Representing ELMo embeddings as two-dimensional text online

Andrey Kutuzov
University of Oslo
andreku@ifi.uio.no

Elizaveta Kuzmenko
University of Trento
lizaku77@gmail.com

Abstract

We describe a new addition to the *WebVectors* toolkit which is used to serve word embedding models over the Web. The new *ELMoViz* module adds support for contextualized embedding architectures, in particular for ELMo models. The provided visualizations follow the metaphor of 'two-dimensional text' by showing lexical substitutes: words which are most semantically similar in context to the words of the input sentence. The system allows the user to change the ELMo layers from which token embeddings are inferred. It also conveys corpus information about the query words and their lexical substitutes (namely their frequency tiers and parts of speech). The module is well integrated into the rest of the *WebVectors* toolkit, providing lexical hyperlinks to word representations in static embedding models. Two web services have already implemented the new functionality with pre-trained ELMo models for Russian, Norwegian and English.

1 Introduction

In this demo paper we describe a new module recently added to the free and open-source *WebVectors* toolkit (Kutuzov and Kuzmenko, 2017)[1]. *WebVectors* allows to easily deploy services to demonstrate the abilities of static distributional word representations (word embeddings) (Bengio et al., 2003; Mikolov et al., 2013) via web browsers. It currently powers at least two embedding model hubs:

- *NLPL WebVectors*[2], featuring models for English, Norwegian and other languages, trained within the Nordic Language Processing Laboratory initiative.
- *RusVectōrēs*[3], featuring models for the Russian language.

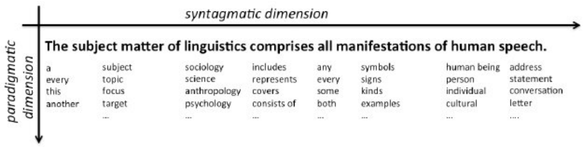

Figure 1: Metaphor of two-dimensional text; borrowed from (Biemann and Riedl, 2013).

The new module (we name it *ELMoViz*) adds the functionality to study, probe and compare recently introduced contextualized embedding (or 'token-based') models (Melamud et al., 2016). In particular, at this point we provide support for the ELMo architecture (Peters et al., 2018a) based on deep recurrent neural networks. In the future, we plan to add support for Transformer-based models like BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020). ELMo architecture is significantly less computationally expensive than Transformers, while being almost on par in terms of performance. Thus, it yields rich possibilities in the context of non-commercial web services.

For analyzing ELMo representations of an arbitrary input text, we offer the metaphor of 'two-dimensional text' first proposed in (Biemann and Riedl, 2013) (see Figure 1). This allows a sort of 'visualization' for contextualized embeddings through finding words which are most semantically similar to the input words in their current contexts. From the linguistic point of view, these are 'paradigmatic replacements' (Saussure, 1916) – words that can to some extent substitute target words. The two dimensions here are the *syntagmatic* one (horizontal) which describes the linear order of the sentence, and the *paradigmatic* one (vertical) which describes semantic classes to which

[1] A screencast is available at https://www.youtube.com/watch?v=dDugoV1r_wk.

[2] http://vectors.nlpl.eu/explore/embeddings/

[3] https://rusvectores.org/

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, pages 143–148
April 19 - 23, 2021.

the words in the sentence belong to. The generated substitutes in the vertical axis can also be thought of as ‘semantic variations’ of the input sentence.

The rest of the paper is organized as follows. In Section 2 we describe the background for this work, including the *WebVectors* framework, and explain the need to develop additional functionality in order to handle contextualized embeddings. Section 3 describes in detail this functionality, both from the point of view of the end user and from the point of view of deployment logistics. In Section 4, we conclude and outline future work.

2 Background

Since the widespread adoption of prediction-based word embeddings (Mikolov et al., 2013) started, there has always been a need to efficiently serve and demonstrate these representations over the Web. Researchers and practitioners need this for quick experimentation and testing hypotheses by comparing different distributional models. Those who teach natural language processing and computational linguistics need ways to show the students how dense distributional representations capture lexical semantics without installing any software or downloading any models (often it is desirable that this is shown for a particular language or domain).

In turn, language teachers value tools to demonstrate lexical variety and degrees of similarity for words in a foreign language. To this extent, serving word embeddings over the Web can help both the teachers with preparing educational materials and the students with grasping the concepts in a foreign language.

The *WebVectors* framework we presented in (Kutuzov and Kuzmenko, 2017) is aimed at all these purposes. It allows to quickly deploy a stable and robust web service featuring operations on vector semantic models, including querying, visualization and comparison, all available to users of any computer literacy level. It extended already existing embedding visualization services like Embedding Projector[4] by providing users with the ability to find nearest semantic neighbors of query words, perform vector math operations over embeddings, etc. Since being first presented in 2016, *WebVectors* keeps adding new functionality, and now it offers filtering nearest associates by part of speech tags or corpus frequency, and can generate semantic ego graphs, among other features (see Figure 2).

[4]https://projector.tensorflow.org/

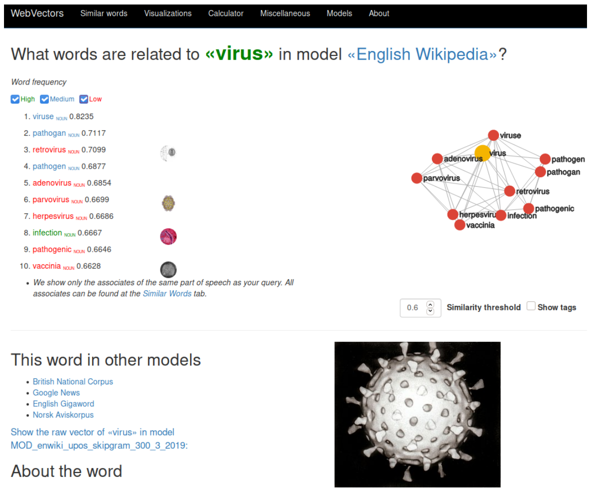

Figure 2: Screenshot of a *WebVectors* instance at http://vectors.nlpl.eu/explore/embeddings/

Until the introduction of *ELMoViz*, these features were limited to the so-called ‘static word embeddings’, that is, architectures like *word2vec* (Mikolov et al., 2013), *fastText* (Bojanowski et al., 2017) or *GloVe* (Pennington et al., 2014). In these architectures, after the training is finished, each word type in the vocabulary is rigidly associated with a single dense vector. However, in the recent years NLP saw a surge of pre-trained ‘contextualized’ embedding architectures, like ELMo (Peters et al., 2018a), BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020) and many others. One of the changes these deep learning models brought was that even at inference time, each word token representation (embedding) depends on its immediate context. This means that ambiguous words will receive different representations depending on the sense in which they are used, which opens rich new possibilities for natural language understanding.

Libraries used in *WebVectors* to deal with static word embeddings (*Gensim*, (Řehůřek and Sojka, 2010)) were not fit to power operations on contextualized models. That is why we decided to implement an entirely new *WebVectors* module, which would take a query phrase as an input, and produce paradigmatic replacements (lexical substitutions) for each content word in this phrase, based on a given pre-trained contextualized ELMo language model.

One can find a number of existing frameworks for online experimentation with contextualized models: among others, we should mention Language Interpretability Tool (Tenney et al., 2020), exBert by (Hoover et al., 2019) and the hosted inference API at the HuggingFace Community Model Hub (Wolf et al., 2020). However, these projects are aimed exclusively at the Transformer-based architectures. The system we present in this demo paper, on the other hand, is aimed more towards RNN-based architectures like ELMo. As it was shown, for example, in the field of semantic change detection (Kutuzov and Giulianelli, 2020), ELMo can often outperform BERT or be on par with it, while requiring significantly less computational resources. We believe it is especially important for teaching activities.

Additionally, our system is more lexically oriented and is integrated with the existing *WebVectors* functionality, as we will show in the next section.

3 System description

After turning on the contextualized embedding related functionality in the *WebVectors* configuration file,[5] the person deploying the service has to provide three data sources for each ELMo model:

1. a pre-trained *ELMo model* itself in the standard format (`*.HDF5` file with the weights and `options.json` file with the model architecture description);

2. a tab-separated *frequency dictionary* file to use when determining the frequency tier of word types (it is recommended to derive it from the same corpus the ELMo model was trained on, but technically this is not required);

3. a set of *static (type-based) word embeddings* produced by averaging contextualized token embeddings inferred with the same ELMo model.

The last item of this list requires some explanation. Our aim is to provide the end user with a set of lexical substitutes for each word token in context from the input sentence (see Figure 3). With static embedding architectures, this boils down to looking up the vector of the target word x and then finding n other words in the model vocabulary with the vectors closest to x. However, this is obviously impossible with contextualized language models: there are no static vector lookup tables to begin with. One can easily infer contextualized representations for each word in the input sentence: but

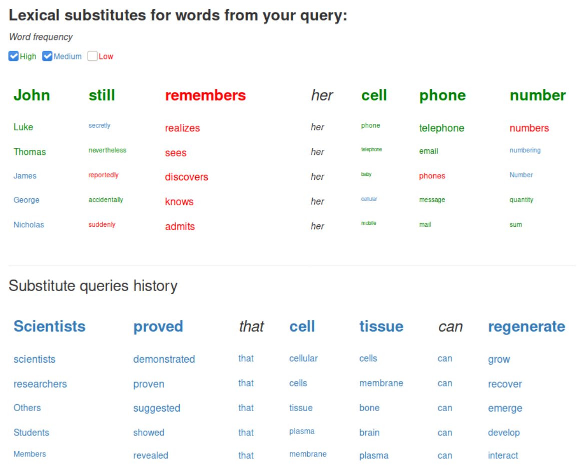

Figure 3: Examples of two-dimensional text inferred from an ELMo model ($n = 5$).

what to compare them with in order to illustrate their meaning?

To cope with this issue, we adopted the approach described in (Liu et al., 2019). They employed the so called type-level context averaging in order to align pre-trained contextualized models cross-linguistically. In our case, we needed only the first stage of their workflow. The idea is to obtain static type-level word representations located in the same vector space as the contextualized embeddings. Given a large enough reference text corpus and a pre-trained contextualized language model, one takes the average of all token representations for each target word occurrence in the corpus. This averaged type embedding is comparable to contextualized token embeddings routinely produced by the model.

In practice, we found that one does not even need to *average* token embeddings: it is enough to *sum* them, and then unit-normalize the resulting summed vector. As for the list of target words, we simply use top 10 000 (or any other amount found suitable) most frequent words from the corresponding ELMo model vocabulary or from a reference corpus (excluding functional parts of speech and digits). Low frequent words are usually not needed in this case anyway, since the quality of their embeddings is also lower. We provide a simple script to extract type embeddings from an ELMo model and a given corpus in our GitHub repository.[6]

As a result, when an end user enters an input phrase or sentence (typically from 5 to 15 words),

[5] In principle, it is also possible to use only ELMoViz, without other *WebVectors* modules.

[6] `https://github.com/akutuzov/webvectors/tree/master/elmo/`

WebVectors produces contextualized token embeddings for each token in the query, and finds top n words in the *type embedding* model, which are the closest (by cosine similarity) to each of the *token embeddings*. These predictions are *lexical substitutes* or *paradigmatic replacements*; they demonstrate what other words could fill these positions in the query, depending on the context.

Another option to produce such substitutes would be to feed the input sentence to the ELMo model and then for each word token choose the strongest activations at the final softmax layer of the language model and map them to words in the model vocabulary. However, in practice we found that this approach is slightly slower than the one described above. Additionally, ELMo models are often published online without the vocabulary they were trained on. Since the input layer of ELMo is purely character-based, it does not hinder inferring token embeddings, but it effectively blocks using these weights as language models *per se*. Our approach allows one to use any given ELMo model with any desired corpus to produce a set of reference type embeddings.

System maintainers can provide several models for the service to work with, including models for different languages; one of the models should be specified in the configuration files as the default one. When entering the query sentence, users can choose the model which will process the input.

Apart from choosing between different models, *WebVectors* also allows users to choose the exact ELMo layer from which token representations will be inferred; it was shown in (Peters et al., 2018b) that different neural network layers convey information related to different linguistic tiers: syntax, semantics, pragmatics, etc. At this point, one can choose between the top ELMo layer and the average of all layers. Note that for all operations with pre-trained ELMo models we use `simple_elmo`: a lightweight TensorFlow-based Python package also developed by us.[7] If need be, `simple_elmo` can also be used as a standalone library to handle ELMo models.

Both the words from the input sentence and the lexical substitutes are colored according to their frequency tier in the reference corpus (green for 'high', blue for 'mid' and red for 'low'), in accordance with other *WebVectors* components. Similarly, each word is hyperlinked to its 'landing page' bound to one of the static embedding models served by a particular *WebVectors* installation (like the one in Figure 2), allowing easy and playful exploration of the semantic space. The font size of the lexical substitute corresponds to cosine similarity between the token embedding and the substitute type embedding: thus, users can instantly see what word tokens the model is unsure about. The service performs fast under-the-hood part-of-speech tagging of the query,[8] so for functional words we always yield themselves as substitutes (see 'her', 'that' and 'can' in Figure 3). They are also uncolored and not hyperlinked, so that a user might focus on content words, while at the same time still having an impression of 'full sentence variations'.

The users should be aware that the lexical substitutes potentially contain all the biases inherited from the corpus the model was trained on. Thus, the paradigmatic axis might include slander words and stereotypes, if they were frequent enough in the data. We did not address this issue in the present work, but we advise the users to take this into account when dealing with any unsupervised language models.

Importantly, we keep a short history of substitute queries, so that it is possible to see at a glance the changes brought by a different context, a different word order or a different contextualized model (if the web service offers several models). Figure 4 shows an example from our Russian live demo at the *RusVectōrēs* web service. In the first sentence, the word закладку '*zakladku*' is used in the newer sense of 'a secret place to store illegal drugs', while in the second sentence it is used in the older sense of 'the act of founding a building'. The generated substitutes reflect the differences in word meaning depending on the context. In the first example the substitutes include such words as 'meeting, sale, operation', and in the second example the substitutes are 'opening, building, repair'.

4 Conclusion

The described system for generating two-dimensional text using pre-trained ELMo models is now deployed at the two model hubs mentioned in Section 1. *NLPL WebVectors* features ELMo models trained on English Wikipedia and on Norwegian corpora[9], while *RusVectōrēs* features a

[7] `https://pypi.org/project/simple-elmo/`

[8] Using UDPipe (Straka and Straková, 2017).

[9] `http://vectors.nlpl.eu/explore/embeddings/en/contextual`

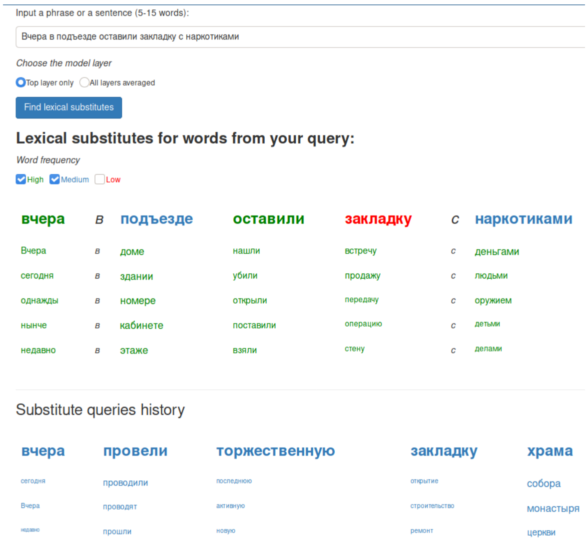

Figure 4: History of lexical substitute queries with a Russian ELMo model.

model trained on concatenated Russian Wikipedia and Russian National Corpus.[10]

The presented component for the *WebVectors* framework allows users to explore pre-trained ELMo models and to visualize contextualized embeddings as a two-dimensional text for faster analysis of early research prototypes. While previously the framework provided interface only to static vector semantic models, introducing support for contextualized architectures allows for more intricate exploration of linguistic phenomena, such as lexical ambiguity and contextual semantic change.

We hope that the new functionality will provide language teachers, NLP researchers and practitioners with a powerful tool to study word meaning in context and at the same time keep the audience up-to-date with recent advances in the field of distributional semantics and deep learning based NLP. A separate important contribution is our `simple_elmo` library which makes using ELMo models in Python much easier, especially for researchers with linguistic background.

In the future, we plan to add support for other contextualized embedding architectures like BERT, to allow inter-architectural comparisons. Another interesting room for future work is integrating with other exploratory services for neural NLP models, like the ones mentioned in Section 2.

[10] `https://rusvectores.org/en/contextual/`

References

Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2d! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2019. exbert: A visual analysis tool to explore learned representations in transformers models. *arXiv preprint arXiv:1910.05276*.

Andrey Kutuzov and Mario Giulianelli. 2020. UiO-UvA at SemEval-2020 task 1: Contextualised embeddings for lexical semantic change detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 126–134, Barcelona (online). International Committee for Computational Linguistics.

Andrey Kutuzov and Elizaveta Kuzmenko. 2017. Building web-interfaces for vector semantic models with the webvectors toolkit. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 99–103.

Qianchu Liu, Diana McCarthy, Ivan Vulić, and Anna Korhonen. 2019. Investigating cross-lingual alignment methods for contextualized embeddings with token-level evaluation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 33–43, Hong Kong, China. Association for Computational Linguistics.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Ferdinand de Saussure. 1916. *Course in general linguistics*. Duckworth.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 107–118. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.