Saint Petersburg State University
Faculty of Applied math and processes of control

Department of programming technologies

Korobkov Nikita Aleksandrovich

Bachelor final qualification work

# Transfer learning for dialog systems

Scientific advisor:
Mishenin A.N.

Saint Petersburg
2019

# Table of Content

# Introduction

In recent years, more and more attention has been paid to solving various problems using machine learning and neural networks. One of the most interesting directions in the field of application of modern neural network methods are dialogue systems.

A dialogue system is a set of programs and algorithms that allow a person to conduct a dialogue with a program in a manner peculiar to man. Sometimes dialogue systems are called conversational artificial intelligence or a chat bot.

The task of building such programs is relevant for the industrial field, because solving user problems through dialogue with a support agent has always been and will remain the simplest and most effective method. Since verbal communication is the most natural way of communication for a person. Building a good and reliable dialogue system replacing call center employees would significantly reduce companies' costs for hiring staff.

Dialogue systems can be divided into general-purpose systems and task-oriented. The narrower the problem is to solve the system, the simpler it can be arranged. In the most trivial case, the system can simply return a previously known answer. For example, the current time.

We would like to build a system that solves a more general problem. To do this, you must first find out in detail what exactly the user wants (ask a question and get an answer). In the most trivial implementation, this scenario may look like a simple choice of a task from the list. With the use of this approach in the industry can be encountered today. When calling the bank or the airline, you can often hear a robot that will offer to press different buttons depending on the purpose of the call ("To check the balance, press 1, to clarify the status of the application, press 2," etc.). This approach is good because it does not require any intelligence from the system and works very reliably. However, it takes a lot of user time and is often unnerving. Ideally, we would like to receive a request in the form of a sentence in a natural language, for example, "What is the weather now on the street?", Immediately recognize the user's intention.

This is one of the subtasks in the construction of interactive systems, on which I would like to focus our attention in this work. This task is quite relevant and has been standing for a long time. So, many methods have been proposed for its solution. They will be discussed in more detail in the "Literature Review" section. Most of the methods are focused on working with the English language. Basically, because the most data is collected for English. In addition, English is simply considered the default language in the scientific community.

While impressive results have been achieved for English, the situation with other languages is somewhat different. The presented models are mostly trained on a huge amount of tagged data. The collection and preparation of similar data sets for other languages, together with the subsequent training of the model, can take significant time and computational resources. Therefore, simply transferring the achieved results by teaching an identical model in another language is not always possible and rational.

However, for almost all world languages there are complete dictionaries and simple translators. Using the general knowledge about the connection of two languages (dictionaries, parallel texts), one can generalize the knowledge of one model into another language without using the marked data for the second language at all (or using very little). This approach in the literature is called Transfer Learning.

The purpose of this work is to build a model for extracting intent from a sentence in Swedish using the knowledge acquired by an English-trained model. Swedish was chosen because of its semantic similarity with English. Both of these languages belong to the Germanic language group. In order to test the dependence of the quality of work of methods on the similarity of languages, we also conducted a series of experiments for the Finnish language. English and Finnish are in different language families and, accordingly, have much less in common than with Swedish.

The developed methodology can be used to build models of approximately the same accuracy for any other language.

## Statement of the Problem

Let there be a set of commands in English $E$ and a finite set of intentions $K$

$$K = \{k_1, k_2, \ldots, k_n\}$$

Each team from $E$ uniquely corresponds to an element of the set $K$. Matching is denoted by $J_e$

$$J_e : E \to K$$

The training data consists of the sets $E, K$ and the matching $J_e$. At the same time, there is also a function-translator $T$, which assigns a command in Swedish to each command in English. Let $S$denote the set of commands in the Swedish language

$$T : E \to S$$

. The goal of this paper is to obtain the function $J_s : S \to K$ that matches any command in the Swedish language with the intention. In this case, two conditions must be met.

1. The intention must coincide with the intention of translating the Swedish team into English

$$\forall s \in S \quad J_s(s) = J_e(T^{-1}(s)) \tag{1}$$

$$\forall e \in E \quad J_s(T(e)) = J_e(e) \tag{2}$$

2. Function $J_s$ should not depend on the function $T$.

The second condition is decisive for this task. If we could use the $T$ function in $J_s$, then we could just define $J_s$ as in (1) and stop at that. But this is impossible, since translation (calculation $T^{-1}$) is too expensive. We would like to receive function which would be rather easily computable for use in mobile applications. Therefore, instead of directly translating a command from Swedish into English and then applying existing algorithms, we will try to highlight some key attributes of the command in Swedish and use them to recognize intent.

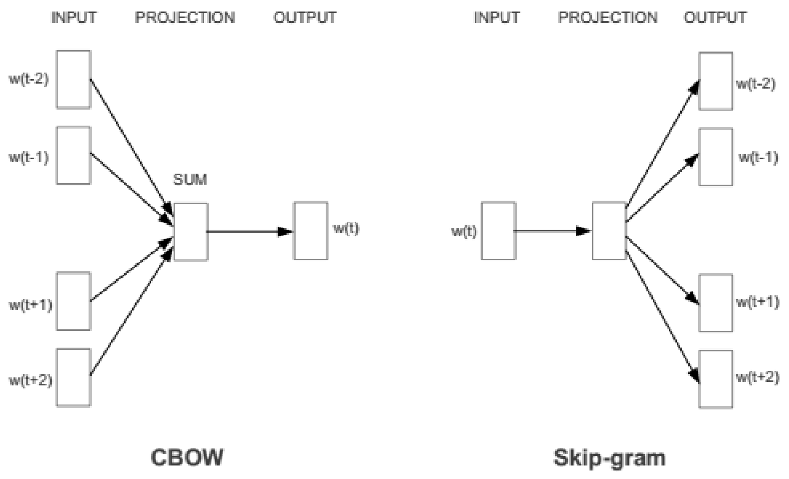
By virtue of requirement 2 it is hardly possible to satisfy requirement 1 completely. Instead, we will set the task to build a system that will be able to predict the user's intentions correctly as often as possible at training data. As a metric of prediction quality, we will use the share of teams in the test sample, according to which the system made the right decision.

# Literature review

The task of constructing interactive systems lies in the field of automatic analysis of text data. One of the main issues of the use of neural networks in this area is the effective representation of words in computer memory. When analyzing texts, it is necessary to replace words with vectors that somehow reflect the semantic meaning of the word.

The most common method for obtaining word vectors is described in [14] In this approach, the learning text corpus is viewed by the $(2h + 1)$ word width window, and for each window a single-layer neural network predicts the central word of the window $w_t$ around $w_{t+i} | i \in [-h, h]$ or vice versa. These architectures are called Continuous Bag-of-words and Skipgram, respectively. Minimizing the prediction error, the neural network builds a projection of words into the vector space of a predetermined dimension. When a given prediction accuracy or a certainreached number of epochs is, the algorithm generates a dictionary with vector representations for words from the learning corpus.

Рис. 1. The neural network architectures represented in [14] for a window width h = 5



This approach allows us to obtain vectors with property of semantic proximity. We can hope that words with a similar meaning will be located side by side in the constructed vector space. After the methods described in [14] showed their effectiveness in a number of word processing tasks [15], several similar methods were developed and proposed for constructing word vectors, such as ELMo [16] and GloVe [?].

At the moment, most natural language processing techniques use word vectors anyway.

In our work, we used the augmented implementation of the original word2vec ”fastText” [2] algorithm. Unlike the original architecture, this approach, in addition to the full words of the context, uses for learning parts of words, which allows predicting word vectors for words that are absent in the training collection.

To work with sequences of words (sentences), recurrent neural networks described in [17], [11] articles are often used. In order to avoid the problem of damped gradients, different architectures of recurrent modules are used in training, which can better “remember” information. For example, LSTM [10] or GRU [5].

In the article ”Attention is all you need” [18], a group of researchers from Google describes a fundamentally new approach to the processing of consistent textual information and, in particular, to translations. Instead of the classical architecture of recurrent neural networks using LSTM or GRU modules, the authors propose a so-called attention mechanism, which makes it possible to better represent the information contained in the proposals at the coding stage. This network is also called ”Transformer” because of the flexibility of the internal structure, which allows to obtain different information about the encoded sentence, depending on the request. Many modern automatic translators use this technology.

In our work, we used a ready-made automatic translation system from Yandex. A sample description of the mechanisms of their translator is available in the article [19].

The idea of transformer networks is developed in the article [7]. The authors propose to train a multi-layer model of transformer modules on the task of determining the coherence of sentences and prediction of a missing word. The resulting model shows exceptional results after additional training on a number of specific tasks.

The problem of predicting intentions from a fixed set can be formulated as a problem of the classification of sentences. An interesting approach to this problem using convolutional networks was proposed in the article [12].

# 1. Data Processing

In our work, we used the collection compiled by SNIPS [6] as training and test data. Data includes over 13,000 user requests in English. Each request falls into one of seven categories according to the user's intention.

Presented intentions:

- Get weather (View weather - 2000 entries)
- Play music (Include music - 2000 entries)
- Book restaurant (Book restaurant - 1973 entries)
- Search creative work (Search for works - 1954 entries)
- Add to playlist (Add to playlist - 1942 entries)
- Rate book (Submit evaluation book - 1956 entries)
- Search movie schedule (1959 entries)

Data provided by SNIPS licensed under Creative Commons Zero v1.0 Universal and available for download via the link `https://github.com/snipsco/nlu-benchmark`.

Таблица 1. Examples of queries from the SNIPS dataset

| **Category** | **Example** |
|---|---|
| Get weather | What is the forecast at 12 am in Sudan. |
| Play music | Play some 1954 songs on my Itunes. |
| Book restaurant | Book four people at a Madagascar bar. |
| Search creative work | Find me the Lace and Whiskey soundtrack. |
| Add to playlist | Add this artist to spring music. |
| Rate book | Give this textbook a 5 out of 6 rating. |
| Search movie schedule | Where is Road to the Stage playing. |

To prepare the training and test data in Swedish and Finnish, we used the Yandex Translator API [1].

Before being used for teaching models, the text was pre-processed. It consisted of several stages.

1) Lowercase all letters;

2) Replaces all whitespace characters with spaces;

3) Removal of all accents, umlauts and similar marks. Delete all non-ascii characters;

4) Disclosure of all abbreviations (for example, "You're -> you are");

5) Delete all special characters with regular expressions;

6) Remove multiple spaces.

After preprocessing, the data set contained 13,784 pairs of sentences. The median query length in English is 8 words. The shortest requests consist of two words, for example: "play pop". The longest consists of 33 words. Total sentences include 11282 unique English words, each word occurs on average 10 times.

The median query length in Swedish is also 8 words. There are 14% more unique words than in English - 12852. The entire data set contains 119529 words and 115914 words in English and Swedish, respectively.

[1] `https://translate.yandex.com/developers`

# 2. Algorithm

## 2.1. Statistical analysis

In order to better understand the data structure and to develop intuition for the selection of models, a small statistical analysis of the data set was made during the work. Some models considered below rely on the word as a unit of information and consider sentences as unordered sets of words of the so-called. "bags of words". It was decided to compare individual words by a certain amount, reflecting the value of this word for the classification of sentences.

### 2.1.1. Method

To describe the method for calculating the usefulness of a word for classification, we introduce some notation. Let E contain $N_e$ unique words. $\{w_1, \dots, w_{N_e}\}$ are all possible unique words of the language $E$. Words are found in sentences, there are $D$ sentences. $\{s_1, \dots, s_D\}$. In addition, each sentence belongs to one or another class $k \in K = \{k1, \dots k_n\}$. Consider two indicators. Class membership indicator:

$$I_k(s,k) = \begin{cases} 1 & \text{if the sentence} s \text{belongs to the class} k, \\ 0 & \text{ if the sentence} s \text{does not belong to the class} k. \end{cases}$$

And the indicator of the occurrence of a word in the sentence:

$$I_s(s,w) = \begin{cases} 1 & \text{if the sentence} s \text{contains the word} w, \\ 0 & quad \text{if the sentence} s \text{does not contain the word} w. \end{cases}$$

Then we introduce the following notation: $c_{i,j}$ is the number of times that the word $w_i$ was encountered in the sentences of the class $k_j$:

$$c_{i,j} = \sum_{0<t<D} I_k(s_t, k_j) * I_s(s_t, w_i)$$

$p_{i,j}$ is the share of sentences with met which $k_j$ among all sentences with the word $w_i$:

$$p_{i,j} = \frac{c_{i,j}}{\sum_{0<t<D} I_s(s_t, w_i)}$$

$l_{i,j}$ is the fraction of sentences in which the word $w_i$ is found among the sentences with the label $k_j$:

$$l_{i,j} = \frac{c_{i,j}}{\sum_{0<t<D} I_k(s_t, k_j)}$$

Using the notation introduced, we write the expression to calculate the proposed statistics $U(w_i)$ reflecting the conditional utility of the word $w_i$ for the classification problem:

$$U(w_i) = \sum_{0<j<n} [(p_{i,j} - \frac{1}{n})^2 * l_{i,j}] * \frac{n}{n-1}$$

The value of $U(w_i)$ tends to zero as the distribution of the word $w_i$ approaches by grade to random.

$$p_{i,j} \to \frac{1}{n} \forall j \in \{1, \ldots, n\} \implies u_i \to 0$$

Moreover, the more degenerate the distribution of the word by class and the more class sentences contain this word, the greater will be the value of the metric. If the word $w_i$ is found exclusively in the sentences of the class $k_j$, and each sentence in the class $k_j$ contains this word, then the value of the metric will reach one. This will mean that this single word allows you to accurately indicate that all sentences containing it belong to a particular class $k_j$.

### 2.1.2. Analysis

By calculating the conditional utility for classification for each word in English and Swedish, we constructed a rating. The table shows the 20 ”most useful for classification in the framework of this task” words and their $U$ value. Some words that are direct translations of each other are highlighted in color. In addition, the table shows the values:

$$mplp = \max_{j=1\ldots n} p_{i,j} (\text{most popular label probability})$$

$$mpll = l_{i,j} | j = \arg\max_{j=1\ldots n} p_{i,j} (\text{most popular label load})$$

$$label = k_j | j = \arg\max_{j=1\ldots n} p_{i,j}$$

Рис. 2. Top-20 words for ”classification utility” in English and Swedish

| # | word | label | u | mplp | mpll | \| | # | word | label | u | mplp | mpll |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | add | AddToPlaylist | 0.686 | 0.996 | 0.809 | \| | 1 | lagg | AddToPlaylist | 0.58 | 0.995 | 0.683 |
| 2 | play | PlayMusic | 0.629 | 0.924 | 0.883 | \| | 2 | boka | BookRestaurant | 0.508 | 0.966 | 0.642 |
| 3 | playlist | AddToPlaylist | 0.464 | 0.931 | 0.64 | \| | 3 | spellista | AddToPlaylist | 0.441 | 0.942 | 0.592 |
| 4 | rate | RateBook | 0.436 | 0.997 | 0.512 | \| | 4 | spela | PlayMusic | 0.415 | 0.782 | 0.867 |
| 5 | weather | GetWeather | 0.376 | 0.999 | 0.44 | \| | 5 | till | AddToPlaylist | 0.388 | 0.775 | 0.829 |
| 6 | movie | SearchScreeningEvent | 0.36 | 0.917 | 0.514 | \| | 6 | bord | BookRestaurant | 0.306 | 0.994 | 0.361 |
| 7 | restaurant | BookRestaurant | 0.313 | 0.993 | 0.372 | \| | 7 | restaurang | BookRestaurant | 0.302 | 0.993 | 0.358 |
| 8 | book | BookRestaurant | 0.28 | 0.764 | 0.622 | \| | 8 | filmer | SearchScreeningEvent | 0.277 | 0.995 | 0.327 |
| 9 | be | GetWeather | 0.256 | 0.909 | 0.373 | \| | 9 | kommer | GetWeather | 0.27 | 0.915 | 0.388 |
| 10 | will | GetWeather | 0.248 | 0.924 | 0.347 | \| | 10 | det | GetWeather | 0.252 | 0.716 | 0.653 |
| 11 | points | RateBook | 0.244 | 0.998 | 0.286 | \| | 11 | poang | RateBook | 0.244 | 1.0 | 0.284 |
| 12 | forecast | GetWeather | 0.243 | 1.0 | 0.283 | \| | 12 | stjarnor | RateBook | 0.243 | 0.986 | 0.293 |
| 13 | out | RateBook | 0.24 | 0.915 | 0.345 | \| | 13 | vader | GetWeather | 0.239 | 0.998 | 0.28 |
| 14 | stars | RateBook | 0.234 | 0.976 | 0.288 | \| | 14 | betygsatt | RateBook | 0.213 | 1.0 | 0.249 |
| 15 | playing | SearchScreeningEvent | 0.225 | 0.978 | 0.276 | \| | 15 | min | AddToPlaylist | 0.209 | 0.766 | 0.459 |
| 16 | table | BookRestaurant | 0.223 | 0.992 | 0.265 | \| | 16 | musik | PlayMusic | 0.176 | 0.912 | 0.255 |
| 17 | my | AddToPlaylist | 0.222 | 0.741 | 0.527 | \| | 17 | film | SearchScreeningEvent | 0.161 | 0.909 | 0.235 |
| 18 | give | RateBook | 0.208 | 0.873 | 0.333 | \| | 18 | priser | RateBook | 0.161 | 0.992 | 0.192 |
| 19 | it | GetWeather | 0.207 | 0.896 | 0.311 | \| | 19 | bli | GetWeather | 0.146 | 0.997 | 0.172 |
| 20 | movies | SearchScreeningEvent | 0.201 | 0.992 | 0.239 | \| | 20 | biograf | SearchScreeningEvent | 0.143 | 0.997 | 0.168 |

First of all, according to the results in the table, we can say that the proposed metric really has meaning. The words in the table are similar to the words that are relevant to the classification. For example, the word ”play” has a high meaning for recognizing the label ”PlayMusic” (play music). Which is quite logical.

You may also notice that words that have the same meaning are not always in the same positions in different languages. For example, the word ”book” in English is lower than one of its translations ”boka” in Swedish. This is due to the fact that in English the word ”book” has several meanings and can mean both ”book” and ”book.” And therefore it has less utility for classification in these classes. A similar situation is observed with the word ”table” (table, table).

## 2.2. Models of sentence vectors

As mentioned earlier, language processing tasks imply the need to represent words and sentences in natural language in any form convenient for processing. As a rule, these are vectors. In our work, we used the representation of sentences as vectors, that is, the function $J_s$ was searched in the form:

$$J_s(s) = J_s(V_s(s)),$$

where $V_s$ is the transformation of the sentence on Swedish to vector.

$$V_s : S \to \mathbb{R}^\lambda$$

In order to study the transfer of knowledge of the model for the English language to another, you must first choose a good architecture that works in the same domain.

In our work, we used several developed methods for representing sentences as vectors in a certain space.

### 2.2.1. FastText - average

For the simplest method of constructing vectors of sentences, we relied on a ready-made model from Facebook - fastText [9]. This is an implementation of the CBOW algorithm for obtaining word vectors with some additions. To train the model, fastText uses not only context words, as in the classical [14] implementation, but also symbolic n-grams as input parameters. That is, all subwords of a certain length. The resulting word vector is obtained as the sum of vectors of such n-grams. This technique was described by the Facebook AI Research team in the article [2] and showed its effectiveness in the search for similar words and analogies. The availability of information about subwords when teaching a model allows using it to obtain word vectors for words that have not been previously encountered in the dictionary.

In the course of the work, we ourselves did not train the model, but used pre-trained fastText vectors published in the public domain. These data are distributed under the license CC BY-SA 3.0, you can find them at `https://fasttext.cc/docs/en/crawl-vectors.html`. When learning vectors, the CBOW architecture was used with n-grams of length 5, window size 5 and vector size $\lambda = 300$. Texts from Wikipedia and other open sources were used for training.

To obtain the vectors of sentences, we took the average of all the vectors of words obtained from the model. This approach does not take into account the order and mutual arrangement of words, but it works very quickly.

### 2.2.2. FastText - weighted average

When averaging all word vectors to get a sentence vector, the information contained in separate words is noisy. Many words are not useful for the classification of semantic load. I would like to distinguish such words and not use them for averaging.

We tested another method for obtaining fastText-based offer vectors. To obtain the vector of the sentence, we calculated the weighted average of all the words included in it. As a weight, we used the value of the word proposed by us in the previous chapter for the classification of $U(w)$:

$$V_s(s) = \frac{\sum_{w_i \in s} V_{fastText}(w_i) * U(w_i)}{\sum_{w_i \in s} U(w_i)}$$

With pre-calculated values $U(w_i) \forall i \in [1 \ldots N]$This approach also works quickly, but it allows to reduce the amount of noise in the vector and thereby raise the quality of the classification.

### 2.2.3. ELMo - average

As an alternative to fastText, we looked at another model to get offer vectors - ELMo (Embeddings from Language Models) [16]. This technique is based on the use of internal layers of a trained bidirectional language LSTM network for word prediction. On the first layer of the bidirectional network, word vectors obtained from the convolutional network on symbols are fed. Then two biLSTM layers follow. The resulting word vector is obtained as a linear combination of the output values on the previous three layers.

In our work, we, like in the case of fastText, did not train the model on our own, but used the pre-trained one. The data used in [4] [8] is available at `https://github.com/HIT-SCIR/ELMoForManyLangs#downloads`. This model was trained on a body of 20 million words, randomly typed from Wikipedia and other open sources. The dimension of the resulting vector word $\lambda = 1024$.

This model naturally uses the information about the location of words when calculating vectors, since it includes bidirectional LSTM modules. However, the computation takes significantly more time compared to fastText.

As in the case of fastText vectors, we took the average value of the vectors of all its words as a sentence vector.

### 2.2.4. Universal Sentence Encoder

USE [3] is the most modern of the three technologies reviewed, proposed in April 2018 by the Google research group. The construction of sentence vectors, considered in the article, is based on the use of a deep neural network to transform individual word vectors to a sentence vector. This network is trained on various tasks of language processing, which allows to achieve significant results in different areas. However, obtaining vectors requires significant computational costs.

In our work, we used a pre-trained model implemented using the TensorFlow [1] framework located at `https://tfhub.dev/google/universal-sent`
2. The model is licensed under CC BY 3.0. The dimension of the resulting vectors of sentences $\lambda = 512$.

To evaluate the quality of sentence vectors in the context of the problem of recognizing user intent, we trained a single-layer neural network to predict the class probabilities. Network error was considered using the negative log likelihood function. The Adam [13] method was used to change the weights. The accuracy of the classifier obtained was measured as a percentage of correctly predicted labels for the test sample using cross-validation.

## 2.3. Linear transformation

Suppose we have a model $J_e$, which predicts intent $K$ well according to the query vector in English. Then, to solve the problem of predicting intentions for the second language, it suffices to construct a transformation of the request vectors from the second language to English and then use $J_e$. Note that such a conversion will not be a translation, since the restoration of the proposal for its vector is almost impossible.

As the first test method to search for a transform, we chose a simple linear model. The function $L(V_s(s))$ was searched in the form $L(V_s(s)) =$

$V_s(s)A$. To find the transformation matrix A, a linear system of equations was solved on a training data subset.

$$V_s A = V_e,$$

where $V_s$ and $V_e$ are matrices of sentence vectors of dimension $N_{train}$ rows on $\lambda$ columns, $\lambda_l$ - the length of the sentence vector in the representation $V_l$, $N_{train}$ - the number of examples of the training sample

Since $N_{train}$, as a rule, is much larger than $\lambda_s$ then the system is overridden. By solution we mean a matrix A that minimizes the Euclidean distance between the columns in the right and left parts of the expression. To find the columns of matrix A, the least squares method was used.

$$A^{(i)} = (V_s^T V_s)^{-1} V_s^T V_e^{(i)}, i \in [1, \lambda_e]$$

In our experiment to select the predicted label we found the distance from the image of the vector $V_s(s_i)$ to all the vectors $V_e(e_j)$ from the test sample, except for the direct translation $s_i$, and took the class of the predicted class $e_j$, the distance to which vector was minimal:

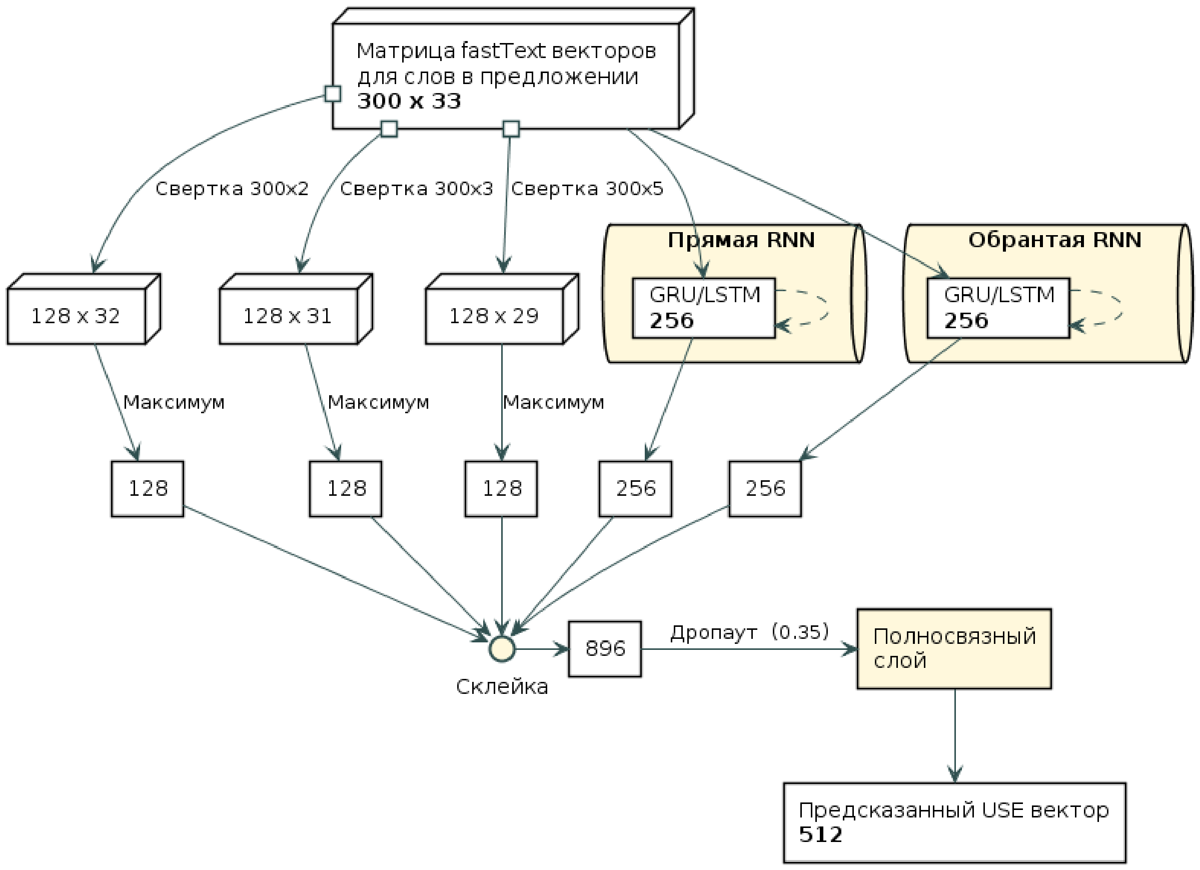
$$J_s(s_i) = J_e(\operatorname*{arg\,min}_{\substack{j=1\ldots N_{train} \\ j\neq i}}(\|L(V_s(s_i)) - V_e(e_j)\|^2))$$

## 2.4. Neural Network

Linear transformation is a very limited way of constructing a connection between the vectors of different languages. In addition, by averaging the fastText vector for Swedish, we lose some of the information. In addition to the linear transformation, we also tried to predict the USE sentence vector in English using a neural network with recurrent and convolutional layers. As input parameters, the network received a matrix of separate vectors for each Swedish word in the sentence. Thus, the algorithm had the opportunity to catch information about the relationship between words and from the mutual arrangement.

The overall network architecture was as follows. Convolution filters that capture two, three or five words were applied to the matrix of word vectors. In parallel, words were fed one at a time to the input of a recurrent neural network with LSTM modules in the forward and reverse order. The two vectors of the latent state of the last layer of the recurrent network were glued to the vectors of the maximum values for each filter. Then one full mesh layer followed. When training before a fully connected layer, 35

Рис. 3. The architecture of the neural network used in the work



Besides LSTM modules, we also tested a similar architecture with GRU modules in a recurrent network.

For training, we input a matrix from the fastText word vectors of the Swedish sentence to the network and select weights to minimize the square of the distance between the network output and the USE vector of the corresponding English sentence.

To predict the class label, we tried two methods.

1) As in the case of the linear transformation, the predicted class of the Swedish sentence was considered the class of the English sentence from the test sample, whose USE vector is closest to the network output.

2) The vector obtained by the recurrent network was fed to the linear classifier trained on the USE vectors. The label predicted by the classifier was considered the label of the Swedish proposal.

# 3. Results

To test the hypothesis that the quality of knowledge transfer strongly depends on the semantic similarity of languages, we conducted part of experiments not only for Swedish, but also for Finnish.

## 3.1. Offer vector models

This table shows the accuracy of classifying user requests according to classes, depending on the technology used to obtain offer vectors.

Таблица 2. Classification accuracy for linear models based on sentence vectors for different languages

| **Model** | **English** | **Swedish** | **Finnish** |
|---|---|---|---|
| FastText-avg | 91.9 % | 88.3 % | 84.4 % |
| FastText-uw-avg | 96.2 % | 94.2 % | 94.3 % |
| ELMo-avg | 97.7 % | 96.0 % | 95.3 % |
| USE | 96.8 % | - | - |

It can be noted that the proposed method of weighting fastText vectors before averaging (fastText-uw-avg) has significantly improved the quality of classification. The resulting supply vectors are almost equal in quality with ELMo-avg vectors.

## 3.2. Linear transformation

In the course of the work, 6 different pairs of representations of sentences in English and Swedish were tested. For each pair, we calculated a matrix for converting vectors from Swedish to English, and used it to predict the class of the Swedish sentence.

The table shows the average percentage of correctly predicted labels and the mean square error after cross-validation depending on the used sentence vectors for different languages.

Таблица 3. Accuracy of a model with linear transformation of sentence vectors

| **Swedish** | **English** | **Accuracy** | **Error** |
|---|---|---|---|
| FastText-avg | ELMo-avg | 86.2 | 0.00745 |
| FastText-avg | USE | 89.6 % | 0.00083 |
| hline FastText-uw-avg | ELMo-avg | 88.8 % | 0.00881 |
| FastText-uw-avg | USE | 91.2 % | 0.00095 |
| ELMo-avg | ELMo-avg | 89.1 % | 0.00682 |
| hline ELMo-avg | USE | 91.5 % | 0.00081 |

We also checked the two most interesting pairs for the Finnish language.

Таблица 4. Accuracy of a model with linear transformation fi-en

| **Finnish** | **English** | **Accuracy** | **Error** |
|---|---|---|---|
| hline FastText-avg | USE | 88.6 % | 0.00089 |
| FastText-uw-avg | USE | 91.8 % | 0.00093 |

## 3.3. Neural Network

The Swedish-language fastText vector architecture described above predicted USE for English. Two class prediction methods were tested. Using the ”nearest neighbor”, and using the USE classifier. The table shows the percentage of correctly predicted labels of the Swedish sentences and the mean square error between the predicted and real USE vectors.

Таблица 5. Accuracy of the model with a neural network for the Swedish language

| **Recurrent modules** | **Prediction method** | **Accuracy** | **Error** |
|---|---|---|---|
| GRU | Nearest Neighbor | 94.2 % | 0.00068 |
| GRU | USE Classifier | 96.4 % | 0.00068 |
| LSTM | Nearest Neighbor | 93.7 % | 0.00069 |
| LSTM | USE classifier | 96.1 % | 0.00069 |

Since the network with GRU modules worked slightly better for a couple of English-Swedish, when testing the Finnish language, we limited ourselves only to this architecture.

Таблица 6. Accuracy of a model with a neural network with GRU modules for the Finnish language

| **Prediction method** | **Accuracy** |
|---|---|
| Nearest neighbor | 94.0 % |
| USE classifier | 96.4 % |

The mean square deviation of the predicted USE vectors from the real values on the test sample was **0.00069**.

It can be noted that the resulting architecture predicts the label of the sentence on fastText vectors better than the previously considered linear classifier, which does not use knowledge of related English sentences and their USE vectors.

For comparison, we also tried to predict the class label by the USE vector based on the nearest neighbor label in the USE space of vectors without any transformations. On the entire data set, we obtained accuracy in **94.3 %**

## Corollary

In this paper, the task was to evaluate the effectiveness of transferring model experience from one language to another using the example of the prediction of intent. During the work, several models were built using this approach. The tested algorithms showed accuracy comparable to the accuracy of simple models working in English for Swedish and Finnish. Contrary to expectations, accuracy for the Finnish language was at the same level as for Swedish. This allows us to conclude about the effectiveness of the proposed methods and the possibility of their use in real-world problems for a wide range of languages.

It was noticed that GRU modules in the neural network work slightly better than LSTM. Probably simpler architectures will also achieve comparable results. It makes sense to continue the experiments to find the most simple model, sufficient for complete transfer of knowledge.

In addition, the paper proposed a method for evaluating the usefulness of words for classification in a specific problem. The results of this method were applied to improve the results of the classification using one of the algorithms.

# Conclusion

In this paper, several models of constructing vectors of sentences were considered and the features of various approaches were studied.

A method for ranking words in a sentence on utility for classification has been proposed and it has been shown that such a ranking can be used to improve some models for constructing a vector of sentences.

Two methods of knowledge transfer from one language to another were considered: using linear transformation and using a neural network with recurrent and convolutional layers.

Both techniques were tested in different configurations on different input data. The results were obtained and it was shown that the transfer of knowledge through the adaptation of word vectors by the neural network can increase the accuracy compared to a linear model that works in the same language.

# Список литературы

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, 2016. USENIX Association.

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

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[4] Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[5] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. *CoRR*, abs/1502.02367, 2015.

[6] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre

Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190, 2018.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[8] Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden, May 2017. Association for Computational Linguistics.

[9] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[11] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015.

[12] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning*

*Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints*, January 2013.

[15] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting Similarities among Languages for Machine Translation. *ArXiv e-prints*, September 2013.

[16] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[17] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[19] Yandex. Технологии - Машинный перевод, 2019. Дата обращения - 4 апреля 2019.