An Extensible Framework for Verification of Numerical Claims

James Thorne
Department of Computer Science
University of Sheffield, UK
j.thorne@sheffield.ac.uk

Andreas Vlachos
Department of Computer Science
University of Sheffield, UK
a.vlachos@sheffield.ac.uk

Abstract

In this paper we present our automated fact checking system demonstration which we developed in order to participate in the Fast and Furious Fact Check challenge. We focused on simple numerical claims such as "population of Germany in 2015 was 80 million" which comprised a quarter of the test instances in the challenge, achieving 68% accuracy. Our system extends previous work on semantic parsing and claim identification to handle temporal expressions and knowledge bases consisting of multiple tables, while relying solely on automatically generated training data. We demonstrate the extensible nature of our system by evaluating it on relations used in previous work. We make our system publicly available so that it can be used and extended by the community.[1]

1 Introduction

Fact checking is the task of assessing the truthfulness in spoken or written language. We are motivated by calls to provide tools to support journalists with resources to verify content at source (Cohen et al., 2011) or upon distribution. Manual verification can be too slow to verify information given the speed at which claims travel on social networks (Hassan et al., 2015a).

In the context of natural language processing research, the task of automated fact checking was discussed by Vlachos and Riedel (2014). Given a claim, a system for this task must determine what information is needed to support or refute the claim, retrieve the information from a knowledge base (KB) and then compute a deduction to assign a verdict. For example, in the claim of Figure 1 a system needs to recognize the named entity (Germany), the statistical property (population) and the year, link them to appropriate elements in a KB, and deduce the truthfulness of the claim using the absolute percentage error.

Input: Around 80 million people were inhabitants of Germany in 2015.
Data Source: `data/worldbank_wdi.csv`
Property: `population(Germany, 2015)`
Value: `81413145`
Absolute Percentage Error: 1.7%
Verdict: `TRUE`

Figure 1: Fact checking a claim by matching it to an entry in the knowldge base.

We contrast this task against rumour detection (Qazvinian et al., 2011) – a similar prediction task based on language subjectivity and growth of readership through a social network. While these are important factors to consider, a sentence can be true or false regardless of whether it is a rumour (Lukasik et al., 2016).

Existing fact checking systems are capable of detecting fact-check-worthy claims in text (Hassan et al., 2015b), returning semantically similar textual claims (Walenz et al., 2014); and scoring the truth of triples on a knowledge graph through semantic distance (Ciampaglia et al., 2015). However, neither of these are suitable for fact checking a claim made in natural language against a database. Previous works appropriate for this task operate on a limited domain and are not able to incorporate temporal information when checking time-dependent claims (Vlachos and Riedel, 2015).

In this paper we introduce our fact checking tool, describe its architecture and design decisions, evaluate its accuracy and discuss future work. We

[1] https://github.com/sheffieldnlp/numerical-fact-checking-eacl2017

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 37–40

highlight the ease of incorporating new information sources to fact check, which may be unavailable during training. To validate the extensibility of the system, we complete an additional evaluation of the system using claims taken from Vlachos and Riedel (2015). We make the source code publicly available to the community.

2 Design Considerations

We developed our fact-checking approach in the context of the HeroX challenge[2] – a competition organised by the fact checking organization FullFact[3]. The types of claims the system presented can fact check was restricted to those which require looking up a value in a KB, similar to the one in Figure 1. To learn a model to perform the KB look up (essentially a semantic parsing task), we extend the work of Vlachos and Riedel (2015) who used distant supervision (Mintz et al., 2009) to generate training data, obviating the need for manual labeling. In particular, we extend it to handle simple temporal expressions in order to fact check time-dependent claims appropriately, i. e. population in 2015. While the recently proposed semantic parser of Pasupat and Liang (2015) is also able to handle temporal expressions, it makes the assumption that the table against which the claim needs to be interpreted is known, which is unrealistic in the context of fact checking.

Furthermore, the system we propose can predict relations from the KB on which the semantic parser has not been trained, a paradigm referred to as zero-shot learning (Larochelle et al., 2008). We achieve this by learning a binary classifier that assesses how well the claim "matches" each relation in the KB. Finally, another consideration in our design is algorithmic accountability (Diakopoulos, 2016) so that the predictions and the decision process used by the system are interpretable by a human.

3 System Overview

Given an unverified statement, the objective of this system is to identify a KB entry to support or refute the claim. Our KB consists of a set of un-normalised tables that have been translated into simple Entity-Predicate-Value triples through a simple set of rules. In what follows we first describe the fact checking process used during testing (Section 3.1) and then how the relation matching module is trained and the features used for this purpose (Section 3.2).

[2] http://herox.com/factcheck

[3] https://fullfact.org

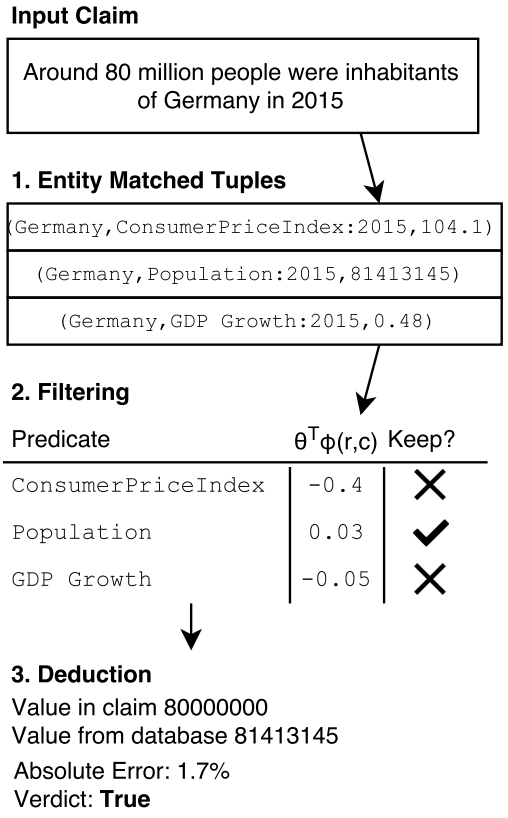

Figure 2: Relation matching step and filtering

3.1 Fact Checking

In our implementation, fact checking is a three step process, illustrated in Figure 2. Firstly, we link named entities in the claim to entities in our KB and retrieve a set of tuples involving the entities found. Secondly, these entries are filtered in a relation matching step. Using the text in the claim and the predicate as features, we classify whether this tuple is relevant. And finally, the values in the matched triples from the KB are compared to the value in the statement to deduce the verdict as follows: if there is at least one value with absolute percentage error lower than a threshold defined by the user, the claim is labeled true, otherwise false.

We model the relation matching as a binary classification task using logistic regression implemented in `scikit-learn`, predicting whether a predicate is a match to the given input claim. The aim of this step is to retain tuples for which the predicate in the Entity-Predicate-Value tuple can be described by the surface forms present in the

claim (positive-class) and discard the remainder (negative-class). We chose not to model this as a multi-class classification task (one class per predicate) to improve the extensibility of the system. A multi-class classifier requires training instances for every class, thus would not be applicable to predicates not seen during training. Instead, our aim is to predict the compatibility of a predicate w. r. t. the input claim.

For each of the candidate tuples $r_i \in R$, a feature vector is generated using lexical and syntactic features present in the claim and relation: $\phi(r_i, c)$. This feature vector is inputted to a logistic regression binary classifier and we retain all tuples where $\theta^T \phi(r_i, c) \geq 0$.

3.2 Training Data Generation

The training data for relation matching is generated using distant supervision and the Bing search engine. We first read the table and apply a set of simple rules to extract subject, predicate, object tuples, and for each named entity and numeric value we generate a query containing the entity name and the predicate. For example, the entry `(Germany,Population:2015,81413145)` is converted to the query *"Germany" Population 2015*. The queries are then executed on the Bing search engine and the top 50 web-page results are retained. We extract the text from the webpages using a script built around the BeautifulSoup package.[4] This text is parsed and annotated with co-reference chains using the Stanford CoreNLP pipeline (Manning et al., 2014).

Each sentence containing a mention of an entity and a number is used to generate a training example. The examples are labeled as follows. If the absolute percentage error between the value in the KB and the number extracted from text is below a threshold (an adjustable hyperparameter), the training instance is marked as a positive instance. Sentences which contain a number outside of this threshold are marked as negative instances. We make an exception for numbers tagged as dates where an exact match is required.

For each claim, the feature generation function, $\phi(r, c)$, outputs lexical and syntactic features from the claim and a set of custom indicator variables. Additionally, we include a bias feature for every unique predicate in the KB. For our lexical and syntactic features, we consider the words in the span between the entity and number as well as the dependency path between the entity and the number. To generalise to unseen relations, we include the ability to add custom indicator functions; for our demonstration we include a simple boolean feature indicating whether the intersection of the words in the predicate name and the sentence is not empty.

4 Evaluation

The system was field tested in the HeroX fact checking challenge - 40 general-domain claims chosen by journalists. Given the design of our system, we were restricted to claims that can only be answered by a KB look up (11 claims in total), returning the correct truth assessment for 7.5 of them according to the human judges. The half-correct one was due to not providing a fully correct explanation. To fact check a statement, the user only needs to enter the claim into a text box. The only requirement is to provide the system with an appropriate KB. Thus we ensured that the system can readily incorporate new tables taken from encyclopedic sources such as Wikipedia and the World Bank. In our system, this step is achieved by simply importing a CSV file and running a script to generate the new instances to train the relation matching classifier.

Analysis of our entry to this competition showed that two errors were caused by incorrect initial source data and one partial error caused by recalling a correct property but making an incorrect deduction. Of numerical claims that we did not attempt, we observed that many required looking up multiple entries and performing a more complex deduction step which was beyond the scope of this project.

We further validate the system by evaluating the ability of this fact checking system to make veracity assessments on simple numerical claims from the data set collected by (Vlachos and Riedel, 2015). Of the 4,255 claims about numerical properties about countries and geographical areas in this data set, our KB contained information to fact check 3,418. The system presented recalled KB entries for 3,045 claims (89.1%). We observed that the system was consistently unable to fact check two properties (undernourishment and renewable freshwater per capita). Analysis of these failure cases revealed too great a lexical difference

[4] `https://www.crummy.com/software/BeautifulSoup/`

between the test claims and the training data our system generated; the claims in the test cases were comparative in nature (e. g. country X has higher rate of undernourishment than country Y) whereas the training data generated using the method described in Section 3.2 are absolute claims.

A high number of false positive matches were generated, e. g. for a claim about population, other irrelevant properties were also recalled in addition. For the 3,045 matched claims, 17,770 properties were matched from the KB that had a score greater than or equal to the logistic score of the correct property. This means that for every claim, there were, on average, 5.85 incorrect properties also extracted from the KB. In our case, this did not yield false positive assignment of truth labels to the claims. This was because the absolute percentage error between the incorrectly retrieved properties and the claimed value was outside of the threshold we defined; thus these incorrectly retrieved properties never resulted in a true verdict, allowing the correct one to determine the verdict.

5 Conclusions and Future Work

The core capability of the system demonstration we presented is to fact check natural language claims against relations stored in a KB. Although the range of claims is limited, the system is a field-tested prototype and has been evaluated on a published data set (Vlachos and Riedel, 2015) and on real-world claims presented as part of the HeroX fact checking challenge. In future work, we will extend the semantic parsing technique used and apply our system to more complex claim types. Additionally, further work is required to reduce the number of candidate relations recalled from the KB. While this was not an issue in our case, we believe that ameliorating this issue will enhance the ability of the system to assign a correct truth label where there exist properties with similar numerical values.

Acknowledgements

This research is supported by a Microsoft Azure for Research grant. Andreas Vlachos is supported by the EU H2020 SUMMA project (grant agreement number 688139).

References

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193.

Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. 2011. Computational journalism: A call to arms to database researchers. In *CIDR*, volume 2011, pages 148–151.

Nicholas Diakopoulos. 2016. Accountability in algorithmic decision making. *Communications of the ACM*, 59(2):56–62.

Naeemul Hassan, Bill Adair, James T Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015a. The quest to automate fact-checking. *world*.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015b. Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1835–1838. ACM.

Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. 2008. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3.

Michal Lukasik, Kalina Bontcheva, Trevor Cohn, Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2016. Using gaussian processes for rumour stance classification in social media. *arXiv preprint arXiv:1609.01962*.

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 ACL System Demonstrations*, pages 55–60.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*, pages 1003–1011.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of EMNLP*, pages 1589–1599.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. *ACL 2014*, page 18.

Andreas Vlachos and Sebastian Riedel. 2015. Identification and verification of simple claims about statistical properties. In *Proceedings of EMNLP*.

Brett Walenz, You Will Wu, Seokhyun Alex Song, Emre Sonmez, Eric Wu, Kevin Wu, Pankaj K Agarwal, Jun Yang, Naeemul Hassan, Afroza Sultana, et al. 2014. Finding, monitoring, and checking claims computationally based on structured data. In *Computation+ Journalism Symposium*.