Stance Detection for the Fake News Challenge: Conditional Encoding With Composite Weight-Sharing Long Short-Term Memory

Chengyao Fu
Cheriton School of Computer Science
University of Waterloo
chengyao.fu@uwaterloo.ca
Github: https://github.com/chuckfuv/MSCI641

Abstract

The paper is designed for the project for MSCI641(text analytic) at University of Waterloo. The project is proposed by the Fake News Challenge organizers and it asks participants to perform stance detection – i.e. identifying whether a particular news headline agrees with, disagrees with, discusses, or is unrelated to a particular news article. In this paper, I implemented 3 LSTM-based models to perform conditional encoding for the headline and body texts.

1 Introduction

Fake news, defined by the New York Times as a made-up story with an intention to deceive 1, often for a secondary gain, is arguably one of the most serious challenges facing the news industry today. With the development of more sophisticated language model, the model like GPT-2(Radford and Wu, 2019) now is able to generate fake news that even human cannot distinguish.

For my MSCI641 final project, I decided to do some research on this subject and try to find a better solution for the text classification task posted by the Fake New Challenge(FNC). The goal of this challenge is to explore how the natural language processing model, especially with the presence of deep learning, will combat this problem. However, detecting the fake news is a complicated task and It is challenging even for trained experts, so the task is broken down into several stages. The very first stage of this task is to understand what other news organizations are saying about his topic, called stance detection(FNC-1). Stance detection of FNC-1 focuses on estimating the relative perspective between a headline and a body of text from news articles and determine what relationship exits between the two. The possible classes are as followed:

- The body text **agrees with** the headline
- The body text **disagrees with** the headline
- The body text **is a discussion on** the headline, but does not take a position
- The body text **is unrelated** the headline

The success of this classifier can be used as a building block for future more complicated model to detect the truthfulness of the articles.

2 Background and Literature Review

With the prevalence of deep neural network, NLP researchers are able to develop more robust language models to understand the natural language. Typically, language models can learn latent representations for words, such as word2vecword2vec, GloVe(GloVe, 2014) where it learns vector representations for words and preserve their semantic relationships between words. Another popular model to learn representations for sequences of words is recurrent neural network(RNN) where it acts like memory unit to keep the information of the order and contents of words in a sequence. The 2 popular variations of RNN are long short-term memory(LSTM)(LSTM, 1997) and gated recurrent unit(GRU).(GRU, 2014)

A simple baseline model is introduced by the Fake News Challenge organizers. It used hand-coded features and a GradientBoosting classifier and it is also available on Github.

Ali K. Chaudhry (stanford, 2017) also utilizes different models to solve this problem including multi-layer feed-forward network, LSTM with independent, conditional, and bidirectional conditional encoding, and also seq2seq with attention.

(seq2seq, 2015) They found that LSTM with conditional encoding outperforms all other models, while the addition of bidirectionality seems to only have marginal improvements.

3 Approach

The work of previous study (stanford, 2017) shows that LSTM-based conditional encoding performs well on the stance detection task so I will try to improve the performance by adding more complexity to the structure. I have built four models to compare their performance on the FNC dataset, including:

- Baseline model implemented by Fake News Challenge organizers
- Conditional encoding with LSTM
- Conditional encoding with composite LSTM
- Conditional encoding with composite weight-sharing LSTM

The following subsections give an overview of the models that I have mentioned.

3.1 Baseline Model

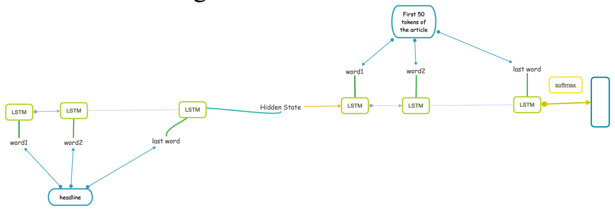
A simple baseline using hand-coded features and a GradientBoosting classifier is available on Github.

The baseline implementation also has codes for pre-processing text, splitting data carefully to avoid bleeding of articles between training and test, k-fold cross-validation, scorer. The hand-crafted features include word/n-gram overlap features, and indicator features for polarity and refutation.

3.2 Conditional encoding with LSTM

This is the basic model that is introduced by Ali K. Chaudhry.(stanford, 2017) It uses different recurrent units(LSTM) to encode headlines and body text separately. I first sent the headline to the first LSTM($LSTM^{headline}$) and then initialize another LSTM($LSTM^{bodyFirst}$) with the final cell and hidden state of the $LSTM^{headline}$(conditional encoding) and also sent the first 50 tokens of the body to the $LSTM^{bodyFirst}$ while training, followed by a simple softmax layer to do the classification. In this way, the LSTM somehow learns the encoding for the heading and combine it with the encoding for the article body and understands the relationship between them during training The structure shown in fig 1:

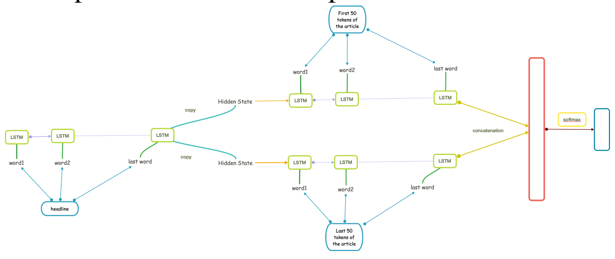
Figure 1: This is the structure of the simple conditional encoding with LSTM. The initial state of the $LSTM^{bodyFirst}$ comes from the last state of the $LSTM^{headline}$ and this structure is called conditional encoding.



3.3 Conditional encoding with Composite LSTMs

The simple conditional encoding only considers the first 50 tokens of the body articles, this may lose a lot of information about the body test since the average number of tokens is 350, so I introduce another LSTM, $LSTM^{bodyLast}$to learn the encoding for the last 50 tokens of the body articles and initialize it with the last state of the headline $LSTM^{headline}$ and concatenate the outputs of both $LSTM^{bodyFirst}$ and $LSTM^{bodylast}$ (composite LSTMs) followed by a softmax layer to do the classification. Compared to the model above, this new model takes more information as inputs so It is expected to learn more details about the body text and the relationship. The model structure is described in fig 2:

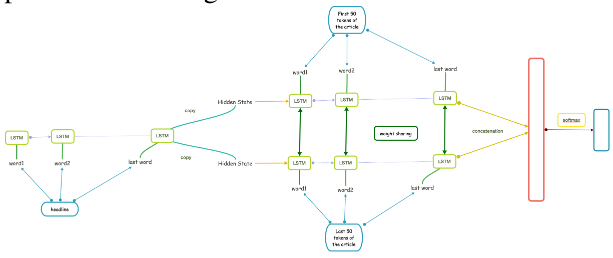
Figure 2: This is the structure of the conditional encoding with composite LSTM. The initial state of both the $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$ come from the last state of the $LSTM^{headline}$ and the outputs of both the $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$ are concatenated as a latent vector to predict the relationship.



3.4 Conditional encoding with Composite weight-sharing LSTMs

The composite LSTMs model in the last section introduced an additional LSTMs to learn the last 50 tokens of the body article separately. However, this also doubled the parameters of the model, which can easily lead to overfitting and prevent the model from learning anything. One way to tackle this problem is to make $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$ sharing their weights so that the number of parameters shrinks by half but inputs remain the same and the first 50 tokens and the last 50 tokens of the body article should come from the same distribution so it makes sense to have $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$ having the same weights. The model is expected to be more robust than one introduced in the last section. The model structure is described in fig 3:

Figure 3: This is the structure of the conditional encoding with composite weight-sharing LSTMs. The initial state of both the $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$ come from the last state of the $LSTM^{headline}$ and the outputs of both the $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$ are concatenated as a latent vector to predict the relationship. The weights of $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$ are shared between each others to prevent overfitting



4 Experiment

This section will discuss the dataset I use to train the model, the hyperparameters I used for the model, the evaluation metric and the results for different model.

4.1 Dataset

The dataset I used is provided by the Fake News Challenge organization, and it is available at Github. The training set contains 49972 samples. Each sample contains:

- A headline that is to compare with a corresponding article body. The average length of the headline is 11.
- The ID number of an article body text corresponding to the headline and the ID can be used to look up the actual text in a separate file. The average length of the article is round 300 tokens.
- The true stance of the headline concerning the article. The stance is one of the four classes mentioned above(agree, disagree, discuss, and unrelated.)

The detailed breakdown is as followed:

rows	**unrelated**	**discuss**	**agree**	**disagree**
49972	0.73131	0.17828	0.0736012	0.0168094

Table 1: Breakdown of the true class for training set.

4.1.1 Validation set

In this paper, I used 2 kinds of validation sets

- **SET 1** I randomly split the original training set into training and validation sets, which means the headlines in the validation set can have overlap with training set as well.
- **SET 2** I purposely split the original training set into training and validation sets where validation set and training set do not have overlapped headline.

4.2 Hyperparameters

There are a lot of hyperparameters in this model to be tuned but LSTM is really expensive train so I have only tried some combinations of the hyper parameters and so this is guaranteed to be the optimal solution.

4.2.1 Token Length

I choose the length for headline to be 15 since 90% of the headline is covered by this length and I choose the first 50 tokens of the body text and last 50 tokens of the body text because 90% of the body texts have a token length that is larger than 101. I have also tried body length of 40, 50 and 60 but 50 gives the best result. Another intuition is that people only focus on first several sentences when they are doing this pairing job since they provided the most information. I truncated or padded headlines that are longer or shorter than the 15 tokens and truncated or padded body texts that are longer or shorter than the 50 tokens.

4.2.2 Hidden Units for the LSTMs

This hyper parameter seems to influence the performance the most. Large number of hidden units make the model easily to overfit the training dataset while small dimension cause the model underfits, so I have tried 80, 100, 150, 200, 300 and 800 units and the 300 gives the best results within a reasonable training time. However, this does not mean that the 150 units are the best choice for our task since I have only tried limited number of choice due to the intensive demand for computing power for even training one LSTM.

4.2.3 Learning Rate and Running Epoch

I did not specify the learning rate since I used adam(adam, 2014) as the way of optimization and it uses an adaptive learning rate to train the model, preventing it from taking too large or small steps. In all the experiments I had during training, all of the models seems to converge after 300 epochs.

4.3 Evaluation Metric

4.3.1 Weighted accuracy

The evaluation I used is the same as the one that proposed by the Fake News Challenge organization: a weighted, two-level scoring system:

- **Level 1**: Classify headline and body text as related or unrelated 25% score weighting.
- **Level 2**: Classify related pairs as agrees, disagrees, or discusses 75% score weighting.

The reasons behind this metric are deciding whether the pair is related or unrelated is both easier and less relevant to Fake News Detection, which has been given fewer weights while detecting the stance is both harder and more relevant in the Fake News Detection taskṪhus, it has been given more weights.

4.4 Results

This section will present the results for each model I introduced above(including baseline) and compare their performance on the set 1, set 2 and competition set using the evaluation metric mentioned in the last section.

4.4.1 Baseline Model

Results on set 2 With the features and a gradient boosting classifier in the baseline model gives Score: 3538.0 out of 4448.5 (79.53%)

	agree	**disagree**	**discuss**	**unrelated**
agree	118	3	556	85
disagree	14	3	130	15
discuss	58	5	1527	210
unrelated	5	1	98	6794

Table 2: Confusion Matrix for the Baseline model implemented by the Fake News Challenge organizers across the four class labels, with raw counts and percentages respectively.

The following is the results of the competition set using the model: Score: 8761.75 out of 11651.25 (75.20%)

	agree	**disagree**	**discuss**	**unrelated**
agree	173	10	1435	28
disagree	39	7	413	238
discuss	221	7	3556	680
unrelated	10	3	358	17978

Table 3: Confusion Matrix of the competition set for the baseline model across the four class labels, with raw counts and percentages respectively.

4.4.2 Conditional Encoding with Simple LSTMs(SimLSTMs)

Results on set 1 The Conditional Encoding with Simple LSTMs introduced above gives Score: 2217.5 out of 2253.75 (98.39156960621187%) on set 1 The confusion matrix is in table 4:

	agree	**disagree**	**discuss**	**unrelated**
agree	343	12	6	1
disagree	8	71	0	0
discuss	3	1	889	5
unrelated	7	1	23	3628

Table 4: Confusion Matrix of set 1 for the Conditional Encoding with Simple LSTMs across the four class labels, with raw counts and percentages respectively.

The following is the results of the competition set using the model: Score: 7767.25 out of 11651.25 (66.66452097414441%). The confusion matrix is in table 5

	agree	**disagree**	**discuss**	**unrelated**
agree	909	41	318	635
disagree	196	80	98	323
discuss	807	80	2284	1293
unrelated	999	119	794	16437

Table 5: Confusion Matrix of the competition set for the Conditional Encoding with simple LSTMs across the four class labels, with raw counts and percentages respectively.

Results on set 2 The results on set 2 are: Score: 3788.75 out of 4448.5 (85.16915814319434%) The confusion matrix is in table 6

	agree	**disagree**	**discuss**	**unrelated**
agree	555	11	129	67
disagree	62	70	23	7
discuss	235	70	1394	101
unrelated	170	18	161	6549

Table 6: Confusion Matrix of set 2 for the Conditional Encoding with Simple LSTMs across the four class labels, with raw counts and percentages respectively.

The following is the results of the competition set using the model: Score: 7570.25 out of 11651.25 (64.9737152666023%). The confusion matrix is in table 7

	agree	**disagree**	**discuss**	**unrelated**
agree	904	22	310	667
disagree	215	47	103	332
discuss	838	54	2154	1418
unrelated	1104	54	875	16319

Table 7: Confusion Matrix of the competition set for the Conditional Encoding with Simple LSTMs across the four class labels, with raw counts and percentages respectively.

4.4.3 Conditional Encoding with Composite LSTMs (CompositeLSTMs)

Results for set 1 Score: 2211.5 out of 2253.75 (98.12534664448143%)

The confusion matrix is in table 8

The following is the results of the competition set using the model: Score: 7602.0 out of 11651.25 (65.24621821692952%). The confusion matrix is in table 9

	agree	**disagree**	**discuss**	**unrelated**
agree	342	8	8	4
disagree	9	69	0	1
discuss	7	1	886	4
unrelated	9	2	23	3625

Table 8: Confusion Matrix of the set 1 for the Conditional Encoding with Composite LSTMs across the four class labels, with raw counts and percentages respectively.

	agree	**disagree**	**discuss**	**unrelated**
agree	847	35	333	688
disagree	210	36	110	341
discuss	980	121	2163	1200
unrelated	1043	56	815	16435

Table 9: Confusion Matrix of the competition set for the Conditional Encoding with Composite LSTMs across the four class labels, with raw counts and percentages respectively.

Results for set 2 The Conditional Encoding with Composite LSTMs introduced above gives overall 84.33% accuracy for the hold-out set. The confusion matrix is in table 10

	agree	**disagree**	**discuss**	**unrelated**
agree	488	33	177	64
disagree	37	80	32	13
discuss	221	102	1393	84
unrelated	135	36	167	6560

Table 10: Confusion Matrix of set 2 for the Conditional Encoding with Composite LSTMs across the four class labels, with raw counts and percentages respectively.

The following is the results of the competition set using the model: Score: 7507.75 out of 11651.25 (64.4372921360369%). The confusion matrix is in table 11

4.4.4 Conditional Encoding with Composite Weight-sharing LSTMs (WeightSharingLSTMs)

Results for set 1 Score: 2220.25 out of 2253.75 (98.51358846367165%). The confusion matrix is in table 12.

The following is the results of the competition set using the model: Score: 7805.5 out

	agree	**disagree**	**discuss**	**unrelated**
agree	770	48	339	746
disagree	194	35	101	367
discuss	757	78	2135	1494
unrelated	813	86	696	16754

Table 11: Confusion Matrix of the competition set for the Conditional Encoding with Composite LSTMs across the four class labels, with raw counts and percentages respectively.

	agree	**disagree**	**discuss**	**unrelated**
agree	346	6	5	5
disagree	6	72	0	1
discuss	6	1	889	2
unrelated	8	1	21	3629

Table 12: Confusion Matrix of the set 1 for the Conditional Encoding with Composite weight sharing LSTMs across the four class labels, with raw counts and percentages respectively.

of 11651.25(66.99281193005042%).The confusion matrix is in table 13

	agree	**disagree**	**discuss**	**unrelated**
agree	871	38	354	640
disagree	174	67	102	354
discuss	949	84	2305	1126
unrelated	959	74	767	16549

Table 13: Confusion Matrix of the competition set for the Conditional Encoding with Composite weight sharing LSTMs across the four class labels, with raw counts and percentages respectively.

results for set 2 The Conditional Encoding with composite weight-sharing LSTMs introduced above gives overall 86.01% accuracy for the set 2. The confusion matrix is in table 14

The following is the results of the competition set using the model: Score: 7904.75 out of 11651.25 (67.84465186138826%). The confusion matrix is in table 15

4.4.5 Comparison

I also summarize the results in the table 16

The table shows that the Conditional Encoding with Composite Weight-sharing LSTMs outperforms all other models we mentioned on set 1, set 2 and the scores for competition dataset.

	agree	**disagree**	**discuss**	**unrelated**
agree	528	15	174	45
disagree	24	100	26	12
discuss	218	66	1421	95
unrelated	127	12	172	6587

Table 14: Confusion Matrix of set 2 for the Conditional Encoding with Composite Weight-sharing LSTMs across the four class labels, with raw counts and percentages respectively.

	agree	**disagree**	**discuss**	**unrelated**
agree	871	38	354	640
disagree	174	67	102	354
discuss	949	84	2305	1126
unrelated	959	74	767	16549

Table 15: Confusion Matrix of the competition set for the Conditional Encoding with Composite weight sharing LSTMs across the four class labels, with raw counts and percentages respectively.

It won out the conditional encoding with simple LSTMs because it added additional 40 tokens for the LSTMs to learn, which benefits the performance by feeding more information to the model. I also expected the conditional encoding with composite weight-sharing LSTMs performs better than the one without weight sharing, and the results confirmed my hypothesis: the weight-sharing model has slightly higher accuracy than the non-weight-sharing one. The possible explanations for that might be the nature of weight sharing reduced the trainable parameters by half immediately; in other words, this technique implicitly double the dataset since more parameters require more data to train. Using weight sharing also makes sense in the perspective of understanding language. The first 50 tokens and last 50 tokens are both from the body texts so it is reasonable to assume they are from the same distribution and can be encoded using the same set of parameters.

5 Conclusion

This section includes discussion for the models, limitation and future works.

5.1 Discussion

The results we got above through the experiments showed clearly that RNN-based(LSTMs) models have performance edge over the baseline model on

	accuracy for set 1	**accuracy for set 2**	**scores on leader-board**
Baseline		79.53%	8761.75
Sim LSTMs	98.39%	84%	7767.25
Composite LSTMs	98.12%	84.33	7805.5
Weight Sharing LSTMs	98.51%	86.01%	7904.75

Table 16: Comparison for all the models

the stance detection task since the model reserves the order of the sequence of words and learned a robust memory unit for a certain language distribution. Moreover, the paper (stanford, 2017) suggests that adding conditional encoding delivers a significant boost in prediction performance. The conditional encoding allows the model to learn not only the representations of headline and body texts individually but also the relationships between them, which boosts the performance largely compared to the other forms of the model where it learns the representation in parallel.
Based on the works they had, I firstly introduced the conditional encoding on composite LSTMs by simply adding another LSTM, $LSTM^{bodyLast}$ to encode the last 50 tokens of the body test and the $LSTM^{bodyLast}$ is also condition on the encoding for the headline (output of $LSTM^{headline}$).Finally, I concatenated outputs of $LSTM^{bodyFirstst}$ and $LSTM^{bodyLast}$ and feed it to a softmax classifier. The new model further improves the accuracy by acquiring more information from the content in body text, but this model also increases the trainable parameters enormously, which can make the mode prune to overfit the training data.
In order to solve the problem mentioned above, I made a modification on the previous model simply by sharing the weights between $LSTM^{bodyFirst}$ and $LSTM^{bodyLast}$. In this way, the size of the parameters shrinks by half immediately, which enlarge the training set implicitly and made the model more robust.

5.2 Limitation

In this subsection, I will mention the limitations of the project.

- **RNN is slow to optimize:** The computation of the hidden state depends on the hidden state from the last time step. The optimization method for this structure is called 'back propagation through time', which optimize the model step by step in a sequence and this made it hard for parallel computing and cannot utilize the GPU to its full potentials.
- **LSTM tends to forget long-distance relationship:** The forget cell in the LSTM can easily lose information for relationship between distant words when the sequence is too long.
- **Training set is small:** Neural network model seems to performs better on bigger dataset and within the scope of this project, the dataset is limited and any data augmentations are forbidden so the results might not fully show the potentials of the models.
- **pre-trained word embedding model:** For this project,I downloaded the GloVe model online with 300 dimensions. This model is trained on a large corpus, which produces a robust representation of words, however, this also leads a lot of out-of-vocabulary words and this may lose a lot of information of the original texts.
- **Hyperparameters tuning:** Training an RNN is computationally expensive and time-consuming, so I did not quite explore many combinations of the hyperparameters due to the limited time of the course schedule.

5.3 Ablation Study

Since the model did not perform well on the competition set but it did very well on the validation set, it might be the sign that the model overfits the training set. I just reduced the number of the hidden units to 150, 100, and 50 respectively. The results did not improve a lot so the hypothesis I have is the competition set and the training set are from 2 completely different distribution so the LSTM learns the distribution in training set but it does not apply well on a different distribution.

5.4 Future Work

In this subsection, I will discuss some possibilities for future work

- **Transfer to Transformer-based model:** RNN is slow to train and cannot fully utilize the benefits of GPU, so it is reasonable to use transformers as the base model for conditional encoding. In this way, the model may capture more long-distance relationship.
- **Customized word embedding:**The project used pre-trained word embedding and it is worth trying to train a customized word embedding to see whether it will improve the performance
- **Collect more data:**Machine learning model is proven to perform better by training on a larger dataset so it is definitely beneficial to acquire more data for the training set, especially for the less frequent classes, like agree and disagree.
- **Integrate other models to vote for the class:**In this project, I have only tried the deep neural network method. It is also reasonable to train several traditional NLP models with word frequency, topics or other features and representation for documents and every model votes for a class and the final model decides a final class in an ensemble way.
- **Use sub-word embedding instead of word embedding:**Sub-word embedding like BPEmb (BPEmb, 2017) is base on byte-pair encoding and trained on wikipedia. The method is designed to overcome the out-of-vocabulary issues in the traditional word2vec model. It splits a word into subwords and transforms this character sequence into a vector representation as same as what the word embedding does.
- **Use the most relevant sentences in body text instead of the first 50 tokens**In this project, I use the first 50 and the last 50 tokens to represent the body text,however, this can not be true for all text body, so instead, we can tokenize the body article into sentences and find the top k sentences that are most relevant to the headline by calculating the cosine similarity between the tf-idf representations of the headline and sentences.

References

[Radford and Wu2019] A Radford, J Wu, R Child, D Luan, D Amodei 2019. *Language models are unsupervised multitask learners*, OpenAI, US.

[GloVe2014] Jeffrey Pennington and Richard Socher and Christopher D. Manning 2014. *GloVe: Global Vectors for Word Representation.*

Empirical Methods in Natural Language Processing (EMNLP)

[LSTM1997] Sepp Hochreiter and Jrgen Schmidhuber. 1997. *Long Short-Term Memory.*

Neural Comput. 9, 8 (November 1997), 1735-1780.

[GRU2014] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. 2014. *On the properties of neural machine translation: Encoder-decoder approaches.*

arXiv preprint arXiv:1409.1259, 2014

[adam2014] Diederik P. Kingma, Jimmy Ba 2014. *Adam: A Method for Stochastic Optimization.*

Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015

[seq2seq2015] Thang Luong, Hieu Pham, Christopher D. Manning 2015. *Effective Approaches to Attention-based Neural Machine Translation.*

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing

[stanford2017] Ali K. Chaudhry. 2017. *Stance Detection for the Fake News Challenge : Identifying Textual Relationships with Deep Neural Net.*

[BPEmb2017] Benjamin Heinzerling, Michael Strube. 2017. *BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages.*