# Accelerated Natural Language Processing
# Tagging parts of speech based on word embeddings

## 1. Introduction

The goal of the project is to train and analyze part-of-speech classifiers based on word pre-trained embeddings. We use a dataset of various English sentences, which are tokenized into words in verying ways depending on the method (see section 2). The dataset's sample counts can be found in table 2.

We wish to explore which models perform the best on the test dataset – what are the differences between static and contextual embeddings, and whether all types of contextual embeddings exhibit similar behavior, or are there any unique properties to some of them. We would also like to dive deep into cases where some of the models go wrong using specific examples, and put forward theories as to why that might be.

## 2. Methods

In the project as input we use two kinds of word embeddings: GloVe ("static") embeddings from (Pennington et al., 2014), and BERT ("contextual") embeddings from (Devlin et al., 2018). "Contextual" here means that they take into account the information carried by the sentence around the word. Internally, the latter embeddings use the Transformer architecture introduced by (Vaswani et al., 2017). As output we use the tags from the universal dependencies project (Nivre et al., 2016). The only exception to this is the baseline model we call BASIC, which always predicts the most common tag for a given word (*not* word embedding), or in the case of unseen words, the most common tag overall.

For static embeddings, the text is always split into words as they are separated by the space character (with punctuation marks also being separate). In the case of contextual embeddings, the BERT tokenizer might split a word into several tokens (e.g. "ambulance" into "amb", "ul" and "ance"). When that happens, we use one of the following strategies to combine the token embeddings into one word embedding: first token (FT), last token (LT), reduce max (RX), reduce mean (RM), or reduce sum (RS).

We used two methods to train models on the word embeddings:

- Logistic regression (LGR) as it's available in the Scikit-learn library (Pedregosa et al., 2011), trained using the L-BFGS algorithm (Liu & Nocedal, 1989) and using the cross-entropy loss.
- Multi-layer perceptron (MLP) – a fully connected neural network; we got best results when using two hidden layers of 500 neurons each. We trained it using Stochastic Gradient Descent (learning rate 0.01, divided by 10 every 30 epochs, trained for 100 epochs) and Cross-Entropy loss as they are in the `Torch` library in Python (Paszke et al., 2019).

## 3. Results

### 3.1. Model performance

The accuracy of each model (combination of model type and embedding type used) on the test set can be found in table 3.

It is interesting to note that the more complex models are extremely close, when trained on static embeddings, to the baseline that assigns the most common tag to each word. They do, however, generalize much better to unseen words, since the former just assigns the most common tag (NOUN in this case) to all unseen words, yielding poor results.

We note that while the MLP model outperforms the LGR model on both partitions of the testing dataset for the static embeddings, and is on par with it for most other embedding types, we were not able to train it to that standard on the RS embeddings. We theorize that it might be because the highly varying values of the embeddings (for a word with $L$ subword tokens, the sum of the values in its word embedding is expected to be $L$ times higher than that of a single-subword token). The relatively simple, first-order SGD method, which we picked because it's easy to understand and interpret its result, doesn't perform as well as the second-order L-BFGS method in this unstable environment.

It is clear that, barring one outlier mentioned above, the contextual embeddings provide much more powerful part-of-speech taggers than the static embeddings. This matches our expectations, since they contain much more information not only about the word, but also about its context – and there are plenty of examples in English where words occur as different parts of speech depending on the context (e.g. "strike" – which, as our analysis in subsection 3.2 found, is one of the many words static embeddings struggle with).

### 3.2. Analysis of models' mistakes

Notation remark: in this section we will use the notation LAX→LAY to mean that the true label LAX has been misclassified as the label LAY.

#### 3.2.1. Proper nouns

Firstly, we notice that *all* models struggle with proper nouns. For each model we consider here, both PROPN→NOUN and NOUN→PROPN are in the top 5 mistake types (see table 4). We think that, as far as mistakes ago, these are the most understandable, since the line between the two tags is very thin and it would often be unclear even to a human annotator whether a noun is a proper noun or not – since after all proper nouns are, most of the time, just regular nouns with special meaning attached to them (e.g. *United Nations*, *Supreme Court*, etc.)

#### 3.2.2. Contextual vs. Static

We note that, as per table 4, both types of models trained on static embeddings exhibit high numbers of mistakes. The lgr variant struggles mostly with verbs that also have noun meanings (e.g. *project*, *delays*, *share*, *strike*), which account for 12.5% of its overall mistakes., The mlp variant struggles with auxiliary verbs that can also function as standalone verbs (e.g. *have*, *do*, *is*). Both variants struggle a lot with the "to" particle (e.g. *to get*) or adposition (e.g. *get to*), always assigning it the former tag, thus misclassifying it all of the 208 times it appears as an adposition.

We note that neither the verb/noun and verb/aux pairings, nor the word "to", are nearly as problematic for models trained on contextual embeddings.

#### 3.2.3. The X tag

In the universal dependencies project, the X tag is described as the tag to be used in the last resort when an annotator has no idea what to tag a word as, so we thought it might be interesting to look at how our models handle it. Unsurprisingly, we found out on test set that the tag X was usually classified as tag NOUN by the basic model. When static embeddings were used, the lgr model only achieved 2.2% class accuracy, while the mlp model achieved 73.8%. In case of contextual embeddings with different combinations we observe that this problem is handled even better and the tag is correctly classified as X with a class accuracies of up to 68.6% (lgr) and 79.6% (mlp).

Upon inspection of the data, we can see that most of the words with this tag in the test set are URLs, and due to their large variety, only 23/137 of these words are also in the training set, and to make matters worse, we believe that most of these were misclassified by the annotators, since a majority of this "known" group is proper names and punctuation. Of the unknown group, over 70% are URLs. We theorize that the mlp static embedding model learned to assign the tag X to out-of-vocabulary words (represented in static embeddings as zero vectors), and the contextual models learned to assign it to words associated with the subword tokens corresponding to URLs, whereas the rest of the models failed to learn these relationships due to their more limited architecture.

## 4. Conclusion and Future Scope

In our report, we performed quantitative analysis of the models in subsection 3.1, and then qualitatively dove into specific examples in subsection 3.2. We discussed the limitations of the basic and static classifiers, and how the various contextual models overcome them. The contextual classifiers reached a point where their most common mistakes *by far* come from the ambiguity of proper nouns, so the major direction in taking them further would have to address this flaw.

## References

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Liu, Dong C and Nocedal, Jorge. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Nivre, Joakim, De Marneffe, Marie-Catherine, Ginter, Filip, Goldberg, Yoav, Hajic, Jan, Manning, Christopher D, McDonald, Ryan, Petrov, Slav, Pyysalo, Sampo, Silveira, Natalia, et al. Universal dependencies v1: A multilingual treebank collection. In

*Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1659–1666, 2016.

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Kopf, Andreas, Yang, Edward, DeVito, Zachary, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie, and Chintala, Soumith. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

## 5. Appendix

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| NOUN | 0.6741 | 0.9292 | 0.7813 | 4136 |
| PRON | 0.9678 | 0.9333 | 0.9502 | 2158 |
| PUNCT | 0.9941 | 0.9855 | 0.9898 | 3098 |
| VERB | 0.8844 | 0.8087 | 0.8449 | 2640 |
| accuracy | | | 0.8576 | 25098 |
| macro_avg | 0.8411 | 0.7798 | 0.7916 | 25098 |
| weighted_avg | 0.8725 | 0.8576 | 0.8546 | 25098 |

*Table 1.* Requested results from the preliminary task

| | Train | Val-All | Val-Unk | Test-All | Test-Unk |
|---|---|---|---|---|---|
| ADJ | 13114 | 1872 | 188 | 1782 | 187 |
| ADP | 17698 | 2029 | 4 | 2028 | 6 |
| ADV | 9697 | 1180 | 44 | 1147 | 26 |
| AUX | 12424 | 1509 | 4 | 1508 | 1 |
| CCONJ | 6696 | 779 | 2 | 737 | 1 |
| DET | 16280 | 1895 | 3 | 1898 | 2 |
| INTJ | 688 | 115 | 19 | 118 | 14 |
| NOUN | 34754 | 4196 | 621 | 4135 | 709 |
| NUM | 4108 | 382 | 109 | 540 | 204 |
| PART | 5554 | 631 | 0 | 629 | 1 |
| PRON | 18562 | 2216 | 8 | 2156 | 3 |
| PROPN | 12292 | 1787 | 737 | 1984 | 800 |
| PUNCT | 23573 | 3075 | 23 | 3096 | 23 |
| SCONJ | 4490 | 464 | 6 | 443 | 6 |
| SYM | 697 | 79 | 9 | 106 | 5 |
| VERB | 22958 | 2757 | 208 | 2638 | 185 |
| X | 709 | 144 | 100 | 137 | 114 |
| **Total** | **204294** | **25110** | **2085** | **19762** | **2287** |

*Table 2.* Counts of samples in the dataset, plus the sizes of the non-training datasets with words from the training dataset removed. Note that these counts are not of unique values, they may count repeated values multiple times.

| Name | All | Unknown |
|---|---|---|
| BASIC | 0.8576 | 0.3095 |
| LGR-STATIC | 0.8673 | 0.6571 |
| MLP-STATIC | 0.8705 | 0.6903 |
| LGR-FT | 0.9534 | 0.8373 |
| MLP-FT | 0.9542 | 0.8263 |
| LGR-LT | 0.9574 | 0.8522 |
| MLP-LT | 0.9574 | 0.8518 |
| LGR-RX | 0.9568 | 0.8526 |
| MLP-RX | **0.9578** | 0.8557 |
| LGR-RM | 0.9580 | **0.8645** |
| MLP-RM | 0.9569 | 0.8526 |
| LGR-RS | 0.9565 | 0.8535 |
| MLP-RS | 0.6686 | 0.5409 |

*Table 3.* Accuracy values for the models when evaluated on the entire test set, and on the words in the test set that do not appear in the training set, respectively

| LGR-STATIC | | MLP-STATIC | | MLP-RM | | LGR-RM | |
|---|---|---|---|---|---|---|---|
| mistake type | count | mistake type | count | mistake type | count | mistake type | count |
| PROPN→NOUN | 349 | PROPN→NOUN | 243 | NOUN→PROPN | 196 | PROPN→NOUN | 150 |
| VERB→NOUN | 245 | VERB→AUX | 228 | PROPN→NOUN | 67 | NOUN→PROPN | 111 |
| NOUN→PROPN | 208 | ADP→PART | 208 | NOUN→ADJ | 64 | ADJ→NOUN | 65 |
| ADP→PART | 208 | NOUN→PROPN | 207 | VERB→AUX | 43 | NOUN→ADJ | 43 |
| NOUN→VERB | 172 | PUNCT→SYM | 197 | NOUN→X | 42 | VERB→AUX | 36 |
| MLP-FT | | MLP-LT | | LGR-RS | | MLP-RX | |
| mistake type | count | mistake type | count | mistake type | count | mistake type | count |
| NOUN→PROPN | 200 | NOUN→PROPN | 197 | PROPN→NOUN | 150 | NOUN→PROPN | 204 |
| NOUN→ADJ | 74 | PROPN→NOUN | 66 | NOUN→PROPN | 109 | PROPN→NOUN | 69 |
| PROPN→NOUN | 69 | NOUN→ADJ | 59 | ADJ→NOUN | 59 | NOUN→ADJ | 61 |
| VERB→AUX | 47 | VERB→AUX | 43 | NOUN→ADJ | 50 | VERB→AUX | 44 |
| NOUN→NUM | 40 | NOUN→X | 34 | VERB→AUX | 36 | ADJ→NOUN | 33 |

*Table 4.* Most common mistake types for selected models