UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Statistical and neural language models for the Ukrainian Language

Author:
Anastasiia KHABURSKA

Supervisor:
Igor TYTYK

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Anastasiia KHABURSKA, declare that this thesis titled, "Statistical and neural language models for the Ukrainian Language" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Statistical and neural language models for the Ukrainian Language

by Anastasiia KHABURSKA

Abstract

Language Modeling is one of the most important subfields of modern Natural Language Processing (NLP). The objective of language modeling is to learn a probability distribution over sequences of linguistic units pertaining to a language. As it produces a probability of the language unit that will follow, the language model can be viewed as a form of grammar for the language, and it plays a key role in traditional NLP tasks, such as speech recognition, machine translation, sentiment analysis, text summarization, grammatical error correction, natural language generation. Much work has been done for the English language in terms of developing both training and evaluation approaches. However, there has not been as much progress for the Ukrainian language. In this work, we are going to explore, extend, evaluate and compare different language models for the Ukrainian language. The main objective is to provide a balanced evaluation dataset and train a number of baseline models.

Acknowledgements

First of all, I would like to thank my supervisor Igor Tytyk who directed me throughout the research, for his enthusiasm for the project, for his support, valuable ideas and pieces of advice, encouragement and patience.

Special thanks to Artem Chernodub (Ukrainian Catholic University, Grammarly) and Grammarly itself for the motivation, computational resources and consultation.

This work would not be possible without Ukrainian Brown Corpus. Thanks to Brown-Uk enthusiastic group [1] for the meticulous work on constructing the corpus. I sincerely wish them success in the future development of their work.

Many thanks to Dmitry Chaplinsky, who kindly agreed to share with me the data from Uber Text corpora. [2]

Last, but not least, I am grateful to Ukrainian Catholic University and Oleksii Molchanovskyi personally for the first master program in data science in Ukraine and to Ciklum for covering my tuition fees and made my study possible here.

[1]Brown-Uk: `https://r2u.org.ua/corpus`

[2]Uber Text: `http://lang.org.ua/en/corpora/`

Contents

List of Figures

List of Tables

List of Abbreviations

BPC	**B**its **P**er **C**haracter
BPE	**B**yte **P**air **E**ncoding
LM	**L**anguage **M**odel
NLP	**N**atural **L**anguage **P**rocessing
OOV	**O**ut **O**f **V**ocabulary
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong**S**hort**T**erm**M**emory
GRU	**G**ated **R**ecurrent **U**nit

List of Symbols

S	sequence of linguistic units
P	probability
PP	perplexity
V	vocabulary of tokens
G	vocabulary of character n-grams
w_i	ith word in a sequence
u_i	ith linguistic unit in a sequence
M	million
K	thousand
lr	learning rate

Dedicated to my loving family who always supports me

Chapter 1

Introduction

1.1 Importance of Language Modeling

The objective of Language Modeling is to learn a probability distribution over sequences of linguistic units pertaining to a language. As **linguistic units**, we can consider any natural units into which linguistic messages can be divided, for example, characters, words or phrases. The linguistic units seen by the model compose model's dictionary U.

$$P(S) = P(u_1, u_2, \ldots, u_n) \tag{1.1}$$

where S- sequences of linguistic units and u_i - i-th unit.

Typically, this is achieved by providing conditional probabilities $p(u|c)$, where c is the context of linguistic unit u. For example, the probability of a particular unit in the sequence:

$$P(u_i | u_{i-k_1}, u_{i-k_1+1}, \ldots, u_{i+k_2-1}, u_{i+k_2}) \tag{1.2}$$

Most fixed-vocabulary language models employ a symbol $< unk >$ that represents all units that are not presented in vocabulary U. These units are termed out-of-vocabulary (OOV).

In this Master Thesis, we recall to LMs depending on the learned linguistic unit, as word-level, subword-level and character-level LMs respectively.

As it produces a probability of the following language unit, the language model (LM) can be viewed as a grammar of the language, and it plays a key role in traditional NLP tasks, such as automatic speech recognition (Mikolov et al., 2010, Arisoy et al., 2012), machine translation (Schwenk, Rousseau, and Attik, 2012, Vaswani et al., 2013), sentiment analysis (Hu et al., 2007), text summarization (Rush, Chopra, and Weston, 2015, Filippova et al., 2015), grammatical error correction (Bryant and Briscoe, 2018), natural language generation (Edunov, Baevski, and Auli, 2019).

1.2 Motivation

Language Modeling is one of the central tasks to Natural Language Processing and Natural Language Understanding. Thus, in order to elaborate upon an NLP task for the language, this language needs to have a well-designed high-quality language model(LM).

To train and evaluate language models, it is required to have a well-composed corpus (corpora). In linguistics and NLP, corpus refers to a collection of texts. In case of language modeling, it is very important to evaluate and benchmark the models on the data with balanced genres and topics.

For the English Language, language modelling coherently evolves in time and has already gone through multiple stages of elaboration. Much work has been done in terms of collecting (building) corpora as well as developing both training and evaluation approaches. Firstly, count-based approaches (based on statistics of N-grams), such as Kneser-Ney smoothed N-gram models Kneser and Ney, 1995 were a fairly strong baseline. In recent years, much progress has been made by neural methods Bengio et al., 2003; Mikolov et al., 2010, character-aware Neural Language Models Kim et al., 2015, based on LSTMs Jozefowicz et al., 2016, gated convolutional networks N. Dauphin et al., 2017 and self-attentional networks Al-Rfou et al., 2018.

At the same time, there hasn't been as much progress for the Ukrainian language. Overall, building a baseline language model and a gold standard corpus for the Ukrainian language is a crucial step in the evolution of Ukrainian NLP. This is the main motivation for our work.

1.3 Goals of the master thesis

In this master thesis, we want to explore, extend, develop, evaluate and compare a set of language models for the Ukrainian language. The main objective is to offer an evaluation corpus and set a number of baselines.

1. To provide an overview of existing approaches to language modeling for other languages.
2. To compose a data corpus sufficiently large for training neural language models for the Ukrainian language and conduct an appropriate preprocessing of the gathered data.
3. To experiment with different linguistic units that can potentially better model the sequential information in Ukrainian data.
4. To explore, reproduce and extend the set of main approaches that where considered effective for other languages.
5. To evaluate and compare language models trained for the Ukrainian language. To offer an evaluation corpus and set a number of baselines.

1.4 Structure of the thesis

In Chapter 2, we provide an analytical summary of what has been done by the scientific community for the English language in terms of language modeling. The chapter comprises a review of related publications and the background theory, which forms the basis of the work presented later in the thesis. In Chapter 3, we describe the methodology we used to address language modeling for the Ukrainian language. Here we describe the process of constructing training and evaluation corpora, the experimental setup, the implementation details and the obtained results. Chapter 4 summarizes our contributions and set the directions for further research.

Chapter 2

Literature Review

2.1 English Language Modeling

English is the most widely spoken language in the world, and, consequently, most of the NLP research has been mainly evolving around it. Countless papers were written, vast linguistic resources were collected, numerous benchmarks were set and approaches tested. This way, it can be claimed that NLP for the English Language is the most mature among the natural languages.

Hence, it is essential to start with a deep dive into the research that has been done for English, including the relevant text corpora and models. So, the knowledge and experience can be reapplied for the Ukrainian language.

2.1.1 Corpora

In this section, we will list and compare the most used general-specific English corpora, which were used for evaluation of the sentence based LMs, similar to the ones we are going to use.

A common evaluation corpora is Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) – large annotated collection of texts. The material annotated includes such wide-ranging genres as IBM computer manuals, nursing notes, the Wall Street Journal articles, and transcribed telephone conversations, etc. Its original version consists of over 4.5 million words of American English. The PTB portion of the Wall Street Journal corpus, preprocessed by Mikolov et al., 2011a has been used by numerous researchers. The dataset consists of three partitions: training (929k tokens), validation (73k tokens), and the test partition (82k tokens). As part of the preprocessing, the corpus was stripped of any punctuation, words were lower-cased, numbers were replaced with N, newlines were replaced with <eos>. The vocabulary is limited to 10k most frequent words and the rest of the tokens being discarded and replaced with a special token, <unk> .

The One-Billion Word benchmark introduced by Chelba et al., 2013 is a larger benchmark corpus extracted from a news-commentary site. The dataset consists of 829,250,940 tokens with a vocabulary of 793,471 words. All words with a frequency below 3 were replaced by the <unk> token. The sentences in this model are shuffled, hence only sentence-level models can be trained and evaluated on the corpus.

The WikiText dataset was proposed as a more realistic benchmark by Merity et al., 2016. Overall, the corpus consists of around 103 million words derived from knowledgeable and featured Wikipedia articles with punctuation, original casing, and numbers. That allows for capturing and utilization of longer-term dependencies within longer text spans. The dataset is available in two different sizes: WikiText-2 and WikiText-103.

A comparison of the four described datasets is provided in Table 3.2. The values for the PTB, WikiText-2 and WikiText-103 are taken from Merity et al., 2016: "The out-of-vocabulary (OOV) rate notes what percentage of tokens have been replaced by an $< unk >$ token. The token count includes newlines which add to the structure of the WikiText datasets". The values for the One Billion Word are computed from the Description of the Benchmark Data in Chelba et al., 2013. $829,250,940$ is the size of the whole dataset, counting sentence boundary markers $< S >, < /S >$. Then, sentence order was randomized, and the data was split into 100 disjoint partitions. One such partition was split again into 50 disjoint partitions, one of them is used as a test set and amounts to $159,658$ words without counting the sentence beginning marker $< S >$. Then, training part contains $829,250,940 - 1\%$ and validation data approximately contains $159,658 * 49$ word-level tokens.

	Penn Treebank	1 Billion Word	WikiText-2	WikiText-103	
Tokens	929,590	829,250,940 - 1%	2,088,628	103,227,021	Train
	73,761	$\approx 7,823,242$	217,646	217,646	Valid
	82,431	159,658	245,569	245,569	Test
Articles	-	-	600	28,475	Train
	-	-	60	60	Valid
	-	-	60	60	Test
Vocab size	10,000	793,471	33,278	267,735	
OOV rate	4.8%	0.28%	2.6%	0.4%	

TABLE 2.1: Statistics and comparison of the Penn Treebank, 1 Billion Word, WikiText-2, and WikiText-103.

Zipf's law was popularized as a rather good approximation to the word probabilities in many different languages (Zipf, 1935, Zipf, 1949). According to it, the frequency of any word is inversely proportional to its rank order in the frequency list sorted from the most common to the less common words. Thus, the probability $p(w_i)$ that in a random sample we find the i-th most common word w_i can be calculated with $Z \approx 1$ and K the normalization constant, i.e., $K^{-1} = \sum_{i \leq n} \frac{1}{i^Z}$

$$p(w_i) = \frac{K}{i^Z} \tag{2.1}$$

Word frequencies for any language corpora could be approximated with a Zipfian distribution. This statement is mostly used by plotting the data on a log(rank order) - log(frequency) graph. Zipfian plot over the training partition in Penn Treebank and WikiText-2 datasets are shown on the 2.1. For the PTB corpus, the curve abruptly stops when hitting the 10k rank, because it is the size of its vocabulary.

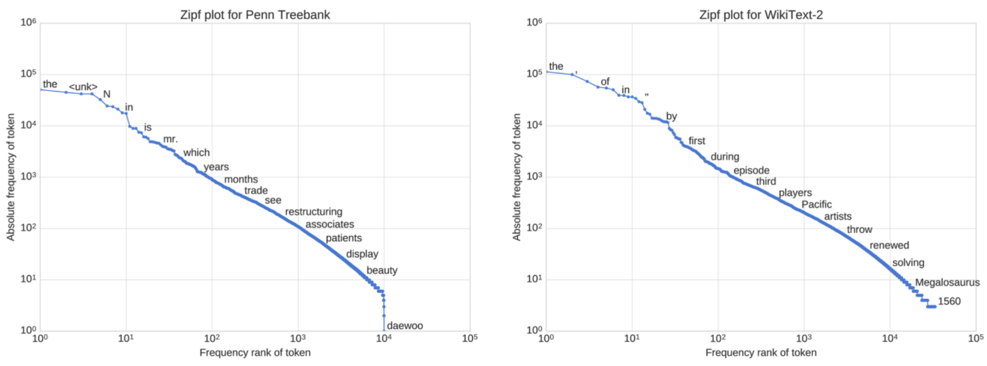


FIGURE 2.1: Zipfian plot over the training partitions from Penn Treebank and WikiText-2 datasets. Source: (Merity et al., 2016)

2.1.2 Models

N-gram models

N-gram models is a widely used type of language models. They are very straightforward to construct except for the smoothing techniques developed to estimate probabilities of out-of-vocabulary words or, in general, when there is insufficient data.

The generic equation for computing probability of a sentence, using n-gram model is:

$$p(s) = \prod_{i=1}^{l+1} p(u_i | u_{i-n+1}^{i-1}) \tag{2.2}$$

where n is the order of the model and u_i^j denotes the units $u_i \cdots u_j$. Also, additional units $u_0 =< BOS >$ and $u_{l+1} =< EOS >$ are initiated. To estimate the probabilities:

$$p(u_i | u_{i-n+1}^{i-1}) = \frac{c(u_{i-n+1}^{i})}{\sum_{u_i} c(u_{i-n+1}^{i})} \tag{2.3}$$

where $c(u_{i-n+1}^{i})$ denotes the number of times the n-gram $u_i...u_{i-n+1}$ occurs in the given corpus. The units u_{i-n+1}^{i-1} preceding the current unit u_i are called the history. The sum $\sum_{u_i} c(u_{i-n+1}^{i})$ is equal to the count of the history $c(u_{i-n+1}^{i-1})$.

Smoothing is a technique used to adapt the maximum likelihood estimate of probabilities and to make distribution more uniform, by adjusting low probabilities such as zero probabilities upward and high probabilities downward. At the same time, smoothing methods are designed to prevent zero probabilities.

While sparse data is one of the central issues in language modeling with n-grams, a large variety of techniques has been proposed for "smoothing" n-gram models. Chen and Goodman (1998) carried out an extensive empirical comparison of the most widely used smoothing techniques, including those described by Jelinek and Mercer (1980), Katz (1987), Bell, Witten, and Cleary (1990), Ney, Essen, and Kneser (1994) and Kneser and Ney (1995). They introduced methodologies for analyzing smoothing algorithm performance in detail, and, using these techniques, they motivate a novel variation of Kneser-Ney smoothing that consistently outperforms all other algorithms evaluated.

This backoff-smoothed model estimates the probability based on the observed entry with longest matching where the probability $p(u_i|u_f^{i-1})$ and back-off penalties $b(u_n^{i-1})$ are provided by the already-estimated model.

$$p(u_i|u_1^{i-1}) = p(u_i|u_f^{i-1}) \prod_{n=1}^{f-1} b(u_n^{i-1}) \tag{2.4}$$

Open-source KenLM library proposed by Heafield (2011) efficiently uses two data structures (*PROBING* and *TRIE*) to query n-gram language model with modified Kneser-Ney smoothing, reducing both time and memory costs.

Neural Language models

First feed-forward neural language model (NLM) was proposed by Bengio et al., 2003. Distributed word representations ("feature vectors") were processed by a single hidden layer with hyperbolic tangent activation and then a softmax output layer. This simple architecture for learning of both word feature vectors and the parameters of joint probability function of word sequences simultaneously was a breakthrough solution for dealing with a fundamental problem of N-gram language modelling, "the curse of dimensionality". Also, leveraging the fact that each word was associated with a point in a vector space, NLM was able to take into account the "similarity" (distance) between words.

Their experiments have shown perplexity reduction between 20% and 35% compared to, at that time state-of-the-art, a smoothed trigram model.

Bengio et al., also introduced so-called "cycling architecture", that was a glance into the future of the Recurrent Neural Networks described further.

Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of Artificial Neural Network used in NLP. RNN is designed to be able to exhibit sequential data inputs. By introducing architecture with hidden state $h^{(t)}$ and loops, this type of neural networks allows the previously processed information to persist. A general visualization of such a model is presented in Figure 2.2

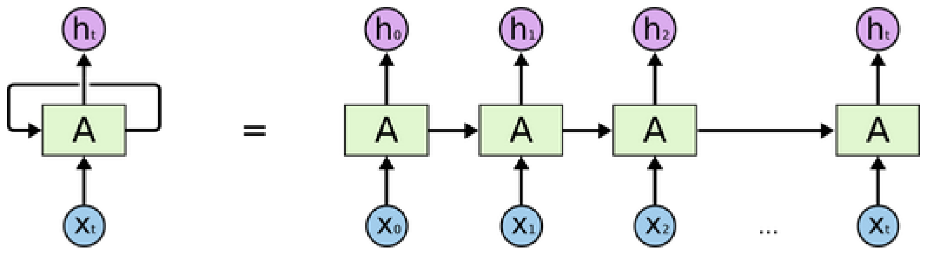


FIGURE 2.2: The repeating module in a standard RNN.
Source: Olah, 2015

Assuming that the RNN is used to predict words or characters, and the output is discrete, the mathematical formulation of forward propagation at each time step of a Vanilla RNN is as follows (Goodfellow, Bengio, and Courville, 2016):

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{2.5}$$

$$h^{(t)} = tanh(a^{(t)}) \tag{2.6}$$

$$o^{(t)} = c + Vh^{(t)} \tag{2.7}$$

$$\hat{y}^{(t)} = softmax(o^{(t)}) \tag{2.8}$$

Where $x^{(t)}$ is an input vector, $\hat{y}^{(t)}$ is an output vector representing probability distribution over the vocabulary at time step t. The bias vectors b and c and the weight matrices U, V and W are the model parameters. Hidden state $h^{(0)}$ is initialized at first iteration. Matrices U, V and W respectively, compute input-to-hidden, hidden-to-output and hidden-to-hidden connections.

The main advantage over N-gram LMs, where only a finite context size window would be considered, is the RNN's capability of conditioning the model on all previous words in the sentence.

Even though the preliminary results on small corpora were promising, Mikolov et al., 2010 observed that it is impossible for a vanilla RNN trained with gradient descent to capture long context information.

When training very deep recurrent neural networks two common problems that arise are the *vanishing* and *exploding gradients*. Due to the process of backpropagation, which in essence is the application of the chain rule, the gradient from the latest layers is multiplied by matrices in order to be propagated to the beginning of the network. If the largest eigenvalues of these matrices are less than one, then the value of the gradient will go to zero if the network has many layers, which is called the vanishing gradients problem. On the other hand, if the largest eigenvalues are greater than one, then the gradients will go to infinity — the exploding gradients problem.

Long Short Term Memory networks

Long Short Term Memory (LSTM) network is a gated RNN, designed by Hochreiter and Schmidhuber, 1997 to mitigate the problems of vanishing and exploding gradients for long-term dependencies during the back-propagation phase. The *memory cell*, which replaces the *hidden unit* in vanilla RNNs, integrates four main elements: *input gate*, *forget gate*, *output gate* and *selfrecurrent connection*.

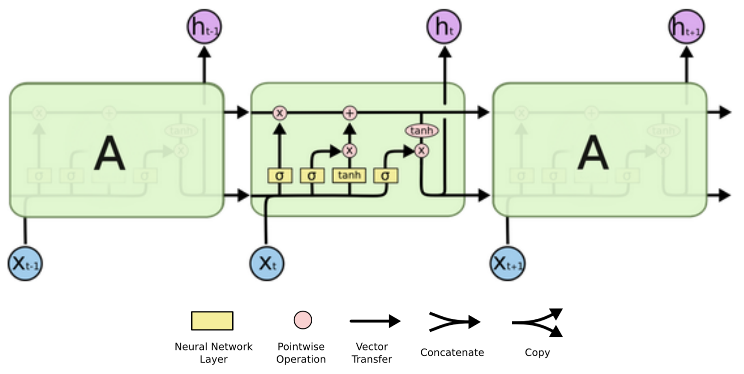


FIGURE 2.3: Four interacting layers in LSTM repeating module (memory cell).
Source: Olah, 2015

The following equations define a forward propagation for each time step t:

$$i^{(t)} = \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)} + b^{(i)}) \quad \text{Input gate} \tag{2.9}$$

$$f^{(t)} = \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)} + b^{(f)}) \quad \text{Forget gate} \tag{2.10}$$

$$o^{(t)} = \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)} + b^{(o)}) \quad \text{Output gate} \tag{2.11}$$

$$\hat{c}^{(t)} = \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)} + b^{(c)}) \quad \text{New memory} \tag{2.12}$$

$$c^{(t)} = \tanh(f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \hat{c}^{(t)}) \quad \text{Final memory cell} \tag{2.13}$$

$$h^{(t)} = o^{(t)} \circ \tanh(c^{(t)}) \quad \text{Updated hidden state} \tag{2.14}$$

Where $x^{(t)}$ is an input vector, $\hat{h}^{(t)}$ is an updated hidden state at time step t. The bias vectors $b^{(i)}$, $b^{(f)}$, $b^{(o)}$ and $b^{(c)}$ and the weight matrices $U^{(i)}$, $U^{(f)}$, $U^{(o)}$, $U^{(c)}$, $W^{(i)}$, $W^{(f)}$, $W^{(o)}$ and $W^{(c)}$ are the model parameters. σ is the non-linearity function.

LSTMs are at the core of language modeling evolution. Even though they solve the vanishing gradient problem, the exploding gradients problem still prevails. In spite of the introduction of such techniques as gradient clipping Pascanu, Mikolov, and Bengio, 2013 and dropout-regularization Zaremba, Sutskever, and Vinyals, 2014 might not be sufficient to adequately address this issue. Many variants of basic LSTM have been proposed in terms of language modeling. In particular, experiments with gates (Melis, Kočiský, and Blunsom, 2020) and incorporation of multiple cells – Multi-cell LSTM (Cherian, Badola, and Padmanabhan, 2018).

Currently, the best perplexity 2.20 score on PTB and WikiText-2 benchmarks where achieved by the Mogrifier LSTM modification (Melis, Kočiský, and Blunsom, 2020). So, even after the introduction of Transformers, venerable LSTMs can still be considered state-of-the-art.

Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a simpler version of LSTM introduced by Cho et al., 2014. GRU has only two gates (reset and update) and does not have the memory cell. The equations 2.15 to 2.18 describe how a GRU cell is updated at every time-step t.

$$z^{(t)} = \sigma(W^{(z)}x^{(t)} + U^{(z)}h^{(t-1)} + b^{(z)}) \quad \text{Update gate} \tag{2.15}$$

$$r^{(t)} = \sigma(W^{(r)}x^{(t)} + U^{(r)}h^{(t-1)} + b^{(r)}) \quad \text{Reset gate} \tag{2.16}$$

$$\hat{h}^{(t)} = \tanh(r^{(t)} \circ U^{(h)}h^{(t-1)} + W^{(h)}x^{(t)} + b^{(h)}) \text{New memory} \tag{2.17}$$

$$h^{(t)} = (1 - z^{(t)}) \circ \hat{h}^{(t)} + z^{(t)} \circ h^{(t-1)} \quad \text{Hidden cell} \tag{2.18}$$

The bias vectors $b^{(z)}$, $b^{(r)}$, $b^{(h)}$ and the weight matrices $U^{(z)}$, $U^{(r)}$, $U^{(h)}$, $W^{(z)}$, $W^{(r)}$ and $W^{(h)}$ are the model parameters. σ is the non-linearity function.

GRU is less famous then LSTM, but having fewer parameters shows comparatively good results on speech recognition LMs Irie et al., 2016 and sequential data generation Chung et al., 2014.

2.1.3 Evaluation

Perplexity

Traditionally, LM performance is measured by such intrinsic metrics as perplexity, cross-entropy, and bits-per-character (BPC). All three of them are highly correlated,

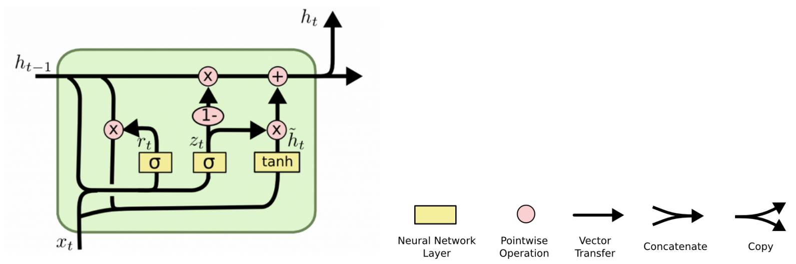


FIGURE 2.4: GRU repeating module.
Source: Olah, 2015

but it is not always reasonable to compare them between LMs with different vocabulary sizes, language units and OOV words handling. The theory comes from estimation of the "entropy" of an English language, proposed by Shannon, 1948 Shannon, 1951, measuring the average amount of information conveyed in a message.

A language model aims to learn the unknown distribution of language units in a language from the sample text. Thus, minimizing the cross-entropy $H(P,Q)$ of learned distribution Q with respect to the empirical distribution P of the language.

$$H(P,Q) = -\sum_{i=1}^{n} P(u_i) log Q(u_i) \tag{2.19}$$

In terms of LM objective defined in 1.1, a better model for a text sequence is the one that assigns a higher probability to the word that actually occurs. The metric perplexity is often used to compare word-level LMs. It is computed as the inverse probability of the text, normalized by the number of words:

$$PP(S) = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}} \tag{2.20}$$

Where N stands for the number of words.

After applying a chain rule, the formula can be written as:

$$PP(S) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 w_2 ... w_{i-1})}} \tag{2.21}$$

Usually, in order to minimize the computational complexity, a logarithm of the perplexity is computed by normalizing the sum over the negative logarithms of predicted probabilities:

$$log_2(PP(S)) = \frac{1}{N}\sum_{i=1}^{N} (-log_2(P(w_i|w_1 w_2 \dots w_{i-1})) \tag{2.22}$$

$$PP(S) = 2^{-\frac{1}{N}\sum_{i=1}^{N} log_2(P(w_i|w_1 w_2 ... w_{i-1}))} \tag{2.23}$$

By minimizing perplexity, the probability is maximized. Therefore, a better LM is the one that has a lower perplexity value.

Character-level LMs are usually compared with Bits Per Character (bpc) metric.

$$BPC(S) = -\frac{1}{T}\sum_{i=1}^{T} log_2(P(c_i|c_1 c_2 ... c_{i-1})) \tag{2.24}$$

Where T is the number of characters.

Al-Rfou et al., 2018 proposed to compare character-level models with word-level using the approximation:

$$PP(S) = 2^{BPC\frac{T}{N}} \tag{2.25}$$

Comprehensive evaluations

Commercial attractiveness of English has given rise to different NLP tasks. Prominent minds from both industry and science cooperated together to solve those issues with reasonable efforts, time and computational resources. By now, most LMs are designed for a specific task and struggle with out-of-domain data.

However, due to expensiveness of the computational resources, it is highly desirable to train the LMs that are able to perform reasonably well independently of the context in which they are used, in order to easily be used as pretrained models for some extrinsic NLP tasks. In other words, LMs should have enough flexibility to represent complex distributions and to be capable of dealing with various peculiarities of human languages. In an attempt to achieve this goal, Wang et al., 2018 introduced the General Language Understanding Evaluation (GLUE) benchmark, a collection of tools for evaluating the performance of models across a diverse set of existing tasks.

2.1.4 Tokenization levels

LMs typically operate with a fixed vocabulary of language units. During the training process, the model learns the distribution of the units, their similarities and relations. Subject to the tokenization level of the linguistic message, the model would also encounter challenges like size of the vocabulary, treating unknown words (open or closed vocabulary), number of parameters in order to be able to model relations over long distances, the length of the input sequence and the number of required computations.

Word level

Most statistical LMs use words as their atomic units. This architecture approach is the most intuitive, as the model is supposed to learn word similarities and dependencies. Word-level models are closed-vocabulary and usually, introduce an $< unk >$ token to represent OOV words. The number of parameters highly depends on the size of the vocabulary.

Character level

Character-level models (Mikolov et al., 2011a, Jozefowicz et al., 2016, Al-Rfou et al., 2018, Krause et al., 2016, Kim et al., 2015) operate over vocabularies comprised of distinct characters seen in the training corpus. This type of LM must learn a large vocabulary of words "from scratch". Also, character sequences and dependencies are longer, thus require significantly more steps of computation. The main advantage is the ability of the model to generate new, unseen before words. Thus, a character-level NLM, which provides comparatively good results, should be deep and with a huge amount of trainable parameters.

Subword level

Subword tokenization is widely used in machine translation models (Sennrich, Haddow, and Birch, 2016, Kudo, 2018, Mikolov et al., 2011b). In general, subword-level LMs perform better for inflectional and agglutinative languages, because they are able to separate the meaningful morphological units and learn to construct new words that were not seen at training time. This models advantage both configurable size of vocabulary and being open-vocabulary simultaneously. Subword-level LMs typically follow a common template: 1) A segmentation algorithm is applied to train corpora in order to form a subword vocabulary. 2) Train and test data is tokenized with respect to formed vocabulary. 3) The output of the model is detokenized.

In this work, we consider the most used word separation algorithm — Byte Pair Encoding (BPE).

Byte Pair Encoding

BPE was initially introduced as a data compression technique (Gage, 1994) and then adapted to languages by Sennrich, Haddow, and Birch, 2016. The algorithm incrementally finds a set of symbols such that the size of the dictionary for encoding the text is minimized.

The algorithm (Sennrich, Haddow, and Birch, 2016) (the hyperparameter N - number of merge operations):

- Step 0: The vocabulary is initialized with the character vocabulary. Each word is represented as a sequence of characters, plus a special end-of-word symbol.
- Step 1: All symbol pairs frequencies occurrences are counted.
- Step 2: The most frequent pair ('A', 'B') is merged and replaced with a new symbol 'AB'. A new token is added to the initial vocabulary.

Repeat step 1 and step 2 N times. The final vocabulary size equals to the size of the vocabulary initialized on Step 0, plus the number of merges (Step 2).

Another frequently used dictionary substitution encoder is the *unigram language model* Kudo, 2018. Its data compression principle is to minimize the total code length for the text. The unigram language model encodes sentence into not unique subword sequence, thus imposing spurious ambiguity. Subword regularization technique (Kudo, 2018) trains the model on different data inputs randomly sampled from the original input sequences and can be regarded as a variant of ensemble training.

Bags of character n-grams

This word representation, introduced by Bojanowski et al., 2017, takes into account word's internal structure.The dictionary of n-grams of size G is generated and each word w is represented as a bag of character n-gram $w_i \subset \{1, \dots G\}$ appearing in w. Word feature vector is computed as a sum of the feature vectors of its n-gram components. As a result, embedding could be generated even for unknown words. Bojanowski et al., 2017 observed that using word representations trained with subword information outperforms the plain word-level model and " ...this improvement is most significant for morphologically rich Slavic languages such as Czech (8% reduction of perplexity) and Russian (13% reduction)."

2.2 What has been done for the Ukrainian Language

Sazhok and Robeiko, 2013 concerns the development of real-time speech recognition system for the Ukrainian language. In terms of language modeling part of the speech-to-text system structure, they compare word-based (Hsu and Glass, 2008) and class-based (Martin, Liermann, and Ney, 1998) statistical language models. According to their experimental set-up, class-based LMs occupy less space and potentially outperform a 3-gram word-based model. Also, implemented class-based LM automatically obtains classes for Ukrainian, that in general correspond to syntactic, semantic and phonetic features.

Maučec, Rotovnik, and Zemljak Jontes, 2003 also tackled the task of building an automatic speech recognition system but for the languages of the same group as Ukrainian – Slavic languages. They used a data-driven language-independent tokenization technique to separate stems and endings as new basic units. They also suggest, that the proposed technique can be applied to other highly inflected Slavic languages. Their approach resulted in the reduction of both OOV rate (by 64%) and vocabulary size, compared to word-level N-gram model, and the recognition accuracy was improved by 4.34%.

Lang-uk [1] open community computed Ukrainian word embeddings (Word2Vec and GloVe) based on the collected corpora of newswire, articles, fiction and juridical texts. At the moment, their GloVe-like embeddings show the best scores on the test sets for quality evaluation of vector representations developed by Tetiana Kodliuk [2].

[1]Lang-uk open community: `https://lang.org.ua/en/models/#anchor4`
[2]Test set for valuation of embeddings: `https://github.com/lang-uk/vecs`

Chapter 3

Methodology

Having all the mentioned above work done for the English language, it's predictable that the approaches which show good results there would perform comparatively good for other languages. Indeed, as pointed out by Cotterell et al. (2018) "Most methods are portable in the following sense: Given appropriately annotated data, they should, in principle, be trainable on any language. However, despite this crude cross-linguistic compatibility, it is unlikely that all languages are equally easy, or that our methods are equally good at all languages."

There are three important differences between Ukrainian and English considering statistical language modeling.

Firstly, Ukrainian is a Slavic language and is famous for its rich inflections. It is noted by Pavliuk (2018), that the number of inflections in Ukrainian by far exceeds their number in English since every notional part of speech has a variety of endings. Those endings express case and gender of nominal parts of speech (nouns, adjectives, numerals, pronouns) and tense, aspect, person, number, voice and mood forms of verbs. Also, in the Ukrainian language, many part of speech may form diminutive forms of the word, while in English only nouns have this possibility.

Secondly, in most agreement word-combinations (when the form of a headword determines the form of a dependent word) in English, there is no concord. In contrast, in Ukrainian, dependent words agree in number, gender and case, if the headword is a noun and adjunct words are adjectives and pronouns.

Thirdly, Ukrainian words often exhibit clearer morphological patterns than can be found in English words. A simplified model of a word can be determined as the stem responsible for the lexical meaning and an inflection appended to the stem. In an idealised example of Ukrainian words, stem is basically a constituent combination of three main parts: a root which is responsible for the main lexical meaning of the word, attached to which may be zero or more derivational prefix(es) and zero or more suffix(es). The presence of these affixes usually makes a contribution to the change in lexical meaning of the root.

Forth, English imposes strict constraints on the relative order of words in a sentence. On the contrary, in Ukrainian, the subject and object of the sentence can be determined not only by the order of the words themselves but also by the word's inflection and by its agreement with the verb.

These differences strongly influenced our research problem formulating and further decision making regarding the selection of the set of experiments.

3.1 Data

As pointed out by Jozefowicz et al., 2016: "Models that can accurately place distributions over sentences encode not only complexities of the language such as grammatical structures, but also distil a fair amount of information about the knowledge that a corpora may contain".

Thus, it was very important for our research to compose a corpus comparable with the ones presented for the English language, a corpus that would well represent Ukrainian vocabulary and grammatical patterns.

As a rule, for data-science tasks, the available data is split into training, validation and test partitions according to the proportion of 90%, 5% and 5% respectively. In order for other researchers to easier compare their results with ours, we decided to pick an open-source well-balanced Ukrainian Brown corpus as a baseline test set. As for the training partition we couldn't meet the same restrictions and requirements in terms of diversity and being a well-composed and proofread data set for the lack of such data.

3.1.1 Training data

Korrespondent

This is a 2.1Gb dataset downloaded from the Korrespondent news source [1]. The articles represent social, national, political, science, sport, lifestyle&fashion, business, showbiz and other news. Other then prose texts, it also contains weather forecasts, sport news, cinema and theatre programs including tables, comparisons, lists of metrics and indices.

We preprocessed (described later in this article) this dataset manually and composed a corpus sufficient as the train set for our experiments.

After the preprocessing it consists of 8 640 598 sentences separated by the Tokenize Text tool from API NLP UK[2].

Collection of Ukrainian fiction

This collection is an 846Mb dataset of Ukrainian literature (prose and poems) kindly shared by Dmytrii Chaplynskyi.

Unfortunately, it is common for the Ukrainian literature texts to contain original pieces of Russian texts, dialogues and citations.

We preprocessed (described later in this article) this dataset manually and composed a corpus of the size sufficient for a training set for our experiments.

After the preprocessing it consists of 5 746 840 sentences separated by Tokenize Text tool from API NLP UK[3].

3.1.2 Evaluation set

BrUk

As a gold standard test corpus for all the experiments in this master thesis, we chose Ukrainian Brown Corpus (BrUk) [4]. BrUk is a well balanced and proofread collection of original Ukrainian texts from different regions of Ukraine. These texts were

[1]Korrespondent: https://korrespondent.net
[2]API NLP UK: https://github.com/brown-uk/nlp_uk
[3]API NLP UK: https://github.com/brown-uk/nlp_uk
[4]Ukrainian Brown Corpus (BrUk): https://github.com/brown-uk/corpus

published between 2010 and 2018 years. BrUk contains only prose texts that do not belong to any specific dialectic group and contains data from such domains as 1) news media; 2) religious media; 3) professional literature; 4) aesthetic-informative literature; 5) administrative documents; 6) popular science; 7) science literature; 8) educational literature; 9) fiction writing.

We conducted a descriptive analysis [5] of "Good" (proofread fragments without mistakes) and "So-so" (proofread fragments with some minor mistakes) parts of this corpus before the preprocessing. It consists of 924 texts, of length from 80 to 2195 words (not tokens). Overall, it comprises 600810 words and 38728 unique lemmas 3.1 (the lemmatization was done with Language Tool API NLP UK [6]).

Domain	Percentage out of all corpus	Number of words	Number of Sentences	Number of unique lemmas
news media	24.0%	144276	9333	18293
religious media	3.3%	19548	1220	4710
professional literature	5.3%	32160	2393	7356
aesthetic-informative literature	7.9%	47491	2820	10858
administrative documents	2.0%	11974	563	2272
popular science	5.4%	32545	2137	7404
science literature	11.3%	67690	3114	9252
educational literature	13.5%	81245	4913	11010
fiction writing	27.3%	163891	13527	21405

TABLE 3.1: The Ukrainian Brown Corpus (BrUk) composition.

3.1.3 Data preprocessing and preparation

We spent a substantial amount of time on data preprocessing and preparation, especially while forming the training set. Firstly, we removed a large number of Russian and other language citations, dialogues and poems or replaced them with the *_#foreign_* token where they were only a part of the sentence. For the non-Cyrillic languages, we also replaced separate foreign words with this token. It was harder to do so with Cyrillic characters, so our train data still contains some small non-Ukrainian language insertions.

Secondly, we ponder the question of replacement of the digits. To represent a statistical distribution over words in a language, usually, digits are replaced with some token and not included in the vocabulary separately. For example, Penn Treebank corpus described in 2.1.1 . As a news-media source text, the "Korrespondent" part of our corpus contains a lot of digits denoting some table values, lists of metrics and indices. We decided to replace digits that mark years, dates, time with *_#year_*, *_#date_* and *_#time_* tags respectively. In the written language, digits are sometimes written instead of the numerals and, as a part of the sentence, may perform the same function as the noun (cardinal and collective numerals), adjectives (ordinal and multiplier numerals) and adverbs (multiplicative numerals and distributive numerals). Numerals that belong to a declinable class of words in collocations with other parts of speech change morphological endings. In the Ukrainian language, numerals have

[5] Git-Hub : https://github.com/Anastasiia-Khab/LMForTheUkrainianLanguage/blob/master/UkrBrownCorpusAnalysis_good%26soso.ipynb

[6] API NLP UK: https://github.com/brown-uk/nlp_uk

number, case and partly gender categories. The cardinal numerals "one" and "two" have three gender distinctions, while others have a common form for masculine and feminine genders and an individual form of the neuter gender. Also, in government type of word-combinations with nouns, numerals "two", "three" and "four" assumes a certain grammatical form of the modified word (noun), exceptional from the scheme of cases governed by numerals "five" and higher. That is why, we decided to replace the digits 1, 2, 3 and 4 where it was appropriate with a relevant word in a right grammatical form. All remaining digits we replaced with the tokens *_#number_* (non-split digits were replaced by one token) and *_#float_* respectively. Almost all Roman numerals were also replaced by the *_#number_* token.

The "Korrespondent" part of corpus contained lots of references to other media sources. We replaced 70 most common of them to the *_#media_* token.

As in any other text preprocessing, we also polished the training and test corpora by removing noisy unknown symbols, special characters and HTML tags, and converting punctuation mark to a single scheme 3.2 .

Punctuation mark sign	Punctuation mark name
"	quotation marks
'	apostrophe
-	hyphen
–	dash
()	parentheses (open and close)

TABLE 3.2: Punctuation marks standard scheme.

Examples of the sentences from the training and test corpus

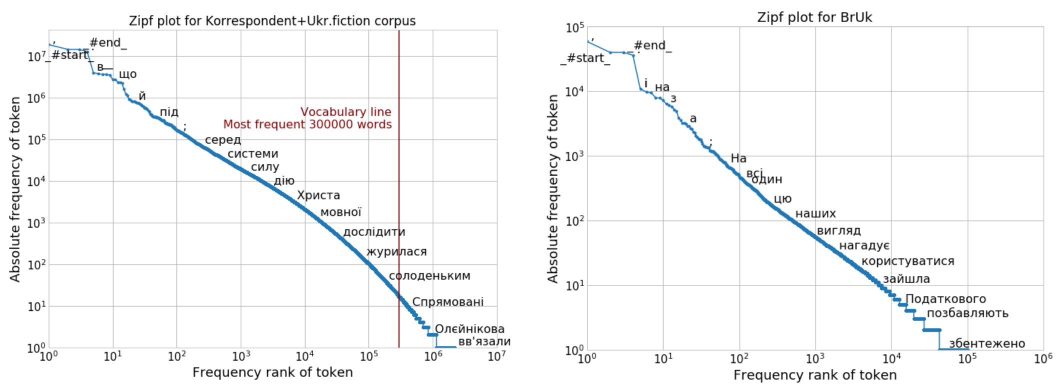
- За кордоном , з усіх дітей , які мають особливі потреби , від *_#number_* до *_#number_*% залучено до інклюзивного навчання .
  Україна має цей показник , за різними даними , від чотирьох до *_#number_*%.

- Постановка проблеми Впровадження інклюзивного навчання в Україні розпочалось Канадсько-Українським проектом , який тривав протягом *_#year_*-*_#year_* рр. , у *_#year_* році розпочався всеукраїнський експериментальний проект з розроблення інклюзивного середовища , яким до *_#year_* р. має бути охоплена уся Україна .

- Дом'є колись сказав : "Людина — це жест ".

- Він відчував , що усередині закипає злість .
  "Марічко , не бійся , дитино , я тебе не здам , не здам ".
  — Я не буду розповідати щось людині , яка не розуміє й крихти того , що відбувається .
  — Чому ж не розумію , — посміхнувся військовий , — добре розумію .

- До "Плуга "входили Дмитро Бедзик , Володимир Гжицький , Андрій Головко , Наталя Забіла , Василь Минко , Іван Сенченко , Павло Усенко та інші . Видавали журнал "*_#media_* "(*_#year_*-*_#year_*) , а також часопис "Плуг "(*_#year_*-*_#year_*) .



(A) (B)

FIGURE 3.1: Zipfian plots over (A) Korrespondent+Ukrainian fiction training corpus and over (B) Brown Ukrainian corpus.

Training and testing corpora

In the next Figure 3.1, we reproduced Zipfian plot over Korrespondent+Ukrainian fiction training corpus and over Brown Ukrainian corpus. Here the axes are *log(rank order in frequency table)* and *log (token frequency)* and sentences are tokenized as described before. Thus, separate tokens include words, punctuation marks and special tokens. Also, we removed sentences longer than 60 tokens (112 119 sentences), with the purpose of training neural network LMs with complex architectures on future steps.

Overall number of unique tokens in train corpus is 2189477. As it is claimed by Zipf's Law and it can be observed from the plot, there are few very high-frequency tokens that account for most of the tokens in text ("", "*_#start_*", "—", "*і*" etc.) and many low-frequency words (surnames, words with mistakes, and other extraordinary words). It would be computation impossible with our resources to train a neural network over all the words, that occur at least once, thus we decided to reduce the vocabulary to the size of 300000 the most frequent tokens. Our vocabulary is larger than it is usually formed for the English language. Thus, all the tokens from the tail after the red line on the Figure 3.1 (A) were replaced with *_#unknown_* token. After this transformation, the least frequent tokens in our vocabulary are the ones, that occurred 17 times.

Statistics 3.3 of Brown Ukrainian Corpus (Test) and Korrespondent + Ukrainian Fiction collection (Train and Validation). The numbers were obtained after adding *_#start_* and *_#end_* tokens (at the beginning and end of each sentence) and after removing all sentences longer than 60 tokens, including the special tokens.

	BrUk	Korrespondent +Ukr.fiction
Tokens	779,001	262,598,163
Sentences	39,900	14,335,495
Vocab size	300,000	300,000
OOV rate	3.84%	1.96%

TABLE 3.3: Statistics of the Brown Ukrainian Corpus (Test) and Korrespondent + Ukrainian Fiction collection (Training and Validation).

3.2 Methods

3.2.1 KenLM

Kneser-Ney LM was computed as a benchmark for father comparison of more complicated LMs. The main advantage of the modified N-gram LM is that with comparatively small amount of time it estimates a statistical distribution over all the n-gram occurrences in the training text.

Using open-source KenLM library [7] and *bin/lmplz* command we estimated the ARPA file on the training corpus. ARPA file represents a model and contains n-grams and their statistics. Then we queried the model with command *bin/query Model.arpa* < *BrUk.txt* on the BrUk test data, to calculate perplexity.

KenLM library automatically includes tokens $< s >$, $< /s >$, and $< unk >$, thus we didn't add *_#start_* and *_#end_* tokens to our sentences manually. Also, the model does not struggle to estimate longer sentences, so sentences longer than 60 tokens where not deleted before this set of experiments.

Pruning is a technique used to reduce the size of the ARPA file, without significantly degrading the model. It is the parameter of the estimation command in KenLM library. By setting $pruning = 0\ 0\ 1\ 1\ 1\ 1$ we specify estimator to remove singletons of order 3 and higher and by setting $pruning = 0\ 0\ 0\ 0\ 0\ 1$ only singletons of order 6 are deleted.

We experimented with different N-grams and different pruning (Table 3.4). Taking into account a trade-off between size of build ARPA files and the perplexity on the test set, we propose to consider 6-gram KenLM with $pruning = 0\ 0\ 0\ 0\ 1\ 1$ as a baseline LM for BrUk corpus.

Kenlm N-gram	Pruning	Training time (minutes)	Size of ARPA file	Perplexity including OOV	Perplexity excluding OOV
3-gram	-	03m:55	6.4G	814.93	697.82
6-gram	0 1 1 1 1 1	07m:24	5.0G	871.97	749.91
6-gram	0 0 1 1 1 1	08m:45	6.0G	800.39	686.53
6-gram	0 0 0 0 1 1	11m:51	18.2G	748.81	641.11
6-gram	0 0 0 0 0 1	23m:25	29.4G	742.42	635.54
6-gram	-	40m:05	41.1G	736.83	630.69

TABLE 3.4: Results for KenLM trained on Korrespondent+Ukrainian fiction corpus, with different pruning values. Perplexity computed on the BruK corpus.

3.2.2 RNNLM (GRU, LSTM)

Our Recurrent neural network LMs follow a common template:

1. Sentence is represented as a sequence of integers, where a token u_i is identified by its index in vocabulary V. ($u_i \in \{1, ...N\}$, $N = vocabulary\ size$)

2. An embedding layer encodes N integers into N unique feature vectors of *embedding size*.

[7]KenLM library: `https://kheafield.com/code/kenlm/`

3. Each token is propagated sequentially through the RNN layers described previously 2.1.2 - 2.1.2.

4. A fully connected layer applies a linear transformation in order to gain a result vector in dimension, which equals to the size of the vocabulary. A softmax function is computed over the result vector, assigning probabilities for each token in vocabulary to be the next token in the sequence.

5. The loss function is computed with the same function as perplexity.

All the models were trained with PyTorch (1.3.1 version) machine learning library.

Word-level models

The following Table A.2 represents training parameters, and results of word-level trained LMs. The models were trained on Korrespondent+Ukrainian Fiction corpus formed in 3.1.3 with vocabulary size 300000 and pruned sentences longer the 60 tokens. Sentences were processed into the model in batches of size 40 packed with *torch.nn.utils.rnn.pack_sequence*. Before each epoch, the sentences were randomly shuffled, and the validation set (of 4%) was selected.

It would be enormously time-consuming to grid-search through different sets of hyperparameters of the models. Thus, we firstly chose hyperparameters randomly, then based on the previous results. Except for the models shown in Table A.2, we also held a number of experiments with SimpleRNN models and different parameters of clipping, dropout and learning rate. Many of the experiments soared with perplexity inf on the train and validation set as in the first model from Table A.2 because of either vanishing or exploding gradients 2.1.2. Also, the architecture of the preferred model was highly restricted by the available computational and memory resources.

FastText pretrained vectors

Motivated by Bojanowski et al., 2017 and their results on language modeling for Slavic languages, we had the aim to evaluate the performance of bags of character n-grams 2.1.4 on the Ukrainian language.

FastText [8] pretrained 300 dimensional word embeddings for 157 languages (including Ukrainian) were received and made public by Facebook AI Research in 2018 Grave et al., 2018. The model was trained using CBOW (Mikolov et al., 2013) with position-weights and bags of character n-grams 2.1.4.

The FasText public vocabulary includes 2M tokens. Unfortunately, there were several mistakes in tokenization of Ukrainian texts (for example lots of words are not separated at all; apostrophes are considered as separation symbols etc.) Overall, 763578 unique tokens from our train corpus were presented in FastText vocabulary. Although, FastText embedding vectors are the sum of character n-gram representations and vectors for out-of-vocabulary words can be easily obtained using FastText tools, for this experiment we decided to use only vocabulary size of 300K most frequent FastText pretrained embeddings. We replaced all OOV tokens with *_#unknown_* token and calculated its embedding as the mean of all vectors not used in our vocabulary. Also, we calculated embedding for *_#number_* token as the mean of all numerical tokens in FastText vocabulary.

[8]FastText vectors for 157 languages: `https://fasttext.cc/docs/en/crawl-vectors.html`

Further, we used 300K pretrained FastText vectors instead of the embedding layer in our LSTM LM and froze it. With this approach, we can approximately compare gained results with the results of the previously trained models, as both support the exact same set of tokens.

3.2.3 BPE

BPE incrementally substitutes a dictionary with a set of tokens such that the total number of tokens for encoding the text is minimized. We computed segmentation with the varying number of BPE merges (Step 2: 2.1.4) in order to see how the size of the encoded corpus depends on the number of BPE merge steps (Figure 3.2). The vocabulary size formulated by BPE encoding equals to the number of unique characters in the text plus the number of merge operations.

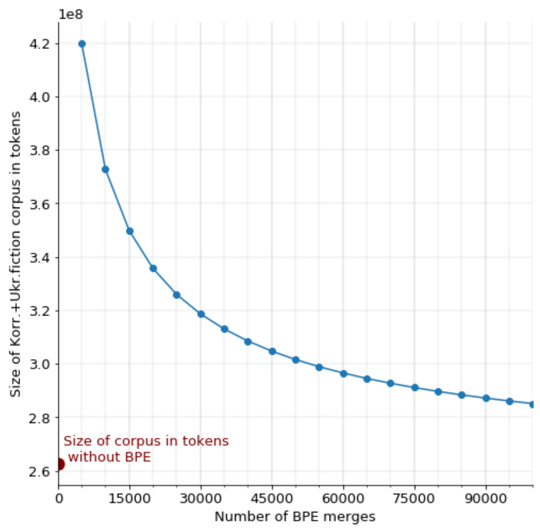


FIGURE 3.2: Dependence of the total number of tokens for encoding the text on the number of BPE merges.

For our task, not only the number of tokens for encoding the text and the vocabulary size is important, but also the significance of the morphological information encoded in individual separated tokens. In this experiment, we wanted to observe the influence of cutting words into the sequence of root , affixes and ending. In general, BPE is not invented specifically for this. It just gives a good balance between vocabulary size and decoding efficiency. But usually, frequently occurring parts of the word turn out to be affixes. Thus, out of gained vocabularies, we manually selected the one where not too many entire words were already selected, but a large portion of the vocabulary consists of small parts of words. Then we round the size of this vocabulary up to 20K tokens.

We implemented segmentation using Google's SentencePiece library [9], which is also the official code of Kudo and Richardson, 2018 paper. SentencePiece specifies the final size of vocabulary as an input parameter for BPE tokenization algorithm (we selected 20K). Also, the tool reserves vocabulary ids for special meta symbols, that were easily changed to our corresponding tokens (unknown symbol *_#unknown_*, beginnning of sentence *_#start_* and end of sentence *_#end_*). Another great feature of SentencePiece is its ability to handle user-defined symbols as one

[9]Sentencepiece library: https://github.com/google/sentencepiece

piece in any context. We specified our special tokens (*_#foreign_*, *_#year_*, *_#date_*, *_#time_*, *_#media_*, *_#number_*, *_#float_*) as user-defined symbols.

Then, we just encoded our text with the series of ids corresponding to the unique tokens in a generated vocabulary and feed packed sequence to the RNN as it was done for the word-level (3.2.2).

We decided to compute perplexity for the subword-level LM normalizing not by the number of tokens in predicted sequence, but by the number of words as it was proposed in 2.25.

Table A.1 depicts the results of training RNNs with BPE-formulated vocabulary of 20K tokens and different hyperparameter set-ups. Overall, compared to word-level RNNs the size of the model, number of trainable parameters and, as a result, computational resources requirements substantially decrease. Even though the numerical values of the perplexity metric are higher, we don't consider this as an indication that the results are worse. Mainly, because of two important reasons:

1) The subword-level model is open-vocabulary, and it tries to predict from the whole theoretically possible set of words. Thus, it spreads its probability mass thin over a near-infinite number of cases, in contrast to the comparably small size of word-level vocabulary (300K).

2) If the word-level model correctly predicts that it hasn't seen an *_#unknown_* token before, its perplexity decreases, while the subword- level model does not have such a possibility because its *_#unknown_* token represents just individual symbols that never occurred in training corpus. But with well composed corpora such symbols shouldn't exist.

For the reasons mentioned above and the performance of pretrained FastText vectors described in 3.2.2, we believe that subword-level approaches have a strong potential for language modeling of highly inflectional languages, such as Ukrainian.

Chapter 4

Conclusions

4.1 Contribution

In this work, we considered the task of Language Modeling for the Ukrainian language. During the work, we made the following contributions:

1. We conducted a comprehensive scientific literature review on language modeling for the English Language and for Slavic languages of the same language group as Ukrainian, explored the main benchmarks and most promising approaches. This all gave a good understanding of the field for further reapplying the acquired knowledge to the Ukrainian language.

2. We composed, preprocessed and described a dataset (262M tokens, 14M sentences) sufficient for training neural LMs for the Ukrainian language. The Korrespondent news-commentary corpus and a collection of Ukrainian literature served as the primary source here. As a baseline for training word-level LMs with a limited vocabulary, we propose to use a vocabulary of 300K words. This general-purpose training corpus makes it possible for researchers to reproduce our experiments and train new LMs.

3. As a benchmark evaluation corpus, we propose to use publicly available Brown Ukrainian corpus (BrUk) (779K tokens, 39K sentences). As a baseline language model, we consider modified Kneser-Ney 6-gram model with the pruning of n-grams (n>4) occurred less than 2 times trained on our training corpus, which estimated on BrUk gives a perplexity 748.81, and the model file of size 18.2G.

4. We ran a number of experiments using two types of neural architectures: GRU and LSTM. We explored different hyperparameters and applied two types of text tokenization: word-level and subword-level. Out of all the models trained from scratch the best perplexity (268.5) on BrUk test corpus was achieved with a three-layered LSTM word-level model (with the hidden size of 500).

5. When using BPE as a word separation technique, we formed a vocabulary of 20K subword units and trained a three-layered LSTM model (hidden size of 2048). The perplexity calculated on the BrUk corpus was 393.6. However, taking into account the fact that this approach generates open-vocabulary models, at the same time, takes up less memory and consumes less computation time, we consider the obtained results promising for the morphologically rich and highly inflectional Ukrainian language.

6. We used 300K pretrained FastText vectors (300 dimensions) as a "frozen" embedding layer in a word-level LSTM with two hidden layers of size 500. This approach provided the best result from the very first epoch, compared to the

same model architecture trained independently. (Perplexity on BrUk test corpus 235.7).

7. Also, we contributed some improvements to the BrUk GitHub project and, currently, we are in the process of making our preprocessed train set publicly available on the LangUk page.

4.2 Future work

1. Due to the time restriction, we couldn't explore all of the known data sources. For example, as an immediate next step, it makes sense to incorporate data from the Ukrainian Web Corpus [1] from the TenTen corpus family. This is a corpus made up of texts collected from the Web.

2. The Mogrifier LSTM extension (Melis, Kočiský, and Blunsom, 2020) in the form of mutual gating of the current input and the previous output. Currently achieves state-of-the-art [2] language modelling results on PTB and WikiText-2 benchmarks. The paper was published on September 4th and we should consider experimenting with it on our data.

3. Perplexity is a relatively easy way to evaluate language models, but it has its limitations. As we learned during this research, perplexity doesn't allow proper evaluation of certain neural architectures or comparison of all varieties of LMs in a fair way. Hence, we believe that creating extrinsic evaluation toolkit similar to GLUE is paramount for Ukrainian language modeling. Measuring language model's performance against a diverse hand-crafted set of linguistic tasks can make the evaluation more objective and motivate the Ukrainian NLP community for contributions. At the same time, we realize that such an undertaking requires dedicated efforts from experts and goes well beyond the scope of our research goals here.

[1]SketchEngine: `www.sketchengine.eu/corpora-and-languages/ukrainian-text-corpora/`
[2]NLP-progress: `http://nlpprogress.com/english/language_modeling.html`

Appendix A

Tables of experiments with GRU and LSTM LMs

All the models were trained on Korrespondent+Ukrainian corpus and tested on Brown Ukrainian corpus 3.1.1. The experimental setup for Tables A.1, A.2 are described in sections 3.2.3 and 3.2.2 respectively.

Model type	LSTM				
BPE vocabulary	20 000 sub-word tokens				
Embedding	300		500		
Layers	1: **300** 2: **300**	1: **1024** 2: **300**	1: **1024** 2: **512**	1: **1024** 2: **1024** 3: **1024**	1: **2048** 2: **2048** 3: **2048**
Number of parameters	13 464 800	19 042 496	29 660 320	53 544 096	139 011 232
Size	155Mb	218Mb	340Mb	613Mb	1591Mb
Learning rate	lr(epoch$_1$)=0.001				
lr(epoch$_n$)=	$lr_{n-1} * 0.8$	$lr_{n-1} * 0.8$	$lr_{n-1} * 0.8$	$lr_{n-1} * 0.85$	$lr_{n-1} * 0.85$
Test perplexity for each epoch					
Epoch 1	927.2	782.7	742.5	620.9	529.3
Epoch 2	840.5	700.9	657.6	547.1	460.8
Epoch 3	796.2	652.8	618.9	512.4	426.8
Epoch 4	770.2	620.6	580.9	487.0	408.0
Epoch 5	748.6	604.8	558.5	470.1	393.6
Epoch 6	745.1	588.2	543.6	458.5	-
Epoch 7	730.2	575.9	533.0	445.6	-
Epoch 8	723.6	567.4	523.9	438.8	-
Epoch 9	712.8	563.2	518.8	433.3	-
Epoch 10	704.3	559.9	516.3	426.2	-
Epoch 11	698.9	-	-	-	-
Epoch 12	692.2	-	-	-	-
Epoch 13	690.9	-	-	-	-
Comput.engine	GeForce RTX 2080 Ti (11GB RAM)				
Approximate training time of one epoch	02h:05	02h:50	03h:30	06h:30	17h:00

TABLE A.1: Results of sub-word-level (BPE) LSTM

Model parameters	Training parameters	Model size	Test PP for each epoch	
GRU Embed.: 300 Layers: 1: 300	Clipping: - Dropout: - Optimizer: RMSProp lr = 0.001 Compute engine: 1	Size: 1380Mb NParams: 180 841 800 Time1epoch: $\approx 7h : 40$	1: 1123.3 2: 878.8 3: 835.6 4: 793.2	5: 666.9 6: 676.1 7: 682.5 8: inf
GRU Embed.: 300 Layers: 1: 300 2: 300	Clipping: 0.2 Dropout: 0.2 Optimizer: **RMSProp** lr = 0.001 Compute engine: 1	Size: 1384Mb NParams: 181 383 600 Time1epoch: $\approx 13h : 15$	1: 796.2 2: 730.2 3: 716.7 4: 723.9	- - - -
GRU Embed.: 300 Layers: 1: 300 2: 300	Clipping: 0.2 Dropout: 0.2 Optimizer: Adam lr(epoch)= $\text{lr}_{n-1} * 0.1$ Compute engine: 1	Size: 2076Mb NParams: 181 383 600 Time1epoch: $\approx 18h : 35$	1: 426.6 2: 373.7 3: 372.5 4: 372.1	- - - -
GRU Embed.: 300 Layers: 1: 300 2: 300 3: 300	Clipping: 0.2 Dropout: **0.4** Optimizer: Adam lr(epoch)= $\text{lr}_{n-1} * 0.75$ Compute engine: 2	Size: 2082Mb NParams: 181 925 400 Time1epoch: $\approx 20h : 55$	1: 505.6 2: 467.2 3: 438.7 4: 416.9	5: 401.0 - - -
LSTM Embed.: 300 Layers: 1: 300 2: 300	Clipping: 0.2 Dropout: 0.2 Optimizer: Adam lr(epoch)= $\text{lr}_{n-1} * 0.25$ Compute engine: 2	Size: 2080Mb NParams: 181 744 800 Time1epoch: $\approx 19h : 10$	1: 354.7 2: 315.6 3: 305.9 4: 298.4	5: 296.2 6: 294.1 7: 293.1 -
LSTM Embed.: 300 Layers: 1: **500** 2: **500**	Clipping: 0.2 Dropout: 0.2 Optimizer: Adam lr(epoch)= $\text{lr}_{n-1} * 0.85$ Compute engine: 2	Size: 2792Mb NParams: 243 908 000 Time1epoch: $\approx 23h : 50$	1: 334.5 2: 324.1 3: 305.8 4: 299.2	- - - -
LSTM Embed.: 300 Layers: 1: **500** 2: **500** 3: **500**	Clipping: 0.2 Dropout: 0.2 Optimizer: Adam lr(epoch)= $\text{lr}_{n-1} * 0.85$ Compute engine: 2	Size: 2815Mb NParams: 245 912 000 Time1epoch: $\approx 25h : 30$	1: 331.5 2: 298.2 3: 287.4 4: 279.2	5: 269.4 6: 268.5 - -
LSTM **FastText** Embed.: 300 freezed Layers: 1: **500** 2: **500**	Clipping: 0.2 Dropout: 0.2 Optimizer: Adam lr(epoch)= $\text{lr}_{n-1} * 0.85$ Compute engine: 1	Size: 2105Mb NParams: 153 908 000 Time1epoch: $\approx 16h : 20$	1: 266.9 2: 250.1 3: 243.6 4: 235.7	- - - -

TABLE A.2: Results of word-level GRU, LSTM. Vocabulary size 300K. Perplexity computed for BrUk corpus. Computation engines 1: GeForce RTX 2080 Ti (11GB RAM) ; 2: TITAN X(Pascal) (12GB RAM)

Bibliography

Al-Rfou, Rami et al. (2018). "Character-Level Language Modeling with Deeper Self-Attention". In: *CoRR* abs/ 1808.04444. arXiv: 1808.04444. URL: http://arxiv.org/abs/1808.04444.

Arisoy, Ebru et al. (2012). "Deep Neural Network Language Models". In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. WLM '12. Montreal, Canada: Association for Computational Linguistics, pp. 20–28. URL: http://dl.acm.org/citation.cfm?id=2390940.2390943.

Bell, Timothy C., I. H. Witten, and John G. Cleary (1990). *Text compression*. English. Prentice Hall Englewood Cliffs, N.J, xviii, 318 p. : ISBN: 0139119914.

Bengio, Yoshua et al. (2003). "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3, pp. 1137–1155.

Bojanowski, Piotr et al. (2017). "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. DOI: 10.1162/tacl_a_00051. URL: https://www.aclweb.org/anthology/Q17-1010.

Bryant, Christopher and Ted Briscoe (June 2018). "Language Model Based Grammatical Error Correction without Annotated Training Data". In: *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 247–253. DOI: 10.18653/v1/W18-0529. URL: https://www.aclweb.org/anthology/W18-0529.

Chelba, Ciprian et al. (2013). "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling". In: *CoRR* abs/1312.3005. arXiv: 1312.3005. URL: http://arxiv.org/abs/1312.3005.

Chen, Stanley F. and Joshua Goodman (1998). *An Empirical Study of Smoothing Techniques for Language Modeling*. Tech. rep.

Cherian, Thomas, Akshay Badola, and Vineet Padmanabhan (2018). "Multi-cellLSTM Based Neural Language Model". In: *CoRR* abs/1811.06477. arXiv: 1811.06477. URL: http://arxiv.org/abs/1811.06477.

Cho, Kyunghyun et al. (Oct. 2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://www.aclweb.org/anthology/D14-1179.

Chung, Junyoung et al. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop. URL: http://arxiv.org/abs/1412.3555.

Cotterell, Ryan et al. (June 2018). "Are All Languages Equally Hard to Language-Model?" In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*

(Short Papers). New Orleans, Louisiana: Association for Computational Linguistics, pp. 536–541. DOI: 10.18653/v1/N18-2085. URL: https://www.aclweb.org/anthology/N18-2085.

Edunov, Sergey, Alexei Baevski, and Michael Auli (2019). "Pre-trained Language Model Representations for Language Generation". In: *CoRR* abs/ 1903.09722. arXiv: 1903.09722. URL: http://arxiv.org/abs/1903.09722.

Filippova, Katja et al. (Sept. 2015). "Sentence Compression by Deletion with LSTMs". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 360–368. DOI: 10.18653/v1/D15-1042. URL: https://www.aclweb.org/anthology/D15-1042.

Gage, Philip (1994). "A new algorithm for data compression". In: *The C Users Journal* 12.2, pp. 23–38. URL: https://ci.nii.ac.jp/naid/10027597010/en/.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Book in preparation for MIT Press. MIT Press. URL: http://www.deeplearningbook.org.

Grave, Edouard et al. (2018). "Learning Word Vectors for 157 Languages". In: *CoRR* abs/1802.06893. arXiv: 1802.06893. URL: http://arxiv.org/abs/1802.06893.

Heafield, Kenneth (2011). "KenLM: Faster and Smaller Language Model Queries". In: *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland, United Kingdom, pp. 187–197. URL: https://kheafield.com/papers/avenue/kenlm.pdf.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hsu, Bo-June and James Glass (Jan. 2008). "Iterative language model estimation: efficient data structure algorithms." In: pp. 841–844.

Hu, Yi et al. (2007). "A Language Modeling Approach to Sentiment Analysis". In: *International Conference on Computational Science*.

Irie, Kazuki et al. (Sept. 2016). "LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition". In: *Interspeech*. San Francisco, CA, USA, pp. 3519–3523.

Jelinek, Fred and Robert L. Mercer (1980). "Interpolated estimation of Markov source parameters from sparse data". In: *Proceedings, Workshop on Pattern Recognition in Practice*. Ed. by Edzard S. Gelsema and Laveen N. Kanal. Amsterdam: North Holland, pp. 381–397.

Jozefowicz, Rafal et al. (2016). "Exploring the limits of language modeling". In: *arXiv*, 1602.02410v2 [cs.CL]. URL: https://arxiv.org/abs/1602.02410.

Katz, Slava M. (1987). "Estimation of probabilities from sparse data for the language model component of a speech recognizer". In: *IEEE Trans. Acoustics, Speech, and Signal Processing* 35, pp. 400–401.

Kim, Yoon et al. (2015). "Character-Aware Neural Language Models". In: *CoRR* abs/1508.06615. arXiv: 1508.06615. URL: http://arxiv.org/abs/1508.06615.

Kneser, Reinhard and Hermann Ney (1995). "Improved backing-off for M-gram language modeling". In: *1995 International Conference on Acoustics, Speech, and Signal Processing* 1, 181–184 vol.1.

Krause, Ben et al. (2016). "Multiplicative LSTM for sequence modelling". In: *CoRR* abs/1609.07959. arXiv: 1609.07959. URL: http://arxiv.org/abs/1609.07959.

Kudo, Taku (July 2018). "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Melbourne, Australia: Association for Computational Linguistics, pp. 66–75. DOI: 10.18653/v1/P18-1007. URL: https://www.aclweb.org/anthology/P18-1007.

Kudo, Taku and John Richardson (2018). “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing”. In: *CoRR* abs/1808.06226. arXiv: 1808.06226. URL: http://arxiv.org/abs/1808.06226.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini (June 1993). “Building a Large Annotated Corpus of English: The Penn Treebank”. In: *Comput. Linguist.* 19.2, pp. 313–330. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972470.972475.

Martin, Sven, Jörg Liermann, and Hermann Ney (Apr. 1998). “Algorithms for Bigram and Trigram Word Clustering”. In: *Speech Commun.* 24.1, 19–37. ISSN: 0167-6393. DOI: 10.1016/S0167-6393(97)00062-9. URL: https://doi.org/10.1016/S0167-6393(97)00062-9.

Maučec, Mirjam, Tomaž Rotovnik, and Melita Zemljak Jontes (Jan. 2003). “Modelling Highly Inflected Slovenian Language”. In: *International Journal of Speech Technology* 6, pp. 245–257. DOI: 10.1023/A:1023466103841.

Melis, Gábor, Tomáš Kočiský, and Phil Blunsom (2020). “Mogrifier {LSTM}”. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SJe5P6EYvS.

Merity, Stephen et al. (2016). “Pointer Sentinel Mixture Models”. In: *CoRR* abs/1609.07843. arXiv: 1609.07843. URL: http://arxiv.org/abs/1609.07843.

Mikolov, Tomas et al. (2011a). “Extensions of recurrent neural network language model”. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531.

Mikolov, Tomas et al. (2011b). “Subword Language Modeling With Neural Networks”. In:

Mikolov, Tomas et al. (2013). “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Mikolov, Tomáš et al. (2010). “Recurrent neural network based language model”. In: *INTERSPEECH* 2, pp. 1045–1048. URL: https://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

N. Dauphin, Yann et al. (2017). “Language Modeling with Gated Convolutional Networks”. In: *arXiv*, 1612.08083v3 [cs.CL]. URL: https://arxiv.org/abs/1612.08083.

Ney, Hermann, Ute Essen, and Reinhard Kneser (1994). “On structuring probabilistic dependences in stochastic language modelling”. In: *Computer Speech Language* 8, pp. 1–38.

Olah, Christopher (2015). *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Online; posted on August 27, 2015.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). “On the difficulty of training recurrent neural networks”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, pp. 1310–1318. URL: http://proceedings.mlr.press/v28/pascanu13.html.

Pavliuk, Nataliia (July 2018). *Contrastive Grammar of English and Ukrainian*.

Rush, Alexander M., Sumit Chopra, and Jason Weston (2015). “A Neural Attention Model for Abstractive Sentence Summarization”. In: *CoRR* abs/1509.00685. arXiv: `1509.00685`. URL: `http://arxiv.org/abs/1509.00685`.

Sazhok, Mykola and Valentyna Robeiko (2013). “Language Model Comparison for Ukrainian Real-Time Speech Recognition System”. In: *Speech and Computer*. Ed. by Miloš Železný, Ivan Habernal, and Andrey Ronzhin. Cham: Springer International Publishing, pp. 211–218.

Schwenk, Holger, Anthony Rousseau, and Mohammed Attik (June 2012). “Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation”. In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Montréal, Canada: Association for Computational Linguistics, pp. 11–19. URL: `https://www.aclweb.org/anthology/W12-2702`.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: `10.18653/v1/P16-1162`. URL: `https://www.aclweb.org/anthology/P16-1162`.

Shannon, Claude Elwood (July 1948). “A Mathematical Theory of Communication”. In: *The Bell System Technical Journal* 27.3, pp. 379–423. DOI: `10.1002/j.1538-7305.1948.tb01338.x`. URL: `https://ieeexplore.ieee.org/document/6773024/`.

— (Jan. 1951). “Prediction and Entropy of Printed English”. In: *Bell System Technical Journal* 30, pp. 50–64. URL: `http://languagelog.ldc.upenn.edu/myl/Shannon1950.pdf`.

Vaswani, Ashish et al. (Oct. 2013). “Decoding with Large-Scale Neural Language Models Improves Translation”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 1387–1392. URL: `https://www.aclweb.org/anthology/D13-1140`.

Wang, Alex et al. (Nov. 2018). “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, pp. 353–355. DOI: `10.18653/v1/W18-5446`. URL: `https://www.aclweb.org/anthology/W18-5446`.

Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals (2014). “Recurrent Neural Network Regularization”. In: *CoRR* abs/1409.2329. arXiv: `1409.2329`. URL: `http://arxiv.org/abs/1409.2329`.

Zipf, George K. (1949). *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.

Zipf, George Kingsley (1935). *The Psychobiology of Language*. New York, NY, USA: Houghton-Mifflin.