UniMelb at SemEval-2019 Task 12: Multi-model Combination for Toponym Resolution

Haonan Li[♠] **Minghan Wang**[♠] **Maria Vasardani**[♡]
Martin Tomko[♡] **Timothy Baldwin**[♠]
♠ School of Computing and Information Systems
♡ Department of Infrastructure Engineering
The University of Melbourne
{haonanl5,minghanw}@student.unimelb.edu.au,
{maria.vasardani, tomkom}@unimelb.edu.au, tb@ldwin.net

Abstract

This paper describes our submission to *SemEval-2019 Task 12 on toponym resolution in scientific papers*. We train separate NER models for toponym detection over text extracted from tables vs. text from the body of the paper, and train another auxiliary model to eliminate mis-detected toponyms. For toponym disambiguation, we use an SVM classifier with hand-engineered features. Our best model achieved a strict micro-F1 score of 80.92% and overlap micro-F1 score of 86.88% in the toponym detection subtask, ranking 2nd out of 8 teams on F1 score. For toponym disambiguation and end-to-end resolution, we officially ranked 2nd and 3rd, respectively.

1 Introduction

Toponym resolution (TR) refers to the task of automatically assigning geographic references to place names in text, which has applications in question answering and information retrieval tasks (Leidner, 2008; Daoud and Huang, 2013; Vasardani et al., 2013), user geolocation prediction (Roller et al., 2012; Han et al., 2014; Rahimi et al., 2015), and historical research (Grover et al., 2010).

This paper describes our system entry to the *Toponym resolution in scientific paper task of SemEval 2019* (Weissenbacher et al., 2019). The task consists of three subtasks: toponym detection, toponym disambiguation, and end-to-end toponym resolution.

For the toponym detection task, we extract tables from the full text and train separate BiLSTM-ATTN models for each. For tables, the model captures the horizontal row-wise structure of the table. For non-table content, the model can capture syntactic and semantic features. In both cases, we use a deep contextualized word representation — ELMo (Peters et al., 2018) — to represent each token. After detecting toponyms, we use an organization name detection model to eliminate mis-detected toponyms that are actually part of an organization name. For the toponym disambiguation task, we first construct a candidate set by searching toponyms on GeoNames.[1] Then, we manually construct features based on search results, and finally, train an SVM model to disambiguate the locations. For the end-to-end resolution task, we pipeline the two aforementioned steps.

Our work makes the following contributions:

- we show that training separate models for table and non-table portions of the paper is better than simply training one model over the full text;
- we show that contextualized word representation boosts performance;
- we show that auxiliary organization name recognition model is helpful for toponym detection, and better than training a single named entity recognizer (NER).

2 Toponym Detection

Figure 1 shows our workflow on the toponym detection task, which consist of 4 parts: (a) pre-processing, which contains tokenization, table extraction and sentence segmentation; (b) training and inference for the toponym detection model; (c) post-processing, to combine detected words into toponyms; and (d) refinement of the results by incorporating an auxiliary model.

2.1 Pre-processing

Tables are ubiquitous in scientific articles, and differ in structure to text in the body of the paper, in

[1]GeoNames, https://www.geonames.org/ is a freely available global placename database.

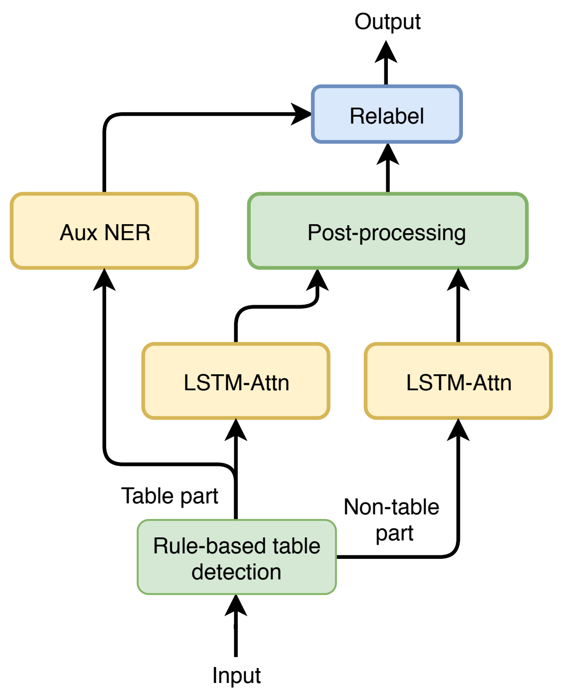

Figure 1: Toponym detection workflow

terms of syntactic structure. As such, training a single text embedding model over both the main body of text and tables will likely lead to suboptimal representations, leading us to train separate models for: (1) tables, and (2) the remainder of the text content of the paper. To extract tables from the plain text dump provided by the shared task organisers, we use a rule-based table detection method.

We first tokenize the entire article, as part of which we treat all punctuation as a separator. In the process of table extraction, we process the raw text line-by-line rather than performing sentence tokenization. We treat numbers, OOV tokens (using GloVe vocabulary), |, and - as table elements, and consider lines with more than 70% of table elements to be table rows. Three or more consecutive table rows are considered to make up a table. In this way, we extract tables from the plain text dump of the articles. Note that the original PDF versions of papers were not made available by the task organizers, meaning that it wasn't possible to use vision-based methods to identify tables.

For the remainder of the text dump not detected as tables, we perform tokenization, remove hyphens caused by line breaks, and then perform sentence segmentation using SpaCy.[2] Sentences that are shorter than 5 tokens in length are concatenated with the preceding and proceeding sentences to make up a single sentence. By expanding short sentences, richer context can be exploited by both ELMo and the RNN-based model.

[2]https://spacy.io

2.2 Contextual Representation

We use ELMo (Peters et al., 2018) word representations in this paper, which are learned from the internal states of a deep bidirectional language model (biLM), pre-trained on a large text corpus. ELMo representations are purely character-based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training. They are also robust to syntactic disfluencies caused by the fine-grainedness of word segmentation. For the purposes of empirical comparison, we also report on experiments using GloVe (Pennington et al., 2014) embeddings.

2.3 Models

For toponym extraction in the table part, we experimented with two kinds of models. The first is a token-level model which is described in Magge et al. (2018). In this model, each training instance consists of an input word, the word's context, and a label indicating whether the word is a part of a toponym. The context of the word is formed by the words in its neighbourhood, which is a window of words centred on the given word. We experimented with two- and three-layer feed-forward models.

The second model is built with RNN and self-attention (Vaswani et al., 2017). Although an RNN is able to make predictions over long sequences, the documents in this task are too long for an RNN, and at the same time, the size of the training data is not sufficient to train an RNN. As such, we split each document with several sentences and make predictions on separate sentences (hidden states are not passed through sentences). We use a two-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to capture the sequential information of the body and table contents, and use self-attention to enhance the connection of each token in the line. We consider the paper body content to have semantic information which can be captured by a sequential model like a BiLSTM. However, for tables, it is not clear that sequential information across cells in a table row should be processed as a sequence. Therefore, we use self-attention to learn the table structure over an entire line. We consider each sentence as a matrix which we denote as L, where $L \in \mathbb{R}^{t \times h}$; t represents the number of tokens in the line, and h is the dimensionality of the embedding representation. To improve training efficiency, we pack l lines into a single batch, thereby making L become a three-dimensional tensor $L \in \mathbb{R}^{l \times t \times h}$. We pad short lines to make the same length as the longest line in a batch, and set the embedding of each padding word to a zero vector with dimensionality h.

We first encode each line with a two-layer bi-directional LSTM, denoted as:

$$L' = \text{BiLSTM}(L) \tag{1}$$

Then, we feed L' into the attention model to encode structural information. The attention model can be denoted as follows:

$$\begin{aligned} &\text{Attn}(Q, K, V; \theta_Q, \theta_K, \theta_V) = \\ &\text{Softmax}\left(\frac{f(Q;\theta_Q) f(K;\theta_K)^\top}{\sqrt{h}}\right) f(V;\theta_V) \end{aligned} \tag{2}$$

This style of attention is named scaled dot-product attention by Vaswani et al. (2017), where $Q, K, V \in \mathbb{R}^{t \times h}$ represent the query, key, and value, respectively, and can be described as mapping a query and a set of key–value pairs to an output, where the query, keys, values, and output are all vectors. In this model, we use tokens in the same line to represent the query, key, and value, and use the attention function Attn to find self-correlations among them. Meanwhile, in Eqn 2 we define f to be a one-layer feed-forward network with different parameter sets $\theta_Q, \theta_K, \theta_V$, which we denote as $f(X;\theta)$. This allows us to learn the correlation with these three parameter sets. We use the following attention function:

$$L'' = \text{Attn}(L', L', L') \tag{3}$$

Finally, we pass L'' into a 3-layer feed-forward network denoted as g, using layer normalization in each layer to increase the training speed. The output of the feed-forward block is passed into the output layer with a residual connection with L'', denoted as:

$$\hat{y} = \phi(g(L'') + L'') \tag{4}$$

The architecture of the model is shown in Figure 2.

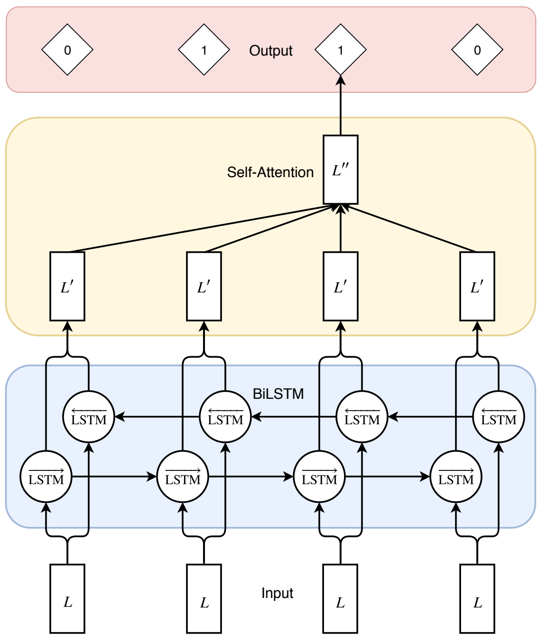

Figure 2: RNN with self attention

2.4 Post-processing

Since we are training a sequence labelling model, result segmentation and combination is necessary. For instance, the sentence *AIV H9N2 was spread to New York, Washington DC and Ottawa* contains three toponyms, and 5 tokens which are contained in those toponyms (e.g. the words *New* and *York* are combined into one toponym). An external gazetteer[3] downloaded from GeoNames, and an in-house place name abbreviation library were used.

We first restore all abbreviations in order to facilitate matching in the gazetteer. We then combine all consecutive tokens that were labelled as a toponym. After this, two different segmentation methods were used: (1) longest string match in the gazetteer; and (2) no segmentation. The result shows that the second method is better because of the limitation of string matching. We think using a better toponym match method like searching via Geonames rather than string matching could achieve better results.

2.5 Auxiliary Model

The single NER model picks up on features such as the word-initial character being uppercase, that are also common in non-toponym named entities, possibly resulting in toponym named entity FPs.

[3] http://download.geonames.org/export/dump/allCountries.zip

Model		Overlap Micro			Strict Micro		
		Precision	Recall	F1	Precision	Recall	F1
Table	BiLSTM	78.11	69.54	73.58	74.91	65.77	70.04
	BiLSTM+Attn	80.27	73.06	76.50	76.63	69.14	72.69
Non-table	BiLSTM	93.37	**90.69**	92.01	89.64	84.55	87.02
	BiLSTM+Attn	93.65	90.38	91.99	90.11	**84.78**	87.36
	BiLSTM+Attn+Aux	**94.66**	90.23	**92.39**	**90.98**	84.50	**87.62**
Single	BiLSTM+Attn+Aux	90.15	85.22	87.62	85.73	71.67	78.07
Combined	BiLSTM+Attn	90.77	86.27	88.46	85.02	72.59	78.31
Combined	BiLSTM+Attn+Aux	91.35	82.83	86.88	84.69	77.48	80.92

Table 1: Performance of different models. “Table” refers to the performance on the table part, and “Non-table” the non-table part; “Single” refers to the single model on the entire article; “Combined” refers to the combination of the two models on table and non-table parts; and “+Aux” refers to the use of the auxiliary model to eliminate misdetected toponyms.

For example, in the phrase *The Royal Melbourne Hospital*, the word *Melbourne* should not be detected as toponym according to the competition setting. This issue was also identified by Dredze et al. (2009).

In this paper, we use two methods to tackle this. The first is to train a single NER to detect toponyms and organization names together. The second is to train an organization name recognizer to correct misdetected toponyms in organization names.

We used the WikiNER (Nothman et al., 2012) dataset to train an organization detection model, and applied it to our dataset. Then we build an organization type set containing *Institute, School, Hospital* etc.. Finally, we re-label toponyms that are part of a corresponding organization name as non-toponyms.

3 Toponym Disambiguation

We used a support vector machine to disambiguate toponyms. For each detected toponym, we first search for it on Geonames, and keep the top 20 records as candidate results. Features are constructed from this, as follows:

- **History Result**: If the toponym appears in the training set, history result refers to the ranking of the number of times the Geonames ID appears as a standard answer. For instance, the toponym *Melbourne* appears 13 times in training, of which 12 occurrence have Geonames ID 2158177 and 1 has ID 7839805, so the history result feature for 2158177 is 1, 7839805 is 2, and all other Geonames IDs are 3.
- **Population**: The ranking of the population of the candidate.
- **GeoNames Feature Codes**: The feature class codes of Geonames records, e.g. *A* represents country, state, region,...; *P* city, village,...; etc.
- **Name Similarity**: The ranking of the string similarity of the toponym and Name item in each record.
- **AncestorsNames Correlation**: The ranking of matching words of the AncestorsNames item in each record.

4 Experiments and Results

4.1 Experiment Setting

The model architecture used for the toponym detection task is depicted in Figure 2. We use the Adam (Kingma and Ba, 2015) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-9}$ and an initial learning rate of $1e^{-3}$. A dropout (Srivastava et al., 2014) rate of 0.5 is used to prevent overfitting. The hidden size (d) of the model is 300, and cross-entropy loss is used for training.

To compare different word embeddings, we use pre-trained 300-dimensional GloVe embeddings and pre-trained 1024 dimensional EMLo embeddings, respectively. We do not update the word embeddings during training.

Embedding	Precision	Recall	F1
GloVe	80.47	75.03	77.65
ELMo	**84.69**	**77.48**	**80.92**

Table 2: Performance of different word representations

4.2 Results

We randomly selected 10 articles from the training set to manually evaluate the table extraction method: 26 out of 27 tables were detected, and 3 non-table parts were misidentified as tables. 62% of tables are exactly accurate, or in other words, 38% tables have some lines misidentified.

Table 1 shows the subtask 1 performance (precision, recall and F1 score) of different models on the table and non-table parts. Strict and overlapping micro measures results are reported. In the strict measure, model outputs are considered to match with the gold standard annotations if they cover the exact same span of text; whereas in the overlapping measure, the model output is considered to match if it overlap in span with the gold-standard. From Table 1, we see that self-attention improves the results on both table and non-table parts, but is particularly effective for the table part. Experimental results on the non-table part are further improved by incorporating the auxiliary model. Finally, the combined model perform is better than a single model on entire articles.

Table 2 shows the subtask 1 performance of different word representations. From that, we find that using ELMo representation is much better than using GloVe embeddings. The reason is that our tokenization method separates many words like *I'm*, *let's*, which ELMo can generate a contextualized representation for, while GloVe cannot. Furthermore, there are many numbers and OOVs in the tables, the GloVe embedding for which is a random 300-dimensional vector that does not provide useful context information.

5 Conclusions

In this work, we presented a method for toponym detection and disambiguation in scientific papers, in the context of Sem-Eval 2019 Task 12, using an LSTM model and SVM model respectively. We extract tables from plain text, and train a dedicated model for each to improve overall performance due to the different structures of tables and the body of text. We also demonstrated the performance of the different models for toponym detection, with our final submission coming in 2nd (among 8).

References

Mariam Daoud and Jimmy Xiangji Huang. 2013. Mining query-driven contexts for geographic and temporal search. *International Journal of Geographical Information Science*, 27(8):1530–1549.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. 2009. Sequence learning from data with multiple labels. In *Proceedings of the ECML/PKDD 2009 Workshop on Learning from Multi-Label Data*.

Claire Grover, Richard Tobin, Kate Byrne, Matthew Woollard, James Reid, Stuart Dunn, and Julian Ball. 2010. Use of the Edinburgh geoparser for georeferencing digitized historical collections. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1925):3875–3889.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based Twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*.

Jochen L Leidner. 2008. *Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names*. Universal-Publishers.

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

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2012. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194:151–175.

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 2014)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2015. Twitter user geolocation using a unified text and network prediction model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 630–636.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1500–1510.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Maria Vasardani, Stephan Winter, and Kai-Florian Richter. 2013. Locating place names from place descriptions. *International Journal of Geographical Information Science*, 27(12):2509–2532.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Annual Conference on Neural Information Processing Systems (NIPS 2017)*, pages 6000–6010.

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