Generating Phonetic Embeddings for Bulgarian Words with Neural Networks

Lyuboslav Karev
Faculty of Mathematics and Informatics
Sofia University, "St. Klimenth Ohridski"
lkarev@uni-sofia.bg

Ivan Koychev
Faculty of Mathematics and Informatics
Sofia University, "St. Klimenth Ohridski"
koychev@fmi.uni-sofia.bg

Abstract

Word embeddings can be considered the cornerstone of modern natural language processing. They are used in many NLP tasks and allow us to create models that can understand the meaning of words. Most word embeddings model the semantics of the words. In this paper, we create phoneme-based word embeddings, which model how a word sounds. This is accomplished by training a neural network that can automatically generate transcriptions of Bulgarian words. We used the Jaccard index and direct comparison metrics to measure the performance of neural networks. The models perform nearly perfectly with the task of generating transcriptions. The model's word embeddings offer versatility across various applications, with its application in automatic paronym detection being particularly notable, as well as the task of detecting the language of origin of a Bulgarian word. The performance of this paronym detection is measured with the standard classifier metrics - accuracy, precision, recall, and F1.

Keywords: neural networks, word embeddings, transcriptions, phonemes, grapheme-to-phoneme

1 Introduction

In natural language processing, automatically generating transcriptions of words is a formidable challenge, particularly when confronted with languages like Bulgarian. Renowned for its near-perfect phonemic orthography, wherein each letter typically corresponds to a single sound, Bulgarian presents an intriguing paradox. While its orthographic structure promises clarity, exceptions within the system thwart the straightforward application of bijective mapping algorithms. These anomalies underscore the need for innovative approaches that seamlessly reconcile orthographic and phonetic representations.

In this work, we create a neural network that generates a phoneme transcription from a word. The model is trained on five books, transforming each into a pair of words and transcriptions.

One particularly noteworthy application of our model lies in automatic paronym detection. By seamlessly integrating phonetic representations into the detection process, our model demonstrates good results for paronym detection.

Another application of phoneme embeddings is the detection of loanwords in the Bulgarian language. A classifier is trained using the embeddings as input, which classifies the language of origin of a given word.

1.1 Main concepts

We will define a few concepts that will be used throughout the paper.

- Phoneme: The smallest sound unit in a language that can differentiate words.
- Transcription: The process of representing spoken language in written form.
- Grapheme: The smallest unit of a writing system that represents a phoneme in the spelling of a word.
- Syllable: In the Bulgarian language, a syllable is a collection of sounds containing exactly one vowel.
- International Phonetic Alphabet: Standardized phonetic notation system representing spoken language sounds.

1.2 Rules for generating phonetic transcriptions

In the Bulgarian language, generating the phonetic transcription for a word is almost straightforward. We need to know the word and which syllables are stressed to get the transcription. We can check if the letter is a consonant or a vowel by going through each letter. We can replace the letter with the matching phoneme if the letter is a consonant. If the letter is a vowel, we have to check if the vowel is stressed and then pick the correct phoneme based on that.

The result is a string written in the International Phonetic Alphabet (IPA). As an example, the Bulgarian word ”здравей” (meaning ”hello”) will look something like this: ”zdrʌvɛj”

1.3 Syllabic transposition rules

In the Bulgarian language, the structure of a syllable depends on the amount and placement of consonants around a vowel. From this, we can distinguish four types of syllables Весела Кръстева (2009):

- Syllables that have only a single vowel.
- Syllables that have one or more consonants followed by a vowel.
- Syllables that have a vowel followed by one or more consonants.
- Syllables that have one or more consonants, followed by a vowel, followed by one or more consonants.

This structure is not enough to split a word into syllables. To solve this problem, we can use the hyphenation rules in the Bulgarian language, described below. БАН (2011)

1. The consonant letter, which is between two vowels, is hyphenated on the next line.
2. Two or more consecutive consonant letters between two vowels are hyphenated such that at least one consonant is after the first vowel and at least one before the second vowel.
   (a) Two repeated consonants are split equally on the first and second part of the hyphenation.
3. Two consecutive vowels are hyphenated such that one is on the first line while the second is on the second line.

A word can automatically be split into syllables with this set of rules.

2 Related work

Generating the phoneme transcriptions from the grapheme representation is not new, and multiple approaches have been proposed.

Black and Lenzo (2003) and Elovitz et al. (1976) both describe a rule-based approach where graphemes are directly replaced with their corresponding phonemes. On an English language dataset, Elovitz et al. (1976)'s approach achieves around 80% of correctly generated transcriptions, while Black and Lenzo (2003)'s approach results in around 70% of correctly generated transcriptions.

A statistical model is described in Bisani and Ney (2008), which relies on modeling the translation process as a linear sequence of operations. The model generates a phonemic transcription from the orthographic form of a word. This is achieved by approximating the sum of the joint probabilities of all possible graphone sequences that match the given spelling.

Rao et al. (2015) describes an LSTM-based model for performing grapheme-to-phoneme conversion with a 25.8% error rate on a standard English dataset.

Li et al. (2022) describes multiple approaches that work on a single language - a joint n-gram model, a sequence-to-sequence LSTM model, and a transformer model. The models handle unknown languages by finding the closest language based on the language family tree and using the nearest k language models. The downside of this approach would be that if a model is not trained on Bulgarian data, it might not yield sufficient results.

Yolchuyeva et al. (2019) and Engelhart et al. (2021) propose using transformer-based models to automatically generate the transcriptions from words and apply the model to different tasks for the English language.

In this paper, a similar transformer-based network to Yolchuyeva et al. (2019) and Engelhart et al. (2021) will be used, with the main difference being that the network will be trained exclusively on Bulgarian words and transcriptions to solve the task of automatic transcription generation for the Bulgarian language.

3 Data gathering, analysis, and transformation

3.1 Data gathering

The initial data set was collected from the website https://chitanka.info/. Five books, each in a separate text file, were downloaded from this website.

The website of ”Chitanka” also provides an SQL database containing information about the stressed syllables in a word. All words and their stressed syllables are downloaded from the database. A small transformation was applied to the words. The stressed syllables are marked with the ‘ symbol. For this to be understood by the model, We replaced the ‘ symbol with the index of the stressed vowel. For example, the word ”авиобранш” has two stressed vowels - the two ’а’s. The entry in the database for that word would be ”а‘виобра‘нш”. After the transformation, the indices of the stressed syllables would be at indices 0 and 6.

3.2 Transformation of the data

For each of the books, the following transformation is applied:

1. Read all lines of the book.
2. Split each line into separate sentences.
3. For each sentence, remove any special formatting from the website.
4. Split each sentence into words.
5. For each word, generate its transcription.
6. Each word, along with its transcription, is written in a file.

This transformation of the five books results in a list of 606,102 pairs of words and transcriptions.

3.3 Analysis of the data

A short analysis of the data shows us that the amount of unique words is 39,405. The ten most common words are listed under Table 1

From this, we can see that the most common words are the conjunctions ”на”, ”да”, ”и”, ”се”, etc.

A word is seen 15.381 times on average in the dataset, making the dataset imbalanced in terms of words. However, it’s vital to note that natural language is also imbalanced, so this dataset reflects real-life usages of the words. The average length of a word is 4.673 characters. The distribution of the lengths of the words can be seen in Figure 1.

Word	Count	% count
на	20369	3.36%
да	20311	3.35%
и	20048	3.31%
се	16139	2.66%
в	10114	1.67%
от	9666	1.59%
не	8151	1.34%
си	7617	1.26%
с	7160	1.18%
че	6724	1.11%

Table 1: 10 most common words

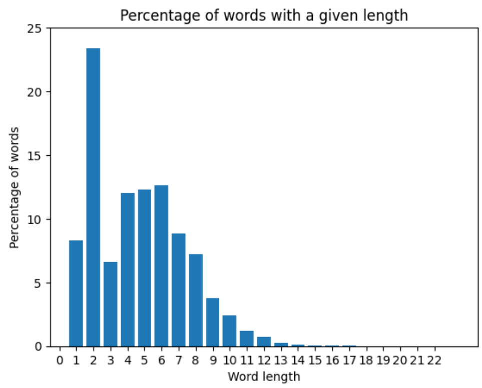

Figure 1: Word length count

Regarding the syllables, the dataset contains a total of 1,222,793 syllables. The unique syllables are 5,469. The 10 most common syllables are listed under Table 2. The average length of a syllable is 2.31 characters. The distribution of the lengths of the syllables can be seen in Figure 2. There, we can see that the 2-character syllables are the most common, with 61.482%, followed by 3-character syllables, with 23.695%.

3.4 Data segmentation

The dataset is split into three subsets: one for training, one for validation, and one for testing. The subsets are split in the following way:

- Training set - 80% of the data (484,881 pairs)
- Validation set - 10% of the data (60,610 pairs)
- Testing set - 10% of the data (60,611 pairs)

Syllable	Count	% count
на	53107	8.76%
то	34352	5.67%
та	32000	5.28%
и	30720	5.07%
да	28596	4.72%
ни	23501	3.88%
те	22548	3.72%
ка	22094	3.65%
се	21358	3.52%
е	20169	3.33%

Table 2: 10 most common syllables

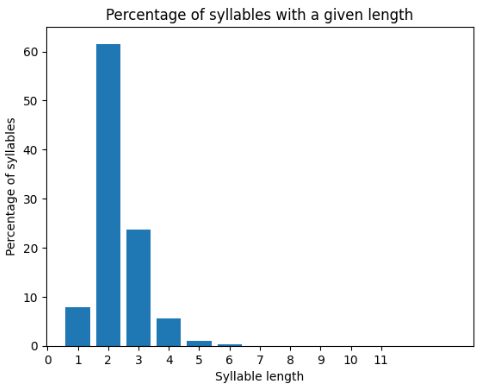

Figure 2: Syllable length count

4 Model overview and training

4.1 Transformer model

The idea is to train a neural model based on the *Transformer* architecture Vaswani et al. (2023).

We introduce the following notation:

- s - syllable.
- $W = s^+$ - word, containing at least one syllable.
- s^W - word W, split into its syllables.
- l_W - the amount of syllables in the word W.
- t_s - the transcription of the syllable s.
- t_W - the transcription of the word W.
- $f(s) : s \mapsto \mathbb{N}$ - function, which maps a syllable to an index.
- $f'(s) : \mathbb{N} \mapsto t_s$ - function, which maps an index to a transcription of a syllable.
- i_s - the index of the syllable s, $f(s) = i_s$.
- i_W - the indices of the syllables from the word W.
- i_{t_s} - the index of the transcription of the syllable t_s.
- i_{t_W} - the indices of the transcription of the word W.
- v_{i_W} - the input vector of the model.
- o_{i_W} - the output vector of the model.
- Embedding - a vector representation of a word, or in this case, a syllable.
- emb - the size of the resulting embeddings from the model.
- <bos> - tag for the start of a word.
- <eos> - tag for the end of a word.

The neural model accepts a vector v_{i_W} with size l_W and returns a new vector o_{i_W} with size l_W.

Before each word W can be sent to the model, it must be transformed. This transformation is described in Section 4.2

The model contains an embedding layer, an encoder layer, a decoder layer, and a linear layer. The input vector is transformed into an input embedding augmented with positional encodings, which provide information about the order of the syllables in a word. From there, the positional encoded input is passed to the encoder and decoder parts of the Transformer model. The encoder encodes the input into a series of hidden representations, which pass through the layers of the encoder. The encoder outputs a series of embeddings, which are then fed to the decoder, which generates the transcription of the word. The final part of the model is a linear layer, which transforms the output of the transformer model into unnormalized probabilities for each transcription token.

4.2 Transforming a word W

The transformations applied over a given word w so that it can be used as input for the model are as follows:

The word W is split into its syllables s^W. For each syllable s, $f(s)$ is applied, resulting in a vector of indices of the syllables of W, i_W. At the start and end of this vector, the unique tags <bos> and <eos> are added. As a result, the vector v_{i_W} is created and can be used by the model.

The model's output is a vector o_{i_W}, which has a size of l_W. It contains the special symbols <bos> and <eos>. After their removal, the vector i_{t_W} is left. We apply f' for every index i_{t_s}, to get the transcription t_s for the syllable s. Once we have all the syllables, we get the transcription t_W of the word W.

4.3 Training of the model

For training the module, the samples are passed in batches of 128.

The parameters of the model chosen during training are the following:

- Amount of Encoder layers: 3
- Amount of Decoder layers: 3
- Embedding size (E): 512

The loss function used for training is the Cross entropy loss function. It is used when training models that solve multi-class classification problems. With this function, we can quantify how well the model performs. It evaluates the output vector of the model against the expected result vector and returns a scalar. The lower the number, the better the model performs.

The training is done for 25 epochs. Figure 3 shows the loss value change across each training epoch.

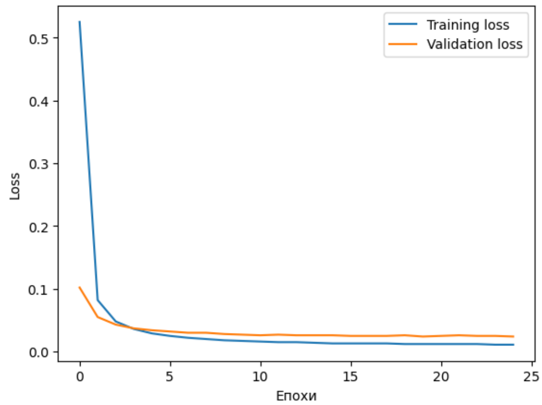

Figure 3: Results of the loss function on the train and validation datasets across the epochs

On the 25th epoch, the value of the loss function is as follows: Train set - 0.011, validation set - 0.024.

5 Usage of the phonetic embeddings

In this section, we will use the following notation:

- E_W - the resulting vector from the encoder layer of the model. Size: (l_W, emb)
- $wt_s(W)$ - the function that transforms the word W to the valid input for the model. This function is described in Section 4.2.

The embeddings result from using only the encoder part of the model. The user enters a word W, which is transformed (4.1) into the input vector for the model v_{i_W}. From there, the vector is passed to the encoder part of the model, which returns the vector E_W. As the model works on syllables, the returned embedding is not a single vector, but l_W vectors, each of size emb (which in this case is 512), containing floating-point numbers.

6 Experiments

Apart from the notation used in Section 4.1, we will introduce the following symbols:

- t_W^p - the transcription of the word w, which results from the model working on the word W.
- t_W^a - the transcription of the word W, taken from the dataset.

6.1 Result over random words

The model is run over a random set of Bulgarian words. The results of this experiment can be seen in Table 3.

word	generated trasctiption
здравей	zdrʌvɛj
благодаря	blʌgodʌrjɐ
лято	ljɐto
зима	zimʌ
книга	knigʌ
кафе	kʌfɛ
синьо	sinjo
часовник	tʃʌjɐjɐsnik
градина	grʌdinʌ
слънце	slɐntsɛ
месец	mɛsɛts
живот	ʒivot
река	rɛkʌ
музика	mozikʌ
храна	xrʌnʌ

Table 3: Results on a random set of Bulgarian words

As we can see, most of the words are correct. One of the examples that is wrong is the word ”часовник”, the transcription of which is generated as tʃʌjɐjɐsnik. Converting this transcription back to a written form would result in the word ”чаяясник”, which is incorrect.

6.2 Jaccard index

The Jaccard index is the first objective metric to evaluate the model's performance. This metric evaluates the similarity between two sets and is defined over sets and multisets.

Let's define the function $J(A, B)$, which measures the Jaccard index over two sets.

For two sets, A and B, $J(A, B)$ is defined as follows:
$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A|+|B|+|A \cap B|}$.

As the transcription t_W of the word W can be considered an ordered multiset containing the transcriptions t_s of the syllables s, we can use the Jaccard index as an evaluation metric. This metric can evaluate what percentage of the generated syllables are correct (ignoring their ordering). As the transcription of a single syllable can appear multiple times in the transcription of the word, we must modify the Jaccard index to support multisets.

We can define a function $J_m(A, B)$, where A and B are multisets, in the following way:

$J_m(A, B) = \frac{|A \cap B|}{|A|+|B|}$. It's important to mention that $J_m(A, B)$ returns values in the $[0; 1/2]$ range. To normalize this interval, the result is multiplied by 2.

We can now define the metric Jaccard index for two transcriptions t^a_W and t^p_W in the following way:
$J_t(t^a_W, t^p_W) = 2 * (\frac{|t^a_W \cap t^p_W|}{|t^a_W|+|t^p_W|})$

Evaluating the test set with the Jaccard index metric yields a result of 99.571% match

6.3 Direct comparison

The Jaccard index only evaluates whether the generated transcriptions of the syllables are correct, ignoring their ordering. Another metric must be defined to get a metric that includes the ordering. A good candidate is the direct comparison of the two transcriptions.

We can define a function that compares the syllables of two transcriptions t^a_W and t^p_W pairwise.

$$c(t^a_W, t^p_W) = \begin{cases} 1 & \text{If } |t^a_W| = |t^p_W| \\ & \text{and for } \forall i \in 0..|t^a_W| \\ & \text{the following is true: } t^a_{Wi} = t^p_{Wi} \\ 0 & \text{otherwise} \end{cases}$$

We are evaluating the test set with the direct comparison metric, which yields a result of 99.285% match.

Comparing this result to the Jaccard index metric, we can see that 0.286% of the generated transcriptions have all the correct syllables but in a different order than in the correct transcription.

7 Paronyms detection

7.1 Introduction

In the Bulgarian language, two words are paronyms if they are close in sound but different in meaning. As an example, the words ”статист” и ”статистик” sound close (in this case, they share syllables) but have vastly different meanings.

We hypothesize that the embeddings from the model described above can be used to detect if two words are paronyms. This can be done by classifying two words based on their ”phonemic distance”—the distance that would model how the words sound.

The metric used to calculate this ”phonemic distance” is the Cosine similarity metric. This metric gives us the level of similarity between two vectors. It's defined the following way:

$S_C(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$. The cosine similarity returns values in the $[-1; 1]$ range. A value of -1 means the vectors are opposite, while one means the vectors are proportional.

Based on this, we can define cosine similarity between two embeddings.

Let E_1 and E_2 be the embeddings of the words W_1 and W_2. Let l_{W_1} and l_{W_2} be the amount of syllables in the two words and $l = min(l_{W_1}, l_{W_2})$ Then:

$S_E(E_1, E_2) = \frac{\sum_{i=0}^{l} S_C(E_{1i}, E_{2i})}{l}$

S_E again belongs to the $[-1; 1]$ range. Result -1 will be interpreted as the words don't sound alike, while 1 will mean that the words do sound alike.

7.2 Experiments and results

From Върбанова and Весела Кръстева (2009) 92 pairs of paronym words were collected. Ninety-three pairs of non-paronym words were added, resulting in a dataset of 185 pairs. This dataset was split into train and test sets in an 80:20 ratio—148 pairs for training and 37 for testing.

A logistic regression classifier is used to determine whether two words are paronyms. The model uses the cosine similarity between the embeddings of the two input words as input.

The following notation and metrics are used to evaluate the model: accuracy, precision, recall, and F1 score.

As a comparison, we used the word embeddings from the fastText library Bojanowski et al. (2016), Joulin et al. (2016) to train the same classifier on the same dataset. The results are shown in Table **??**

Metric	Score	
	Phonetic	fastText
accuracy	0.892	0.892
precision	0.850	0.938
recall	0.944	0.833
F1	0.895	0.882

Table 4: Results over the paronyms dataset comparing phonetic embeddings with fastText word embeddings

As we can see, our phonetic embeddings show the same accuracy as the fastText ones. fastText performs better on the precision metric, however loses a bit in the recall. Comparing the F1 scores shows that our phonetic embeddings perform around the same as the fastText embeddings.

8 Language of origin detection for Bulgarian words

8.1 Introduction

As the phoneme embeddings mentioned in Section 5 model the way a word sounds, we propose that the embeddings can be used to detect the language of origin of a Bulgarian word. We assume that loanwords in Bulgarian will sound differently than regular Bulgarian words. As part of this section, an RNN-based classifier is trained using data from loanword dictionaries. The dataset consists of pairs of words and their language of origin. Table 5 shows the amount of loanwords in the dataset. From there, three datasets are created, with different amounts of Bulgarian words - with 5 000 Bulgarian words, with 13 395 Bulgarian words, and with 30 875 Bulgarian words. Each model accepts Bulgarian words as input, gets its phoneme embedding, and returns the probability of the word belonging to a certain language.

Language	Word count	% of total
Latin	1504	29.10%
Greek	984	19.04%
French	958	18.51%
Turkish	658	12.73%
English	478	9.25%
German	240	4.64%
Italian	156	3.02%
Russian	102	1.97%
Spanish	33	0.64%
Dutch	29	0.59%
Hebrew	11	0.21%
Arabic	7	0.14%
Serbian	4	0.08%
Persian	3	0.06%

Table 5: Words from a given language

From there, three models are trained, depending on the number of Bulgarian words used - we'll call these models *Phoneme-5k*, *Phoneme-13k* and *Phoneme-30k*.

8.2 Experiments and results

The standard metrics—accuracy, precision, recall, and F1—are used to evaluate the classifiers. For comparison, a classifier using the fastText word embeddings was trained on the same datasets. The fastText-based classifiers will be referred to as *fastText-5k*, *fastText-13k* and *fastText-30k*.

Model	Metric			
	Accuracy	Precision	Recall	F1
Phoneme-5k	0.590	0.553	0.590	0.565
Phoneme-13k	0.736	0.727	0.736	0.730
Phoneme-30k	0.806	0.807	0.806	0.806
fastText-5k	0.672	0.697	0.672	0.653
fastText-13k	0.794	0.797	0.794	0.789
fastText-30k	0.868	0.864	0.868	0.861

Table 6: Comparsion between our phoneme embeddings and fastText embeddings for the language of origin task

The results from Table 6 show that the fastText embeddings perform a bit better than the phoneme embeddings, although it's not a sizeable difference.

9 Limitations

While the results of the model are looking good, there are some limitations on it. The input of the model does not include information about the emphasis of the word. This information is only used to generate the proper transcription for the training data. This impacts the phonemes generated by the model.

The phonetic embeddings also don't seem to exhibit any of the properties present in other word embeddings. Embeddings like word2vec Mikolov et al. (2013) represent the semantic and syntactic relationships between the words. For example, the distance between the words ”man” and ”woman” is similar to the distance between the words ”king” and ”queen”. The phonetic embeddings however don't exhibit such connections. For example, if we have two words, which differ only at the suffix, the distance between their embeddings varies from small to a large, and is not consistent through different pairs of words.

10 Conclusion

A dataset containing 600,000+ words and their transcription was created. A transformer-based model was created to solve the Bulgarian language's grapheme-to-phoneme task. The model performs with very high accuracy. Embeddings extracted from this model were used in a simple classifier that checks if two words are paronyms. The classifier performs also with a high accuracy percentage.

As a result, there is now an automatic system for paronym detection and automatic generation of phonemic transcriptions of Bulgarian words. These approaches can be applied to other languages that are different from Bulgarian. In future research, we intend to use the embeddings to detect if a word is a loanword.

11 Acknowledgments

The work is partially financed by the European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project No BG-RRP-2.004-0008.

References

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Commun.*, 50:434–451.

Alan Black and Kevin Lenzo. 2003. Issues in building general letter to sound rules.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

H. S. Elovitz, Rodney W. Johnson, Astrid McHugh, and John E. Shore. 1976. Automatic translation of english text to phonetics by means of letter-to-sound rules (nrl report 794.

Eric Engelhart, Mahsa Elyasi, and Gaurav Bharaj. 2021. Grapheme-to-phoneme transformer model for transfer learning dialects.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Xinjian Li, Florian Metze, David Mortensen, Shinji Watanabe, and Alan Black. 2022. Zero-shot learning for grapheme to phoneme conversion with language ensemble. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2106–2115, Dublin, Ireland. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks | IEEE Conference Publication | IEEE Xplore.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need.

Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. 2019. Transformer based grapheme-to-phoneme conversion. In *Interspeech 2019*, pages 2095–2099.

Институт за български език при БАН. 2011. Правопис и пунктуация на българския език. [Institut za balgarski ezik. Pravopis i punktuatsiya na balgarskiya ezik. Prosveta, 2011.].

Павлина Върбанова. 2021. Списък на пароними. Как се пише. [Pavlina Varbanova. Spisak na paronimi. Kak se pishe.].

Весела Кръстева. 2009. Практическа граматика на съвременния български език. Кръгозор. [Vesela Krasteva. Prakticheska gramatika na savremenniya balgarski ezik. Kragozor, 2009.].