Unknown word handling in structured text classification

Anastasiya Silaeva¹ and Igor Balk²

¹ MIREA — Russian Technological University, Vernadsky avenue, Moscow, 119454, Russia
² Global Innovation Labs LLC, 258 Harvard Str #352 Brookline MA 02446 USA

Email: info@innovationlabs.net

Abstract. In the NLP task, high priority was accorded to the accumulation of vocabularies. To complement them, you need to find unknown words. Unique words with the help of an expert can be determined in the dictionary. This paper presents a technique for finding unknown words in Named Entity Recognition (NER).

1. Introduction

Named Entity Recognition (NER) is an important NLP task, and for stable operation of this method, you need to consider unknown phrases. The problem is that for marking up texts it is necessary to compose task-specific dictionaries. This is needed to structure the text. If you disregard unknown words when classifying, for recognition, it will be given the wrong class. To avoid this, you need to define it in advance for the class as unknown. As a result, at the end of recognition, unknown words are added to a separate dictionary, and they are marked up depending on the subject area. After that, dictionaries for NER are already supplemented. But it is possible to deal with such words if the recognition system has the feature of additional training and additional data labeling.

Traditional methods of this area are CRF, SVM, and perceptron models [1–3]. However, neural network models such as:
1. RNN [4],
2. LSTM [5]
3. bi-LSTM [6].
4. bi-LSTM + CRF [7]

2. Methodology

We present methodology which includes the bi-LSTM neural network architecture with embedding layer and bi-LSTM without embedding layer. For both neural networks, used pre-trained vectors from fasttext. The model fasttext was trained on a corpus of 144 million records with a vector dimension of 300, letter n-grams of length 5, a window of size 3. It includes a dictionary of 123,473 unique words and n-grams.

The dataset in research was collected by hand and has about 2 million records. To train the neural network, 1.6 million records were used, and the test sample consisted of 400 thousand records.

In order to structure the data for NER, the auto-markup algorithm was used. Auto-markup includes dictionaries with:
1. Lemmas and words with typos (words that were misspelled and required to be replaced with the original)
2. Stop words (words that have no value for the task)
3. Stable expressions (unique phrases that are included in the subject area)
4. Unknown words (unknown words include punctuation marks, special characters, numbers)
5. Parts of speech
6. Common for the neural network

Further, after creating dictionaries, a part of speech was established for each word. The grammar rules for extracting features from the text were applied and tags were tagged to the extracted features.

BIO (short for inside, outside, beginning) was used for NER [1]. Markup, in which B is the beginning of the phrase, I is inside of a chunking task, O is all the other words. The unknown tag was also added to recognize unknown words.

The dictionary of unique words and phrases has a dimension of 343635. A unique index was set for each word. For unknown words, a constant index was assigned, which made it possible to correlate that the given word was not added to the dictionary.

The number of unique tags of words and phrases is shown in Table 1. From this table, you can see that there are the largest number of other tags - this is due to the fact that during the markup these words were used to find named entities. Examples of tagged sentences are shown in Figure n.

| Tag    | Count of unique words |
|--------|-----------------------|
| I-obj  | 1526                  |
| unknown| 1849                  |
| B-action| 1877                |
| B-obj  | 2910                  |
| other  | 118074                |

Figure n examples of marked up records from the corpus.

Since the dataset is imbalanced, a weighted loss function was used, which was [156.3965, 100.8784, 192.3697, 1.0001, 158.7649].
3. Results

3.1. Bi-LSTM with embedding layer

For speed of learning, only those vectors of words that are present in the dictionary for the neural network were extracted.

Bi-LSTM with pretrained embedding vectors has the following result (see Fig. 1). The Embedding layer has not been trained further in order for the neural network to concentrate on training weights.

![Normalized Confusion Matrix](image)

**Figure 1.** Decision matrix based on the results of the bi-lstm neural network with an embedding layer

In Figure 2, the classification report shows that f1 was 0.96.

```
| Tag       | precision | recall | f1-score | support   |
|-----------|-----------|--------|----------|-----------|
| B-action  | 0.97      | 0.98   | 0.97     | 756787    |
| B-obj     | 0.92      | 0.93   | 0.93     | 881316    |
| I-obj     | 0.84      | 0.90   | 0.87     | 48949     |
| other     | 0.97      | 0.96   | 0.97     | 2966190   |
| unknown   | 0.98      | 0.99   | 0.99     | 866632    |

| tag       | accuracy  | precision | recall | f1-score | support   |
|-----------|-----------|-----------|--------|----------|-----------|
| accuracy  | 0.96      | 0.96      | 0.96   | 5517874  |
| macro avg | 0.94      | 0.95      | 0.94   | 5517874  |
| weighted avg | 0.96 | 0.96      | 0.96   | 5517874  |
```

**Figure 2.** Full report for each tag and overall prediction for all tags
3.2. Bi-LSTM without embedding layer
This method used benefits of fasttext. There was a converter in the method: word = vector from fastText. In this case, you do not need to save separately the training dictionary and the pre-trained vectors from fasttext. To predict unfamiliar words, the unknown tag was added, the words of which were assigned a random vector when receiving the result of the converter.

The predicted tag decision matrix is illustrated in Figure 3:

Figure 3. Decision matrix based on the results of the bi-lstm neural network without embedding layer

Figure 4 shows a classification report on predicted tags:

Figure 4. Classification report for each tag and general prediction for all tags
The results show that the neural network often makes mistakes on the other tag and associates many tags with B-obj. This is due to the fact that words with B-obj tags sometimes fall under the other tag when auto-tagging. The overall result for bi-LSTM without embedding f1 is 0.86.

4. Conclusion
According to the results, it turned out that the bi-LSTM architecture with an embedding layer has a better result in predicting all tags and the f1 measure was 0.96, and the bi-LSTM without an embedding layer was 0.86, the comparison is presented in table 2. The advantage of the first model is that it has greater accuracy, finds unique words that were not used when training the neural network. The advantage of the second model is that the weights are 394 kb in contrast to the first model, which weights are 436 mb. Another plus is that the model finds fewer unfamiliar words due to the fasttext capabilities. If you remove the words in the markup that are defined as other, but are included in the B-obj dictionary, the results will improve. Also, for better results, you need to supplement all dictionaries.

| Model                      | Precision, % | Recall, % | F1-score, % | Accuracy, % | Size weight, mb |
|----------------------------|--------------|-----------|-------------|-------------|-----------------|
| Bi-lstm with embedding     | 94           | 95        | 95          | 96          | 436             |
| Bi-lstm without embedding  | 84           | 92        | 86          | 84          | 0,394           |

References
[1] Lev Ratino and Dan Roth 2009 Design challenges and misconceptions in named entity recognition In Proceedings of the Thirteenth Conference on Computational Natural Language Learning pp 147–155
[2] Alexandre Passos, Vineet Kumar and Andrew McCallum 2014 Lexicon infused phrase embeddings for named entity resolution Proceedings of the Eighteenth Conference on Computational Natural Language Learning pp 78–86
[3] Gang Luo, Xiaojiang Huang, Chin-Yew Lin and Zaiqing Nie 2015 Joint entity recognition and disambiguation Proceedings of the Conference on Empirical Methods in Natural Language Processing pp 879–88
[4] Christoph Goller and Andreas Kuchler 1996 Learning task-dependent distributed representations by backpropagation through structure IEEE International Conference on Neural Networks 1 pp 347–52
[5] G Petasis, S Petridis, G Paliouras, V Karkaletsis, S J Perantonis and C D Spyropoulos 2000 Symbolic and neural learning for named-entity recognition Proceedings of the Symposium on Computational Intelligence and Learning pp 58–66
[6] Alan Graves, Abdel-rahman Mohamed and Geoffrey Hinton 2013 Speech recognition with deep recurrent neural networks IEEE International Conference on Acoustics, Speech and Signal Processing pp 6645–49
[7] Anh L T, Arkhipov M Y and Burtsev M S 2017 Application of a Hybrid Bi-LSTM-CRF model to the task of Russian Named Entity Recognition arXiv (preprint arXiv:1709.09686)