Annotated Span Normalization as a Sequence Labelling Task

Daniil Anastasyev
Yandex
Moscow, Russia
`dan-anastasev@yandex-team.ru`

Abstract

In this paper, we describe a way to perform span normalization as a sequence labelling task. Our model predicts the modifications that should be applied to the span tokens to normalize them. This prediction is performed via sequence labelling, which means that each token is normalized independently. Despite the simplicity of the approach, we show that it can lead to the state-of-the-art results. We compare different pretraining schemas in application to this task. We show that the best quality can be achieved when the normalizer is trained on top of a BERT-based morpho-syntactic parser's representations. Moreover, we propose some additional features useful in the task and prove that auxiliary morpho-syntactic losses can help the model. Furthermore, we show that the model compares favourably with other contestant models of the RuNormAS competition.

Keywords: span normalization, lemmatization, pretrained language models, multitask learning
DOI: 10.28995/2075-7182-2021-20-8-15

Нормализация аннотированного спана как задача разметки последовательности

Анастасьев Д. Г.
Яндекс
Москва, Россия
`dan-anastasev@yandex-team.ru`

Аннотация

В этой статье мы описываем способ осуществления нормализации аннотированного спана как задачу разметки последовательности. Наша модель предсказывает модификацию, которую нужно произвести над токенами спана, чтобы получить его нормализованный вариант. Это предсказание осуществляется для каждого токена независимо. Данный подход является простым, однако способен давать высокое качество. Мы сравниваем различные схемы предобучения в контексте данной задачи и показываем, что наилучшие результаты могут быть достигнуты с применением модели предобученного морфо-синтаксического парсера на основе BERT. Мы предлагаем признаки, которые полезны в данной задаче, в дополнение к представлениям, получаемым из BERT. Итоговая модель показала одни из самых высоких результатов в рамках соревнования RuNormAS.

Ключевые слова: Нормализация аннотированного спана, лемматизация, предобученные языковые модели, многозадачное обучение.

1 Introduction

Annotated span normalization is the task of converting an annotated span of text to its normalized form. Such normalization is highly context-dependent. For instance, span ”Александра Пушкина” will be normalized differently in the following contexts: ”Александра Пушкина видели…”[1] (to ”Александр Пушкин”) and ”Александра Пушкина видела…”[2] (to ”Александра Пушкина”).

[1]”Alexander Pushkin was seen...”
[2]”Alexandra Pushkina has seen...”

In a way, it is similar to lemmatization because you are required to perform disambiguation of a word to obtain its normal form. But in contrast to it, span normalization has to be syntactically correct: the syntactic dependencies between the span's words should be retained after the normalization.

The most common approach to perform it is based on a morpho-syntactic parser (e.g., it is achieved this way in Natasha library[3]). Each word of a span is converted to the correct form based on the syntactic relationships between the word and other words in the sentence. It is a suitable way to perform a span normalization without any specific train data available. However, it may be hard to obtain a normalization that would follow some external guidelines because you have to learn them first and implement them using some (maybe quite extensive) set of rules.

An alternative approach may be designed using language models or seq2seq framework as in [5]. In this case, a model is trained to predict the normalized sequence given the source sequence. It is quite straightforward to formulate the task this way and language models are known to lead to the state-of-the-art results in tasks such as machine translation. Another reason to utilize this approach is the fact that the model learns the normalization guidelines itself without additional supervision from your part.

In our approach, we consider something in between those two ways mentioned above. We perform the normalization as a sequence labelling task predicting a modification for each word in a span. The model is based on the word representations obtained from a morpho-syntactic parser. To this extent, our approach is similar to the first one. Nevertheless, we use only word embedding from the parser, not its final predictions. We assumed that such representations should contain all useful information that does not have to be converted to the parses. Our approach should also be faster than the one based on a language model because it predicts the normalization in a non-autoregressive manner.

2 Model

The model we used in our approach closely follows the model from [1] which has shown the state-of-the-art performance in morpho-syntactic parsing on the GramEval-2020 dataset [6].

The model consists of three main parts: a BERT-based embedder [2], an LSTM-based encoder and a simple classifier that predicts the modification which should be applied to the given word to obtain its normalized form. As a result, the model predicts the modifications for each word in each span in the sentence independently, which means that we can normalize all sentence spans in a single forward pass. Conversely, it may also lead to some inconsistencies between the normalized words.

Such a model would have been a simplified version of the model from [1], however, we added a few improvements to address the differences between the lemmatization and annotated span normalization tasks.

Figure 1 gives an overview of our model.

2.1 Span Features

The main difference between those tasks is the fact that the normalized form of a word depends not only on the disambiguated word itself but also on all the other words in the span. Let's consider the following sentence: ”Правительство Российской Федерации рассмотрит…”[4]. The normalized form of the word ”Российской”[5] depends on the span: it should be left unchanged when the first three words form the span, it is equal to ”Российская” in the normalization of ”Российской Федерации”[6] and it is normalized to ”Российский” in a single word span.

To address such a difference, we used special features to mark those words that have to be normalized. The idea is based on BIO-notation in NER. We marked with the B-feature the first word of each span, while all the rest words of the span were marked with the I-feature and other words (not corresponding to any span) were marked with the O-feature. We trained three embeddings corresponding to those BIO features and concatenated them to the word embedding obtained from BERT.

[3]`https://natasha.github.io/`

[4]”The Government of the Russian Federation will consider…”

[5]”Russian”

[6]”The Russian Federation”

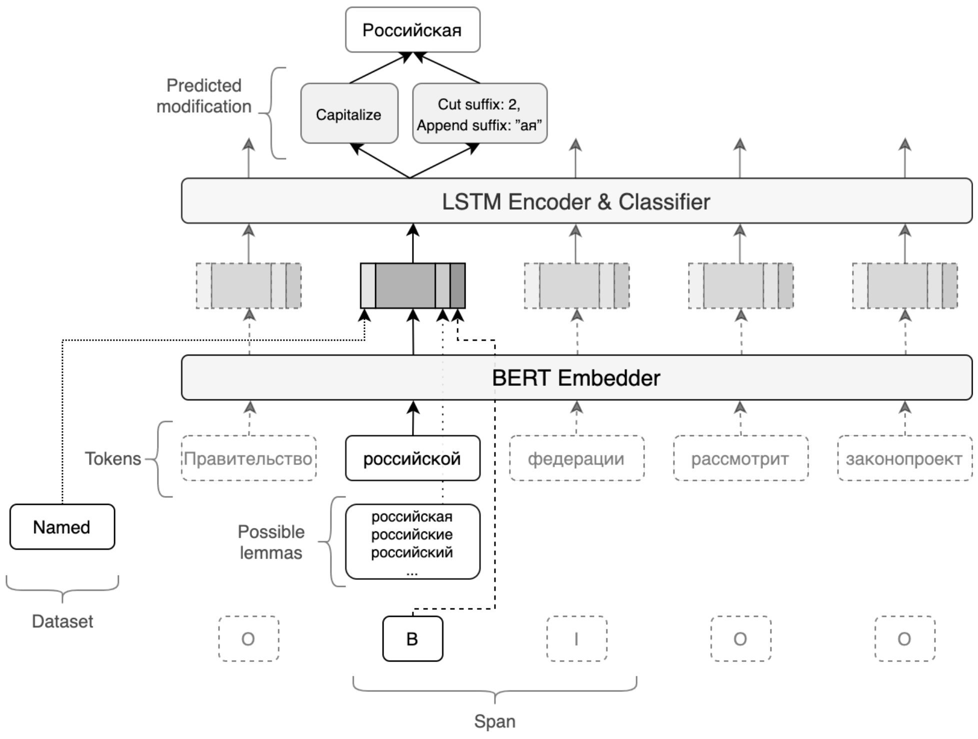

Figure 1: Schematic representation of our model

2.2 Prediction Factorization

We predicted the word normalization with a two-head classifier. One head was aimed to predict a set of modifications which should be applied to a word to obtain its normalized form, e.g.:

{"cut_prefix": 0, cut_suffix": 1, "append_suffix": "ый"}

Another head predicted the word capitalization: lower- or uppercased, or capitalized. Such a process differs from the one in [1] in the way that the sets of spelling and capitalization modifications are now independent. We call it prediction factorization. Such factorization leads to a richer set of modifications because each spelling modification can be used together with each capitalization modification. It also should be easier to be learned by the model because of a significantly lower number of classes (approximately by the factor of three).

2.3 Lemma Features

We also conditioned the prediction on the set of spelling modifications allowed by a dictionary. We collected all the possible forms of each word and built a bag-of-spelling-modifications binary vector where 1s corresponded to indices of those modifications that can be obtained from the dictionary and 0s corresponded to all other modifications. This way, the model learned to prefer the allowed forms when it is possible but wasn't constrained by them.

2.4 Auxiliary Morpho-Syntactic Loss

As we mentioned before, the span normalization can be performed based on a set of rules on top of morpho-syntactic parsing. We assumed that the model could benefit from an additional loss which will help it to learn morpho-syntactic relations better. For that purpose, we used a pretrained morpho-syntactic parser to predict the grammar values and syntactic relations in the train data. We trained our normalizer model to predict those (possibly dirty) values. We show that such auxiliary loss leads to some improvement of the normalization quality.

2.5 Pretrained Models

Pretrained models' usage is known to lead to great improvement in the model quality (e.g., [2]). We considered the RuBERT model and RuBERT fine-tuned for morpho-syntactic parsing in our work. The latter model should be more aware of morpho-syntactic relationships, so we hypothesized that it may suit our task better.

We also considered two modes of training. In the first variant, we trained only the LSTM-based encoder and classifier, while in the second one, we also fine-tuned the RuBERT's parameters.

However, it should be noted that the first approach on top of a morpho-syntactic parser is the best one in terms of its applicability: such a model may be used for normalization as well as parsing with the slowest part of computations (BERT representations obtaining) performed only once.

2.6 Ensembling

We also considered an ensemble of few models trained with a similar set of parameters. To select the prediction from such an ensemble, we used a majority vote with additional filtering. Such filtering worked in the following way: whenever a known word existed among the predicted normalization, we removed all unknown options from the voting. After the filtering, we selected the most popular predicted normalization.

3 Data

We trained our model on the data from the RuNormAS competition. We used random 90% of the data for training and the rest for validation purposes.

The data consisted of two subsets: generic spans and named entities. In the former subset, the model was required to normalize an arbitrary span, while in the latter, it should have to normalize only named entities. Consequently, generic spans generally were longer (in the number of tokens) and the named spans contained much more infrequent words (which would have been hard to find in a general-purpose dictionary).

	Generic	Named
Span count	54360	39575
Token count	136357	70997
Avg span length	2.5	1.8
Unknown token count	1376	5776
Unknown token rate	1%	8%

Table 1: Dataset statistics

Table 1 provides the train dataset details. We called *unknown* those tokens that were not available in the pymorphy2 dictionary (from [4]).

The main issue was to convert the data to the sequence-labelling-friendly format. To achieve it, we aligned the original texts with the corresponding normalized sequences and removed those which could not be aligned correctly.

The chief source of misalignments was a different number of tokens in the source and target sequences. Usually, it was a result of an additional spelling correction performed during the normalization of the training data. E.g., in some cases, the original text contained two words ”анти террористической”[7] that should have been normalized to a single word ”антитеррористический”. Unfortunately, our model was not designed to fix such misspellings, so we did not have any option but to filter such sequences out of the training dataset. It led to the loss of about 2% of the training data.

We trained every model on all available data and did not try any specialized model for the generic or named subsets. However, we noted a slightly different format of normalization between the generic and named subtasks. E.g., a simple postprocessing rule that selects an infinitive form instead of a participle or

[7]”antiterrorist”

gerund improved the quality on the generic subtask but decreased it on the named one. To give the model an ability to learn such rules itself, we tried to concatenate a dataset embedding to each word embedding.

4 Experiments

4.1 Experimental Setup

As it was mentioned before, we highly reused the model from [1] in our setup. The whole experimental setup was also based on the provided code[8]. We utilized allennlp [3] and transformers [7] libraries for our implementation. The parameters of the BERT-based embedder and LSTM-encoder were the same. We used 8-dimensional span and dataset embedding features. We used the trainable_bert checkpoint to initialize the morpho-syntactic parser.

The code used for those experiments was published in a separate branch[9].

4.2 Embedders' Comparison

Table 2 summarizes the results achieved by different combinations of embedders, their finetuning modes and the applied auxiliary losses. *Syntax-BERT* models are the models initialized from a morpho-syntactic parser's checkpoint. *Multitask* models are the models trained with the auxiliary losses, while single corresponds to the models trained only to normalize words. *Frozen* or *trainable* relates to the BERT parameters' finetuning.

We also implemented a baseline based on the syntax parser predictions (*Syntax Baseline*). We normalized span tokens using the Natasha library implementation mentioned above, but we utilized syntax predictions from the same morpho-syntactic parser as in the *Syntax-BERT* models. Natasha predicts lowercased normalization so we used the source capitalization for the normalized tokens to achieve better quality. There still is some room for improvement with a capitalization predictor: we could have achieved 91.27% quality on the *Named* subset, if capitalization had not been considered.

	Syntax Baseline	RuBERT				Syntax-BERT			
		Frozen		Trainable		Frozen		Trainable	
		Single	Multitask	Single	Multitask	Single	Multitask	Single	Multitask
Generic	94,52%	96,81%	96,98%	96,87%	97,43%	97,30%	**97,74%**	97,10%	97,72%
Named	90,23%	96,20%	96,05%	97,47%	97,64%	97,74%	97,58%	97,41%	**97,77%**

Table 2: Embedders' comparison

Our baseline model shows the worst results which proves that good normalization is hard to obtain without additional handwritten rules or special model training.

Clearly, the model benefits from morpho-syntactic pretraining. It is especially noticeable in the case of frozen models comparison. As we mentioned before, we considered the frozen Syntax-BERT model to be the most useful in downstream applications because it can perform lots of tasks at the same time. However, the results show that such a model is also good from the normalization quality point of view.

Surprisingly, in the single-task training mode, the frozen Syntax-BERT is better than the trainable model. We can assume that the model slightly overfits when all parameters are trained. However, this is not the case in the multitask training. The additional losses prevent the model from forgetting useful morpho-syntactic relations learned by the parser.

RuBERT-based models work significantly better in the trainable setup. It may be explained by the fact that BERT's representations should be adjusted for the task (BERT-based morpho-syntactic parser also performed poorer without proper finetuning). However, a multitask trainable model is almost on par with the Syntax-BERT models. It suggests that the model also managed to learn some useful morpho-syntactic information from the automatic markup.

To assess this hypothesis, we compared the qualities of the multitask models on the GramEval-2020 dataset. The results are presented in Table 3.

[8] `https://github.com/DanAnastasyev/GramEval2020`
[9] `https://github.com/DanAnastasyev/GramEval2020/tree/span_normalization`

Model	POS	Feats	LAS
Frozen Syntax-BERT	**96,20%**	**96,40%**	**84,60%**
Trainable Syntax-BERT	95,55%	96,09%	83,70%
Trainable RuBERT	94,76%	95,81%	81,56%

Table 3: Comparison of multitask models on the GramEval-2020 test data

It is expected that the frozen model initialized from the parser trained on the GramEval dataset should achieve the highest scores. However, the results of other models are also promising. As anticipated, the trainable variant of the same model doesn't forget much because of the auxiliary losses. Moreover, the trainable RuBERT-based model learns morpho-syntactic parsing reasonably well without been exposed to a significantly larger and cleaner original dataset.

4.3 Model Features' Ablation

Table 4 shows how different features of the proposed model affected the quality. The Final model is based on Syntax-BERT fine-tuned in the multitask mode. Each subsequent row shows how the model would have performed without one feature.

	Generic	Named
Final model	97,72%	97,77%
- span embedding	84,70%	95,68%
- lemmatization factorization	97,48%	97,54%
- dataset embedding	97,28%	97,46%
- possible lemma embedding	97,10%	97,48%

Table 4: Model features' ablation

Just as expected, the span embeddings have a key role in our model. The model is unable to understand the span boundaries and performs simple lemmatization without them. It especially affects the model's quality on the generic subset where the spans tend to be longer.

Lack of any other feature also reduces the model quality, although much less substantially.

4.4 Comparison on RuNormAS

Finally, we can compare the model quality with other contestants of the RuNormAS competition. Our team was called *qbic* and we ended up in second place in the generic track and first place in the named entity normalization track. The full results are presented in Table 5.

Team	Generic	Named
ksmith	**98,01%**	98,12%
qbic	97,91%	**98,15%**
eindenbom	97,58%	97,92%
king_menin	96,45%	95,75%
baseline	77,32%	88,81%
fateev.da	77,30%	88,97%
shkunkov.a	0,00%	76,80%

Table 5: Comparison with other contestants

We used ensembling of five models to achieve better quality. The ensemble outperformed a single model by 0,2% on the generic subset and by 0,4% on the named subset.

5 Error Analysis

We performed some error analysis of the final ensemble. We identified the following types of errors:

- *Space* error is an error that could have been fixed by correct spacing, e.g., ”В.Филев” was predicted instead of the expected ”В. Филев”. Our algorithm used the original spaces for the normalization, so it was not able to correct such cases. This type of error was ignored in the generic track.
- *Punctuation* error occurred because of the mismatch of the source and normalized texts punctuation, e.g., ”дети - сироты” (with a hyphen) instead of ”дети – сироты” (with en-dash).
- *Capitalization* error is a result of incorrectly predicted capitalization. It is common in some multi-word expressions with ambiguity, e.g., in the normalization ”институт исследований Южной Азии” predicted instead of ”Институт исследований Южной Азии” the word ”институт” may be capitalized or not depending on the context. This type of error was also ignored in the generic track.
- *Alignment* error may be a result of the incorrect word count (e.g., ”будет увеличиваться” instead of ”увеличиваться”) or incorrect word ordering (e.g., ”Юрий Мирошник” instead of ”Мирошник Юрий”). Such errors clearly could not be fixed by our model.
- *Word form* errors. We split them into four categories depending on whether the expected normalization could have been found in the pymorphy2 dictionary (*Known-*/*Unknown-*) and whether the predicted normalization could have been found in the dictionary (*-Known*/*-Unknown*). Most of the errors are in the Known-Known category, which means that the model was not able to select a correct word form of a common word.

We believe that *space*, *punctuation* and *alignment* errors are minor and may be ignored.

The qualitative analysis can be found in Table 6.

	Generic		Named	
Error	Single model	Ensemble	Single model	Ensemble
Space	-	-	11 (4.30%)	11 (5.19%)
Punctuation	15 (7.77%)	17 (9.60%)	25 (9.77%)	22 (10.38%)
Alignment	32 (16.58%)	30 (16.95%)	20 (7.81%)	14 (6.60%)
Capitalization	-	-	39 (15.23%)	38 (17.92%)
Known-Known	136 (70.47%)	121 (68.36%)	121 (47.27%)	95 (44.81%)
Known-Unknown	4 (2.07%)	3 (1.69%)	7 (2.73%)	9 (4.25%)
Unknown-Known	3 (1.55%)	3 (1.69%)	5 (1.95%)	6 (2.83%)
Unknown-Unknown	3 (1.55%)	3 (1.69%)	39 (15.23%)	28 (13.21%)

Table 6: Error analysis of multitask trainable Syntax-BERT model and its ensemble

It is clear that ensembling helps to select a better word form and reduces errors count this way. It shows the instability in the training process because a significant number of errors was fixed by the majority vote over the predictions of similar models.

6 Conclusion

We proposed an alternative way to perform span normalization and showed that it works reasonably well on the RuNormAS dataset achieving the best or almost the best results there. We believe that our method should be considered favourably when enough training data is available for normalizer because it is simpler than language model training or normalization rules designing in such a scenario. The inference speed is bound by the speed of the syntax parser’s encoder, which should be practicable in most applications.

References

[1] Anastasyev D. G. Exploring Pretrained Models for Joint Morpho-Syntactic Parsing of Russian // Proceedings of the International Conference ”Dialog 2020”. — Moscow, Russia, 2020. — P. 1–12.

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // 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). — Minneapolis, Minnesota, 2019. — P. 4171–4186.

[3] Gardner Matt et al. AllenNLP: A Deep Semantic Natural Language Processing Platform // Proceedings of Workshop for NLP Open Source Software (NLP-OSS). — Melbourne, Australia, 2018. — P. 1–6.

[4] Korobov Mikhail. Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts. — 2015. — P. 320–332.

[5] Korzun V.A. Proper Names Normalization without Semantic Parsing // http://www.dialog-21.ru/media/4671/доклад_корзун.pdf. — 2019.

[6] Lyashevskaya O. et al. GRAMEVAL 2020 Shared Task: Russian Full Morphology and Universal Dependencies Parsing // Proceedings of the International Conference ”Dialog 2020”. — Moscow, Russia, 2020. — P. 553–569.

[7] Wolf Thomas et al. Transformers: State-of-the-Art Natural Language Processing // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. — Online, 2020. — P. 38–45.