# A Hybrid Model to Rank Sentences for Check-worthiness

Rudra Dhar[1], Subhabrata Dutta[1], and Dipankar Das[1]

Jadavpur University, West Bengal, India
{rudradharrd,subha0009,dipankar.dipnil2005}@gmail.com

**Abstract.** In this paper we describe a system submitted to the CLEF 2019 CheckThat Shared Task. We implement an ensemble of a logistic regression model and an LSTM-based neural network model to predict the worthiness of a sentence for fact checking. Our key idea is to train two separate models with high precision and high recall on binary classification task, and then use the binary class probability as a check-worthiness score. Our system achieves a reciprocal rank 0.4419 and mean average precision of 0.1162 for ranking sentences according to their check-worthiness.

## 1 Introduction

With the advent of web technology, mass media have achieved a revolutionary new form. People all around the globe share information with each other to construct their opinion about everything, that too at an unprecedented speed. In this age of communication, quality of information becomes utmost important. Rumors, fake news, malicious tampering of reality can cause substantial amount of economic as well as social calamity now more than ever. This makes fact-checking an essential part of media and information systems. To handle such large amount of information sharing, automation of such a system is necessary if not mandatory.

As Hassan et al. [5] suggest, automatic fact checking can be divided into a two-fold task: identification of check-worthy sentences, and checking their trustworthiness based on some reliable source. In this task, we attempt to build a system which can assign a score to an input sentence indicating its check-worthiness. This score can vary from 0 (not check-worthy) to 1 (fully check-worthy). The organizers provided the required data set comprised of 16421 sentences. These sentences are binary labelled, 0 and 1, corresponding to not check-worthy and fully check-worthy respectively. Out of these 16421 sentences, 440 are labelled as fully check-worthy. We insist on building a system using this training data only. We attempt to build a classifier framework which assigns a probability score to each sentence, and hypothesize that this probability score corresponds

**Table 1.** Feature size for each feature.

| Feature | Size |
|---|---|
| Arc | 4341 |
| Bi-arc | 1448 |
| Tri-arc | 479 |
| Quad-arc | 169 |
| Subjectivity score | 1 |
| Cumulative Entropy | 1 |
| Polarity words | 3 |
| LIX score | 1 |

to the check-worthiness of the sentence. Our framework is an ensemble of two separately trained classifiers: one based on Long Short-Term Memory (LSTM) networks, and another based on Logistic Regression classifier. Our key strategy is to build one model predicting check-worthy sentences with high precision (less false positives) and another one with high recall (less false negatives), so that their ensemble will perform more accurately.

## 2 Logistic Regression Component

To train our logistic regression model, we extract the following features for each sentence:

1. Syntactic N-grams [8] computed using Syntactic N-gram builder[1]; we generate the dependency parse tree of the sentence using Stanford CoreNLP [6] and compute syntactic n-grams for dependecy paths of length one (arc), two (bi-arc), three (tri-arc) and four (quad-arc).
2. Subjectivity score of the sentence, using TextBlob[2].
3. Cumulative Entropy of the sentence as,

$$CE = \frac{1}{|T|} \sum_{t \in T} (tf(t) * (\log(|T|) - \log(tf(t)))$$

   where $T$ is the set of terms in the corpus and $tf(t)$ is the frequency of term $t \in T$ in the sentence.
4. Count of negative, neutral and positive polarity words computed using SentiWordNet [2].
5. LIX score [3] representing the readability of the sentence.
6. Count of named entities present in the sentence.

With this feature set, we train a logistic regression classifier using Scikit-learn.

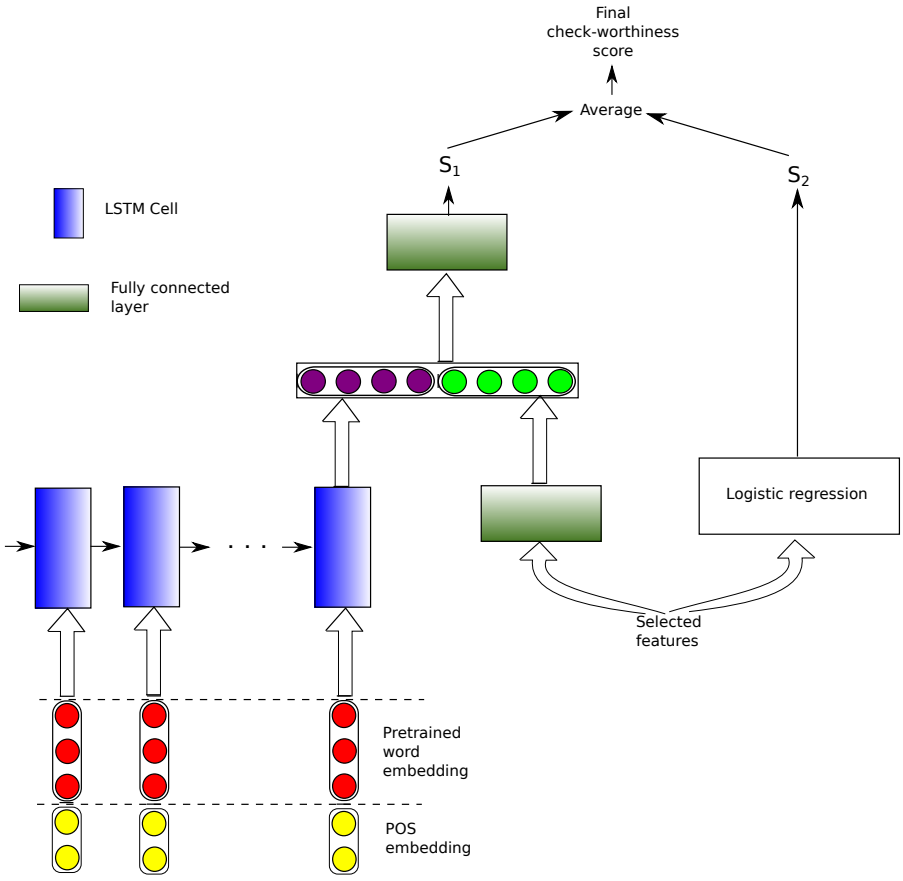


**Fig. 1.** Complete architecture of the system

### 2.1 LSTM component

We use GloVe [7] pre-trained word embeddings of size 300 to be used as word representations. Along with that, we input parts-of-speech tags of the words (tagged using Stanford CoreNLP) as one-hot vectors, which selects randomly initialized POS embeddings. These two embeddings are concatenated and input to an LSTM layer, which learns a many-to-one mapping from the sequence of word-POS tag vectors to a single representation of the sentence. We also use the manually extracted features as an input to a fully connected layer, learning a dense low dimensional representation from the sparse feature vectors. The output of the LSTM layer and the fully connected layer is then concatenated and input to another fully connected layer, which outputs a probability score.

[1] https://github.com/jmnybl/syntactic-ngram-builder
[2] https://textblob.readthedocs.io/en/dev/

**Table 2.** Performance of the individual models on the validation set for positive class (check-worthy sentences).

| | Precision | Recall |
|---|---|---|
| Logistic Regression | 0.07 | 0.78 |
| LSTM | 0.81 | 0.15 |

## 3 Training

As already stated, the LSTM component and logistic regression components are trained separately. For a single sentence, if the output score of the LSTM model is greater than 0.5, then the predicted label is 1, otherwise 0. We train the LSTM model using the Adam optimizer, with a batch size equal to 256.

As the training data-set has high class imbalance, we train the LSTM model with a class-weighted binary cross-entropy loss. We weigh the positive class with weight $w = \log \frac{N_0}{N_1}$ where $N_0, N_1$ corresponds to the number of negative and positive class samples in the training data. The class imbalance automatically puts a bias in the logistic regression model towards the negative class, making its positive predictions highly precise; on the other hand, the weighted loss function forces the LSTM model to bias towards the positive class, resulting in a higher recall.

## 4 Testing

To produce the final check-worthiness score, we average the scores predicted by the two components. We test our system on the test dataset provided by the organizers. Test dataset contains 7080 sentences. In Table 3 we present the evaluation results of our system for various metrics.

**Table 3.** Evaluation results for ranking check-worthiness

| MAP | RR | R-P | P@1 | P@3 | P@5 | P@10 | P@20 |
|---|---|---|---|---|---|---|---|
| 0.12 | 0.44 | 0.11 | 0.29 | 0.19 | 0.17 | 0.17 | 0.13 |

## 5 Conclusion

We made a system to predict the check-worthiness of a sentence for CLEF 2019 CheckThat (Task 1). We did not use any external data. Here we used a combined logistic regression model and an LSTM-based neural network model to get a better probability score for check-worthiness. We achieved a reciprocal rank of 0.4419 and a mean average precision of 0.1162 .

## References


1. Atanasova, P., Nakov, P., Karadzhov, G., Mohtarami, M., Da San Martino, G.: Overview of the clef-2019 checkthat! lab on automatic identification and verification of claims. task 1: Check-worthiness
2. Baccianella, S., Esuli, A., Sebastiani, F.: Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In: Lrec. vol. 10, pp. 2200–2204 (2010)
3. Björnsson, C.H.: Readability of newspapers in 11 languages. Reading Research Quarterly pp. 480–497 (1983)
4. Elsayed, T., Nakov, P., Barrón-Cedeño, A., Hasanain, M., Suwaileh, R., Da San Martino, G., Atanasova, P.: Overview of the CLEF-2019 CheckThat!: Automatic Identification and Verification of Claims. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. LNCS, Lugano, Switzerland (September 2019)
5. Hassan, N., Adair, B., Hamilton, J.T., Li, C., Tremayne, M., Yang, J., Yu, C.: The quest to automate fact-checking. In: Proceedings of the 2015 Computation+ Journalism Symposium (2015)
6. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. pp. 55–60 (2014)
7. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
8. Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., Chanona-Hernández, L.: Syntactic n-grams as machine learning features for natural language processing. Expert Systems with Applications **41**(3), 853–860 (2014)