An Analysis of Deep Contextual Word Embeddings and Neural Architectures for Toponym Mention Detection in Scientific Publications

Matthew Magnusson
Department of Computer Science
University of New Hampshire
mfm2@cs.unh.edu

Laura Dietz
Department of Computer Science
University of New Hampshire
dietz@cs.unh.edu

Abstract

Toponym detection in scientific papers is an open task and a key first step in place entity enrichment of documents. We examine three common neural architectures in NLP: 1) convolutional neural network, 2) multi-layer perceptron (both applied in a sliding window context) and 3) bi-directional LSTM and apply contextual and non-contextual word embedding layers to these models. We find that deep contextual word embeddings improve the performance of the bi-LSTM with CRF neural architecture achieving the best performance when multiple layers of deep contextual embeddings are concatenated. Our best performing model achieves an average F1 of 0.910 when evaluated on overlap macro exceeding previous state-of-the-art models in the toponym detection task.

1 Introduction

The available scientific knowledge is growing every day.[1] Yet, this knowledge is often locked into publications in pdf format, that are not condusive to machine-reading or automated analyses. In this work we take a step towards automated knowledge extraction that is compatible with extraction and visualization frameworks for scientific publications (Ronzano and Saggion, 2016).

Many scientific publications contain geographic references which are commonly confused by extractors with other entities such as people or proteins whose name contains references to places. Extracting such placenames, or toponyms, has several important applications such as the identification of virus outbreak locations (Weissenbacher et al., 2015), treatment adherence (Zhang et al., 2012), and mapping of research findings (Leveling, 2015).

Toponyms are textual spans of text that identify geospatial locations. This can range from the canonical name of populated places, such as "Chengdu" to direct or indirect mentions of geographic entities, including "Cho Oyu" or "5 km south of Mirnyy". The parsing of geographic locations from unstructured text is often addressed with gazeteers. It is generally very difficult to achieve high accuracy due to domain diversity, place name ambiguity, metonymic language and limited contextual cues (Gritta et al., 2018). Furthermore, major challenges to toponym detection in scientific texts come from the fact that names of institutions, viruses and proteins often contain geographic references. Moreover, the extractor needs to handle the overall noisy nature of scientific articles after PDF extraction—with challenges include associating figures and tables as well as handling character encodings.

Task: Toponym detection. Given the text of a scientific publication (as extracted from the PDF), the task is to extract character offset locations of true toponyms. This location is referred to as a toponym mention in the following. A toponym is defined to include proper names and geographic entities but to exclude indirect mentions of places and metonyms. Toponym detection is a first step towards toponym resolution where each toponym mention is to be aligned to a geospatial location.

In this work we focus on toponym detection and evaluate different neural specialization models for word embeddings on this task.[2] This approach has benefitted many natural language processing (NLP) tasks, such as named entity recognition (Collobert et al., 2011). Previous work in toponym detection has mostly focused on non-contextual word embeddings (Magge et al., 2018). Here we study which neural model and which word embed-

[1] In 2016, 2.3 million science and engineering publications were produced globally up from 1.2 million in 2003 for a 5.2% compound annual growth rate (NSF, 2018).

[2] Data and code available in appendix: https://cs.unh.edu/~mfm2/index.html

Proceedings of the Workshop on Extracting Structured Knowledge from Scientific Publications, pages 48–56
Minneapolis, USA, June 6, 2019.

ding types are best suited for the detection of toponyms in scientific publications. We also demonstrate the benefits of neural architectures in comparison to Tagme, a state-of-the-art entity linker, from which we isolate toponym spots based on DBpedia categories.

The contribution of our work lies in answering the following research questions in regards to the task of toponym detection in scientific papers:

RQ1 Independent of the neural model architecture for specialization, which embedding demonstrates better performance: A task-independent deep contextual embeddings or a non-contextual embedding trained on a scientific-domain specific corpus?

RQ2 Given an optimal embedding, which neural specialization architecture is optimal for the task?

RQ3 Given an optimal word embedding and neural architecture, what are the performance impacts of different combinations of the embedding and the classifier?

Our findings show that the best performance on toponym detection is achieved by deep contextual embeddings (even though trained on a non-scientific corpus) when using bidirectional LSTMs with CRFs as the specialization architecture (Peters et al., 2018), while concatenating the layers of the embeddings. However, other deep contextual configurations including weighted average, and single layer selection also yield similar average performance. We also find that handcrafted orthographic features did not impact bi-LSTM model performance, but did positively impact MLP and negatively impacted CNN.

Outline. In Section 2 we discuss related work. Section 3 explains the neural models types included in our analysis and discusses word embedding types. In Section 4, we provide details on the approaches examined in our study. In Section 5 we discuss the data, metrics, and results obtained. We finish with a conclusion about the research questions posed.

2 Related Work

There is significant work in the area of toponym detection (Matsuda et al., 2015; Lieberman et al., 2010) and the closely related fields of named entity recognition (Li et al., 2018) and entity mention detection (Shen et al., 2015) with many different approaches. State-of-the-art named entity detection models have historically employed a combination of hand-crafted features, rules, natural language processing, string-pattern matching, and domain knowledge using supervised learning on relatively-small manually annotated corpora (Piskorski and Yangarber, 2013). A common approach to toponym detection has been to utilize place name gazetteers which are directories of geographic names and their corresponding geolocations to perform string matching of place names in text (Lieberman et al., 2010).

Contemporary approaches in entity detection have included conditional random fields (CRF) (Lafferty et al., 2001) and neural-based architectures. (Collobert et al., 2011) propose a window-based, multi-layer, dense feed-forward neural architecture using word embeddings concatenated with orthographic features and a gazetteer as an input layer with a hard Tanh output layer for superior performance on a standard NER task. Huang et al. (2015) utilise a bi-directional LSTM with a sequential conditional random layer using a gazetteer and Senna word embeddings to obtain superior performance. Magge et al. (2018) achieves state-of-the-art results in toponym detection by utilizing a window-based deep neural network, word embeddings trained on a domain-specific corpus, orthographic features, and a gazetteer.

3 Background

We briefly recap the background of several methods we include in our study.

3.1 Neural Models

Many neural approaches to natural language applications make use of in input layer that consists of tokenized text mapped to a pre-trained word embedding matrix. One common neural architecture is the deep multi-layer perceptron (MLP) which is a densely connected feed forward network with multiple layers. One or more layers of densely connected neurons are combined allowing for complex function approximation. Another common architecture, the convolutional neural network (CNN), uses mathematical cross-correlation to reduce the number of free parameters in deep

models. Pooling layers can be used to combine the output of specific sets of neurons in one layer to a single neuron in a subsequent layer.

Recently, more approaches incorporate a recurrent neural network (RNN) architecture which contrasts with MLP or CNN by using internal state in subsequent processing of input sequences. A bi-LSTM is a variant of a recurrent neural network that processes sequences of input in both directions with a hidden state shared between each "step" of the sequence processing. Many deep models contain mixtures in different layers of these three architecture types.

3.2 Word Embeddings

A word embedding is a popular approach for representing text using a dense vector representation. This contrasts with traditional bag-of-word model encodings where high dimensional one-hot vectors are used to represent each words. A drawback of the bag-of-words approach is that the semantic similarity between words is lost, while dense embeddings have been shown to exhibit semantic similarity with linear relationships (Turney and Pantel, 2010).

Pre-trained embedding models can be applied as the input layer of a neural model which is then specialized for the task at hand. Mikolov et al. (2013) brought the concept of word embeddings to the forefront of natural language research with the continuous skip-gram word2vec model. This method utilizes a feedforward neural net to create a language model. The dense continuous vector representation of words in these models demonstrate superior performance on semantic word relationship tests relative to sparse term vectors. A limitation of feedforward language models including word2vec is that they are non-contextual which means that all senses of a word are merged into one dense vector.

Peters et al. (2018) propose a deep neural model (ELMo) that generates contextual word embeddings which are able to model both language and semantics of word use. ELMo embeddings assign a representation to a token as a function of the entire input token sequence. Devlin et al. (2018) introduce a pre-trained language model transformer architecture called BERT that is jointly conditioned on left and right context in all layers. The model can be fine-tuned or deep contextual embeddings can be extracted from the model layers.

4 Approach

We study three different neural approaches for toponym detection: 1) sliding windows convolutional neural networks, 2) sliding window multilayer perceptrons, and 3) bi-LSTM. Both contextual and non-contextual word embeddings are used and enriched with a limited number of handcrafted features. We run 5 trials for each configuration. Deep embedding variants in the analysis are: first, middle (mid), and last layer; layer concatenation (concat); weighted-average (w-avg); softmax classifier (soft) and no orthographic features (no-ortho).

We study the effects on the performance, when choosing a particular embedding (4.1) in a specialization architecture (4.3), with or without hand crafted features (4.2). The remainder of this section lays out the options we included in our study.

4.1 Embeddings

ELMo: We use deep contextual embeddings from ELMo embeddings (Peters et al., 2018) which represent learned functions of the internal states of a deep bidirectional language model that has been pre-trained on the 1B Word Benchmark (Chelba et al., 2013). In Table 2 ELMo embeddings are abbreviated as EL.

BERT: We use deep contextual embeddings generated by extracting the three uppermost layers of the model (Devlin et al., 2018) using the pre-trained BERT-Base 12-layer Cased model.[3] The BERT model uses WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. We use the WordPiece embedding corresponding to the input source token and concatenate the three upper layers of the model.

w2v: The scientific-domain specific non-contextual word embeddings are provided by Pyysalo et al. (2013) which are generated from Wikipedia, PubMed, and PMC texts using the word2vec tool. They are 200-dimensional vectors trained using the skip-gram model.

For the MLP model an input embedding is generated by concatenating the ELMo vectors with the one-hot encoding of orthographic features and an additional binary encoding indicating if the token was contained within the set of gazetteer tokens. The CNN is not enhanced with either orthographic or gazetteer tokens. The bi-LSTM embedding is only enhanced with orthographic features.

[3] https://github.com/google-research/bert

4.2 Hand-crafted Features

Neural network based approaches have been shown to achieve strong results without the use of hand-crafted features, however, in many cases, hand-crafted features can boost model performance. We use two sets of hand-crafted features that frequently appear in the literature to increase performance in named entity recognition. In both sets of features, their inclusion did benefit performance.

Orthographic Features: a one hot encoding is assigned to each token based on its orthographic structure including presence of digits, alphabetic characters, and upper case characters. The orthographic features assist the model for managing out of vocabulary tokens.

Gazeteer Features: a set of toponynm tokens is generated from the GeoNames entries.[4] For example, for the entry in Geonames, "Gulf of Mexico", the tokens "Gulf", "of", and "Mexico" are added to the toponym set. This approach does include stop words such as "of". The impact of excluding stop words was not examined. This is used as a binary feature for the presence of the parsed token in the constructed Geonames token set. An indicator of inclusion in a gazetteer is a common feature in toponym detection models. Our study shows that this approach yields a small improvement in the MLP model performance.

4.3 Specialization Architectures

MLP: We use a sliding window multi-layer perceptron model with w2v and ELMo embeddings. A sliding window (size = 5) is applied to each tokenized sentence using the corresponding embeddings. The input layer is connected to two fully connected layers with 128 hidden units each and relu activation. The output layer uses a sigmoid with a binary output to indicate if the token is part of a toponym. MLP-EL-max is the maximum run by macro overlap F1 when using ELMo embeddings with orthographic features and gazetter indicator. MLP-w2v-max is the same model only differing by using the w2v embedding.

CNN: We use a sliding window convolutional neural network using w2v and ELMo embeddings. A sliding window (size = 5) is applied to each tokenized sentence using the corresponding embeddings. The input layer is two 1d convolutional layers with filter sizes of 250 and a kernel size of 3. A global 1-d max pooling layer follows the convolutional layers. Two fully connected layers with 100 hidden units each and relu activation follow max pooling. A sigmoid function is applied in output layer to indicate if the token is part of a toponym. CNN-EL-max is the maximum run by macro overlap F1 when using ELMo embeddings with gazetter indicator. CNN-w2v-max is the same model only differing by using the w2v embedding.

Bi-LSTM with CRF: The implementation used is based on the approach develped by Lample et al. (2016) using code adapted from Reimers and Gurevych (2017).[5] Input sentences for the model are generated in IOB representation for labeled toponyms in the training data. Each LSTM has a size of 100 and is trained with a dropout of 0.50. Character embeddings are generated using a convolutional neural network and the maximum character length is 50. We use the w2v, ELMo and BERT embeddings for token encoding. LSTM-w2v uses w2v and orthographic features; LSTM-BERT uses BERT embeddings (top 3-layers concatenated) without orthographic features; LSTM-EL uses concatenated ELMo embeddings with orthographic features. LSTM-EL-concat-w2v is LSTM-EL embeddings concatenated with w2v.

4.4 Baseline

The following two models are included as baselines in the evaluation.

MLP-Baseline-w2v: A sliding window multi-layer perceptron as suggested by Magge et al. (2018). The system has a specific component for toponym detection using a two-layer feedforward neural network (200 hidden units per layer). The baseline features a sliding window (size = 5) over each sentence using the w2v embeddings for token encoding. The baseline did not include a gazetter-based lookup but did incorporate orthographic structure of the tokens.

TagMe: TagMe (Ferragina and Scaiella, 2010) is a state-of-the-art entity linking tool that aligns spans in text to entities in Wikipedia snapshots of April, 2016. We filter entity links to include location entities only. Spots are included as toponyms if their linked Wikipedia entity is associated with a category that contains one of the words: place, capital, province, nations, countries, territories, territory, geography, or continent

[4] https://www.geonames.org/export/

[5] https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf

Table 1: Gold Standard Corpus Statistics.

	Documents	Tokens	Toponyms
Train	72	396,668	3,637
Valid	32	179,443	2,141
Test	45	253,159	4,616
Total	149	829,720	10,394

(TagMe-Baseline). We also run a SVM classifier that takes all categories as phrases and words. It is using LibSVM with the c-SVC algorithm and a linear kernel. The regularizer (aka "C" parameter) is tuned on the tuning split to optimize F1 and the dataset is balanced before training and tuning (TagMe-SVM).

5 Experiment Evaluation

In the following we describe our experimental evaluation using data and metrics from the SemEval Toponym resolution task.

5.1 Data

The experimental evaluation is based on a dataset of 150 full texts of open access journal articles from PubMed Central (PMC) which is provided by Davy Weissenbacher (2019).[6] To create the corpus, they convert PDF to text with the "pdf-to-text" software and then manually annotate toponym spots using the Brat annotator 3. Table 1 details statistics of this dataset.

The text documents are parsed from PDF files as many scientific articles are still not available in well-structured text formats such as XML and therefore annotators need to be adaptable to noisy inputs. The structure of the text demonstrates the challenge of using scientific text for toponym detection as the pdf-to-text conversion process results in text that introduces new line characters at non-sentence boundaries and exhibits hyphenation which splits tokens in the middle of the word. This complicates tokenization and sentence boundary detection. The pdf conversion process also injects header and footer text in the document which interrupts the flow of the documents. Tables and equations add additional noise to the text with irregular line lengths that can further complicate the extraction of toponym mentions from documents.

[6]From the train data set, PMC4009295.txt was not included because of encoding issues

5.2 Metrics

Quality of predictions is evaluated in terms of precision, recall and F1-measure. The model is tuned on F1 with validation on the valid set and prediction on the test set.

The dataset comes with a recommendation for two variants of evaluation: strict boundaries and overlapping boundaries. In the strict evaluation, spots must match the exact span boundaries in the gold standard. In the overlapping evaluation, a match occurs when the spot span and gold standard span overlap.

Furthermore, two options for computing precision and recall are available handling spots quality per publication. In micro-averaging all spans across the corpus treated as one set on which precision and recall is calculated. In macro averaging precision, recall, and F1 are calculated on a per publication basis, and then the results are averaged.

Over all four the evaluations measures provide similar results, we only report results on the overlapping evaluation with macro-averaging. Because the average performance of the CNN and MLP were below the average performance demonstrated by bi-LSTM, we show the maximum value of CNN and MLP to highlight that even best obtained result is less than bi-LSTM.

5.3 Results

The results are provided for precision (P), recall (P), and F1 for overlapping boundaries and macro-averages. Because of small errors in character offset alignment, the performance across all of the models for strict evaluation is slightly lower overall (omitted results will be available online).

Table 2 provides the comparison of different architectures, embeddings, and baselines.

TagMe-SVM obtains the lowest performance of all measures with a F1 of 0.330. TagMe-Baseline achieves a F1 of 0.544 and is the only model not directly trained on the data. The TagMe-SVM has a recall that is similar to that of the CNN and MLP neural methods but with a severe degradation in precision.

The ELMo embeddings enhance the F1 performance of the bi-LSTM model but appear to have limited benefit to the other studied neural models. The convolutional network using the ELMo-based embeddings exhibits higher performance on the F1 score relative to MLP-ELMo.

The CNN exhibits higher precision with similar recall to other methods that are not bi-LSTM. Bi-LSTM with CRF outperforms the MLP and CNN models independent of the embedding type. The best average performance of the bi-LSTM model is achieved when the three ELMo embeddings were concatenated, obtaining 0.910 F1. When word2vec and averaged ELMo embeddings are concatenated, a similar average F1 is achieved (0.909), however this model has the highest average precision (0.909).

Table 3 reports the results of different combinations of the ELMo embeddings based on bi-LSTM with CRF, the best performing neural model in our study. We also examine replacing the CRF classifier with a softmax when the ELMo embeddings are concatenated. The softmax classifier exhibits decreased performance with an F1 of 0.900. This indicates the importance of choosing the right classifier for the task in the bi-LSTM architecture.

We examine the effect of only using one of the three vectors provided in the ELMo embedding. In terms of average F1, the poorest performing layer is the first layer. The middle and last layer exhibit similar F1 performance. Peters et al. (2018) indicates that the lowest layer captures more syntactical information while the upper layers have a higher degree of semantic information, which may explain the performance difference in the layers.

Across all measures, the concatenation of all three ELMo vectors performed the best on average over any layer in isolation. Concatenating these three embeddings performs also slightly better than calculating an average or weighted average of the embeddings. This is based on a sample size of 5 for each measure evaluated.

Orthographic features yields an average absolute performance benefit of 2.4% in the tested MLP-w2v model. But somewhat surprisingly, causes a substantial degradation in CNN-w2v performance (-16.6% absolute). In bi-LSTM, the removal of orthographic features causes a very slight degradation in performance. This indicates that in MLP and CNN models, handcrafted features are a consideration, but may not be necessary in bi-LSTM models for toponym detection.

We also compare the performance between two contextual embeddings BERT and ELMo. Both contextual embeddings exhibit similar average F1 measures with BERT slightly underperforming ELMo. An explanatory factor could be that by only extracting the first WordPiece embedding per corresponding source token (based on the approach (Devlin et al., 2018) undertake for NER task) that information is being lost by not using all WordPiece tokens. We also use the Cased Based model, alternatively the uncased and/or Large models may yield better performance. From an implementation standpoint, the WordPiece tokenization is challenging for maintaining alignment in embedding layer composition approaches other than mapping source-to-head WordPiece token. The additional coding effort complicates the implementation of this approach.

Table 2: Comparison of different architectures and embeddings.

Run	P	R	F1
TagMe-SVM	0.214	0.712	0.330
TagMe-Baseline	0.449	0.692	0.544
MLP-Baseline-w2v	0.864	0.797	0.829
MLP-EL-max	0.886	0.798	0.840
CNN-w2v-max	0.896	0.797	0.843
CNN-EL-max	0.908	0.788	0.844
MLP-w2v-max	0.888	0.835	0.861
LSTM-w2v	0.893	0.871	0.882
LSTM-BERT	0.895	0.913	0.904
LSTM-EL-concat-w2v	**0.909**	0.910	0.909
LSTM-EL-concat	0.904	**0.916**	**0.910**

For implementations using CNN or MLP, the results of this task did not indicate that the implementation of deep contextual embeddings yields superior performance. The appeal of non-contextual embeddings such as word2vec is their ease of implementation, which require only mapping a source token to its corresponding vector in a fixed vocabulary (or unknown if OOV). Deep contextual embeddings require mapping a token to a vector based on the ”key” of its entire sentence. This is reasonable to implement but does require extra effort. The results of bi-LSTM clearly indicate that the additional performance may justify the additional implementation resources.

Figure 1 illustrates the different variations applied to the bi-LSTM with ELMo embeddings after 5 runs for each variation. Using the first layer alone in the embedding appeared to have the most negative impact on performance. Either concatenation or weighted average appear to have the most consistent highest level of performance. This is consistent with Peters et al. (2018) that found that weighted average had the best performance on a NER task using ELMo embeddings and De-

Table 3: Comparison of variations of bi-LSTM with ELMo embeddings.

Run	P	R	F1
first	0.897	0.880	0.889
soft	0.897	0.903	0.900
avg	**0.920**	0.885	0.901
last	0.896	0.912	0.904
mid	0.908	0.903	0.905
no-ortho	0.904	0.911	0.907
w-avg	0.907	0.911	0.909
concat	0.904	**0.916**	**0.910**

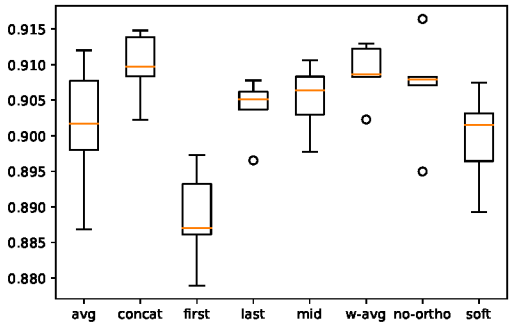

Figure 1: Comparison of variations of bi-LSTM with ELMo embeddings.

vlin et al. (2018) that found that concatenation of deep contextual embeddings (BERT) had the best performance. So either appear to be an appropriate approach given they both possess the overriding characteristic of using all layers for representation. Averaging appears to inject more variablity into performance which limits its appeal. Using softmax instead of CRF as a classifier resulted in a slight decline in performance. This highlights the importance of having a quality classifier at the top layer of the bi-LSTM for structured prediction. The omission of orthographic characters may slightly impair performance, but this is not certain, as the highest score observed out of all trials is without orthographic features (0.916). This analysis also highlights the importance of multiple trials with testing neural models as relying on one trial may sert to under or over state the average performance of a hyperparameter.

5.4 Error Analysis

Figure 2 illustrates a challenging passage of text in the corpus where none of the text should be annotated. The best performing model LSTM-EL-concat (highlighted in yellow) identifies "Britain"

The generated coordinates were then linked by the University of Portsmouth's Great Britain Historical Geographic Information System (GIS) Project to the relevant historical area boundary using county administrative diagrams (22, 23). For the 40 postcodes from 1972 (1.6%) and the 1,101 addresses from 1950 (45.0%) that could not be matched by the SAHSU team, the Great Britain Historical GIS team employed manual methods of assignment (13)

Figure 2: False positives by Tagme-Baseline and LSTM-EL-concat.

or A/Quail/Hong Kong/G1/97 (G1-like, H9N2). More importantly, some of their internal genes are closely related to those of novel H5N1 viruses isolated during the outbreak in Hong Kong in 2001.

Figure 3: False positive and false negative by Tagme-Baseline.

as a mention. While Great Britain is a place, in this context, it is highlighting a character span within an entity that is not a place. Tagme-Baseline correctly does not identify text in the prevously identified span but does incorrectly (highlighted in blue) identifies the character spans for "addresses" (a general concept not a specific location) and "Great Britain Historical GIS" (adjective for the "team" entity) as mentions. These are all examples of false positives for toponym detection.

Figure 3 shows Tagme-Baseline incorrectly identifying "Hong Kong" (highlighted in yellow) as a mention (false postive) and failing to correctly identify the second "Hong Kong" (underlined) which is an annotated mention (false negative). LSTM-EL-concat correctly did not identify the first "Hong Kong" as a mention but did properly identify the second. The first "Hong Kong" mention is part of a virus name and while has a relationship to that place it is not meant to identify the place.

6 Conclusion

In this work, we study the benefits of different neural architecture for the specialization of pre-trained embeddings for the task of toponym detection in scientific publications. We demonstrate superior results using neural models in comparison to a state-of-the-art entity linker. This indicates that general-purpose popular entity linking tools are not the optimum choice for the task. We also show that non-contextual yet domain-specific word embeddings underperform compared to deep contextual embeddings trained on a general large-scale corpus for state-of-art bi-LSTM models. We

believe the increase in performance due to ELMo-based embeddings is due to the richer context and character structure contained in the embeddings. This richer representation did not benefit toponym detection in the CNN and MLP neural models tested and in fact the maximum result for MLP was using the domain specific non-contextual embedding vectors.

Out of all the neural architectures, the neural model with the best performance is bi-LSTM with CRF using concatenated ELMo contextual embeddings. This finding is consistent with other research using bi-LSTM with CRF that has demonstrated state of the art results for named entity recognition tasks. It is noteworthy, that the Bi-LSTM with CRF is able to extract toponym mentions using context from embeddings without relying on the presence of a gazetteer. An open question is if a gazetteer or other knowledge graph resources could be incorporated into a neural model to achieve superior performance.

Areas of future research include exploring the integration of dense, convolutional, or other neural architectures as a top layer of the bi-LSTM to enhance classification. Concatenating contextual and the non-contextual embeddings improved recall and incorporating both into future models could be an area that yield further performance gains.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1846017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

2018. Science and engineering indicators 2018. NSB-2018-1. National Science Board, Alexandria, VA.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Karen O'Connor Matthew Scotch Graciela Gonzalez Davy Weissenbacher, Arjun Magge. 2019. Semeval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1625–1628, New York, NY, USA. ACM.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2018. What's missing in geographical parsing? *Lang. Resour. Eval.*, 52(2):603–623.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Johannes Leveling. 2015. Tagging of temporal expressions and geological features in scientific articles. In *Proceedings of the 9th Workshop on Geographic Information Retrieval*, GIR '15, pages 6:1–6:10, New York, NY, USA. ACM.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2018. A survey on deep learning for named entity recognition. *CoRR*, abs/1812.09449.

Michael D. Lieberman, Hanan Samet, and Jagan Sankaranayananan. 2010. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *Proceedings of the 6th Workshop on Geographic Information Retrieval*, GIR '10, pages 6:1–6:8, New York, NY, USA. ACM.

Arjun Magge, Matthew Scotch, Abeed Sarker, Davy Weissenbacher, and Graciela Gonzalez-Hernandez. 2018. Deep neural networks and distant supervision for geographic location mention extraction. *Bioinformatics*, 34(13):i565–i573.

Koji Matsuda, Akira Sasaki, Naoaki Okazaki, and Kentaro Inui. 2015. Annotating geographical entities on microblog text. In *Proceedings of The 9th Linguistic*

Annotation Workshop, pages 85–94. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Jakub Piskorski and Roman Yangarber. 2013. Information extraction: Past, present and future. In Thierry Poibeau, Horacio Saggion, Jakub Piskorski, and Roman Yangarber, editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 23–49. Springer Berlin Heidelberg.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Francesco Ronzano and Horacio Saggion. 2016. Knowledge extraction and modeling from scientific publications. In *Semantics, Analytics, Visualization. Enhancing Scholarly Data*, pages 11–25, Cham. Springer International Publishing.

W. Shen, J. Wang, and J. Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge & Data Engineering*, 27(2):443–460.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141.

Davy Weissenbacher, Tasnia Tahsin, Rachel Beard, Mari Figaro, Robert Rivera, Matthew Scotch, and Graciela Gonzalez. 2015. Knowledge-driven geospatial location resolution for phylogeographic models of virus migration. *Bioinformatics*, 31(12):i348–i356. Exported from https://app.dimensions.ai on 2019/03/04.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Juan Zhang, Jun Xie, Wanli Hou, Xiaochen Tu, Jing Xu, Fujian Song, Zhihong Wang, and Zuxun Lu. 2012. Mapping the knowledge structure of research on patient adherence: Knowledge domain visualization based co-word analysis and social network analysis. *PLOS ONE*, 7(4):1–7.