Deep Learning for Identification of Adverse Effect Mentions in Twitter Data

Paul Barry and Ozlem Uzuner, PhD, FACMI
George Mason University
pbarry2, ouzuner@gmu.edu

Abstract

Social Media Mining for Health Applications (SMM4H) Adverse Effect Mentions Shared Task challenges participants to accurately identify spans of text within a tweet that correspond to Adverse Effects (AEs) resulting from medication usage (Weissenbacher et al., 2019). This task features a training data set of 2,367 tweets, in addition to a 1,000 tweet evaluation data set. The solution presented here features a bidirectional Long Short-term Memory Network (bi-LSTM) for the generation of character-level embeddings. It uses a second bi-LSTM trained on both character and token level embeddings to feed a Conditional Random Field (CRF) which provides the final classification. This paper further discusses the deep learning algorithms used in our solution.

1 Data

The training data consists of 2,367 unique tweets of which 1,212 are positive examples and 1,155 are negative while the evaluation data consists of 1,000 tweets with 500 positive examples and 500 negatives. Of the 1,212 positive examples in the training set, 345 examples present two or more spans within the tweet that are AEs experienced by the individual. The remaining positive examples contain only one AE span. Spans of AEs are not limited to singular words nor are they required to be whitespace delimited. Because of this, many AEs within the data set consist of multiple words. Spans are not limited to English words or whole words, so abbreviations, portions of words, and concatenations of multiple words are expected. Tweets provided to participants had all alphabetical characters converted to their lowercase form. No other preprocessing steps were performed prior to dataset distribution. We divided the training dataset into subsets with 1,657 tweets used for training, 355 for validation, and 355 for testing. We tuned our parameters on the training set and report final results on the shared task evaluation set.

2 Preprocessing

We preprocessed AEs to consolidate overlapping spans and remove AEs that are a subset of others. Subsequently, we replaced twitter handles with "@person" to reduce the noise inherent to multiple tokens sharing the same meaning and to reduce dimensionality. To further reduce dimensionality, we removed the URLs within tweets as they do not provide contextual value. The hashtag character, "#", was removed so hashtag words could be treated like regular words rather than as separate, unique tokens. Tokenization was performed and tested using several tokenizers to include the Natural Language Toolkit's (NTLK) Word Tokenizer, NLTK's Word Punct Tokenizer, NLTK's Whitespace Tokenizer, and the Stanford Tokenizer (Manning et al., 2014; Bird et al., 2009). Lastly, the removal of all special characters was evaluated in conjunction with each of the above methods.

3 System Structure

The system used in this study is based around a Recurrent Neural Network (RNN) variant known as a bi-LSTM which features the Long Short-term Memory (LSTM) unit (Hochreiter and Schmidhuber, 1997). The system consists of four layers: a character embedding layer, a token embedding layer, a label prediction layer, and a label sequence optimization layer (Dernoncourt et al., 2017). As input, it uses three portions of the dataset for training, validation, and testing. Input to the bi-LSTM consists of word embeddings. We initialized word embeddings based on pretrained GloVe embeddings (Pennington et al., 2014). We then used ELMo to continue training embeddings so they better represent each word's usage within the corpus (Peters et al., 2018). The trained word embeddings are augmented by training a bi-LSTM model on individual characters within a word and concatenating the character embeddings onto the word embedding vector. These character-enhanced token-embeddings are then passed as input into a second bi-LSTM layer in which both directions predict the label. The output from both directions is concatenated and passed to a CRF which provides the model's final prediction (Dernoncourt et al., 2017).

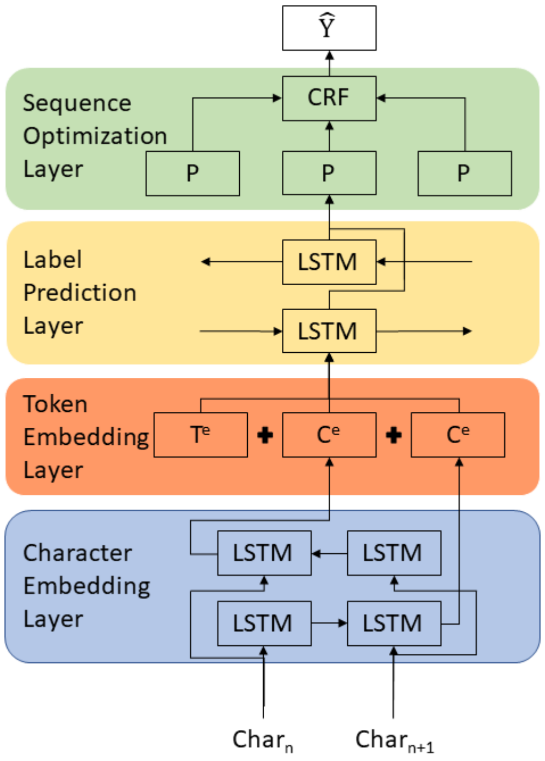

Figure 1: The character-enhanced bi-LSTM CRF system architecture. Where T^e is the token embedding, C^e are the character embeddings, and P is the bi-LSTM's predicted class.

4 Training

The hyperparameters that yield the best results were identified as: character embeddings with 25 dimensions, character level LSTM hidden states that use 25 dimensions, token embeddings with 100 dimensions, token level context embeddings with 1,024 dimensions, and a token level hidden state that uses 100 dimensions. We limited the model to 100 epochs with early stopping when the validation set's F1 score did not improve after 10 epochs. Early stopping was triggered when the model's F1 score on the validation set peaked then failed to achieve a better score within ten more epochs. We used a learning rate of 0.005. We clipped gradients at 5.0 and applied a dropout rate of 0.5. We tested several other hyperparameters with the model to include 200 dimension token embeddings, 2,048 context embeddings, 0.001 learning rate, 0.4 and 0.6 dropout rates. None of these provided significant increases in performance, however, some did cause large increases in training and inference times. Using a 16 core CPU, word embeddings are trained in 8 minutes and 43 seconds and training the model takes 19 minutes and 22 seconds. Due to the small data set size, only 3GB of free RAM is necessary to train the system.

5 Evaluation and Results

We measured performance of the system based on provided gold label AEs. We used Precision, Recall, and F1 Score to monitor a model's performance as it trained and to check that the reported values were reflective of the model's ability to generalize to the test set. Due to the inherently noisy nature of user generated social media text, we found that noise reduction techniques performed during the preprocessing stage had a much higher impact on model performance than hyperparameter tuning. Swapping tokenizers netted performance increases in F1 Score as big as 9.73, when keeping special characters, and 8.07, when not. Table 1 shows that best results on the test set are achieved with NLTK's Word Punct tokenizer and when special characters are kept.

Tokenizers	Special Characters	P	R	F1
Stanford	Yes	35.12%	47.19%	40.27
Stanford	No	33.13%	52.27%	40.55
NLTK Word	Yes	42.08%	58.17%	48.83
NLTK Word	No	**46.09%**	51.46%	48.62
Word Punct	Yes	42.79%	**60.13%**	**50.00**
Word Punct	No	39.52%	53.25%	45.27
Whitespace	Yes	32.90%	58.22%	42.04
Whitespace	No	44.29%	52.92%	48.22

Table 1: System performance on test set with different tokenizers.

The shared task was evaluated using a total of six performance metrics including both strict and relaxed variants of Precision, Recall, and F1 Score. Table 2 shows that our final system provided a 59.7 Relaxed F1 Score and a 40.7 Strict F1 Score on the evaluation set, beating shared task averages by 5.9 and 9.0, respectively.

Metric	Our System	Task Average
Relaxed Precision	59.6%	51.3%
Relaxed Recall	59.9%	61.7%
Relaxed F1 Score	59.7	53.8
Strict Precision	40.6%	30.3%
Strict Recall	40.7%	35.8%
Strict F1 Score	40.7	31.7

Table 2: System Performance on evaluation set.

Error analysis shows that words heavily associated with AEs, such as “withdrawal”, are almost always accurately identified as being AEs. Alternatively, words with neither positive nor negative connotations are frequently missed as being AEs, such as “sleep” in “it could be two months before i sleep well again”. Errors also occurred when tokens frequently associated with AEs were present but not in relation to medication usage. An example would be the identification of “rejection hurts” in “rejection hurts, cymbalta can help”. The model appears to give excessive weight to the specific word being used while not giving enough weight to the word’s context. Future work would explore the use of a larger corpus that includes more negative examples of those words, additional LSTM layers in the label prediction layer, and the use of more recent word embedding algorithms.

References

Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits (2017). De-identification of patient notes with recurrent neural networks. *Journal of the Amerian Medical Informatics Association, volume 24* (Issue 3), 596-606.

Sepp Hochreiter and Jurgen Schmidhuber (1997). Long Short-term Memory. *Neural Computation, volume 9* (Issue 8), 1735–1780.

Edward Loper, Ewan Klein, and Steven Bird (2009). *Natural Language Processing with Python*. O’Reilly Media.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55-60.

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*, 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, volume 1.*

Davy Weissenbacher, Abeed Sarker, Arjun Magge, Ashlynn Daughton, Karen O'Connor, Michael Paul, and Graciela Gonzalez-Hernandez (2019). Overview of the Fourth Social Media Mining for Health (SMM4H) Shared Task at ACL 2019. In *Proceedings of the 2019 ACL Workshop SMM4H: The 4th Social Media Mining for Health Applications Workshop & Shared Task.*