Learning Word Embeddings for Ukrainian: A Comparative Study of FastText Hyperparameters

Nataliia Romanyshyn[*]
Ukrainian Catholic University
Lviv, Ukraine
romanyshyn.n@ucu.edu.ua

Dmytro Chaplynskyi[*]
Lang-uk
Kyiv, Ukraine
chaplinsky.dmitry@gmail.com

Kyrylo Zakharov
Court on the Palm
Kyiv, Ukraine
kirillzakharov13@gmail.com

Abstract

This study addresses the challenges of learning unsupervised word representations for the morphologically rich and low-resource Ukrainian language. Traditional models that perform decently on English do not generalize well for such languages due to a lack of sufficient data and the complexity of their grammatical structures. To overcome these challenges, we utilized a high-quality, large dataset of different genres for learning Ukrainian word vector representations. We found the best hyperparameters to train fastText language models on this dataset and performed intrinsic and extrinsic evaluations of the generated word embeddings using the established methods and metrics. The results of this study indicate that the trained vectors exhibit superior performance on intrinsic tests in comparison to existing embeddings for Ukrainian. Our best model gives 62% Accuracy on the word analogy task. Extrinsic evaluations were performed on two sequence labeling tasks: NER and POS tagging (83% spaCy NER F-score, 83% spaCy POS Accuracy, 92% Flair POS Accuracy).

1 Introduction

Word embeddings (Almeida and Xexéo, 2019) are fixed-length vector representations of words that have a variety of applications in natural language processing (NLP) tasks, including semantic text similarity (Nguyen et al., 2019), word sense disambiguation (Ruas et al., 2019), text classification (Mandelbaum and Shalev, 2016), question answering (Shen et al., 2017). Word embedding techniques rely on the distributional hypothesis – the assumption that the meaning of a word is captured by the contexts in which it appears (Harris, 1954).

Even though unsupervised word embeddings can be learned directly from raw texts, gathering a significant amount of data for their training remains an immense challenge for low-resource languages such as Ukrainian. Moreover, Ukrainian is a morphologically rich language; its nouns decline for 7 cases, three genders, and two numbers. Adjectives agree with nouns in case, gender, and number. Verbs conjugate for four tenses, two voices, and two numbers. Ukrainian verbs come in aspect pairs: perfective and imperfective. Not surprisingly, traditional models that give excellent results for English may not be able to generalize well for highly inflected languages, such as Ukrainian, without special tuning.

To address these challenges, we worked with a high-quality, large dataset of different genres for learning vector representations of words in the Ukrainian language. We identified the best hyperparameters to train fastText language models on this dataset and performed the intrinsic and extrinsic evaluations of the generated word embeddings using firmly established methods and metrics. Furthermore, the obtained vectors were compared with the ones previously published by the Facebook team[1] (Grave et al., 2018) and those trained using the default hyperparameters. Our optimized models outperformed the baseline models by **7.1%** in Accuracy on the word analogy task, and showed a **6.4%** improvement compared to the Ukrainian word embeddings published by Grave et al. (2018).

The novel contributions of this work are:

- Conducted the first study of effects of various hyperparameters for fastText word embeddings in the Ukrainian language.
- Created and made publicly available the largest collection of pre-trained Ukrainian word embeddings. The best models are available on the Hugging Face platform[2], and others upon request.
- Presented a new word analogy dataset for

[*]These authors contributed equally to this work.

[1]https://fasttext.cc/docs/en/crawl-vectors.html

[2]https://huggingface.co/dchaplinsky/fasttext_uk

Ukrainian[3].

- Provided the first formal analysis of Ukrainian word vectors utilizing intrinsic and extrinsic approaches.

The obtained results allow NLP practitioners to reduce the computational resources and time required to develop algorithms for solving applied NLP tasks.

The rest of the paper is organized as follows. Section 2 contains an overview of related work. Section 3 presents the data collection and preprocessing techniques. We describe the methodology for learning word vectors and conducted experiments in Section 4. Section 5 describes the evaluation methods and obtained results. We conclude and present future work in Section 6.

2 Related work

A standard approach (Miaschi and Dell'Orletta, 2020) for learning non-contextual word representations is to train a log-bilinear model based on the continuous bag-of-words (cbow) or the skip-gram architectures. An example of such a language model is word2vec, described by Mikolov et al. (2013a). It learns continuous representations of words using a shallow neural net. Mikolov et al. (2013b) showed that these representations capture syntactic and semantic regularities in language decently, and presented a novel method — word analogies — for evaluating word embeddings based on vector arithmetic.

Word2vec's main drawback is that it ignores the word's internal structure that contains rich information. This knowledge could be beneficial in computing representations of uncommon or misspelled words and for morphologically rich languages like Ukrainian. To address this issue, the fastText method (Bojanowski et al., 2017) proposed to take word morphology into account by representing each word as a bag of character n-grams. They evaluated the fastText model on nine languages exhibiting different morphologies, but Ukrainian was omitted.

Grave et al. (2018) developed and released fastText language models for 157 languages, including Ukrainian, but they did not provide any evaluation of the embeddings for this language. They used Wikipedia, which is of limited diversity, and quite noisy data from the Common Crawl (CC) project[4] for learning their word vectors. Another shortcoming is that they did not optimize subword size and employed character n-grams of size 5–5 for all languages. Also, the authors concluded that the quality of the obtained vectors for low-resource languages is significantly worse than for high-resource ones.

Novotný et al. (2021) discovered that subword sizes have a significant impact on the Accuracy of fastText language models, and their optimization enhanced the Accuracy on word analogy tasks by up to 14%. The authors proposed a simple n-gram coverage model and discovered the optimal subword sizes on the English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian word analogy tasks. We utilized their findings for Czech and Russian as they also belong to the Slavic language group.

In the current study, we evaluate how the corpus size, specific models, and set of hyperparameters affect the quality of Ukrainian word embeddings for different NLP tasks. Also, we conduct a proper evaluation of the proposed by Grave et al. (2018) word embeddings for Ukrainian and compare them with our results.

3 Training Dataset

This section presents our datasets and preprocessing techniques.

3.1 Corpus Selection

Currently available corpora for the Ukrainian language include:

- Zvidusil[5] corpus, which contains 2,8 billion tokens. Around ten text sources of Zvidusil are collected by the specialized parsers. The rest is retrieved automatically using SpiderLing[6];
- General Regionally Annotated Corpus of Ukrainian (GRAC)[7], collected by Shvedova et al. (2017-2022). It consists of approximately 890 million tokens and has various genre coverage, good general quality, and unique texts;
- other corpora[8] of a smaller size.

[3]https://github.com/lang-uk/vecs/blob/master/test/test_vocabulary.txt

[4]https://commoncrawl.org

[5]https://mova.institute/

[6]http://corpus.tools/wiki/SpiderLing

[7]http://uacorpus.org/

[8]http://uacorpus.org/Kyiv/ua/other-ukrainian-corpora

General corpora statistics are summarized in Table 1.

Table 1: Statistical information on Zvidusil and GRAC corpora.

Corpus	**Zvidusil**	**GRAC**
# of tokens	2,848,203,658	889,097,859
# of sentences	155,821,729	55,324,205
# of documents	6,936,227	113,569

Unfortunately, neither Zvidusil nor GRAC datasets are open-source and available for direct download.

Therefore, we decided to use other corpora — UberText 1.0[9] and the developer preview of UberText 2.0[10]. We summarize these datasets in Table 2.

UberText 1.0 is the smaller one and includes 11 news websites spanning 2006-2016, Ukrainian Wikipedia as of 2016, and fiction. UberText 2.0 — the bigger one, at the moment of the experiment, consisted of 30 news websites spanning 2000-2021, Ukrainian Wikipedia as of 2021, and a bigger subcorpus of fiction. The final version of the UberText 2.0 corpus is a subject of a separate paper (Chaplynskyi, 2023); here, we only cover the essential aspects of its composition and preprocessing.

Table 2: Training datasets description. The **# of words** indicates the vocabulary size of the models trained on this dataset and equals to the number of unique tokens.

Corpus	**UberText 1.0**	**UberText 2.0**
# of tokens	665,322,645	1,589,010,407
# of words	1,758,917	2,665,029
# of sentences	48,522,905	126,696,187
Size	8.4 GB	20.1 GB

Both datasets were collected using custom-written spiders for the Scrapy[11] framework to parse publicly available sources of Ukrainian texts.

The selection of sources covers modern vocabulary of the Ukrainian language and, therefore, is helpful for the downstream tasks. All the texts were converted from the HTML markup to Markdown standard using html2text[12] to maintain the basic structure of headers and sub-headers.

In comparison to UberText 1.0, the second version provides the following improvements:

- more sources of texts;
- texts that were added to the existing sources since 2016;
- better internal structure and meta information on texts (authorship, tags, images).

It was a deliberate decision not to include Common Crawl or Oscar[13] corpora data into UberText because of their aggregated nature and instead focus on individual sources of texts rather than deal with noisy input.

3.2 UberText Preprocessing

All the texts were preprocessed using the nlp-uk[14] library, a wrapper for the LanguageTool[15]; the following techniques were applied:

1. cleansing — removal of Markdown tags, fix for broken encodings, normalization of the hyphens, apostrophes, fixes for mixed Latin/Cyrillic texts, fixes for simple word wraps;
2. rating for the number of used Ukrainian and Russian dictionary words and characters specific to Ukrainian and Russian alphabets;
3. tokenization into paragraphs, sentences, and words using the LanguageTool tokenizer for the Ukrainian language;
4. removal of punctuation marks.

No changes were made to the word capitalization in texts.

During the export of the texts, the following filters were applied:

- Texts with a substantial amount of Russian words (over 25%) were removed to exclude articles wholly or partially written in Russian.
- Articles shorter than 100 characters were also removed.

[9] https://lang.org.ua/static/downloads/corpora/ubercorpus.txt.tokenized.noemptylines.no_markdown.txt.bz2

[10] https://lang.org.ua/static/downloads/corpora/ubertext.fiction_news_wikipedia.filter_rus+short.tokens.txt.bz2

[11] https://scrapy.org

[12] https://pypi.org/project/html2text/

[13] https://oscar-corpus.com

[14] https://github.com/brown-uk/nlp_uk

[15] https://github.com/languagetool-org/languagetool

4 Embedding methods

While recent advances in NLP have been dominated by transformer-based language models, there is still a place for simpler models like continuous bag-of-words (cbow) and skipgram (Mikolov et al., 2013a) in certain scenarios. These models offer several advantages over more complex ones, particularly in low-resource settings. For one, they are computationally efficient and can be trained on smaller datasets. Additionally, they offer greater interpretability and transparency, making it easier to understand how the model makes its predictions.

Given these advantages, we choose to use cbow and skipgram methods for obtaining context-independent word embeddings for our study.

Cbow model learns to predict a target word based on its context, using the sum of the background vectors. A predefined window size surrounding the target word represents the neighboring terms taken into account.

Skipgram is another architecture for creating word embeddings. The model uses a target word for predicting the context by summing the log probabilities of the surrounding words to the left and right of the target word.

For the study, we have chosen the fastText[16] implementation of these models, where morphology is taken into account. Each word is represented as a bag of character n-grams (i.e., subwords), and the word vector is obtained by taking the sum of the vectors of the character n-grams appearing in that particular word (Bojanowski et al., 2017). While being the golden implementation, it has two shortcomings that weren't described in the project documentation:

- has a hyperparameter wordNgram that does not impact the training;
- it lacks the implementation of the cbow algorithm with positional weights, called cbow weighted in what follows. Such an algorithm was first described in the work of Grave et al. (2018) and used to train reference word vectors, available for 157 languages. To overcome this issue, we have switched to an alternative implementation described in the paragraph below.

Cbow with positional weights is a variation of the cbow model that modifies the input vectors of context words to better depict the relationship between the target and context words based on their relative positions (Novotný et al., 2022). The authors noted that the positional model more than doubles the training time since, for each gradient update of an input vector, we also need to update the weights of a positional vector. We utilized the implementation of the positional weighted model presented in the paper mentioned above.

4.1 Baseline

In order to compare learned embeddings, we trained the fastText model on our dataset UberText 2.0 (20.1 GB) with default parameters, but the vector dimension was fixed to 300 instead of 100. We introduce this model as a "Baseline".

4.2 Hyperparameter tuning

We decided to add the following modifications to the Baseline model for obtaining high-quality word vectors:

1. more epochs for training; by default, the fastText library trains for five epochs;
2. more negative samples; by default, it samples five negative examples;
3. use different character n-grams size instead of the default range of 3–6;
4. also utilize the cbow model for learning word vectors; the default is the skipgram variant.

The increment in the number of the training epochs and negative samples refers to the results of Grave et al. (2018) experiments, which show that although such adjustments increase training time, they result in a significant increase in Accuracy. Subword ranges were chosen based on Novotný et al. (2021) reported accuracies on word analogies for Czech and Russian, which are the most related to Ukrainian among the studied languages.

In Table 3, we have collected all selected options for training fastText vectors for the current study. To find the best setting, we explored 32 combinations of parameters for learning word representations.

Since the cbow weighted model requires much time to learn, we did not train it on all combinations of parameters but used the best one according to our previous evaluation on other models and their performance on the word analogies. Therefore, subword range 2-5 and 15 negative samples were

[16] `https://fasttext.cc/`

Table 3: Selected parameters to study their impact on Ukrainian vectors' quality. Subword refers to the min and max character n-gram.

Model	Epochs	Subword	Negative Sampling
cbow	10	2-5	10
skipgram	15	2-6	15
		4-6	
		5-6	

used. Also, following the Grave et al. (2018) experiment setup, we trained word vectors with character n-grams of length five only and ten negative samples. As mentioned in Novotný et al. (2022), the positional model can benefit from a larger context window. Therefore, we set it to 15, as the optimal context window size defined by the authors.

To run the training on all the combinations of hyperparameters, we wrote the software[17] that can be quickly deployed on any server with enough RAM, effectively turning that server into a computational node that picks the next available task from the pool. That allowed us to quickly deploy it to the farm of seven servers and parallelize the grid search.

4.3 The impact of training data size

Another experiment was conducted to investigate the result stated by Bojanowski et al. (2017) regarding the impact of training data size on the quality of produced vectors. They argued that high-performing word embeddings can be constructed on a corpus of a restricted size while still performing well on previously unseen data. For the purpose of investigating this claim, we utilized the smaller corpus UberText 1.0 (8.4 GB) and trained with optimized parameters and hyperparameters on intrinsic and extrinsic tasks. See more details in Sections 5.1.2 and 5.2.3.

4.4 Estimating the hyperparameter significance

The quality of calculated word vectors was measured with the Accuracy and F1 score. Both metrics are continuous variables expressed in a range from 0 to 1. Therefore we used a Beta regression to regress the hyperparameters. It is assumed that the model's dependent variable is beta-distributed and that its value is related to a set of independent variables through a linear predictor with unknown coefficients and a link function (logit in our case). Calculations were made with betareg package (Zeileis et al., 2016) in the R environment for statistical computing (R Core Team, 2013).

5 Evaluation

In this section, we describe various evaluation metrics on trained word vectors. Prior work on evaluation of the embeddings can be divided into intrinsic and extrinsic evaluation methods (Torregrossa et al., 2021). We start with the intrinsic evaluation on the word analogy task, then continue with named entity recognition (NER) and part-of-speech (POS) tagging for extrinsic estimation.

5.1 Intrinsic Evaluation

Intrinsic evaluators directly measure the syntactic and semantic relationship between word vectors. For this study, we intrinsically evaluated our models on the word analogy task, also known as analogical reasoning, introduced by Mikolov et al. (2013b). In the test set, a triplet of words `A`, `B`, `C` is given, and the goal is to find the fourth word `D`, where `A` is to `B` as `C` to `D`. An example of such an analogy question is Kyiv relates to Ukraine as Madrid to `prediction`, where the correct answer is Spain.

5.1.1 Word Analogy Dataset

We developed the word analogy dataset for Ukrainian[18], which includes 23,970 questions on 12 topics. More precisely, analogy questions are represented by the relations shown in Table 4. To create a dataset, the following methods were used:

- GraphQL requests to WikiData[19] to obtain information about countries, capitals, nationalities, regions, currencies, and relations between them;
- PyMorphy2[20] library with Ukrainian dictionaries[21] installed to generate singular-plural, opposite, comparative, adjective-adverb, superlative, past tense-present, and verb-noun pairs, inflecting the manually crafted list of popular lemmas to generate a unique pair;

[17] https://github.com/lang-uk/fasttext-vectors-uk

[18] https://github.com/lang-uk/vecs/blob/master/test/test_vocabulary.txt

[19] https://www.wikidata.org/wiki/Wikidata:Main_Page

[20] https://github.com/kmike/pymorphy2

[21] https://pypi.org/project/pymorphy2-dicts-uk/

- the family relations were created manually.

Table 4: Word analogy dataset. Questions denote the number of word pairs to compare, and unique pairs denote the number of unique word pairs.

Relations	**# of questions**	**# of unique pairs**
country : capital	4,271	137
country : region	2,038	117
country : nationality	10,359	2,732
country : currency	729	28
family relations	400	21
singular : plural	1,225	36
adjective : adverb	961	32
opposite	625	26
comparative	400	21
superlative	1,089	33
past tense : present	1,089	33
verb : noun	784	29

To evaluate the vector model, a separate library was written[22] to read the dataset, load vectors, and run the intrinsic tests using gensim utilities [23]. Additional logic was added to the evaluation script to make it case-insensitive. The word vector model is tested on each topic separately and on all questions to get the total analogy score. The answer is considered correct if it occurs in the first n predictions (in our case, n = 4).

5.1.2 Results for Intrinsic Evaluation

Table 5 compares the Accuracy scores for the baseline models with default hyperparameters and the optimized models for different train datasets and algorithms.

Overall, it can be seen that the optimized skipgram model increases Accuracy by **7.1%** compared to the baseline model and by **6.4%** compared to the Grave et al. (2018) Ukrainian word embeddings. Models with larger training data (UberText 2.0) generally outperform models built with UberText 1.0. In terms of architecture selection, the skipgram model shows better results than the cbow one for the word analogy task. The same is supported by the regression analysis. Table 8 (Appendix A) provides estimated coefficients and their significance.

The Accuracy significantly depends on the selected model (skipgram is better) and the corpus size. Also, results can be improved by increasing the number of training epochs. The pseudo R-squared for the regression model is 0.957.

5.2 Extrinsic Evaluation

Extrinsic evaluation measures the contribution of the word vectors to a specific downstream task. The comparison result depends on the nature of these tasks and cannot be used as a metric for the quality of embeddings. Nevertheless, comparing performance across tasks may provide insight into the information encoded by embeddings (Schnabel et al., 2015).

We performed experiments on two sequence labeling models: NER and POS tagging. Named entity recognition is a standard NLP task that can identify entities discussed in a text document. Part-of-speech tagging is the process of labeling a word in the text with a particular part of speech based on both its context and definition.

5.2.1 Data

The datasets for extrinsic evaluation were derived from publicly available GitHub repositories:

NER. Ukrainian corpus for named entities recognition[24] comprises 238,927 tokens from 264 text samples. The primary source of the data is the open Brown Corpus of Ukrainian[25], including texts of different genres. The 6,751 entities are annotated by four classes for recognizing locations (LOC) — 4,390 entities, persons (PERS) — 1,616, organizations (ORG) — 780, and miscellaneous (MISC) — 660. We exploit the standard division into dev/test sets at 70%/30% for the training and validation of our models.

POS tagging. Ukrainian Universal Dependencies (UD) corpus[26] was developed by a non-profit organization Institute for Ukrainian[27]. The data follows the CoNLL-U format (Buchholz and Marsi, 2006). UD Ukrainian consists of 122K tokens in 7,000 sentences of different genres — fiction, news, opinion articles, Wikipedia, legal documents, letters, posts, and comments spanning the previous 15 years and the early twentieth century. The current study utilized the proposed data split between

[22]https://github.com/lang-uk/vecs

[23]https://radimrehurek.com/gensim/models/keyedvectors.html#gensim.models.keyedvectors.KeyedVectors.most_similar

[24]https://github.com/lang-uk/ner-uk

[25]https://github.com/brown-uk/corpus

[26]https://github.com/UniversalDependencies/UD_Ukrainian-IU

[27]https://mova.institute/

Table 5: Evaluated the Accuracy of word embeddings trained with default and optimized hyperparameters. Top Accuracy is marked in bold.

	Training Dataset	Model	Subword	Negative Sampling	Epochs	Intrinsic Accuracy
Grave et al. (2018)	Wikipedia+CC	cbow weighted	5-5	10	10	0.579
Our baselines	UberText 2.0	skipgram	3-6	5	5	0.575
	UberText 2.0	cbow	3-6	5	5	0.449
Our optimized	UberText 2.0	skipgram	2-5	15	15	**0.616**
	UberText 1.0	skipgram	2-5	15	10	0.573
	UberText 2.0	cbow	4-6	15	15	0.492
	UberText 1.0	cbow	5-6	15	15	0.473

Table 6: F1 scores for NER task for word embeddings trained with default and optimized hyperparameters. The top F1 score is marked in bold.

	Training Dataset	Model	Subword	Negative Sampling	Epochs	NER SpaCy F1
Grave et al. (2018)	Wikipedia+CC	cbow weighted	5-5	10	10	0.792
Our baselines	UberText 2.0	cbow	3-6	5	5	0.824
	UberText 2.0	skipgram	3-6	5	5	0.816
Our optimized	UberText 1.0	cbow	5-6	15	15	**0.827**
	UberText 2.0	cbow	2-6	10	10	0.826
	UberText 1.0	skipgram	2-5	15	15	0.824
	UberText 2.0	skipgram	2-5	15	15	0.818

train/dev/test at 75%/10%/15% balanced in genre and complexity.

5.2.2 Models

In order to conduct an extrinsic evaluation, we need to load our learned word vectors as input features into a blank model. In the current study, we exploit spaCy[28] and flair[29] (Akbik et al., 2019) libraries as they both support Ukrainian language and usage of custom word embeddings.

spaCy is a free and open-source software library that provides various practical tools for text processing. We used it for training both NER and POS tagging models. In spaCy, it is implemented by the `ner` and `morphologizer pipeline` components. The `morphologizer` aims to predict morphological features and coarse-grained POS tags following the Universal Dependencies grammar; we used only part-of-speech predictions for our evaluation.

flair is a simple framework for state-of-the-art NLP built directly on PyTorch. We trained a BiLSTM-CRF sequence tagger using the flair for the POS task. In contrast to spaCy, in flair, we can use the ability of custom fastText embeddings to get the representations for out-of-vocabulary (OOV) words, loading word vectors in the `.bin` file that contains the model parameters along with the vectors for all n-grams.

All models were trained with the default hyperparameters using early stopping callback and evaluated on the test set. Metrics were F1 for the NER model and the Accuracy for the POS taggers.

5.2.3 Results for Extrinsic Evaluation

We observed minor improvements in the quality of the word embeddings for the NER task. Table 6 shows no significant advances, and the F1 scores for different models are roughly the same.

This is confirmed by the regression analysis. The estimated coefficients do not significantly differ from zero (see Table 9, Appendix A).

Improvements in optimized models for POS tagging tasks also can be considered modest. SpaCy POS accuracy showed almost no increase with model optimization, and the accuracy of flair POS increased by only **2.8%** compared to the cbow base-

[28]`https://spacy.io/`
[29]`https://github.com/flairNLP/flair`

Table 7: Accuracy scores for POS tagging tasks performed with spaCy and flair. Top Accuracy is marked in bold.

	Training Dataset	**Model**	**Subword**	**Negative Sampling**	**Epochs**	**POS SpaCy Accuracy**
Grave et al. (2018)	Wikipedia+CC	cbow weighted	5-5	10	10	0.824
Our baselines	UberText 2.0	cbow	3-6	5	5	0.825
	UberText 2.0	skipgram	3-6	5	5	0.822
Our optimized	UberText 2.0	cbow	2-6	10	10	**0.827**
	UberText 2.0	skipgram	2-5	15	10	0.826
	UberText 1.0	skipgram	2-5	15	10	0.823
	UberText 1.0	cbow	2-6	10	10	0.823
	Training Dataset	**Model**	**Subword**	**Negative Sampling**	**Epochs**	**POS Flair Accuracy**
Grave et al. (2018)	Wikipedia+CC	cbow weighted	5-5	10	10	**0.94**
Our baselines	UberText 2.0	cbow	3-6	5	5	0.893
	UberText 2.0	skipgram	3-6	5	5	0.881
Our optimized	UberText 2.0	cbow	2-6	15	15	0.918
	UberText 2.0	skipgram	2-6	10	15	0.912
	UberText 1.0	cbow	2-6	10	15	0.911
	UberText 1.0	skipgram	2-5	10	15	0.899

line. Accuracy scores are presented in Table 7. Pseudo R-squared is 0.110.

Nevertheless, regression models for the grid of hyperparameters show that the flair POS model Accuracy is better using cbow and cut down choosing a high minimum subword number and the low subword range (Table 10, Appendix A), and the spaCy POS model can be improved by enlarging the training dataset and shows the same tendency for subwords like flair POS does (Table 11, Appendix A). The pseudo R-squared for both models is 0.304 and 0.918, respectively.

We examined that the performance on downstream models is inconsistent across tasks and with intrinsic evaluations, as was previously discovered by Schnabel et al. (2015).

6 Conclusions and Future work

In this paper, we reviewed various aspects of learning word embeddings, including the quality and the quantity of the corpus texts, the choice of the word embeddings algorithm, and its hyperparameters. Those variations were tested on real-world texts and NLP tasks, and the performance of the resulting word embeddings was carefully measured. During the research, more than forty variants of word vectors were trained and evaluated using the clean framework, which consists of one intrinsic and three extrinsic tests.

The evaluation of the resulting word embeddings has indicated that:

- The best hyperparameters based on intrinsic evaluation are:
  - 2-5 subword size, 15 negative samples and epochs for the skipgram model[30];
  - 4-6 subword size, 15 negative samples and epochs for the cbow model[31].
- While the trained vectors have shown visibly better performance on the intrinsic tests, this performance does not correlate much with the extrinsic evaluation results.
- The correlation between the hyperparameters and the results of the extrinsic tests exists but has no significant impact on the corresponding metrics.

Additionally, we found that the reference implementation of the fastText algorithm misses a vital part: the cbow weighted version, which makes it

[30]https://huggingface.co/dchaplinsky/fasttext_uk

[31]https://huggingface.co/dchaplinsky/fasttext_uk_cbow

hard to reproduce the Grave et al. (2018) results on our corpus.

In the paper, we suggest a methodology to build and test word embeddings for low-resource languages, provide the code for training and evaluation, and describe the required data. Such an approach allows conducting similar experiments for other languages and sets a good performance baseline for Ukrainian, allowing us to revisit the results on an even bigger corpus.

Similar methods can be used to train other word vectors, such as classical word2vec (Mikolov et al., 2013a), GloVe (Pennington et al., 2014), or more recent alternatives like LexVec[32] or Floret[33].

Limitations

During the research, we met some of the limitations which might affect the reproducibility of the paper results:

1. The need for a significantly large corpus of good quality may affect reproducibility for other low-resource languages. Researchers might use one of the existing noisy corpora (such as OSCAR[34]) and apply extensive filtering, use Wikipedia, or collect their corpus using web scraping.
2. As fastText word vectors can be trained only on the CPU and require a lot of RAM, access to the modern server time is needed. For this paper, the farm of 7 servers was utilized for training word vectors and running the evaluation.
3. The implementation of the cbow with positional weights had an issue with the memory allocation for the random weights initialization, so we patched the implementation to make it work on a server with 128GB of RAM.
4. The resulting embeddings for the Ukrainian language require about 8 GB of disk storage; therefore, training and evaluation of tens of thousands of them imposes a visible requirement for data storage.

Ethics Statement

We acknowledge that there is a lack of papers in the ACL Anthology that mention the Ukrainian language or are authored by researchers affiliated with Ukrainian universities.

We believe our paper will increase the visibility of the Ukrainian research community and will help build connections with the ACL community.

Furthermore, we acknowledge the potential broader impact of our research on other low-resource, morphologically rich languages. We believe that our methods and findings are generalizable and can be applied to benefit other languages and communities.

[32]https://github.com/alexandres/lexvec
[33]https://github.com/explosion/floret
[34]https://oscar-project.org/

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Felipe Almeida and Geraldo Xexéo. 2019. Word embeddings: A survey. *CoRR*, abs/1901.09069.

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.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

Dmytro Chaplynskyi. 2023. Introducing UberText 2.0: a corpus of modern Ukrainian at scale. In *Proceedings of the Second Ukrainian Natural Language Processing Workshop*, pages 1–10, Dubrovnik, Croatia. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Zellig Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Amit Mandelbaum and Adi Shalev. 2016. Word embeddings and their use in sentence classification tasks. *CoRR*, abs/1610.08229.

Alessio Miaschi and Felice Dell'Orletta. 2020. Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the*

5th Workshop on Representation Learning for NLP, pages 110–119, Online. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Hien T. Nguyen, Phuc H. Duong, and Erik Cambria. 2019. Learning short-text semantic similarity with word embeddings and external knowledge sources. *Knowledge-Based Systems*, 182:104842.

Vít Novotný, Eniafe Festus Ayetiran, Dalibor Bačovský, Dávid Lupták, Michal Štefánik, and Petr Sojka. 2021. One size does not fit all: Finding the optimal subword sizes for FastText models across languages. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1068–1074, Held Online. INCOMA Ltd.

Vít Novotný, Michal Štefánik, Eniafe Festus Ayetiran, Petr Sojka, and Radim Řehůřek. 2022. When fasttext pays attention: Efficient estimation of word representations using constrained positional weighting. *JUCS - Journal of Universal Computer Science*, 28(2):181–201.

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.

R Core Team. 2013. R: A language and environment for statistical computing.

Terry Ruas, William Grosky, and Akiko Aizawa. 2019. Multi-sense embeddings through a word sense disambiguation process. *Expert Systems with Applications*, 136:288–303.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, Lisbon, Portugal. Association for Computational Linguistics.

Yikang Shen, Wenge Rong, Nan Jiang, Baolin Peng, Jie Tang, and Zhang Xiong. 2017. Word embedding based correlation model for question/answer matching. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3511–3517. AAAI Press.

Maria Shvedova, Ruprecht von Waldenfels, Sergiy Yarygin, Andriy Rysin, Vasyl Starko, and Tymofij Nikolajenko et al. 2017-2022. GRAC: General regionally annotated corpus of Ukrainian.

François Torregrossa, Robin Allesiardo, Vincent Claveau, Nihel Kooli, and Guillaume Gravier. 2021. A survey on training and evaluation of word embeddings. *International Journal of Data Science and Analytics*, 11(2):85–103.

Achim Zeileis, Francisco Cribari-Neto, Bettina Gruen, Ioannis Kosmidis, Alexandre B Simas, Andrea V Rocha, and Maintainer Achim Zeileis. 2016. Package 'betareg'. *R package*, 3(2).

A Regression Analysis

Table 8: Beta regression coefficients for the model predicting the mean Accuracy for word analogy task.

component	term	estimate	std.error	statistic	p.value
mean	**(Intercept)**	**-0.2554787**	**0.0558844**	**-4.5715534**	**0.0000048**
mean	Cbow weighted	-0.0217421	0.0333612	-0.6517183	0.5145829
mean	**Skipgram**	**0.4838693**	**0.0164096**	**29.4869395**	**0.0000000**
mean	**Epochs**	**0.0053332**	**0.0023039**	**2.3148724**	**0.0206199**
mean	Subword 2-6	-0.0081295	0.0230823	-0.3521978	0.7246899
mean	Subword 3-6	-0.0398978	0.0507995	-0.7853973	0.4322207
mean	Subword 4-6	0.0035476	0.0237927	0.1491042	0.8814714
mean	Subword 5-5	-0.0106454	0.0398361	-0.2672294	0.7892926
mean	Subword 5-6	-0.0294700	0.0229863	-1.2820653	0.1998197
mean	**UberText 2.0**	**0.0640098**	**0.0207206**	**3.0891944**	**0.0020070**
mean	Negative sampling	0.0022902	0.0033095	0.6919969	0.4889393
precision	**(phi)**	**1528.2850428**	**305.5581464**	**5.0016177**	**0.0000006**

Table 9: Beta regression coefficients for the model predicting the mean F1 score for NER task.

component	term	estimate	std.error	statistic	p.value
mean	**(Intercept)**	**1.6216440**	**0.0666023**	**24.3481651**	**0.0000000**
mean	Cbow weighted	-0.0075895	0.0397626	-0.1908713	0.8486264
mean	Skipgram	-0.0178466	0.0195245	-0.9140592	0.3606857
mean	Epochs	0.0029815	0.0027425	1.0871598	0.2769662
mean	Subword 2-6	0.0371129	0.0275649	1.3463850	0.1781784
mean	Subword 3-6	-0.0361289	0.0602330	-0.5998193	0.5486267
mean	Subword 4-6	0.0002828	0.0282246	0.0100184	0.9920066
mean	Subword 5-5	-0.0324346	0.0474000	-0.6842734	0.4938025
mean	Subword 5-6	0.0102150	0.0273104	0.3740327	0.7083800
mean	UberText 2.0	0.0108599	0.0246912	0.4398274	0.6600621
mean	Negative sampling	-0.0049084	0.0039411	-1.2454404	0.2129699
precision	(phi)	1902.8778974	380.5287293	5.0006156	0.0000006

Table 10: Beta regression coefficients for the model predicting the mean Accuracy score for SpaCy POS tagging.

component	term	estimate	std.error	statistic	p.value
mean	**(Intercept)**	**1.5293970**	**0.0138662**	**110.2966659**	**0.0000000**
mean	Cbow weighted	-0.0051336	0.0083367	-0.6157876	0.5380347
mean	Skipgram	-0.0049324	0.0040683	-1.2123932	0.2253619
mean	Epochs	0.0003477	0.0005735	0.6062589	0.5443429
mean	**Subword 2-6**	**-0.0118624**	**0.0057226**	**-2.0729047**	**0.0381811**
mean	Subword 3-6	-0.0080235	0.0126195	-0.6357997	0.5249070
mean	Subword 4-6	-0.0099378	0.0058999	-1.6843804	0.0921082
mean	Subword 5-5	-0.0066263	0.0099457	-0.6662503	0.5052511
mean	Subword 5-6	-0.0081829	0.0057037	-1.4346837	0.1513773
mean	**UberText 2.0**	**0.0209113**	**0.0051287**	**4.0773230**	**0.0000456**
mean	Negative sampling	-0.0002691	0.0008203	-0.3280957	0.7428393
precision	**(phi)**	**41970.9373563**	**8394.1353521**	**5.0000310**	**0.0000006**

Table 11: Beta regression coefficients for the model predicting the mean Accuracy score for Flair POS tagging.

component	term	estimate	std.error	statistic	p.value
mean	**(Intercept)**	**2.3223858**	**0.0854556**	**27.1765257**	**0.0000000**
mean	**Cbow weighted**	**-0.2127466**	**0.0544897**	**-3.9043446**	**0.0000945**
mean	**Skipgram**	**-0.2372245**	**0.0246544**	**-9.6219977**	**0.0000000**
mean	Epochs	0.0001504	0.0035905	0.0418964	0.9665813
mean	Subword 2-6	0.0017537	0.0383267	0.0457572	0.9635038
mean	**Subword 3-6**	**-0.1959644**	**0.0775663**	**-2.5264110**	**0.0115235**
mean	**Subword 4-6**	**-0.5094930**	**0.0359537**	**-14.1708221**	**0.0000000**
mean	Subword 5-5	0.0316722	0.0640241	0.4946926	0.6208171
mean	**Subword 5-6**	**-0.5957699**	**0.0344968**	**-17.2702724**	**0.0000000**
mean	UberText 2.0	0.0246145	0.0320123	0.7689059	0.4419491
mean	Negative sampling	0.0063355	0.0049368	1.2833240	0.1993786
precision	**(phi)**	**1631.2103337**	**326.2489973**	**4.9998938**	**0.0000006**