Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference "Dialogue 2020"

Moscow, June 17–20, 2020

# BLENDING FASTTEXT AND BERT PREDICTIONS FOR TAXONOMY ENRICHMENT

**Puzyrev D.** (Dmitrii.Puzyrev@skoltech.ru)‡,
**Artemova E.** (EChernyak@hse.ru)†,
**Shelmanov A.** (A.Shelmanov@skoltech.ru)‡,
**Panchenko A.** (A.Panchenko@skoltech.ru)‡

‡Skolkovo Institute of Science and Technology, Moscow, Russia;
†Higher School of Economics, Moscow, Russia

In this paper, we present one of the solutions to the Taxonomy Enrichment shared task co-located with the Dialogue conference. The proposed method blends distributional information from fastText and BERT word embeddings to predict the most likely parent hypernym node for a new term in a taxonomy. More specifically, we are using both the information on hypernym frequency among the most similar entries in the taxonomy and the similarity of hypernyms themselves. DeepPavlov-based fastText and RuBERT fine-tuned on news texts and Russian Wikipedia achieve a MAP of 0.3939 and MRR of 0.4353.

**Keywords:** distributional semantic models, BERT, taxonomy enrichment, taxonomy refinement

**DOI:** 10.28995/2075-7182-2020-19-1117-1122

## 1. Introduction

This paper describes a hypernym extraction method, submitted to the Dialogue 2020 shared task on Taxonomy Enrichment for the Russian language [4]. Given a list of words that are absent in a taxonomy (addressed further as neologisms), the method associates each word with the appropriate hypernyms from an existing taxonomy. The method is based on two criteria: bottom-up and top-down. A bottom-up criterion helps to find the best parent in the taxonomy by aggregating lower level similarity. The top-down criterion helps to find the best fit of the neologism to the branch of siblings.

The method ensembles two distributional semantic models (DSM's), fastText [1] and BERT [2], which are of completely different nature. fastText is a popular word embedding model trained using negative sampling. BERT is a recent pre-trained language model, which can be treated as a sentence embedding model or a context-aware word embedding model. The similarity of the two DSM's lies in the ability to infer an embedding for an unknown (or out-of-vocabulary, OOV) word, which is essential for dealing with neologisms.

Our submission ranked 11th in the Dialogue 2020 shared task on Taxonomy Enrichment. This shared task exploited RuWordNet [3] as the taxonomy of choice and provides two classes of neologism, being either nouns or verbs. We focused on the former. The format of submission was a ranked list of 10 possible candidates for each neologism in the test set. The submissions were evaluated according to Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

The remainder of this paper is organized as follows. **Section 2** introduces our solution. **Section 3** presents the results. **Section 4** concludes and outlines future word directions.

## 2. Our Solution

Our solution utilizes two distributional semantic models (DSM's) provided by the DeepPavlov project[1], namely:

- fastText model trained on Russian Wikipedia and Lenta.ru corpus;
- RuBERT is a pre-trained language model, treated as a contextual embedding model, for the Russian language.

The solution pipeline consists of the following steps. The first step is to **vectorize** neologisms and taxonomy items. For this aim, we retrieve context sentences both for all neologisms and for all taxonomy items from the news corpus, provided by the shared task organizers. Next, we create two lookup tables: the first is derived from fastText DSM and the second is derived from BERT DSM. The first lookup table maps each word to its fastText embedding. The second lookup table maps each word to its pooled BERT embedding. For pooling, we average all BERT embeddings for the target word over the contexts.

The second step is to **select** neighbors for each neologism. The neighbors are selected from its $k$ nearest taxonomy items ranked according to cosine similarity. The values $k$ were defined tentatively and are equal to 70 and 80 for fastText and BERT DSM's respectively.

In the third step, we **dig** into the taxonomy and extract the hypernyms of the neighbors as the candidate hypernyms.

In the fourth step, we **score** each candidate hypernym using the following features:

1. The frequency $\mathbf{F} = \sum_i \mathbb{I}_{\mathcal{H}(h_i) = \mathcal{H}_j}$ of the candidate hypernym $\mathcal{H}$ among the neighbours of the neologism, i.e *frequency factor*. In other words, we estimate how often $\mathcal{H}_j$ appears to be parental in a taxonomy tree to the word $h_i$. The more frequent $\mathcal{H}_j$ is as a hypernym to a neighbor, the more likely it is a parent to the neologism $\mathcal{N}$.

[1] http://docs.deeppavlov.ai/en/latest/features/pretrained_vectors.html

2. The maximum cosine similarity score $\max_j \cos(h_j, \mathcal{N})$ between $\mathcal{N}$ and the children of $\mathcal{H}_j$, i.e. *max-similarity factor*. This feature helps to estimate the "quality" of the candidate: it shows how similar the neologism is to its potential siblings.
   Together *frequency factor* and *max-similarity factor* contribute to *sibling factor*.

3. The cosine similarity between the candidate and the neologism: $\cos(\mathcal{H}_j, \mathcal{N})$, i.e. *parental factor*. This feature tries to dig deeper into the taxonomy tree and estimates the association between the neologism and the candidate hypernym. It helps to resolve the ambiguity of multiple taxonomy branches being close to the neologism according to neighbor estimation.

The final score is two-fold: it weights sibling and parental factors. This way the features from different taxonomy levels are combined into a single scoring. The sibling factor has two parts as well. It comprises how many children of $\mathcal{H}$ are similar to neologism ($F$) and how close is the most similar among them ($\max_j \cos(h_j, \mathcal{N})$).

The features are combined into the scoring function *score*:

$$\text{score}(\mathcal{H}_j, M) = \underbrace{(F + \max_j \cos(h_j, \mathcal{N}) * c_{sim,M})}_{sibling\,factor} * c_{sib} + \overbrace{\cos(\mathcal{H}_j, \mathcal{N})}^{parental\,factor} * c_{sim,M},$$

where $c_{sim,M}$ is a normalizing coefficient for the values of cosine similarity. The normalizing coefficient helps to scale and balance the terms with different orders of magnitude: the cosine similarity values lie within [0, 1] and the frequency of the hypernym $F$ varies from 0 to 70 or 80. The chosen values of the normalizing coefficient differ for fastText and BERT DSM's and are equal to 60 and 10, respectively. $c_{sib}$ helps to prioritize between the parent and the siblings. In the final submission we set $c_{sib} = 5$.

Finally, in the fifth step, we **rank** the candidate hypernyms according to their scores. The final results are the top 10 candidate hypernyms, achieved from the ranking.

The final ranking is achieved by the weighted voting of DSM's. If one of the DSM's misses a candidate, it is assigned with a zero weight.

$$\text{score}_{final} = \alpha \cdot score(\mathcal{H}_j, fastText) + (1 - \alpha) \cdot score(\mathcal{H}_j, BERT)$$

The coefficient $\alpha = 0.6$ allows us to achieve the best results for the final submission. The detailed analysis of results and hyperparameter tuning is reported in the next section.

## 3. Results and Discussion

Our approach has the following hyperparameters:

1. similarity normalizing coefficient $c_{sim,M}$
2. sibling multiplier factor $c_{sib}$
3. blending coefficient $\alpha$.

In this section, we present how the hyperparameter choice affects the final metrics, namely, Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) scores.

We start by looping over the similarity coefficient $c_{sim,M}$ on both fastText and BERT DSM's excluding parental factor $\cos(\mathcal{H}_j, \mathcal{N})$. The coefficient $c_{sim,M}$ is aimed at balancing between frequency factor $F$ and max-similarity factor $\max_j \cos(h_j, \mathcal{N})$, so that in in either of borderlines cases of $c_{sim,M}$ being equal to 0 or $\infty$ only $F$ and $max - sim$ are used, respectively. The results are presented in Tables 1 and 2.

**Table 1:** fastText DSM performance for different values of $c_{sim,M}$

| $c_{sim,M}$ | MAP Score | MRR Score |
|---|---|---|
| 0 *(use only max similarity factor)* | 0.3175 | 0.3521 |
| 10 | 0.3499 | 0.3858 |
| 20 | 0.3658 | 0.4021 |
| 30 | 0.3768 | 0.4144 |
| 40 | 0.3775 | 0.4150 |
| 50 | 0.3779 | 0.4154 |
| 60 | 0.3783 | 0.4173 |
| 70 | 0.3775 | 0.4154 |
| 80 | 0.3748 | 0.4126 |
| 90 | 0.3740 | 0.4119 |
| $\infty$ *(use only frequency factor)* | 0.3182 | 0.3515 |

**Table 2:** BERT DSM performance for different values of $c_{sim,M}$

| $c_{sim,M}$ | MAP Score | MRR Score |
|---|---|---|
| 0 *(use only max similarity factor)* | 0.2200 | 0.2385 |
| 20 | 0.2462 | 0.2687 |
| 10 | 0.2446 | 0.2654 |
| 30 | 0.2412 | 0.2619 |
| 40 | 0.2344 | 0.2544 |
| 50 | 0.2321 | 0.2512 |
| $\infty$ *(use only frequency factor)* | 0.1806 | 0.1949 |

Both models perform poorly when either the frequency factor or the max-similarity factor are used for scoring when compared to the weighted sum of both factors. The best similarity coefficient value is different for fastText and BERT DSM's, being 60 and 10, respectively. Bearing this in mind, we conclude that the fastText DSM is dominated by the max similarity factor, while BERT DSM relies more on the frequency factor.

Next, we look for the best weight $c_{sib}$ to combine sibling factor and parental factor. We loop over $c_{sib}$ values. The results are presented in Tables 3 and 4.

**Table 3:** fastText DSM performance for different values of $c_{sib,M}$

| $c_{sim,M}$ | MAP Score | MRR Score |
|---|---|---|
| 0 *(parental factor only)* | 0.2297 | 0.2556 |
| 5 | 0.3724 | 0.4099 |
| 10 | 0.3763 | 0.4124 |
| 20 | 0.3783 | 0.4157 |
| 40 | 0.3783 | 0.4171 |
| ∞ *(sibling factor only)* | 0.3783 | 0.4173 |

**Table 4:** BERT DSM performance for different values of $c_{sib,M}$

| $c_{sim,M}$ | MAP Score | MRR Score |
|---|---|---|
| 0 *(parental factor only)* | 0.1200 | 0.1343 |
| 5 | 0.2486 | 0.2723 |
| 10 | 0.2483 | 0.2711 |
| 20 | 0.2470 | 0.2694 |
| 40 | 0.2466 | 0.2690 |
| ∞ *(sibling factor only)* | 0.2462 | 0.2687 |

The parental factor solely is not a strong predictor. However, when combined with the sibling factor, we achieve significant improvement, more noticeably when BERT DSM is exploited. The preferable value of $c_{sib} = 5$.

Finally, we **blend** fastText and BERT predictions by computing a weighted sum with whitening coefficient being equal to $\alpha \in [0, 1]$. The best results so far are achieved when a subtle preference is given to fastText with $\alpha = 0.6$.

Finally, the highest evaluation we were able to achieve using the hyperparameters of choice is a MAP of 0.3939 and MRR of 0.4353.

**Table 5:** Blending fastText and BERT models

| α | MAP Score | MRR Score |
|---|---|---|
| 0 *(only BERT)* | 0.2486 | 0.272 |
| 0.2 | 0.3717 | 0.4084 |
| 0.4 | 0.3838 | 0.4243 |
| 0.5 | 0.3884 | 0.4295 |
| 0.6 | 0.3939 | 0.4353 |
| 0.8 | 0.3865 | 0.4261 |
| 1 *(only fastText)* | 0.3783 | 0.4173 |

## 4. Conclusion

In this paper, we presented our solution to the Taxonomy Enrichment shared task for the Dialogue conference. Our approach blends two DSMs, fastText and BERT, and comprises two criteria. A bottom-up criterion aims at finding the best parent in the taxonomy. The top-down criterion aims to find the most fitting branch of siblings. Being most likely overparameterized, our solution does not outperform the baseline provided by the shared task organizers, but rather offers considerable potential to benefit from hyper-parameter optimization. Another source of failures may be tight to the size of training data, which appears to be not sufficient for proper hyper-parameter tuning. Thus the future work directions include both the development of proper hyper-parameter tuning and data augmentation procedures.

## References

1. *Bojanowski, P. et al.:* Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics. 5, 135–146 (2017).
2. *Devlin, J. et al.:* BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers). pp. 4171–4186 (2019).
3. *Loukachevitch, N. V. et al.:* Creating russian wordnet by conversion. In: Computational linguistics and intellectual technologies: Papers from the annual conference" dialogue. p. 22 (2016).
4. *Nikishina, I. et al.:* RUSSE'2020: Findings of the first taxonomy enrichment task for the russian language. In: Computational linguistics and intellectual technologies: Papers from the annual conference "Dialogue" (2020).