Human Language Technologies – The Baltic Perspective
A. Utka et al. (Eds.)

© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA200608

Intent Detection-Based Lithuanian Chatbot Created via Automatic DNN Hyper-Parameter Optimization

Jurgita KAPOČIŪTĖ-DZIKIENĖ [1]
JSC Tilde Information Technology, Vilnius, Lithuania
Faculty of Informatics, Vytautas Magnus University, Kaunas, Lithuania

Abstract. In this paper, we tackle an intent detection problem for the Lithuanian language with the real supervised data. Our main focus is on the enhancement of the Natural Language Understanding (NLU) module, responsible for the comprehension of user's questions. The NLU model is trained with a properly selected word vectorization type and Deep Neural Network (DNN) classifier. During our experiments, we have experimentally investigated fastText and BERT embeddings. Besides, we have automatically optimized different architectures and hyper-parameters of the following DNN approaches: Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM) and Convolutional Neural Network (CNN). The highest accuracy=~0.715 (~0.675 and ~0.625 over random and majority baselines, respectively) was achieved with the CNN classifier applied on a top of BERT embeddings. The detailed error analysis revealed that prediction accuracies degrade for the least covered intents and due to intent ambiguities; therefore, in the future, we are planning to make necessary adjustments to boost the intent detection accuracy for the Lithuanian language even more.

Keywords. NLU, Intent detection, LSTM, BiLSTM, CNN, hyper-parameter optimization, fastText and BERT embeddings, the Lithuanian language

1. Introduction

The modern society is not imaginable without conversational chatbots that are accurate, fast, tireless and available 24/7. Chatbots can successfully boost human work efficiency, but cannot replace them completely so far. However, the demand for chatbots is increasing exponentially, especially in the customer service space.

Starting from the very first keyword search-based chatbot ELIZA (asking itself instead of answering user's questions) [1], this NLP field has made tremendous progress. Nowadays, different types of chatbots exist: according to the communication channel (voice-enabled or text-based); knowledge domain (closed, general or open domain); provided service (interpersonal or intrapersonal); used learning methods (rule-based, retrieval-based or machine learning-based); and provided answers (extracted or generated answers).

The chatbot as the whole system can consist of several important modules: NLU (dealing with the machine reading comprehension), slot filling (searching for specific

[1] Corresponding Author: Jurgita Kapočiūtė-Dzikienė; JSC Tilde Information Technology, Naugarduko str. 100, LT-03160 Vilnius, Lithuania; E-mail: jurgita.k.dz@gmail.com.

pieces of information with respect to some named entity), dialog management (determining the context and guiding the dialog), even emotion detection (reacting to the user's sentiments accordingly), etc. Various chatbot features as an appearance or a friendly human-like behavior are important, however, the most important function remains the correct comprehension of user's questions and adequate responses to them. In this paper, we are solving the intent detection problem that classifies user utterances into predefined intent categories (related to chatbot answers).

Widely-known intelligent chatbots as Siri, Cortana, Alexa, Google assistant, etc., support only a small group of languages and Lithuanian is not among them. There was an attempt to solve an intent detection problem for morphologically complex languages, including Lithuanian [2]. This research describes experiments with fastText embeddings using the CNN and the Feed Forward Neural Network (FFNN) classifiers (architectures based on expert insights). Despite the significance of this research for the Lithuanian language, the intent detection model was trained and tested on the artificial data (i.e., English benchmark datasets translated into Lithuanian) and a rather small number of intents (varying from 2 up to 8).

In our research, we are solving the intent detection problem for the Lithuanian language by training and testing the NLU model on the real dataset of 41 intent. The contribution of this research is that we seek for the best intent detection model for the Lithuanian language by exploring: 1) two types of word embeddings; 2) three types of classifiers (CNN, LSTM, BiLSTM); 3) different Deep Neural Network architectures; 4) different hyper-parameters. DNN architectures and hyper-parameter values were tuned automatically in the parameter optimization process.

2. Related Work

If all outdated rule-based and keyword-based approaches are excluded, the remaining ones are mainly focused on the Information Retrieval (IR) and Machine Learning (ML) techniques. IR-based chatbots are typically used in cases when it is essential to find relevant information from huge data collections [3] or unstructured documents [4]. In our research, we experiment with the dataset containing question-answer pairs (where questions are grouped into categories representing related answers). This particular shape of the data allows us to focus on the supervised ML approaches [5].

The majority of the intent detection research is focused on the English and Chinese languages having enough resources to experiment and create accurate NLU models (a more detailed review on different intent detection methods can be found in [6]).

The research in [7] compares different chatbot platforms. Authors test the most popular NLU services on a large multi-domain dataset of 21 domains and 25K of utterances. The investigation reveals that *IBM Watson* significantly outperforms other platforms as *Dialogflow*, *MS LUIS* and *Rasa* that also demonstrate very good results. Three English benchmark datasets, i.e., *askUbuntu*, *chatbot* and *webApps* [8] were used in the experiments [9]. Authors introduce a sub-word semantic hashing technique to process input texts before classification. After vectorization, the following classifiers are explored: Ridge Classifier – RC, K-Nearest Neighbors, Multilayer Perceptron, Passive Aggressive – PA, Random Forest – RF, Linear Support Vector – LSV, Stochastic Gradient Descent, Nearest Centroid, Multinomial Naive Bayes, Bernoulli Naive Bayes, K-means. On the *askUbuntu* dataset, RC and LSV classifiers achieved the highest accuracy. PA and RF were the best on *chatbot* and *webApps*, respectively. Authors claim that with the determined classifiers and their sub-word semantic hashing technique, they were able to achieve state-of-the-art performance on these benchmark datasets. The experiments in [10] only confirm the fact that vectorization plays an important role in intent detection. Authors use the BiLSTM classifier with gloVe embeddings enriched with semantic lexicons (containing synonyms, and antonyms) and achieve superiority over the naïve one-hot vectorization on the *ATIS* and *Places* datasets from MS Cortana.

The problems often emerge when some question is out-of-scope the intent detection model is trained to cover. The supervised ML is also the right solution for this type of problems. The research in [11] solves an open intent discovery problem with the dataset of 25K of utterances in two stages: the first stage predicts if the utterance contains some intent and if it does, the second stage tries to predict it. Authors use BiLSTM with the Conditional Random Fields (CRF) method on a top of it and a self-attention mechanism to learn long distance dependencies. The offered approach outperforms state-of-the-art baselines. In the similar work [12], authors use a dataset containing queries that are out-of-scope (covering none of the existing intents) and in-the-scope (covering one of 150 available intents). A range of different approaches has been explored in this research: fastText classifier, BERT classifier, Support Vector Machine, CNN, DialogFlow, Rasa and Multi-Layer Perceptron. The research discovered the BERT classifier to perform well in-the-scope; unfortunately, all approaches struggled to identify out-of-scope queries. Our dataset contains queries only in-the-scope; therefore, we will focus on the closed domain intent detection problems only.

Intent detection problems are sometimes tackled together with the slot filling. Authors in [13] experimentally prove that the proposed joint BERT model outperforms BERT models for the intent detection and slot filling trained separately. Their proposed joint model is trained on the dataset of 72 slots and 8 intents and achieves significant improvement over the attention-based Recurrent Neural Network (RNN) models and slot-gated models. In similar research [14], authors use *SNIPS-NLU* and *ATIS* datasets and propose a Capsule-Based Neural Network model. The architecture consists of three types of capsules: WordCaps (to learn word representations), SlotCaps (to classify words by their slots) and IntentCaps (to determine the intent). The offered method outperforms other popular NN-based approaches. Authors in [15] present an attention-based encoder-decoder NN model for the joint intent detection and slot filling task. The method encodes sentences using the CNN-BiLSTM hybrid approach and decodes using the attention-based RNN with aligned inputs. The authors experimentally prove that the offered approach achieves better performance compared to the other popular DNN-based approaches. Since our dataset does not contain necessary annotations for the slot filling, we will focus on the intent detection problem only; however, the analysis of similar research especially encourages us to test BERT embeddings and DNN classifiers.

3. The Dataset

Our NLU module training and testing was done on the real Lithuanian data: i.e., dataset containing question-answer pairs about the company's Tilde product "Tildes Biuras" (questions about prices for different users, licenses, supported languages, available dictionaries, used technologies, etc.). The whole dataset contains 41 intents (chatbot outputs/answers), each covered by at least 5 instances (user inputs/questions). When solving the intent detection problem, the following Lithuanian language-dependent and spoken language-dependent features have to be considered. The Lithuanian language is morphologically complex, highly inflective, has a rich vocabulary and a rich word-derivation system. Besides, questions (to correspond the real conversation conditions) are formulated in the spoken Lithuanian language: they have relatively free word-order (that is typical for the Lithuanian language), contain stylistically irregular constructions (typical for the spoken language).

The dataset was split into training and testing sub-sets (statistics about these splits can be found in Table 1).

Table 1. Statistics about the training/testing dataset used in our experiments

Number of intents	**Number of questions**	**Number of words**	**Number of distinct words**	**Avg. number of words per question**
Training dataset				
41	365	1,857	699	5.09
Testing dataset				
41	144	751	435	5.22

4. Methodology

The nature of the dataset (presented in Section 3) makes it possible to use it with the supervised ML approaches [5]. A solving task is a typical classification problem and can be formally defined as follows:

Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of instances (user inputs/questions). Let $C = \{c_1, c_2, \ldots, c_m\}$ be a set of classes (chatbot outputs/answers/intents). Each d_i can be attached to only one c_j; besides $m > 2$ that restricts this task to a single-label multi-class classification problem. D is mapped to C according to the logic Γ ($\Gamma : D \rightarrow C$). The main goal is to train a classifier on the training dataset D^T that could find the best approximation of Γ and demonstrate as higher accuracy as possible on the testing dataset D^T.

The formulated NLU problem is considered as AI-hard problem (meaning that the software should be learned to behave as intelligent as a human). The state-of-the-art approaches focus on DNN classifiers applied on a top of neural word embeddings (see Section 2) and for this reason we have chosen to explore these approaches as well. The whole package should contain the correct selection of the vectorization type, classifier, its architecture and hyper-parameter values. Thus, we have investigated:

- FastText[2] [16] and BERT[3] [17] word embeddings. FastText embeddings (offered by Facebook AI) are suitable for the non-normative texts: they are composed of the n-gram vectors; therefore, embeddings of misspelled words are close to their correct equivalents. BERT embeddings (offered by Google) are robust to disambiguation problems (because they have mechanisms to represent homonyms with different vectors depending on their context).
- LSTM [18], BiLSTM [19], CNN [20] (with the 1D convolution adjusted for the text classification [21]) classifiers. Both LSTM and BiLSTM are adjusted to process the sequential data and to overcome limitations of the simple RNN suffering from the short memory problem (due to a vanishing gradient). LSTM

[2] The Lithuanian fastText embeddings (offered by Facebook's AI Research Lab) for our experiments were downloaded from https://fasttext.cc/docs/en/crawl-vectors.html.

[3] We have used the Google's BERT service with the base multilingual cased 12-layer, 768-hidden, 12-heads model for 104 languages (including Lithuanian) downloaded from https://github.com/hanxiao/bert-as-service.

allows the data stream only forward (i.e., from-the-past-to-the-future), whereas BiLSTM process the text in both directions (i.e., forward and backward) and can acquire how succeeding words impact the current time moment. The CNN method is adjusted to seek for the influential text patterns (sequences of words, word n-grams) the most relevant to the intent.

- DNN architectures, having different numbers of hidden layers (i.e., series of convolutional layers in CNN; simple or stacked LSTM and BiLSTM).
- Discrete and real DNN hyper-parameter values: numbers of neurons (100, 200, 300, 400); dropouts (values from an interval [0,1]); recurrent dropouts ([0, 1]); activation functions (*relu*, *softmax*, *tanh*, *elu*, *selu*); optimizers (*Adam*, *SGD*, *RMSprop*, *Adagrad*, *Adadelta*, *Adamax*, *Nadam*); batch sizes (32, 64) and numbers of epochs (20, 30, 40, 50).

All this results in a huge number of options and expert knowledge is not always capable of selecting the most accurate one. The tuning of DNN architectures and hyper-parameters was performed automatically with the parameter optimization library *Hyperas*[4] and two optimization algorithms (the optimization process took 100 iterations):

- *tpe.suggest* (Tree-structured Parzen Estimator) [22], which organizes hyper-parameters into a tree-like space. This Bayesian modelling approach decides which set of hyper-parameters should be tried in the next iteration based on the distribution of previous results.
- *random.suggest* performs random search over a set of hyper-parameters.

All methods were implemented in Python using *Tensorflow* and *Keras*[5].

5. Experiments and Results

All experiments described in Section 4 were carried out with the dataset described in Section 3. The training dataset (see Table 1) was shuffled and used for the DNN hyper-parameter optimization: 80 % was used for training and the rest 20 % for validation. The best determined model (giving the highest accuracy on the validation dataset) was evaluated with the testing dataset. The accuracy (in eq. 1, where $N_{correct}$ and N_{all} represents instances with the correctly predicted intents and all tested instances, respectively) was used as the evaluation metric. The highest achieved accuracies with different DNN methods and vectorization types are presented in Table 2.

$$accuracy = \frac{N_{correct}}{N_{all}} \tag{1}$$

The approach is considered reasonable if the accuracy is above $random = \sum_j P^2(c_j)$ (where c_j is a probability of an intent) and majority=$max(P(c_j))$ baselines in our case, equal to ~0.04 and ~0.09, respectively. The McNemar test [23] with α=0.05 was used to determine if the differences between the obtained results are statistically significant.

[4] More about *Hyperas* is in https://github.com/maxpumperla/hyperas.
[5] https://www.tensorflow.org/ and https://keras.io/.

The architecture of the best determined approach which happened to be the CNN classifier with BERT embeddings is presented in Figure 1 (plotted with the *plot_model* utility function in *Keras*). Since not all determined optimal hyper-parameters can be plotted in the figure, we report them here: *selu* activation function after Conv1D layer; *softmax* activation function after dense; *batch_size*=64; *epochs*=20; *optimizer*=Nadam; *dropout rate*=0.467.

Table 2. The highest achieved accuracies (after hyper-parameter optimization) with different DNN classifiers, vectorization types, and parameter optimization strategies. The best result is in bold, the underlined results determine that they do not differ significantly from the very best

	fastText embeddings		
	LSTM	**BiLSTM**	**CNN**
tpe.suggest	0.243	0.563	<u>0.639</u>
random.suggest	0.278	0.556	<u>0.646</u>
	BERT embeddings		
	LSTM	**BiLSTM**	**CNN**
tpe.suggest	<u>0.681</u>	<u>0.653</u>	<u>**0.715**</u>
random.suggest	<u>0.681</u>	<u>0.694</u>	<u>0.708</u>

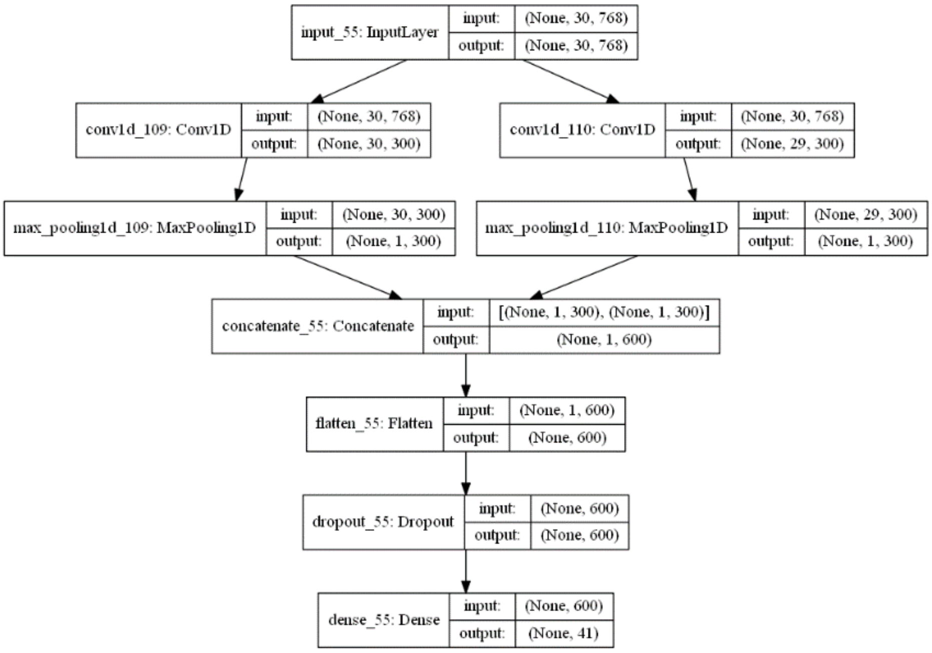

Figure 1. The best determined intent detection classifiers and its architecture: CNN with BERT embeddings

As presented in Table 2, all results are reasonable, because are above random and majority baselines. The highest accuracy=~0.715 is achieved with the BERT embeddings and the CNN classifier after parameter optimization (the best architecture is presented in Figure 1 and the best hyper-parameters are presented in Section 5). The superiority of the CNN classifier over the other approaches as LSTM and BiLSTM reveals that for this intent detection problem, to detect patterns (as word n-grams) is more important than to cope with the sequential nature of the input.

The *tpe.suggest* strategy allowed to optimize parameters even better compared to the *random.suggest*. However, there is no big difference how these parameters were optimized, the most important is that the accurate model has been found.

To see how significantly results differ from the best achieved (*accuracy*=~0.715) with BERT + CNN + *tpe.suggest,* we have performed the McNemar evaluation (calculated p values are in Table 3).

Table 3. The p values indicating how much the results are statistically significant between the best result with BERT + CNN + *tpe.suggest* strategy and other approaches

	fastText embeddings		
	LSTM	**BiLSTM**	**CNN**
tpe.suggest	4.49E-08	0.039	0.300
random.suggest	2.62E-07	0.032	0.348
	BERT embeddings		
	LSTM	**BiLSTM**	**CNN**
tpe.suggest	0.668	0.402	-
random.suggest	0.668	0.828	1.000

We have evaluated the accuracy for each intent separately (see the statistics in $x_y \rightarrow z$, where x, y, and z are number of testing instances, number of intents having x instances and the accuracy, respectively): $2_{10} \rightarrow 1.00$, $2_7 \rightarrow 0.50$, $2_7 \rightarrow 0.00$, $3_1 \rightarrow 0.67$, $3_1 \rightarrow 0.00$, $4_1 \rightarrow 1.00$, $4_2 \rightarrow 0.75$, $4_1 \rightarrow 0.50$, $5_1 \rightarrow 1.00$, $5_2 \rightarrow 0.80$, $5_1 \rightarrow 0.40$, $5_1 \rightarrow 0.40$, $6_2 \rightarrow 1.00$, $6_2 \rightarrow 0.83$, $6_1 \rightarrow 0.33$, $10_1 \rightarrow 0.90$, $14_1 \rightarrow 1.00$. Our solving intent detection problem is very challenging (41 intents, some of them are covered with only a few questions). The error analysis revealed that intents covered by more instances are better predicted. Besides, some intents are ambiguous (leading to very similar answers) and can be aggregated. Still, the accuracy of the NLU model is acceptable; we are planning to increase it even more by filling the training dataset with the new instances (especially by adding more instances to the least covered intents) and uptraining the new version of the model with the parameters already determined in this research.

6. Conclusions

This paper presented the intent detection problem solving for the Lithuanian language. This NLU problem for the Lithuanian language is tackled for the first time by using the real data and by comparing a wide variety of different approaches. We performed the automatic parameter optimization with three classifiers (LSTM, BiLSTM, CNN), two types of word embeddings (fastText, BERT), different DNN architectures (deeper and shallower) and various DNN hyper-parameter values.

The tackled task was especially challenging due to the following reasons: 1) the dataset contains many intents (41); 2) each intent was covered by only a few instances (~8.9 instances of which only ~7.1 are used for training and ~1.8 for validation). The experimental investigation revealed the superiority of the CNN classifier with BERT embeddings over other approaches; the best approach achieved ~0.715 of the accuracy. The error analysis revealed that intents covered by more instances are more reliable and recognized better. It allows us to assume that after adding more instances to the least covered intents and retraining the model, the accuracy will boost even more. In the future research, we are planning to expand the number of intents and instances.

Acknowledgements

The project *Bot Cloud* (No. J05-LVPA-K-03) was funded from EU funds under the measure *General Science and Business Projects* and *1st PRIORITY. Promotion of research, experimental development and innovation*.

References

[1] Weizenbaum, J. ELIZA–a computer program for the study of natural language communication between man and machine. Communications of the ACM 1996; 9:36–45.

[2] Balodis K, Deksne D. FastText-Based Intent Detection for Inflected Languages. Information 2019;10(5): 161.

[3] Lommatzsch A, Katins J. An Information Retrieval-based Approach for Building Intuitive Chatbots for Large Knowledge Bases. Proceedings of the LWDA conference 2019; 343–352.

[4] Zhao Y, Duan N, Bao J, Chen P, Zhou M, Li Z, Zhou J. DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2016;516–525.

[5] Sebastiani F. Machine Learning in Automated Text Categorization. ACM Computing Surveys 2002;34: 1–47.

[6] Liu J, Li Y, Lin M. Review of Intent Detection Methods in the Human-Machine Dialogue System. Journal of Physics: Conference Series 2019; 1267 012059.

[7] Liu X, Eshghi A, Swietojanski P, Rieser V. Benchmarking Natural Language Understanding Services for building Conversational Agents. CoRR 2019;abs/1903.05566.

[8] Braun D, Hernandez-Mendez A, Matthes F, Langen M. Evaluating Natural Language Understanding Services for Conversational Question Answering Systems. Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue 2017;174–185.

[9] Shridhar K, Sahu A, Dash A, Alonso P, Pihlgren G, Pondeknath V, Simistira F, Liwicki M. Subword Semantic Hashing for Intent Classification on Small Datasets. CoRR 2018; abs/1810.07150.

[10] Kim J-K, Tur G, Celikyilmaz A, Cao B, Wang Y-Y. Intent Detection using Semantically Enriched Word Embeddings. IEEE Spoken Language Technology Workshop 2016 Sep; 414–419.

[11] Vedula N, Lipka N, Maneriker P, Parthasarathy S. Towards Open Intent Discovery for Conversational Text. CoRR 2019;abs/1904.08524.

[12] Larson S, Mahendran A, Peper, JJ, Clarke Ch, Lee A, Hill P, Kummerfeld JK, Leach K, Laurenzano MA, Tang L, Mars J. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. Proc. of the 2019 Conference on Empirical Methods in NLP and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) 2019 Nov; 1311–1316.

[13] Chen Q, Zhuo Z, Wang W. BERT for Joint Intent Classification and Slot Filling. CoRR 2019; abs/1902.10909.

[14] Zhang Ch, Li Y, Du N, Fan W, Yu PS. Joint Slot Filling and Intent Detection via Capsule Neural Networks. CoRR 2018;abs/1812.09471.

[15] Wang Y, Tang L, He T. Attention-Based CNN-LSTM Networks for Joint Intent Detection and Slot Filling. Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data 2018;LNAI;11221:250–261.

[16] Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics 2017;5:135–146.

[17] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR 2018; abs/1810.04805.

[18] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation 1997;9(8):1735–1780.

[19] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks 2005;18(5-6):602–610.

[20] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 1998;2278–2324.

[21] Kim Y. Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014;1746–1751.

[22] Bergstra J, Bardenet R, Bengio Y, Kégl B. Algorithms for Hyper-Parameter Optimization. Advances in Neural Information Processing Systems 2011;24:2546–2554.

[23] McNemar Q. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. Psychometrika 1947;12(2):153–157.