormalised cosine similarity mapped to the interval $(0, 1)$ via the sigmoid function:

$$P(+|w, c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{cw})} \tag{8}$$

Every context window contains multiple words. The computation relies on two simplifying assumptions: (1) all context words are independent of each other and (2) their position in the context window does not matter:

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(\mathbf{c}_i \cdot \mathbf{w}) \tag{9}$$

4.1.1 Loss Function

The loss function for a pair of focus and context words aims at maximising their co-occurrence probability and the probability of non-co-occurrence with k negative sample words using *stochastic gradient descent* (SGD) [8]:

$$L_{CE} = -\log\left[P(+|w, c_{pos})\prod_{i=1}^{k} P(-|w, c_{neg_i})\right]$$
$$= -\left[\log\sigma(\mathbf{c}_{pos}\cdot\mathbf{w}) + \sum_{i=1}^{k}\log\sigma(-\mathbf{c}_{neg_i}\cdot\mathbf{w})\right] \tag{10}$$

The embedding weights are randomly initialised and trained as part of the model. Random initialisation most often is done by either sampling the weights uniformly with

$$\mathbf{W} \sim \mathcal{U}(-a, a) \tag{11}$$

for some $a > 0$ or from a normal distribution with standard deviation σ.

$$\mathbf{W} \sim \mathcal{N}(0, \sigma) \tag{12}$$

Kocmi and Bojar [14] have found that initialisation works well irrespective of the distribution as long as the standard deviation is smaller than 0.1.

Through training, the embeddings of word *w* end up more similar to embeddings of words that occur nearby and less similar to embeddings of words that do not [11]. Thus, embeddings of words that occur in similar contexts also end up similar.

Skip-gram actually learns two embeddings for each word: one vector if the word is used as a target and a second vector when using it as a context word. This results in two matrices $\mathbf{W}$ and $\mathbf{C}$, with the rows being the word embeddings [21]. For downstream applications one usually extracts $\mathbf{w}_i$ or adds together $\mathbf{w}_i + \mathbf{c}_i$.

4.2 Continuous Bag of Words

While the objective in skip-gram is to predict context words from a central focus word, the *continuous bag of words* (CBOW) setting features the reverse task: predict the central word from its context. To this end, one sums up the context word embeddings to a context vector $\mathbf{c} = \sum_{i=1}^{k}\mathbf{c}_i$ [8]. The probability that w is the focus for the context c is then $P(+|w, c_{1:L}) = \sigma(\mathbf{w}\cdot\mathbf{c})$. Negative samples are taken for the focus word: $P(-|w_{neg_i}, c_{1:L}) = 1 - P(+|w_{neg_i}, c_{1:L})$.

While skip-gram worked better in empirical tests and can better represent less frequent words, CBOW trains faster and can better represent more frequent words [17].

4.3 Relationship with PMI

It turns out that count-based and neural-based methods are actually closely related. Assume that each row of a word-context matrix M with dimensions $|V_W| \times |V_C|$ corresponds to a word and each column corresponds to a context. Each cell contains a quantity $f(w, c)$ reflecting word-context association strength. Levy and Goldberg [15] have shown and empirically confirmed, that SGNS with k negative examples is implicitly factorising M since the objective is minimised if $M_{i,j} = \mathbf{w}_i \cdot \mathbf{c}_j = PMI(w_i, c_j) - \log k$.

5 FastText

FastText is an extension of word2vec and was developed in 2017 by Bojanowski et al. [2]. The motivation of FastText is to create representations for out-of-vocabulary (OOV) words and to solve the problem of *word sparsity* using a subword model. FastText is remarkable since it provides word embeddings for 157 languages through a unified interface.

FastText

Strengths	Weaknesses
Better performance on syntactic word analogy than word2vec [3]	Worse performance on semantic analogy than word2vec [3]
Better performance on rare words	Slower to train than word2vec [3]
Better performance on OOV words [3]	

5.1 Infrequent Words and Synthesis

According to Zipf's law, the frequency of a word is inverse proportional to its rank when sorting the words by frequency [22], i.e.

$$\text{word frequency} \propto \frac{1}{\text{word rank}} \tag{13}$$

This has the consequence that most words are infrequent and thus, rarely seen during training.

The problem is even more pressing for languages that exhibit a high degree of *synthesis*, i.e. contain a lot of *morphemes* per word. Morphemes are the meaningful subcomponents in words. For the word *dogs*, the morphemes would be *dog* and *s*, expressing the plural. Languages such as Turkish commonly concatenate several grammatical markers

onto words, leading to one word having numerous different forms, leading to observational sparcity.

5.2 Subword Model

The approach to deal with rare and unknown words introduced by FastText is to use a subword model: each word is represented as the sum of embeddings for its *n*-grams and a special embedding for the word itself [12]. The *n*-grams of a string are all substrings of length n. This allows the model to identify morphological units and character sequences with meaning shared across the vocabulary.

5.2.1 Training Objective

FastText uses SGNS. Let G_w denote the set of *n*-grams that occur in w as well as w itself. Let $\mathbf{z}_g$ be the embedding for a $g \in G_w$. Define

$$P(+|w,c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \sigma(\mathbf{c} \cdot \sum_{g \in G_w} \mathbf{z}_g). \quad (14)$$

The subword-level embeddings are not used for context words. These are encoded with standard word embeddings. The training procedure and objective stay the same as with NGNS.

6 GloVe

The *Global Vectors for Word Representation* (GloVe) method was developed by Pennington et al. [18]. It explicitly constructs a word-context matrix by collecting word co-occurrences and then optimises word and context embeddings to serve as a factorisation of the co-occurrence matrix.

GloVe

Strengths	Weaknesses
Fast training: Only requires a single pass through the corpus to collect co-occurrences	Co-occurrence matrix of words needs a lot of memory for storage [3]
Can leverage large amounts of data better than word2vec [18]	Fixed vocabulary size [3]
Performs better than word2vec on word analogy [18]	Ignores internal structure of words
Performs better than LSA [18]	

The co-occurrences are collected in a context-window of 10 preceding and 10 following words. The counts are weighted using the inverse distance d^{-1} between focus and context word in order to give less weight to distant context words [18].

Then, word and context vectors $\mathbf{w}, \mathbf{c}$ are trained, attempting to satisfy

$$\log \#(w,c) = \mathbf{w} \cdot \mathbf{c} + \mathbf{b}_{[w]} + \mathbf{b}_{[c]} \quad (15)$$

where $\mathbf{b}_{[w]}$ and $\mathbf{b}_{[c]}$ are word- and context-specific biases [8] and $\log \#(w,c)$ is an entry in the word-context matrix. To produce the final embeddings for downstream applications, sum the vectors $\mathbf{w}_i + \mathbf{c}_i$ [8]. One can show that if $\mathbf{b}_{[w]} = \log \#(w)$ and $\mathbf{b}_{[c]} = \log \#(c)$ then the objective is similar to factorising the PMI matrix shifted by $\log(|D|)$ with $|D|$ being the length of the document [8]:

$$\mathbf{w} \cdot \mathbf{c} = PMI(w,c) - \log |D| \quad (16)$$

However, in GloVe the bias terms are learned as part of the model and not fixed to allow for more freedom.

6.1 Training Objective

The training objective is optimised using weighted least-squares loss. The loss function is defined as follows [18]:

$$J = \sum_{i,j=1}^{V} f(\#(w,c))(\mathbf{w}_i \cdot \mathbf{c}_j + \mathbf{b}_{[w_i]} + \mathbf{b}_{[w_j]} - \log \#(w,c))^2 \quad (17)$$

The weighting function maps a co-occurrence to a value in the interval $[0,1]$ and cuts off at some height to prevent learning only from very common word pairs:

$$f(x) = \begin{cases} (x/x_{max})^\alpha, & \text{if } x < x_{max}, \\ 1, & \text{otherwise.} \end{cases} \quad (18)$$

The authors suggest values of $\alpha = 3/4$ and $x_{max} = 100$ [18].

7 Evaluation

There are two main methods for evaluating word embeddings. *Extrinsic evaluation* is done by observing the performance in downstream NLP tasks like parsing or sentiment detection when using embedding vectors. *Intrinsic evaluation* methods are used to directly check for desirable properties in the vectors.

7.1 Intrinsic Evaluation

In the following, we will present several methods for evaluating word embeddings intrinsically.

7.1.1 Similarity

In order to check whether word embeddings have desired characteristics with respect to their pairwise similarity, one can check the correlation between embedding (cosine) similarities and similarity ratings assigned by humans. To this end, one can use pre-existing datasets:

- WordSim-353: similarity and relatedness scores from 0 to 10 for 353 noun pairs [7]
- SimLex-999: similarities for 999 noun-noun, verb-verb and adjective-adjective pairs [10]
- TOEFL dataset: 80 questions for synonyms with a target word and four choices [20]

Finally, one can use Spearman's rank correlation to check how well the relationship of human and predicted similarity can be described using a monotonic function. Possible issues arise from the fact that it is quite hard for humans to assess word similarity on a one dimensional scale. Furthermore, the notion of similarity is subjective and often seems to be confused with relatedness [4].

7.1.2 Word Analogy

The second common intrinsic evaluation method is to test performance on an analogy task (cf. Figure 1). Word analogy features questions of the following type [11]:

$$a \text{ is to } b \text{ as } a^* \text{ is to } \_\_\_?$$

abbreviated to the short form:

$$a{:}b{::}a^*{:}b^* \qquad \text{find } b^*$$

The algorithm is given the vectors for a, b and $a*$ and must find $b*$. We expect the word embeddings to capture relational meaning and thus $\mathbf{b}^*$ being closest to the vector $\mathbf{b} - \mathbf{a} + \mathbf{a}^*$ [11]:

$$\hat{\mathbf{b}}^* = \text{argmin}_{\mathbf{x}} \text{ distance}(\mathbf{x}, \mathbf{b} - \mathbf{a} + \mathbf{a}^*) \quad (19)$$

Finally, one can use accuracy as a metric and determine what proportion of predictions correspond to the correct word vectors.

Possible types of relation include morphology (e.g. city:cities::child:children), lexicographic relations (e.g. leg:table::spout:teapot) and encyclopedic relations (e.g. Beijing:China::Dublin:Ireland) [11]. Datasets that that contain such quadruples for testing word embeddings include:

- SemEval-2012 Task 2 dataset of 79 different relations [13]
- Google analogy test set [17]

Problems can arise when the difference between $\mathbf{a}$ and $\mathbf{a}^*$ is small and $\mathbf{b}$ and $\mathbf{b}^*$ are very close since in such cases the correct answer might be returned even though the offsets are different [16].

8 Conclusion

In the preceding sections we have presented and explained several known and established methods for generating static word embeddings. They are a powerful tool for representing natural language words in downstream NLP applications like syntactic parsing and sentiment analysis. Furthermore, they can be used directly to investigate relationships between words. The methods presented are overall easy to interpret, implement and train. A brief comparison of their strengths and weaknesses can be found in Table 1.

A central weak point for all of the approaches above is their shortcoming with respect to *ambiguities*. Ambiguous words are words that have more than one meaning. Take for instance the word *bank* in the sentences *I worked at a bank.* and *She sat near the river bank.* Using static word embeddings, the word *bank* would receive the same representation in both sentences despite referring to a financial institution in the first and to the edge of the river in the second. To solve this problem, contextualised embeddings were developed which take into account the sentential context for generating a representation for a word [8].

References

[1] R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. 1957. *Dynamic Programming*. Rand Corporation research study. Princeton University Press.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[3] Amit Chaudhary. 2020. A visual guide to fasttext word embeddings. https://amitness.com/2020/06/fasttext-embeddings/. Accessed: 2024-07-04.

	Sparse count-based	LSA with SVD	word2vec	GloVe	FastText
dense	–	+	+	+	+
Deals with unseen words	–	–	–	–	+
Leverages subword information	–	–	–	-	+
Easy interpretation	+	–	–	–	–
Strong for semantic analogy		–	–	+	–
Strong for syntactic analogy		–	–	–	+
Does not introduce randomness	+	+	–	–	–

Table 1: Strengths and weaknesses of the methods discussed in the previous sections.

[4] Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 1–6, Berlin, Germany. Association for Computational Linguistics.

[5] Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 76–83, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

[6] Scott C. Deerwester, Susan T. Dumais, George W. Furnas, Richard A. Harshman, Thomas K. Landauer, Karen E. Lochbaum, and Lynn A. Streeter. 1978. Computer information retrieval using latent semantic structure. US Patent US4839853A.

[7] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131.

[8] Yoav Goldberg. 2017. *Neural Network Methods for Natural Language Processing*. Springer Cham, Basel, Switzerland. ISBN: 978-3-031-01037-8.

[9] Zellig S. Harris. 1954. Distributional structure. In *WORD*, volume 10, pages 146–162. International Linguistic Association (ILA).

[10] Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

[11] Dan Jurafsky and James H. Martin. 2024. Speech and language processing. https://web.stanford.edu/~jurafsky/slp3/. 3rd ed. draft.

[12] Dan Jurafsky and James H. Martin. 2024. Speech and language processing, slides. https://web.stanford.edu/~jurafsky/slp3/. 3rd ed. draft.

[13] David Jurgens, Saif Mohammad, Peter Turney, and Keith Holyoak. 2012. SemEval-2012 task 2: Measuring degrees of relational similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 356–364, Montréal, Canada. Association for Computational Linguistics.

[14] Tom Kocmi and Ondřej Bojar. 2017. An exploration of word embedding initialization in deep-learning tasks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 56–64, Kolkata, India. NLP Association of India.

[15] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

[16] Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 13–18, Berlin, Germany. Association for Computational Linguistics.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Preprint*, arXiv:1301.3781.

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[19] KAREN SPARCK JONES. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

[20] Bo-Hsiang Tseng, Sheng-Syun Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Towards machine comprehension of spoken content: Initial toefl listening comprehension test by machine. *Preprint*, arXiv:1608.06378.

[21] Lena Voita. 2020. NLP course for you, word embeddings lecture slides. Yandex School of Data Analysis. https://github.com/yandexdataschool/nlp_course/tree/2020/week01_embeddings. Accessed: 2024-07-04.

[22] George Kingsley Zipf. 1936. *The Psycho-Biology of Language - An Introdution to Dynamic Philology*. Houghton-Mifflin, Boston, Massachusetts.