Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2019”

Moscow, May 29—June 1, 2019

AN APPROACH TO CUSTOMIZATION OF PRE-TRAINED NEURAL NETWORK LANGUAGE MODEL TO SPECIFIC DOMAIN

Dudarin P. V. (p.dudarin@ulstu.ru),
Tronin V. G. (v.tronin@ulstu.ru),
Svyatov K. V. (k.svyatov@ulstu.ru)
Ulyanovsk State Technical University, Ulyanovsk, Russia

Nowadays the majority of tasks in NLP field are solved by means of neural network language models. These models already have shown state-of-the-art results in classification, translation, named entity recognition and so on. Pre-trained models are accessible in the internet, but the real life problem’s domain could differ from the origin domain which the network was learned. In this paper an approach to vocabulary expansion for neural network language model by means of hierarchical clustering is presented. This technique allows to adopt pre-trained language model to a different domain. In the experimental part the proposed approach is demonstrated on specific domain of textual artifacts of software development process. This field is actively studied this days due the expensiveness of the process and its impact on the modern world and society.

Key words: NLP, language model, neural network, RNN, ULMFiT, transfer learning, clustering, fuzzy graph clustering, word-to-vec

ПОДХОД К АДАПТАЦИИ ПРЕДОБУЧЕННОЙ ЛИНГВИСТИЧЕСКОЙ МОДЕЛИ НА БАЗЕ НЕЙРОННОЙ СЕТИ К СПЕЦИФИЧЕСКОЙ ПРЕДМЕТНОЙ ОБЛАСТИ

Дударин П. В. (p.dudarin@ulstu.ru),
Тронин В. Г. (v.tronin@ulstu.ru),
Святов К. В. (k.svyatov@ulstu.ru)
Ульяновский Государственный Технический Университет, Ульяновск, Россия

В современном мире большинство задач обработки текстов (NLP) решаются при помощи лингвистических моделей на базе нейронных сетей. Эти модели уже показатели выдающиеся результаты в классификации, машинном переводе, извлечении именованных сущностей и многих других. Предобученные модели доступны в сети интернет, но в практических задачах связанных с конкретными предметными областями используемый словарь может сильно отличаться от словаря на котором обучалась нейронная сеть. В данной работе представлен подход к адаптации предобученной лингвистической модели на базе нейронной сети к специфической предметной области. Помимо собственно алгоритма, представлен эксперимент демонстрирующий применимость предложенного подхода на примере предметной области процесса разработки программного обеспечения. Эта предметная область активно изучается ввиду высокой стоимости данного процесса и широкого влияния на современный мир и общество.

Ключевые слова: NLP, лингвистические модели на базе нейронной сети, нейронные сети, RNN, ULMFiT, перенос знаний, кластеризация нечетких графов, word-to-vec

1. Introduction

Code production is a complex and expensive process. Many resources are spent to get instruments of monitoring, control and prediction software development results. Many papers are dedicated to this theme for example in [15] an approach to project architecture analysis is proposed. Besides the code itself there are a lot of information produced during the software development process. For example, tasks are tracked in a task tracking system, where each issue could be commented and discussed by team members. During the code review phase the commit content is discussed by developers. All this information is textual, thus NLP methods are required to analyze it and extract knowledge about the effectiveness of communication among team members, about emotional condition of the team, specific relations between

colleagues. This knowledge could be used to monitor software development process, predict quality and timing, reveal conflicts on the early stages.

Traditional methods in information retrieval [13] work well with large texts and text corpuses. But the most information generated during software process is presented as short sentences and short dialogues. Not only in software processing but also in many other domains, the short text processing is becoming a new trend [6]. Since the short text has rarely external information, it is more challenging than document [18]. To coupe with this task different clustering techniques are used [3], [20]. Each clustering procedure needs a similarity measure, like the one used in Serelex system [16] published in 2013. In 2015 word2vec as the most used technique to obtain this measure in NLP tasks was introduced [14].

Although the word embedding approach has shown good efficiency that is shown in [2], lately an approach of construction neural network language models get a leading position in NLP benchmarks [19], almost every state-of-the-art results are obtaining by means of neural networks. But the process of neural network learning is quite long and computationally expensive.

Besides there are a lot of task in specific domains where there is no opportunity to train special neural network. In this case the idea of transfer learning [10] looks very promising. Authors of ULMFiT propose using their universal architecture to train language model and then to tune them for specific NLP tasks. But in ULMFit the tokens list is limited, authors recommend using up to 60,000 tokens. And as long as different word forms are treated as different tokens, ULMFiTs vocabulary is even more limited. On the contrary, modern word embedding models [11] have 250–400 thousand of lemmas. Word embedding technique being combined with thesaurus could demonstrate even higher performance[12]. In case of Russian language with its huge possible word forms language model approach allows to construct general purpose neural networks like casual phases generator only. And does not allow to include specific terms, neologism, swear words, rare used words and so on. In ELMo [4] and BERT [5] words are split into parts and then fed to neural network. But these models take a lot of calculation resources and could be afforded by huge corporations like Google. There are some multilingual pre-trained ELMo [9] and BERT models. But as for now they demonstrate very poor performance for Russian language. For example 'happy birthday to' really common phrase without double meaning could not be continued correctly by available models.

In this paper an approach to customization of pre-trained neural network language model to specific domain is proposed. This technique allows to process word outside the tokens list and thus to get benefits from transfer learning.

The rest of this paper is organized as follows. In **section 2** the detailed technique description is presented. **Section 3** shows an experimental results. And section 4 concludes the paper.

2. Language Model Customization

General idea of proposed approach is to add an extra layer of words pre-processing before the neural network language model (**Figure 1**).

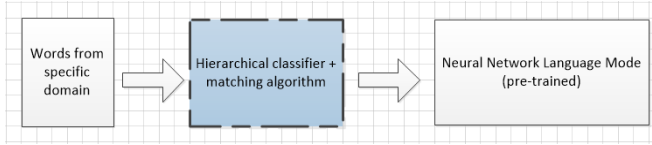

Figure 1: Additional pre-processing layer

This layer consists of two part: hierarchical classifier that groups words from neural network vocabulary and matching algorithm that matches new words to linear combination of words from vocabulary $new\_word = weight_1 * word_1 + ... + weight_N * word_N$.

2.1. Tokens Hierarchical Clustering

The first layer of neural network language model is an embedding layer which transforms one-hot encoded vectors into n-dimensional vectors of the embedding vectors space. Each coordinate of one-hot vector references to a word in a vocabulary of language model.

Lets define W_{lm}—a set of words included in tokens list of neural network. The task is to organize words from tokens list into a tree, where leaf nodes contain single word $w_i \in W_{lm}$, and other nodes are clusters that include all the words below in the hierarchy $w_{kj} \in C_k \subset W_{lm}$. $|W_{lm}| = N$.

This task could be completed by performing procedure which is a hierarchical modification [8] of $\epsilon - clustering$ [1], [17]. This procedure needs to be provided with a similarity measure for objects, let denote it as μ. There are a lot of pre-trained word embedding models for each language. This model provide a vector for each word and than the Euclidean or Manhattan distance could be calculated. In this paper the '*ruwikiruscorpora_upos_skipgram*_300_2_2019'[1] model was used.

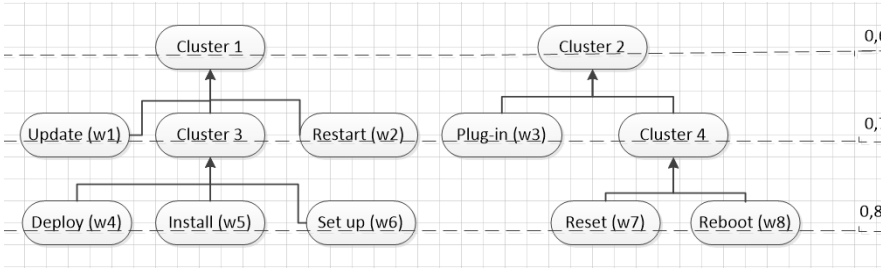

Figure 2: Hierarchical clustering sub-tree sample

One of the main advantages of graph based approach is its ability to be interpreted by human. The classifier could be easily modified by experts to add information domain specifics [7]. At least all the words that are not from domain vocabulary

[1] The model was downloaded from open resource https://rusvectores.org/ru/models.

could be cut of the classifier. On the **Figure 2** a part of sample classifier is shown. This sub-tree consist of two main branches dedicated to software and hardware installation process. Each level has a number (ϵ) that indicates the step of hierarchical clustering procedure when these level was obtained and it means that all the branches on this level has mutual similarity less than ϵ.

Thus a hierarchical classifier wit additional layer information could be obtained.

2.2. Specific Domain Words Matching

The task of the matching step is to construct vectors for words from specific domain in order they could be processed by pre-trained neural network. These vectors should have N components, where N equals to amount of inputs of neural network $N = |W_{lm}|$.

For each word w there are two possible cases. The word is already included into language model tokens list $w = w_i \in W_{lm}$ and in this case corresponding vector $v = (0, 0, ..., 1, 0, ..., 0)$, where component with 1 has i index. Another case when word $w \notin W_{lm}$. In this case there are some possible strategies to get a vector form. The first one, and the most evident, is to replace a given word with the most similar one according to similarity measure μ. Means, to choose $i, \mu(w, w_i) = max(w, w_j)\, \forall j \in [1, N]$. This strategy does not require any classifier, but it is not efficient when there are some equidistant words in the tokens list, especially when they are significantly differ in their semantic meaning. In order to have an alternative way of matching, in the experimental part the first strategy also included.

In general case proposed technique is following:

1. If $max(w, w_j) = \mu(w, w_i) > 0,9$[2] $\forall j \in [1, N]$ then $v = (0, 0, ..., 1, 0, ..., 0)$, with 1 on the i-th place.
2. Start with $\epsilon = 0,9$ and find all the words $W_{nn} = \{w_j | \mu(w, w_j) >= \epsilon, j \in [1, N]\}$. If $|W_{nn}| = 0$ then set $\epsilon = \epsilon - \delta\epsilon$. In this paper $\delta\epsilon = 0,05$, according to hierarchical clustering procedure specifics.
3. Get all the clusters $C_{nn} = \{c_j | \exists i\, w_i \in W_{nn}\}$ i.e. all the parent nodes in classifier for leaf nodes in W_{nn}.
4. Start with layer $l = 0,9$ and get all nodes from this layer $L_l = \{c_j | c_j \in C_{nn} \& layer(c_j) = l\}$. If $|L_l| > 2$[3] then change $l = l + \delta_l$ and move to the previous step. In this paper δ_l has been chosen as 0,05, according to hierarchical clustering procedure specifics.
5. For each node (cluster) define a weight according the distance to the cluster center. $weight = \mu(w, cluster\, center_i) / |\sum_j \in_{Ll} \mu(w, cluster\, center_j)|$
6. For each child node define weight the same as at the previous step and multiply to parent's weight $weight = parent\, weight * children\, weight$.
7. Stop when all the leaf nodes get weights. All the other weights are set to 0 $weight_i = 0\, \forall i \notin W_{nn}$ As a result $v = (weight_1, weight_2, weight_3, ..., weight_N)$

[2] The value $\epsilon = 0,9$ was obtained by experimental way and could differ from model to model.

[3] this threshold is heuristic and need to be surveyed more thoroughly in future studies

This algorithm is illustrated on Figure 3. Firstly the similarity of word 'mount' to other words is calculated. The most similar words 'install', 'set up' and 'plug-in' were detected. Then layer by layer from the bottom to the top parent nodes are detected, until only 2 nodes left. Next, top-to-bottom process starts. Based on the distance to the cluster centers (0.8 and 0.71), node weights are calculated 0.53 and 0.47 respectively. And finally, weights for children nodes of 'cluster 3' are calculated. Thus a vector for word 'mount' will be (0, 0, 0.53, 0, 0.24, 0.23, 0, 0, ...).

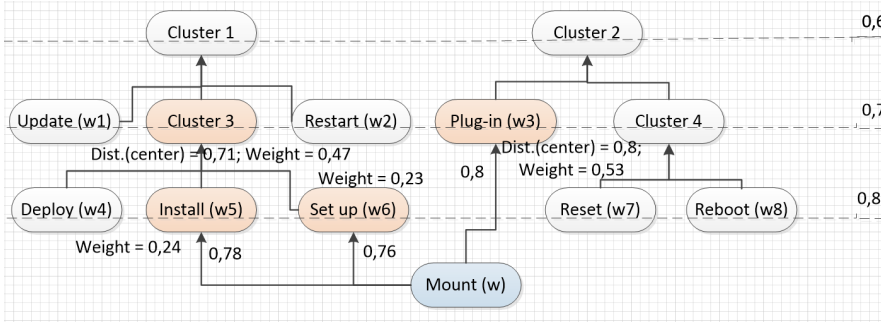

Figure 3: Matching process sample for word 'mount'

Thus each word from the domain is converted into vectors in N-dimensional vector space.

Besides even words that are present in language model token list could be rematched to another words or set of words. This could be useful in case of word's meaning is changed significantly in the domain. For example: 'mount', 'branch', 'bug' in software development domain.

3. Experiment results

For experimental purposes pre-trained neural network language model for Russian language with architecture ULMFiT has been chosen.[4] This model has been trained on news portal (lenta.ru) and has perplexity 36,23.

All the most popular neural network language models take as an input sequence of words, to be more specific - sequence of words indexes in tokens list. This make difficult to use custom input vectors with pre-trained neural network. In this paper hard code solution was used: the 'fastai' library has been modified to change not used input components into hard coded vectors. More thorough and wide experiments will need to make changes into the core of neural network framework where the embedding function is located.

To show the technique some common phrases from developers chats is used (all the experiments were with Russian words, but translation is provided next to the phrases):

[4] https://github.com/ppleskov/Russian-Language-Model

1. ‘кто может **смонтировать** новый жесткий диск?’ (translation: ’who can **mount** a new hard drive?’)
2. ‘тут есть **баг** и тебе нужно исправить его ’ (translation: ’this part has a **bug** you need to fix it’)
3. ‘этот класс не может быть унаследован от этого **интерфейса**’ (translation: ’this class could not be inherited from this **interface**’)

The chosen language model does include ’mount’ in common meaning and does not include words ‘bug’, ’interface’ in its tokens list. The aim is to be able to proceed this sentences with pre-trained neural network.

The first step is to construct a hierarchical classifier. The input layer of the current network has 60,000 neurons. The resulting hierarchy has about 80,000 nodes, 60 levels. The part of hierarchy is shown on **Figure 2**.

The second step is to construct vectors for words that are absent in tokens list. Word ‘смонтировать’ (’mount’) is related to words ‘инициировать’,’установить’ and ‘подключить’ (’install’, ’set up’ and ’plug-in’). This case is shown on **Figure 3**. For the other two words:

1. ‘баг’ (’bug’): ‘неисправность’, ’ошибка’, ‘недочет’ (’failure’, ’error’, ’lack’)
2. ‘интерфейс’ (’interface’): ‘структура’,’правило’,’протокол’ (’structure’, ’rule’, ’protocol’)

Then the sentences could be processed by neural network language model. The first 3–5 generated words has been taken as an output result:

1. Input: ’who can **mount** a new hard drive?’. Output of 15 words contains: ‘сервер’, ‘процессор’, (’server’, ‘processor core’)
2. Input: ’this part has a **bug** you need to fix it’. Output of 15 words contains: ‘приложение’, ‘исправление’ (‘application’, ’patch’).
3. Input: ’this class could not be inherited from this **interface**’. Output of 15 words contains: ‘протокол’, ‘нарушение’ (’protocol’, ‘violation’)

Get some experiments with NN and compare results of two ways. Access perplexity of the resulting NN.

The results below were generated when one the most similar word has been used instead of vector calculation.

1. Input: ’who can **mount** a new hard drive?’. Output of 15 words contains: ‘монтаж’, ‘ремонт’ (‘installation’,’repair’)
2. Input: ’this part has a **bug** you need to fix it’. Output of 15 words does not contains any words related to software domain.
3. Input: ’this class could not be inherited from this **interface**’. Output of 15 words contains: ‘родственники’, ‘матери’ (‘relatives’, ’mothers’)

Neural network output in the first case uses the common word meanings and produces wrong context. In the second and third cases some words were just ignored and the context produced was based on insignificant word.

4. Conclusion

In this paper an attemt to apply transfer learning technique to special domains was made. The proposed approach allows to use not learned words with pre-trained neural network language model. It is important in domains with insufficient amount of texts to train custom language model or when the calculation resources are limited. Also this technique could be used to prototype and check ideas (hypothesis) before starting to teach custom language model.

The results have shown an application capability of proposed approach. But more thorough and wide experiments need to be done. These experiments will need to make changes into the core of neural network framework where the embedding function is located. This will allow to compute perplexity measure of proposed approach and comparison with the other approaches. Further studies will involve comparison of different neural network architectures within proposed approach, searchin a way of fine tuning the language model and comparison of effectiveness in different NLP benchmarks. Besides it is important to develop extension to existing neural network frameworks to support not only a custom head but custom tails also.

5. Acknowledgements

The reported study was funded by RFBR and the government of Ulyanovsk region according to the research project № 18-47-732005 and by RFBR and the government of Ulyanovsk region according to the research project № 18-47-732004.

References

1. *Rosenfeld A,:* Fuzzy graphs. Fuzzy Sets and Their Applications to Cognitive and Decision Processes. Academic Press, New York. pp. 77–95. (1975).
2. *Arefyev, N. et al.:* How much does a word weigh? Weighting word embeddings for word sense induction. CoRR. abs/1805.09209, (2018).
3. *Avendaño, D. P. et al.:* Clustering abstracts of scientific texts using the transition point technique. In: Computational linguistics and intelligent text processing, 7th international conference, cicling 2006, mexico city, mexico, february 19-25, 2006, proceedings. pp. 536–546 (2006).
4. *Che, W. et al.:* 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. pp. 55–64 Association for Computational Linguistics, Brussels, Belgium (2018).
5. *Devlin, J. et al.:* BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. (2018).
6. *Dudarin, P. et al.:* Methodology and the algorithm for clustering economic analytics object. Automation of Control Processes. In: Automation of Control Processes, Vol. 47, № 1. pp. 591–604 RGGU, Ulyanovsk, Russia (2017).
7. *Dudarin, P. et al.:* An approach to feature space construction from clustering feature tree. In: Kuznetsov, S. O. et al. (eds.) Artificial intelligence. pp. 176–189 Springer International Publishing, Cham (2018).

8. *Dudarin, P. V., Yarushkina, N. G.:* An approach to fuzzy hierarchical clustering of short text fragments based on fuzzy graph clustering. In: Abraham, A. et al. (eds.) Proceedings of the second international scientific conference "intelligent information technologies for industry" (iiti'17). pp. 295–304 Springer International Publishing, Cham (2018).
9. *Fares, M. et al.:* Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In: Proceedings of the 21st nordic conference on computational linguistics. pp. 271–276 Association for Computational Linguistics, Gothenburg, Sweden (2017).
10. *Howard, J., Ruder, S.:* Universal language model fine-tuning for text classification. In: Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers). pp. 328–339 Association for Computational Linguistics, Melbourne, Australia (2018).
11. *Kutuzov, A., Kuzmenko, E.:* WebVectors: A toolkit for building web interfaces for vector semantic models. In: Ignatov, D. I. et al. (eds.) Analysis of images, social networks and texts: 5th international conference, aist 2016, yekaterinburg, russia, april 7–9, 2016, revised selected papers. pp. 155–161 Springer International Publishing, Cham (2017).
12. *Loukachevitch, N., Parkhomenko, E.:* Recognition of multiword expressions using word embeddings. In: Kuznetsov, S. O. et al. (eds.) Artificial intelligence. pp. 112–124 Springer International Publishing, Cham (2018).
13. *Manning, C. D. et al.:* Introduction to information retrieval. Cambridge University Press, New York, NY, USA (2008).
14. *Mikolov, T. et al.:* Efficient estimation of word representations in vector space. CoRR. abs/1301.3781, (2013).
15. *Nadezhda, Y. et al.:* An approach to similar software projects searching and architecture analysis based on artificial intelligence methods. In: Abraham, A. et al. (eds.) Proceedings of the third international scientific conference "intelligent information technologies for industry" (iiti'18). pp. 341–352 Springer International Publishing, Cham (2019).
16. *Panchenko, A. et al.:* Serelex: Search and visualization of semantically related words. In: Serdyukov, P. et al. (eds.) Advances in information retrieval. pp. 837–840 Springer Berlin Heidelberg, Berlin, Heidelberg (2013).
17. *Raymond, T. Y., Bang, S.:* Fuzzy relation, fuzzy graphs and their applications to clustering analysis. Fuzzy Sets and their Applications to Cognitive and Decision Pro-cesses. Academic Press. Pages 125-149. (1975).
18. *Tang, J. et al.:* Enriching short text representation in microblog for clustering. Frontiers of Computer Science. 6, 1, 88–101 (2012).
19. *Xu, J. et al.:* Self-taught convolutional neural networks for short text clustering. Neural networks : the official journal of the International Neural Network Society. 88, 22–31 (2017).
20. *Zhao, Q. et al.:* P.: keyword clustering for automatic categorization. In: In: 2012 21st international conference on pattern recognition (icpr). IEEE (2012. (2012).