ClaimLinker: Linking Text to a Knowledge Graph of Fact-checked Claims

Evangelos Maliaroudakis
evamaliar@gmail.com
ICS-FORTH & University of Crete
Heraklion, Greece

Katarina Boland
katarina.boland@gesis.org
GESIS, Cologne & Heinrich-Heine-University Düsseldorf
Cologne, Germany

Stefan Dietze
stefan.dietze@gesis.org
GESIS, Cologne & Heinrich-Heine-University Düsseldorf
Cologne, Germany

Konstantin Todorov
todorov@lirmm.fr
LIRMM, University of Montpellier, CNRS
Montpellier, France

Yannis Tzitzikas
tzitzik@ics.forth.gr
Information Systems Laboratory, ICS-FORTH & University of Crete Heraklion, Greece

Pavlos Fafalios
fafalios@ics.forth.gr
Information Systems Laboratory, ICS-FORTH Heraklion, Greece

ABSTRACT

We present ClaimLinker, a Web service and API that links arbitrary text to a knowledge graph of fact-checked claims, offering a novel kind of semantic annotation of unstructured content. Given a text, ClaimLinker matches parts of it to fact-checked claims mined from popular fact-checking sites and integrated into a rich knowledge graph, thus allowing the further exploration of the linked claims and their associations. The application is based on a scalable, fully unsupervised and modular approach that does not require training or tuning and which can serve high quality results at real time (outperforming existing unsupervised methods). This allows its easy deployment for different contexts and application scenarios.

CCS CONCEPTS

• Information systems → Content ranking; Web services; Web applications; Information systems applications.

KEYWORDS

Claim Linking, Claim Retrieval, Knowledge Graphs, Fact-checking, Fake News

ACM Reference Format:

1 INTRODUCTION

Even though understanding the veracity of statements is important, in particular, when consuming information on the Web, fact-checking of arbitrary statements from Web documents on-the-fly is costly and impractical. On the other hand, the plethora of manually curated fact-checking portals such as Politifact, Snopes or FullFact have led to an abundance of information about claims and their corresponding truth ratings. However, such information is spread across a wide variety of portals. In addition, given that a particular claim proposition may occur in the form of diverse utterances, matching a given statement or utterance to fact-checked claims available from fact-checking portals remains a challenging problem.

This system demonstration presents ClaimLinker, a web application that builds on a knowledge graph of Web-mined fact-checked claims [12]. The system is composed of a set of modular components able to match arbitrary text to more than 30K claims and related data (such as truth ratings, authors, sources, entities and topics) mined from eight popular fact-checking sites and integrated into a knowledge graph. The application thus combines a semantically annotated knowledge base of fact-checked claims (exposed through established schemas such as schema.org), together with an API able to rank matching claim candidates for any given utterance, thereby offering a kind of semantic annotation/tagging of unstructured content that is novel and goes beyond the current entity and relation extraction.

Combining basic principles of the WWW, open standards and interfaces, we provide access to the underlying data and the claim linking API to be integrated into third-party applications. For demonstration purposes, we also provide a Web application as a basic showcase.

2 SYSTEM DESCRIPTION

2.1 Background

ClaimLinker relies on the ClaimsKG knowledge base, an RDF knowledge graph of fact-checked claims that enables structured queries about their truth values, authors, dates, related entities and other contextual metadata [12]. ClaimsKG is generated through a pipeline which periodically harvests data from highly reputable fact-checking portals, including Politifact.com and Snopes.org. It currently contains 34,201 fact-checked claims published since 1996. The claims and their review articles are annotated with entities

1https://data.gesis.org/claimskg/
We consider a use case scenario where real-time claim linking is required. For example, a user reads an online news article and wants to find out if a statement that is mentioned in the article is related to one or more fact-checked claims. To this end, we consider a claim linking approach that comprises one (offline) pre-processing step (for indexing ClaimsKG) and four real-time processing steps.

### 2.2 Problem Definition and Aim

Relating a piece of text to fact-checked claims, as those described in the ClaimsKG knowledge base, essentially comes down to the task of identifying links between utterances. In the context of claims, we isolate three main types of relations:

1. **Identity** (have the exact same meaning) and hence can be linked via an equivalence relation such as \( \text{owl:sameAs} \).
2. **Similarity** in the semantic space, sharing a certain semantic similarity on a scale between identical and dissimilar.
3. **Relatedness** as opposed to similarity, covers “any kind of lexical or functional association”

The aim of the ClaimLinker application is to annotate a given text with fact-checked claims of ClaimsKG, with a focus on the type (b) relation described above (semantic similarity). More formally: given a text or document \( d \) and a knowledge base of fact-checked claims \( K \), ClaimLinker aims at providing a set of claim annotations \( A \) on the form \( (t, p, C) \), where \( t \) is a piece of text in \( d \) (e.g., a sentence), \( p \) is the position of \( t \) in \( d \), and \( C \) is a ranked list of semantically similar fact-checked claims of \( K \). In particular, a claim \( c \in C \) is of the form \( (u, \tau) \), where \( u \) is a claim URI from \( K \) and \( \tau \) is the similarity score representing the strength of the connection between \( c \) and \( t \).

For example, consider the below text:

> “Obama can complain about Republicans in the House as much as he wants. But in the first two years, he had, you know, huge maj wastes in the House and Senate, and did nothing with them to create jobs.”

ClaimLinker links the second sentence (“But in the first two years, he had, you know, ...”) to the ClaimsKG URI (https://data.gesis.org/claims/kg/creative_work/34465e1b-7108-53f7-8960-4595b8dd09b2) to the ClaimsKG URI (Fig. 1). Using this URI, the requestor (e.g., a user or a third-party application) can retrieve additional data from ClaimsKG, like related entities or other claims uttered by the same person.

### 2.3 Implementation Details

We consider a use case scenario where real-time claim linking is required. For example, a user reads an online news article and wants to find out if a statement that is mentioned in the article is related to one or more fact-checked claims. To this end, we consider a claim linking approach that comprises one (offline) pre-processing step (for indexing ClaimsKG) and four real-time processing steps.

### 2.4 Evaluation results

Experiments using the ground truth dataset of CLEF2020 Check-That! 2020 Task 2 (verified claim retrieval) [2] demonstrate the effectiveness ClaimLinker in finding the correct fact-checked claim for a given text. In particular, the correct claim is returned in the first position (Precision@1) in 76% of the test cases, outperforming all unsupervised methods that participated in the challenge [2]. Also, the correct claim is returned in one of the top-3 positions in 80% of the test cases, and in one of the top-5 positions in 83.5% of the test cases. These are promising results given that we consider a fully unsupervised approach which does not require training or tuning and is very efficient. If we consider only the score returned

---

3https://data.gesis.org/claims/kg/explorer
4https://www.elastic.co/
by Elasticsearch, then performance (Precision@1) drops to 71.5%, while if we consider only the eight similarity measures performance drops to 62%. This showcases the benefit of combining the different scores. Finally, we can achieve higher precision by considering a higher Elasticsearch relevance score when filtering out irrelevant claims in the candidate generation step. For instance, by using a threshold score = 20 (instead of 5), Precision@1 is increased to 89.5%, however then recall decreases (the system does not provide any annotations for 28.5% of the test cases).

3 SYSTEM COMPONENTS AND USE CASES

We consider the following use cases (exploitation scenarios), supported by dedicated ClaimLinker services:

**Web application** by Elasticsearch, then performance (Precision@1) drops to 71.5%, while if we consider only the eight similarity measures performance drops to 62%. This showcases the benefit of combining the different scores. Finally, we can achieve higher precision by considering a higher Elasticsearch relevance score when filtering out irrelevant claims in the candidate generation step. For instance, by using a threshold score = 20 (instead of 5), Precision@1 is increased to 89.5%, however then recall decreases (the system does not provide any annotations for 28.5% of the test cases).

**Web application** offering a form where the user can give some text and check if there are fact-checked claims linked to that text.

1. But in the first two years, he had, you know, huge majorities in the House and Senate, and did nothing with them to create jobs.

   Says Barack Obama had ‘huge majorities’ in Congress during his first two years in office and ‘did nothing with them to create jobs in America.’

   The U.S. Constitution states that if a president is impeached by the House but not convicted by the Senate, that person’s first term is nullified and they are ineligible to run for office in any more.

**Bookmarklet** which allows a user to select a piece of text in any web page and check if there are fact-checked claims linked to the selected text.

**Java library** communicating with an Elasticsearch service and offers the claim linking functionality.

**Web service** which accepts HTTP requests and returns the results in JSON.
and the SPARQL endpoint of ClaimsKG, one can retrieve additional information about the linked claims.

The source code of all services is available on GitHub.9

4 CONCLUSION AND FUTURE WORK

We presented ClaimLinker, a Web service and API that links arbitrary text to a knowledge graph of fact-checked claims. Given a text snippet, for instance, a statement in a news item, ClaimLinker first obtains candidate claims from the knowledge graph through a preliminary blocking step and then ranks them using a range of similarity metrics, outnumbering a number of unsupervised baselines on the claim retrieval task.

As applications of our work, we consider any use case where on-the-fly fact-checking is of importance and has to be streamlined by exploiting already fact-checked claims, for instance, as part of online journalism, social media or news platforms. Exploiting the semantic annotations of the underlying knowledge graph [12] facilitates further exploration, for instance, to identify claims related to the same or similar entities, or claims originating from the same sources.

Considering that our current approach utilises basic text similarity metrics for re-ranking candidate futures, future work will be concerned with more elaborate approaches, for instance, using self-supervised, transformer-based language models [6] to obtain contextual vector representations of both claim and text candidates in order to better cater for vocabulary mismatch problems. In addition, while we opted for an efficient unsupervised approach which determines weights of distinct metrics based on heuristics, a natural step is to attempt learning optimal weights, leading into supervised models for claim linking. Given that both approaches have advantages and disadvantages, thorough evaluation has to provide experimental insights into over-/underfitting problems and potential to generalise well across claims and types of corpora.

REFERENCES


9https://github.com/malvag/ClaimLinker