SimpleLSTM: A Deep-Learning Approach to Simple-Claims Classification

Piyush Chawla[1,2], Diego Esteves[2,3(✉)], Karthik Pujar[1], and Jens Lehmann[2,4]

[1] The Ohio State University, Columbus, USA
{chawla.81,pujar.4}@osu.edu
[2] SDA Research, University of Bonn, Bonn, Germany
{esteves,lehmann}@cs.uni-bonn.de
[3] Farfetch, Porto, Portugal
diego.esteves@farfetch.com
[4] Fraunhofer IAIS, Sankt Augustin, Germany

Abstract. The information on the internet suffers from noise and corrupt knowledge that may arise due to human and mechanical errors. To further exacerbate this problem, an ever-increasing amount of fake news on social media, or internet in general, has created another challenge to drawing correct information from the web. This huge sea of data makes it difficult for human fact checkers and journalists to assess all the information manually. In recent years *Automated Fact-Checking* has emerged as a branch of natural language processing devoted to achieving this feat. In this work, we give an overview of recent approaches, emphasizing on the key challenges faced during the development of such frameworks. We test existing solutions to perform claim classification on simple-claims and introduce a new model dubbed SimpleLSTM, which outperforms baselines by 11%, 10.2% and 18.7% on FEVER-Support, FEVER-Reject and 3-Class datasets respectively. The data, metadata and code are released as open-source and will be available at https://github.com/DeFacto/SimpleLSTM.

Keywords: Fact-checking · Trustworthiness · Evidence extraction

1 Introduction

With the increase in false-fact circulation across different social media platforms, it has become pertinent to validate the claims and statements released online [8, 34]. The terms *Fake-News* and *Junk-News* have gained importance in the last few years, mainly in the context of electoral activities in North America and Western Europe. The false facts (or rumors), in the past, have led to situations like stock price drops and large scale investments [35]. Though social media platforms are the most common breeding grounds for fake-news, it sometimes finds its way into the mainstream media too [15]. The constant skepticism about inflammatory news headlines and rapid growth in the facts that are shared online led to a new

P. Moura Oliveira et al. (Eds.): EPIA 2019, LNAI 11805, pp. 1–12, 2019.
https://doi.org/10.1007/978-3-030-30244-3_21

form of journalism that validates political facts made on the web. Political fact-checking [25] has emerged as an effort to fight against the fake news and validate the claims and alternative facts [15] made in political discourse. Organizations like *PolitiFact*[1], *Snopes*[2] and *Factcheck*[3], for instance, employ journalists that manually check facts and hand annotate them publicly. Though this is a good effort towards curbing false information, the human annotators cannot compete with the speed with which these facts are generated. The content on the internet is increasing every day which also means an increase in the false and incorrect facts. This makes manual validation of facts almost impossible (although its high precision). *Automated Fact-Checking (or Fact-Validation)* is a (relatively) recent effort that tries to automate the process of manual fact-checking. Many different branches of natural language processing have emerged that try to achieve this goal [28]. The fact-checking task involves different perspectives. For instance, *stance detection* aims at detecting whether the author of a piece of text is in favor of the given target or against it. Recent challenges like the Fake News Challenge 2017[4] [13] tackle this problem. Yet another task in the fact-checking domain is the *claim classification* which deals with deciding whether a given claim is true or not based on the evidence. FEVER 2018[5] [31] have proved to be important milestone in this direction. Overall these challenges have worked as a catalyst for the automated fact-checking community, releasing new datasets and defining strong baselines to fuel future research. In this paper we focus our attention on the latter case (*claim classification*).

Fact-Checking at its core can be treated as a *claim classification* problem. It is a holistic approach that extracts *evidence* and then uses its *arguments* to make a decision regarding the claim. Given a claim and a corpus containing a set of documents, the problem boils down to predicting the veracity of the claim. In practice, however, searching, extracting and processing evidence is a complex underlying task (discussed in Sect. 3). In this work we describe the automation process and its nuances, we also propose a LSTM-based model (dubbed SimpleLSTM) for claim classification phase of the pipeline.

2 Related Work

Ciampaglia et al. [5] formulate the fact-checking problem as a special case of link prediction in knowledge graphs (*DBpedia*[6]). However, this approach is problematic as the Knowledge Graphs are often outdated and lack status-quo of the world. Vlachos and Riedel [34] defined the problem of fact-checking as the truthfulness of claims in the form of a binary classification problem. They provide two datasets constructed from political fact-checking websites. Their work

[1] https://www.politifact.com/.
[2] https://www.snopes.com/.
[3] https://www.factcheck.org/.
[4] http://www.fakenewschallenge.org/.
[5] http://fever.ai.
[6] https://wiki.dbpedia.org/.

tries to define the problem of fact-checking as a one-to-one automation mapping of the human fact-checking process. Thorne et al. [28] give an overview of fact-checking automation for natural language claims, bridging the gaps between fact-checking and related research areas. Starting with the fact-checking in journalism, they define basic terminology and then go on drawing a parallel between Fake news research, fact-checking, textual entailment etc. Lee et al. [16] propose a neural-ranker-based evidence extraction method, an important part of the fact-checking pipeline, extending the baseline method [31]. Taniguchi et al. [27] give a three-component pipeline consisting of document retrieval, sentence selection and recognizing textual entailment (RTE). Popat et al. [24] design an end2end model for fake-news detection. They use pre-retrieved articles related to a single fact and aggregate the veracity score from each article to make the final decision about the claim. Yang et al. [37] propose a convolution neural network-based approach for fake news detection. They combine the text in the articles with the image cues. Recently many studies have also proposed deep learning solutions to the fact-checking problem [6,17,24,38]. DeclarE [24] combines the evidence extraction and claim classification in a single end2end model. Sizhen et al. [17] select relevant Wikipedia entities using an online available tool - *S-MART* - and use their model to perform both *evidence selection* and *claim classification* in a combined fashion, reaching baseline results *evidence retrieval*. Conforti et al. [6] propose an approach for fake-news detection by focusing on the stance of the claims. They use the dataset from Fake-News-Challenge (FNC-1) to test their model. The model yields better results than the top performers of FNC-1. Yin and Roth [38], give a two-wing-optimization strategy and combine the last two steps of Fact-Checking pipeline. Their model beats the baseline [29] on evidence identification and claim verification by a good margin. Baly et al. [2] use the approach of bias detection in the news media and predict the "factuality" in news reported by the media source. They use the following variants of veracity assessment: fact-checking, fake news detection, troll detection and source reliability estimation. Popat et al. [23] add source trend and language to the credibility assessment approach, and also provide user interpretable explanations of the final decision.

3 Fact-Checking Pipeline: Automating the Task

Fact-checking is a relatively new research area in NLP, but is not a new task in journalism. The problem was first discussed in the early 1920s, evolving into standard practice at many American magazines later on [36]. However, only recently the task has spread over different communities and countries [12]. Most recent ideas in fact-checking revolve around automation of the human (or journalist) fact-checking process. Currently, this is broadly translated into a 3-step process which involves (1) collecting articles about the claim, (2) selecting prospective evidence and finally (3) performing a final judgment. Figure 1 exemplifies this in a nutshell, delineating these three components: (1) *document retrieval*, (2) *evidence selection* and (3) *claim classification*. The flow is detailed in the following.

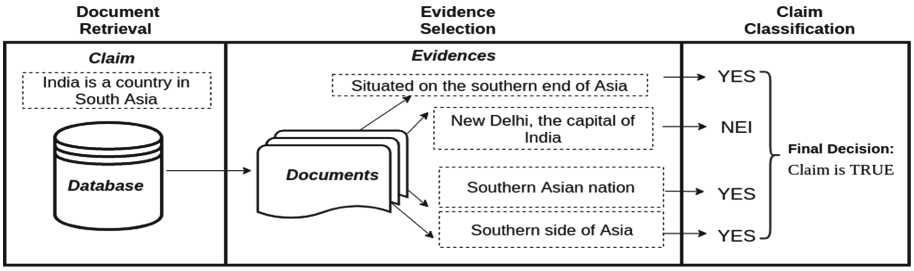

Fig. 1. The figure shows three-component fact-checking pipeline: Document Retrieval, Evidence Selection and Claim Classification.

Document Retrieval: The first step focuses on obtaining relevant documents [2, 31]. The module selects documents from a large corpus that are related to a given claim. The relatedness is determined by selecting a matching metric. Throne et al. [29] use DrQA [4] for document retrieval that selects documents from Wikipedia[7] corpus based on TF-IDF. An alternative approach could be using web search APIs [3,11,18,23] (e.g., Bing API[8]) for collecting related webpages from the internet. This approach is a better approximation of the human fact-checking process, since human fact-checkers do not restrict their research to a single corpus (like Wikipedia).

Evidence Selection: The next step of the pipeline is to select potential evidences from the documents or collection of sentences that we retrieved in the first step. This component does not differentiate between a piece of evidence that refutes the claim, from one that supports it. The main goal at this phase is to collect sentences that could potentially be used to run inference on the veracity of the claim.

Source Classification: This step, although not strictly necessary in order to perform the task, has major importance in order to weight all of the extracted claims (proofs) according to the trustworthiness of source [7,23].

Claim Classification: In the last step, as the name suggests, the model makes final decision on the claim by taking all the collected information into account [24], producing scores for each evidence, and then finally making decision on aggregation of all the scores (textual entailment on the claim by using all pieces of evidence [29]).

3.1 Claim Classification Methods

In the following section we describe common (*weak* and *strong* baselines) methods to perform *claim classification*. The methods are detailed in Sect. 4. *Feature-based*

[7] https://www.wikipedia.org/.
[8] https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/.

*(*XI-FEATURE*)*: the first (weak) baseline method relies on a standard feature engineering flow. The extracted features are mostly based on lexical and string similarity features, for instance [26]. The final classification is obtained through supervised models. We used SVM, MLP and RandomForest models. *Gradient Boosting Classifier (*XGBoost*)* Tosik et al. [32] propose a feature based model for *stance detection*. The model uses a gradient boost method for classification which combines a number of weak learners into a single learner on an iterative basis in the form of decision trees. The feature set contains twenty features based on various distance measures such as *cosine distance* and *hamming distance*, *relative entropy* between topic model probability distributions, *sentiment scores* of the claim and sentence (or evidence) and etc. We use this model as one of the baselines for the claim classification task. *Textual Entailment (TE)* is under the umbrella of Argumentation Mining in NLP and is a natural language processing task to find directional relation between texts [33]. Given a text fragment, the task is to determine if this text is a consequence of another text. The first text fragment is called a *hypothesis* and the second reference text *entailing text*, where the *entailing text* and *hypothesis* can be seen as the evidence and the claim, respectively. We use the TE model implementation by AllenNLP [9] as the second baseline for the claim classification task. The model is an implementation of the decomposable attention model given by Parikh et al. [21].

3.2 Proposed Architecture: SimpleLSTM

We employ recent deep learning techniques for claim-classification step in the automated fact-checking task. To this aim, we propose SimpleLSTM, a Long-Short-Term-Memory (LSTM) based model that extracts semantic information from claims and evidences and then combines these representations. The combined layer is then fed to a fully-connected neural network, where the final decision making (classification) is done. We use stacked-LSTM layers for both, claim and evidence. Figure 2 gives a schematic representation of SimpleLSTM.

The inputs to the model are a claim, evidence and a target label. The claim is represented as $[C_1, C_2, \cdots, C_m]$ where m ranges from 10–20 words, and the evidence is represented as $[E_1, E_2, \cdots, E_n]$ where n ranges from 100–200 words. The claim and evidence vectors are first passed through a pre-trained embedding layer to get corresponding $d \times m$ and $d \times n$ matrices for claim and evidence respectively, where d is the size of each word embedding. The embedding matrices are then fed to two parallel stacked-LSTM layers. Last LSTM outputs for the both the LSTM stacks give feature representations for claim and evidence. We call them *claimvec* and *sentvec*. The feature vectors are passed through a merge function. It should be noted that both the evidence and claim share the embedding layer which facilitates the merging of *claimvec* and *sentvec* vectors. In practice any binary function `MERGE(sentvec,claimvec)` can be used to produce a merged representation. We experimented with (1) Cosine distance, (2) Concatenation and Multiplication of *sentvec* and *claimvec*. We found element-wise multiplication to be most effective. Finally the fully connected layer makes the decision.

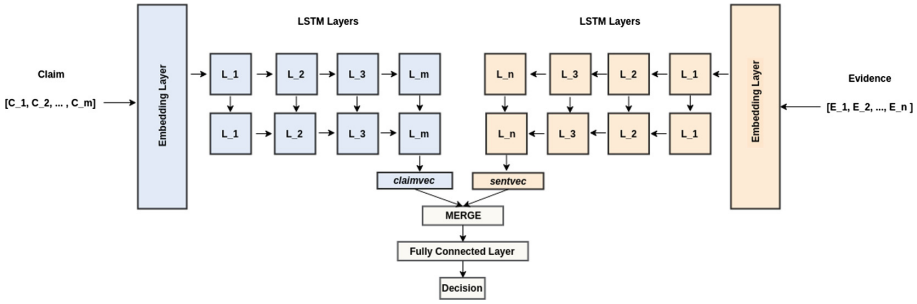

Fig. 2. SimpleLSTM model. The inputs are claim and evidence. Both, the evidence and the claim are fed to an embedding layer (common for both) that outputs embedding representation for each word. These embeddings are then passed through LSTM layers. The final output of LSTM, `sentvec` and `claimvec`, are merged and fed to the fully connected layer.

4 Experiments

For the experiments, we used the most relevant fact-checking dataset publicity available: FEVER [30]. In this section we give details of the experiments we performed on FEVER-Simple datasets for the claim classification task. In this work, we mainly focused on *simple claims*, i.e., claims which do not exceed one sentence in length. Therefore, we extracted from FEVER only those claims which are represented by a *subject-predicate-object* triple. We refer to this extracted subset as FEVER-Simple. We implemented two strong baselines to compare the performance with our models. We further divide the claim classification task into three tasks: FEVER-Support, FEVER-Reject and FEVER-3-Class. FEVER-Support uses only those claims which are *true* and have corresponding evidences that support them. Similarly, FEVER-Reject contains claims which are *false* and have corresponding evidences. The last task uses three class classification *Support*, *Reject* and *NEI* (Not enough information). Table 1 gives details about the number of instances for each dataset. For training the models, we divide the

Table 1. FEVER-Simple subsets

Dataset	Label	Count
FEVER-Support	Support	2761
	NEI	2761
FEVER-Reject	Reject	2955
	NEI	2955
FEVER 3-Class	Support	2761
	Reject	2847
	NEI	2804

datasets into the train (80%), validation (10%) and test (10%) split. The subsections below describe the setup and hyper-parameters chosen for the models and datasets.

Table 2. List of various features used in XI-FEATURE.

Feature	Definition
is sub	Checks if the document contains subject
is obj	Checks if the document contains object
is pred	Checks if the document contains predicate
dist sub obj Text follows	Distance between subject and object
pred between	Does predicate occur between subject and object
sub relax	Checks whether subject is present in partial form
obj relax	Checks whether object is present in partial form
pred relax	Checks whether predicate is present in partial form
Jaccard distance	Maximum Jaccard coefficient
Cosine similarity	Maximum cosine similarity
Semantic similarity	Similarity score of most semantically similar sentence

4.1 Baselines

XI-FEATURE. The XI-FEATURE experiments involve tuning the hyper-parameters for all the three classifiers: MLP, RF and SVM. We used cross-validation as sampling method with grid search for hyper-parameter optimization[9]. We generate a set of eleven features that incorporate the morphological, syntactic and semantic information of the claim and evidence pair. The features are generated by extracting claim specific information from the evidence. We utilize subject, predicate and object (spo) triples (pre-extracted from the claim) in our datasets. A claim like "That 70s show is a sitcom" can be broken down into a spo triple ['That 70s Show', 'is a', 'sitcom']. Given a *spo* triple for each claim, a simple method to find similarity between *claim* and *evidence* is to find whether subject, predicate, and object (or their synonyms) appear in the sentence. The first eight features utilize the triples to extract morphological information from evidence sentence. We later added semantic similarity, cosine similarity and Jaccard similarity between the claim and evidence sentence as three features. These features represent the similarity measure between the *claim* and the most similar sentence in the evidence. Table 2 lists all the eleven features.

Pre-trained word embeddings were used to compute the similarity based metrics using spaCy[10] python library. The extracted features are trained on three classifier models: Support Vector Machine (SVM), Random Forest (RF) and

[9] GridSearchCV from scikit-learn to obtain the best hyper-parameters.
[10] github.com/explosion/spaCy.

Multi-Layer Perceptron (MLP). According to our experiments, Random Forest yields the best results with a maximum depth of 10 and *entropy* as splitting criterion. For SVM penalty of 100, *gamma* 0.001 and *rbf* kernel gave the best performance. The neural network has two hidden layers with 44 perceptrons in each. The model provides best results with *Adam* optimizer and *ReLU* activation function.

XGBoost. *XGBoost* model is trained on FeverSimple datasets and the best hyperparameter settings for each model are selected using a grid search with cross-validation on the training set. A max_depth of 8 and 1000 estimators provide the best performance overall on the three datasets.

TE. *TE* represents the pre-trained textual entailment model with decomposable attention [21] by AllenNLP [9]. We ran this on the testing data by using claim as the hypothesis and evidence as the entailing text.

4.2 Results

Table 3 depicts the accuracy, precision, recall and F1 scores for all the models. It can be observed that the our feature models outperform the gradient-boost (XGBoost) [32] and TE [9]. The MLP models gives better performance than RF

Table 3. Performance comparison of different models on FEVER support, FEVER Reject and FEVER 3-class

Dataset	Classifier	Accuracy	Precision	Recall	f1 Score
FEVER Support	XGBoost [32]	0.766	0.766	0.766	0.762
	TE [9]	0.691	0.835	0.655	0.734
	XI-FEATURE RF	0.79	0.76	0.83	0.79
	XI-FEATURE SVM	0.79	0.71	0.85	0.77
	XI-FEATURE MLP	0.79	0.76	0.81	0.78
	SimpleLSTM	0.850	0.834	0.856	0.845
FEVER Reject	XGBoost [32]	0.74	0.738	0.736	0.73
	TE [9]	0.548	0.759	0.533	0.626
	XI-FEATURE RF	0.73	0.73	0.81	0.76
	XI-FEATURE SVM	0.642	0.73	0.78	0.75
	XI-FEATURE MLP	0.74	0.69	0.78	0.73
	SimpleLSTM	0.816	0.836	0.811	0.824
FEVER 3-class	XGBoost [32]	0.535	0.54	0.534	0.539
	TE [9]	0.418	0.372	0.622	0.465
	XI-FEATURE RF	0.55	0.60	0.61	0.60
	XI-FEATURE SVM	0.55	0.54	0.56	0.53
	XI-FEATURE MLP	0.59	0.61	0.62	0.61
	SimpleLSTM	0.635	0.643	0.620	0.642

and SVM on the FEVER-Simple datasets. The SimpleLSTM model beats all the baselines with significant margin.

We found that out of cosine distance, element-wise multiplication, and plain concatenation, element-wise multiplication works the best for SimpleLSTM, so we decided to go with it. We chose GoogleNews vectors*GoogleNews vectors*[11] [19] for pre-trained word embeddings. This is a word2vec [19] model has been trained on Google News dataset that has about 100 billion words. It contains word embeddings of dimension 300 for 3 million words and phrases. We fixed batch size of 64 and Adam Optimizer, with learning rate of 0.001, for all the datasets. The loss is *binary cross-entropy* loss for binary classification tasks, and *categorical cross-entropy* loss for the 3-class problem. The input size for LSTM stack is 300, and the output size is 150 for FEVER-Support and FEVER-Reject, and 100 for 3-Class.

4.3 Challenges

On the computational side, there are different fundamental challenges w.r.t. the execution of the underlying tasks in this pipeline. We extend the definition of [14] (items 1 and 4) by highlighting two more challenges (items 2 and 3) we argue are crucial to bridge the gap between *automated fact-checking* approaches and human fact-checkers, as follows. (1) to understand what one says [14] (NLU) (2) to have the ability to generate equivalent arguments and counter-arguments (NLG) (3) to have the ability to distinguish credible and non-credible information sources (Credibility) (4) to have the ability to obtain plausible evidence [14] (Argumentation Mining)

Firstly, algorithms need to have the ability to understand what is being said. This refers to a specific research area called Natural Language Understanding (NLU), which encompasses several NLP sub-tasks, such as Named Entity Recognition (NER), and Part-of-speech (POS). Although significant leaps have been made in this area, these technologies are far from human performance, especially in more noisy contexts such as microblogs [1].

Secondly, algorithms need to process similar content accordingly, i.e., to collect equivalent and related content which are potentially useful in the fact-checking process. This is part of a branch in NLP called Natural Language Generation (NLG) [10]. This is of utmost importance in order to have a broader coverage when checking claims. In structured fact-checking [11], this is a crucial step towards interpreting and transforming the input *claim* into natural language. Moreover, in free text, for instance, the sentences "*he was born in USA.*" and "*he is a Yankee.*" share the same meaning. Given a claim "*he is American.*", algorithms should be capable to perceive that, in this context, "USA" and "Yankee" are synonyms for "American", thus the information extraction phase should generate similar content automatically in order to increase *recall*.

Thirdly, the level of trustworthiness of authorities and sources must be checked and taken into account. This has been studied in a topic known as

[11] https://code.google.com/archive/p/word2vec/.

(Web) Credibility [7]. This step is important both in assessing the credibility of sources isolated, as well as when confronting opposite claims made by two or more authorities [33]. For instance, consider a scenario of researching information about a dietary supplement that potentially helps in certain disease treatment. One may find websites from reputable agencies (e.g., NCI USA) alongside sites from private organizations which sell dietary supplements (which may serve as advice hub whilst pointing to their own products). Discerning which sources are trustworthy and which are not is a crucial step forward automated fact-checking systems [7]. Finally, besides collecting sufficient evidence for asserting a given claim, explicit and implicit relations among extracted arguments (as well as possible counter-arguments) should (ideally) be labeled and linked. This is studied in another branch of NLP called Argumentation Mining [20]. The generated graph allows achieving a richer level of metadata in order to better perform the final fact-checking task [22].

5 Conclusion

In this work, we give an overview of the fact-checking problem and its automation in the context of natural language processing. We discuss the fact-checking pipeline that consists of document retrieval, evidence selection, source classification, and claim classification, shedding light on existing challenges. Most notably the claim classification phase, which consists of classification of claims as support, reject or not related. In order to solve this task, we propose two new models: SimpleLSTM and XI-FEATURE, comparing the results with two strong baselines. Our experiments show that SimpleLSTM outperforms all the baselines. Compared to the best baseline (XGBoost [32]), it outperforms it by 11%, 10.2% and 18.7% on the FEVER-Support, FEVER-Reject and 3-Class tasks respectively. However, we show that the task is far from being solved and performance is only reasonable even in more simple scenarios (simple claims). As future work, we will study the impact of such architectures in complex claims, explore other languages and investigate further architectures to minimize the gaps in the fact-checking automation task.

Acknowledgement. This work was partially funded by the European Union Marie Curie ITN Cleopatra project (GA no. 812997).

References

1. Akbik, A., Blythe, D., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 1638–1649 (2018)
2. Baly, R., Karadzhov, G., Alexandrov, D., Glass, J.R., Nakov, P.: Predicting factuality of reporting and bias of news media sources. CoRR abs/1810.01765 (2018). http://arxiv.org/abs/1810.01765

3. Baly, R., Mohtarami, M., Glass, J.R., Màrquez, L., Moschitti, A., Nakov, P.: Integrating stance detection and fact checking in a unified corpus. CoRR abs/1804.08012 (2018). http://arxiv.org/abs/1804.08012
4. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051 (2017)
5. Ciampaglia, G.L., Shiralkar, P., Rocha, L.M., Bollen, J., Menczer, F., Flammini, A.: Computational fact checking from knowledge networks. PLoS ONE **10**(6), e0128193 (2015)
6. Conforti, C., Pilehvar, M.T., Collier, N.: Towards automatic fake news detection: cross-level stance detection in news articles. In: Proceedings of the First Workshop on Fact Extraction and VERification (FEVER), pp. 40–49 (2018)
7. Esteves, D., Reddy, A.J., Chawla, P., Lehmann, J.: Belittling the source: trustworthiness indicators to obfuscate fake news on the web. In: Proceedings of the First Workshop on Fact Extraction and Verification (FEVER), pp. 50–59. Association for Computational Linguistics (2018). http://aclweb.org/anthology/W18-5508
8. Fridkin, K., Kenney, P.J., Wintersieck, A.: Liar, liar, pants on fire: How fact-checking influences citizens' reactions to negative advertising. Polit. Commun. **32**(1), 127–151 (2015). https://doi.org/10.1080/10584609.2014.914613
9. Gardner, M., et al.: AllenNLP: a deep semantic natural language processing platform (2017)
10. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: core tasks, applications and evaluation. J. Artif. Intell. Res. **61**, 65–170 (2018)
11. Gerber, D., et al.: Defacto–temporal and multilingual deep fact validation. Web Semant.: Sci. Serv. Agents World Wide Web **35**, 85–101 (2015)
12. Graves, L., Cherubini, F.: The rise of fact-checking sites in Europe (2016)
13. Hanselowski, A., et al.: A retrospective analysis of the fake news challenge stance detection task. arXiv preprint arXiv:1806.05180 (2018)
14. Hassan, N., et al.: The quest to automate fact-checking. World (2015)
15. Himma-Kadakas, M., et al.: Alternative facts and fake news entering journalistic content production cycle. Cosmopolitan Civil Soc.: Interdisc. J. **9**(2), 25 (2017)
16. Lee, N., Wu, C.S., Fung, P.: Improving large-scale fact-checking using decomposable attention models and lexical tagging. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 1133–1138 (2018)
17. Li, S., Zhao, S., Cheng, B., Yang, H.: An end-to-end multi-task learning model for fact checking. In: Proceedings of the First Workshop on Fact Extraction and Verification (FEVER), pp. 138–144. Association for Computational Linguistics (2018). http://aclweb.org/anthology/W18-5523
18. Mihaylova, T., et al.: Fact checking in community forums. CoRR abs/1803.03178 (2018). http://arxiv.org/abs/1803.03178
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR abs/1310.4546 (2013). http://arxiv.org/abs/1310.4546
20. Palau, R.M., Moens, M.F.: Argumentation mining: the detection, classification and structure of arguments in text. In: Proceedings of the 12th International Conference on Artificial Intelligence and Law, pp. 98–107. ACM (2009)
21. Parikh, A.P., Täckström, O., Das, D., Uszkoreit, J.: A decomposable attention model for natural language inference. In: EMNLP (2016)
22. Peldszus, A., Stede, M.: From argument diagrams to argumentation mining in texts: a survey. Int. J. Cogn. Inform. Nat. Intell. (IJCINI) **7**(1), 1–31 (2013)

23. Popat, K., Mukherjee, S., Strötgen, J., Weikum, G.: Where the truth lies: explaining the credibility of emerging claims on the web and social media. In: Proceedings of the 26th International Conference on World Wide Web Companion, pp. 1003–1012. International World Wide Web Conferences Steering Committee (2017)
24. Popat, K., Mukherjee, S., Yates, A., Weikum, G.: Declare: debunking fake news and false claims using evidence-aware deep learning. CoRR abs/1809.06416 (2018). http://arxiv.org/abs/1809.06416
25. Rashkin, H., Choi, E., Jang, J.Y., Volkova, S., Choi, Y.: Truth of varying shades: analyzing language in fake news and political fact-checking. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2931–2937. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/D17-1317. http://aclweb.org/anthology/D17-1317
26. Reddy, A.J., Rocha, G., Esteves, D.: DeFactoNLP: fact verification using entity recognition, TFIDF vector comparison and decomposable attention. CoRR abs/1809.00509 (2018)
27. Taniguchi, M., Taniguchi, T., Takahashi, T., Miura, Y., Ohkuma, T.: Integrating entity linking and evidence ranking for fact extraction and verification. In: Proceedings of the First Workshop on Fact Extraction and Verification (FEVER), pp. 124–126 (2018)
28. Thorne, J., Vlachos, A.: Automated fact checking: task formulations, methods and future directions. arXiv preprint arXiv:1806.07687 (2018)
29. Thorne, J., Vlachos, A., Christodoulopoulos, C., Mittal, A.: FEVER: a large-scale dataset for fact extraction and verification. CoRR abs/1803.05355 (2018). http://arxiv.org/abs/1803.05355
30. Thorne, J., Vlachos, A., Christodoulopoulos, C., Mittal, A.: Fever: a large-scale dataset for fact extraction and verification. arXiv preprint arXiv:1803.05355 (2018)
31. Thorne, J., Vlachos, A., Cocarascu, O., Christodoulopoulos, C., Mittal, A.: The fact extraction and verification (fever) shared task. arXiv preprint arXiv:1811.10971 (2018)
32. Tosik, M., Mallia, A., Gangopadhyay, K.: Debunking fake news one feature at a time. arXiv preprint arXiv:1808.02831 (2018)
33. Villata, S., Boella, G., Gabbay, D.M., van der Torre, L.: Arguing about the trustworthiness of the information sources. In: Liu, W. (ed.) ECSQARU 2011. LNCS (LNAI), vol. 6717, pp. 74–85. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22152-1_7
34. Vlachos, A., Riedel, S.: Fact checking: task definition and dataset construction. In: Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science, pp. 18–22 (2014)
35. Vosoughi, S., Roy, D., Aral, S.: The spread of true and false news online. Science **359**(6380), 1146–1151 (2018)
36. Witschge, T., Nygren, G.: Journalistic work: a profession under pressure? J. Media Bus. Stud. **6**(1), 37–59 (2009)
37. Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z., Yu, P.S.: TI-CNN: convolutional neural networks for fake news detection. CoRR abs/1806.00749 (2018). http://arxiv.org/abs/1806.00749
38. Yin, W., Roth, D.: TwoWingOS: a two-wing optimization strategy for evidential claim verification. CoRR abs/1808.03465 (2018). http://arxiv.org/abs/1808.03465