Chemlistem - chemical named entity recognition using recurrent neural networks

P.Corbett[1], J.Boyle[2]

Data Science Group, Technology Dept, The Royal Society of Chemistry.

*[1]pcorbett@rsc.org; [2]boylej@rsc.org

Abstract. Chemical named entity recognition has traditionally been dominated by CRF (Conditional Random Fields)-based approaches but given the success of the artificial neural network techniques known as "deep learning" we decided to examine them as an alternative to CRFs. We present here three systems. The first system translates the traditional CRF-based idioms into a deep learning framework, using rich per-token features and neural word embeddings, and producing a sequence of tags using bidirectional Long Short Term Memory (LSTM) networks – a type of recurrent neural net. The second system eschews the rich feature set – and even tokenisation – in favour of character labelling using neural character embeddings and multiple LSTM layers. The third system is an ensemble that combines the results of the first two systems, achieving an F score of 0.9032 on the test data (precision 0.9002, recall 0.9062).

Keywords. Chemicals, Named Entity Recognition, Deep Learning.

1 Introduction

At the Royal Society of Chemistry the data science group undertakes a variety of text mining data to enrich both our data offerings and our corpus. One common task is chemical named entity recognition, and the group has spent considerable time applying different machine learning algorithms to extract such information. This paper discusses one of these approaches, which uses structured deep learning.

The CEMP (Chemical Entity Mention recognition) task of BioCreative V.5 [1] addresses recognition of chemical named entities in patent text, using a training set of 21,000 patent abstracts and a test set of 9,000 patent abstracts. The corresponding task in BioCreative V [2] was dominated by systems employing Conditional Random Fields (CRF) – there were two rule-based non-CRF systems but no other methods employing non-CRF machine learning approaches to the sequence labeling problem. CRF-based systems, such as the highly successful tmChem system [3] treat a sentence or paragraph as a sequence of tokens, and assign a tag to each token to indicate whether it is part of a chemical name, and its position in the name.

The recent resurgence of artificial neural network techniques known as "deep learning" [4] suggest that these may provide an alternative or a complement to CRFs. Recurrent neural networks offer an approach to sequence labeling, a common approach to natural language processing (NLP) tasks such as part-of-speech (POS) tagging and named entity recognition. One type of network – a variety of Long Short-Term Memory (LSTM) known as a Bidirectional LSTM has achieved state-of-the-art performance on common NLP tasks [5]. In this paper we demonstrate how Bidirectional LSTMs, implemented using the Keras toolkit [6], can be applied to chemical named entity recognition.

Here we discuss two different approaches to LSTM-based chemical named entity recognition, and an ensemble system that combines both. The first system – the "traditional" system - works similarly to traditional CRF-based systems, in that it assigns tags to a sequence of tokens, each token bearing features from a rich feature set. Our "traditional" system differs from those that are CRF-based in a number of ways – for example, our traditional system supplements the feature set with neural word embeddings, and (with a few minor exceptions) does not include information about neighboring tokens in the feature set, instead relying on the neural network structure to carry the information from neighboring tokens to the right place.

The second system – the "minimalist" system - labels a sequence of characters, rather than words (i.e. it does not use a tokeniser), and does not use a rich feature set, instead using character embeddings and multiple LSTM layers in order to induce the equivalent of a feature set internally. In related work, character embeddings have been used in domains where word segmentation is difficult, for example Chinese NLP [7] and text containing programming language snippets [8] – suggesting that this may be particularly suitable for chemical text, where tokenization presents particular difficulties. Finally, the ensemble system examines to what extent the two approaches are complementary.

2 System description and methods

For each of our approaches there was a three step process, involving pre-processing, a neural network step, and finally post-processing.

2.1 Pre-processing

Tokenisation in the traditional system was performed using a modified version of a Python translation of the Oscar4 tokeniser [9]. On the training data only, when an entity boundary was in the middle of a token, the token was split in two. The minimalist system does not use tokenization as such – however it is equivalent to ‘tokenizing’ the data to ‘tokens’ a single character long (including whitespace characters). Tokens (or in the minimalist system, characters) in the training data were assigned SOBIE (sometimes known as BIOES) tags – "O" marking a token not part of an entity, "S" marking a token that is the whole of an entity (a "singleton"), "B" marking a token at the beginning of an entity, "I" marking one inside an entity, and "E" marking one at the end.

For both systems the data was split 80:20 for training and testing.

The traditional system starts with finding those tokens in the corpus that occur more than two times, and assigning initial embedding vectors based on GloVe (a set of pre-trained word vectors based on Wikipedia) [10] – tokens not found in GloVe are given initial embedding vectors full of zeros. Tokens that occur two times or less are all given a single "unknown token" vector, again initialized to zeros.

The traditional system uses a "preclassifier" [11] to judge how likely a token is to be chemical – i.e. assigned an S, B, I or E tag as opposed to O. To train this, the system finds tokens only ever tagged O or only ever tagged SBIE, generates binary features for each of these, selects the 1000 binary features with highest mutual information with O-only vs SBIE-only, and uses those to train a random forest (using scikit-learn [12]) with 100 trees. This is the "preclassifier", and is used for producing scores (probability predictions) for tokens it was not trained on. The system trains an additional 5 preclassifiers each using four fifths of the available tokens, and uses each to produce a score for the tokens in was not trained on. The features for the preclassifier are: word shape, character 4-, 3-, 2- and 1-grams (including start and end markers, so this gets prefixes and suffixes), tests against various regular expressions, and tests to see if the token is in various lexicons (a list of chemicals derived from ChEBI [13], another list from ChemSpider [14], and a standard English word list).

Additionally, there are two sets of features that are sent directly to the neural network. One set includes length-based measures (including the number of all non-lowercase characters, the number of all non-letter characters and the number of digit characters) as numerical features, and binary features for the lexicons and regular expressions above. This set is passed to the network in its entirety. The second set of features consists of the 100 most common binary features selected from 2- and 3-character suffixes and word shapes. Note that none of these features look at neighbouring tokens; unlike in a CRF where features for neighboring tokens are included explicitly, in this system we rely on the neural network to combine features from different tokens.

The features for each token in a sentence (excluding the embeddings) consist of the score from the preclassifier and the three sets of features from the paragraph above.

The minimalist system uses only character embeddings – a set of 90 characters (letters, digits, common punctuation) is used, and unknown characters acting as the 91st character.

2.2 Neural Network

The traditional system has two inputs that merge together. One branch is an embedding layer, with 300 units per token. The other branch takes the various features mentioned in the section above, passes them to a 1D convolution layer, with a window size of 3 (i.e. for each token the layer takes inputs from the previous token, the token itself and the next token), and 256 outputs per token, with a ReLu activation function. These two branches are merged, and the 300+256 outputs per token are used as inputs for a bidirectional LSTM layer, with 64 outputs per token per direction. These outputs are inputs for the final layer – a time-distributed dense layer, with 5 outputs per token (corresponding to S, O, B, I and E tags), with a softmax activation function – this ensures that the outputs for each token sum to 1.

The system was trained for 20 epochs, with the model being saved after each epoch, and evaluated against the remaining 20% of the data. Each epoch was trained in mini-batches, drawn from batches of sentences all the same length. The model from epoch that gave the best F score – the 16th epoch – was selected.

The minimalist system has a single input layer – an embedding layer with 200 outputs per character, followed by three bidirectional LSTM layers – the first with 128 outputs per character per direction, the second and third with 64 outputs per character per direction. The final layer was identical to the final layer in the traditional system.

This system was trained for 30 epochs. As before, the model from the highest-scoring epoch – the 27th – was selected. The same mini-batch training procedure was used, except that for the first four epochs, the system was trained in order of sequence length, with the shortest sequences first.

Both networks were trained with the RMSProp optimizer, using the categorical cross-entropy loss function. The LSTMs in both systems, and the convolutional layer in the first system, were trained using dropout [15] with the dropout probability set to 0.5.

2.3 Post-processing

The neural network assigns five scores to each token (for the minimalist system, for "token" read "character") – one for each of the S, O, B, I and E tags. To convert this to a list of entities, the system scans for possible entities, looking up the value for each tag in each possible entity in each position, taking the minimum value, and, if this is above a threshold, accepting the entity and assigning it that value as a score. The thresholds were 0.5 for the traditional system, 0.6 for the minimalist system, and were chosen by investigating several thresholds and selecting those that maximized the F scores.

We experimented with two variants of the minimalist system – a high-recall run, with the threshold adjusted to 0.05 (the threshold that gave a recall around 0.95 when overlapping entities are counted), and a run with different thresholds depending on whether the possible entities appeared in various lists. The results of these experiments are given in the table below, and are labeled Min+hr and Min+pp respectively.

The ensemble system works by running both systems with a low threshold, and generating two lists of entities. If an entity appears in only one list, its score is the score from that list, otherwise it is the sum of the scores from the two lists. The possibly combined score is then divided by 2, and a threshold of 0.475 is applied – this threshold is just below 0.5, so entities that get a high score from one system but no score from the other may nevertheless be recognized.

This challenge does not allow overlapping entities to be submitted, so in runs where this is a danger, checks are done and the lower-scoring entities are discarded.

3 Results and Discussion

The raw results are as below:

System	**Official Test**			**Internal Evaluation**		
	F	**Preci-sion**	**Recall**	**F**	**Preci-sion**	**Recall**
Traditional	.8919	.8867	.8971	0.8703	0.8648	0.8758
Minimalist	.8901	.8865	.8936	0.8664	0.8479	0.8858
Ensemble	**.9032**	**.9002**	.9062	**0.8807**	**0.8646**	0.8976
Min+pp	.8899	.8960	.8837	0.8701	0.8588	0.8816
Min+hr	.8398	.7777	**.9126**	0.8166	0.7386	**0.9131**

Internal evaluations were performed on the 1/5 of the training data not used for training. It is notable that the official test tended to give better results – presumably due the quality of the test data being higher than that of the training data.

Despite the different methods involved, the traditional and the ensemble system performed similarly. It is possible that the minimalist system could be improved by adding additional layers, adding more outputs per layer, or otherwise increasing the amount of computing power required. The system is slow to train and the parameters were not fully optimized. Due to lack of time, we have not investigated the inner workings of the minimalist system – however we think that the memory in the LSTMs should make them useful for spotting character n-grams.

It is possible to include CRF layers as the final layer of an LSTM-based neural network [5] – however this was not investigated as the toolkit that we were using [6] did not support it.

During development, the postprocessing system had looked promising – however this did not survive the final evaluation. The high recall run was also a disappointment – the BeCalm evaluation script does not allow the submission of overlapping entities, and this puts a cap on the level of recall available. Without this constraint, our internal evaluation gets a recall of 0.9551 (precision 0.6900, F 0.8012). One feature of the minimalist system is that there is no hard limit on recall imposed by the tokeniser – with the traditional system there are some entities with boundaries in the middle of tokens, impossible for the system to get even with the lowest possible threshold, setting a hard limit on recall.

The ensemble system has achieved an F of 0.9032 – above the symbolic "90% barrier". This is not yet human-level performance – for example an inter-annotator agreement study of chemical named entity annotation found that an F of 0.93 is possible [16]. This score was achieved without extensive training or post processing on a relatively simple model. For this reason we feel that this approach has demonstrated considerable promise, and we will continue to investigate. Our systems and their source code available on-line [17].

4 Acknowledgment

We would like to thank Adam Bernard for the initial Python translation of the Oscar4 tokeniser, and Colin Batchelor, Nicholas Bailey and Jeff White for valuable discussions.

REFERENCES

1. Martin Pérez Pérez, Obdulia Rabal, Gael Pérez Rodríguez, Miguel Vazquez, Florentino Fdez-Riverola, Julen Oyarzabal, Alfonso Valencia, Analia Lourenco and Martin Krallinger. Evaluation of chemical and gene/protein entity recognition systems at BioCreative V.5: the CEMP and GPRO patents tracks., p. 3-11
2. Martin Krallinger, Obdulia Rabal, Analia Lourenco, Martin Perez Perez, Gael Perez Rodriguez, Miguel Vazquez, Florian Leitner, Julen Oyarzabal, Alfonso Valencia. "Overview of the CHEMDNER patents task" Proceedings of the Fifth BioCreative Challenge Evaluation Workshop (2015): 63-75.
3. Robert Leaman, Chih-Hsuan Wei, Zhiyong Lu. "tmChem: a high performance approach for chemical named entity recognition and normalization. Journal of Cheminformatics 7(suppl 1):S3 (2015).
4. Yann LeCun, Yoshua Bengio, Geoffrey Hinton. "Deep Learning". Nature 521 (2015): 436-444.
5. Zhizheng Huang, Wei Xu, Kai Yu. "Bidirectional LSTM-CRF Models for Sequence Tagging". arXiv preprint (2015) arXiv:1508.01991.
6. François Chollet. "Keras" (2015) https://github.com/fchollet/keras.
7. Yanan Lu, Yue Zhang, Donhong Ji. "Multi-prototype Chinese Character Embedding" Language Resources and Evaluation Conference (2016).
8. Grzegorz Chrupała. "Text segmentation with character-level text embeddings". Workshop on Deep Learning for Audio, Speech and Language Processing, ICML (2013).
9. David Jessop, Sam Adams, Egon Willighagen, Lezan Hawizy, Peter Murray-Rust "OSCAR4: a flexible architecture for chemical text-mining". Journal of Cheminformatics (2011) 3:41.
10. Jeffrey Pennington, Richard Socher, Christopher Manning. "GloVe: Global Vectors for Word Representation" (2014).
11. Peter Corbett, Ann Copestake. "Cascaded classifiers for confidence-based chemical named entity recognition", BMC Bioinformatics (2008) 9(Suppl 11):S4.
12. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. "Scikit-learn: Machine Learning in Python". Journal of Machine Learning Research 12 (2011), 2825-2830.
13. A., Alcantara, R., Darsow, M., Guedj, M., Ashburner, M.: ChEBI: a database and ontology for chemical entities of biological interest. Nucleic acids research 36(suppl 1), D344{D350 (2008)
14. http://www.chemspider.com/
15. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". Journal of Machine Learning Research 15 (2014), 1929-1958.
16. Peter Corbett, Colin Batchelor, Simone Teufel. "Annotation of Chemical Named Entities". Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing, 57-64.
17. https://bitbucket.org/rscapplications/chemlistem