ProVe: A Pipeline for Automated Provenance Verification of Knowledge Graphs against Textual Sources

A Preprint

Gabriel Amaral$^{1\,[0000-0002-4482-5376]}$ Odinaldo Rodrigues$^{1\,[0000-0001-7823-1034]}$
Elena Simperl$^{1\,[0000-0003-1722-947X]}$

October 27, 2022

Abstract

Knowledge Graphs are repositories of information that gather data from a multitude of domains and sources in the form of semantic triples, serving as a source of structured data for various crucial applications in the modern web landscape, from Wikipedia infoboxes to search engines. Such graphs mainly serve as secondary sources of information and depend on well-documented and verifiable provenance to ensure their trustworthiness and usability. However, their ability to systematically assess and assure the quality of this provenance, most crucially whether it properly supports the graph's information, relies mainly on manual processes that do not scale with size. ProVe aims at remedying this, consisting of a pipelined approach that automatically verifies whether a Knowledge Graph triple is supported by text extracted from its documented provenance. ProVe is intended to assist information curators and consists of four main steps involving rule-based methods and machine learning models: text extraction, triple verbalisation, sentence selection, and claim verification. ProVe is evaluated on a Wikidata dataset, achieving promising results overall and excellent performance on the binary classification task of detecting support from provenance, with 87.5% accuracy and 82.9% F1-macro on text-rich sources. The evaluation data and scripts used in this paper are available in GitHub and Figshare.

Keywords Fact Verification · Data Verbalisation · Knowledge Graphs

1 Introduction

A Knowledge Graph (KG) is a type of knowledge base that stores information in the form of semantic triples formed by a subject, a predicate, and an object. KGs represent both real and abstract entities internally as labelled and uniquely identifiable entities, such as *The Moon* or *Happiness*, and can amass information from a multitude of domains and sources by connecting such entities amongst themselves or to literals through relationships, coded via uniquely identified predicates. KGs serve as sources of both human and machine-readable semantically structured data for various crucial applications in the modern web landscape, such as Wikipedia infoboxes, search engines results, voice-activated assistants, and information gathering projects [30].

Developed and maintained by ontology experts, data curators, and even anonymous volunteers, KGs have massively grown in size and adoption in the last decade, mainly as secondary sources of information. This means not storing new information, but taking it from authoritative and reliable sources which are explicitly referenced. As such, KGs depend on well-documented and verifiable provenance to ensure they are regarded as trustworthy and usable [56].

Processes to assess and assure the quality of information provenance are thus crucial to KGs, especially measuring and maintaining verifiability, i.e. the degree to which consumers of KG triples can attest these are truly supported by their sources [56]. However, such processes are currently performed mostly manually, which does not scale with size. Manually ensuring high verifiability on vital KGs such as Wikidata and DBpedia is prohibitive due to their sheer size.

ProVe (Provenance Verification) is proposed to assist data curators and editors in handling the upkeep of KG verifiability. It consists of an automated approach that leverages state-of-the-art Natural Language Processing (NLP) models, public datasets on data verbalisation and fact verification, as well as rule-based methods. ProVe consists of a pipeline that aims at automatically verifying whether a KG triple is supported by a web page that is documented as its provenance. ProVe first extracts text passages from the triple's reference. Then, it verbalises the KG triple and ranks the extracted passages according to their relevance to the triple. The most relevant passages have their stances towards the KG triple determined (i.e. supporting, refuting, neither) and finally ProVe estimates whether the whole reference supports the triple.

This task is a specific application of Automated Fact Checking (AFC), also known as AFC on KGs. AFC is a currently well explored topic of research with several published papers, surveys, and datasets [47, 57, 15, 43, 29, 27, 16, 58, 28, 59, 49, 48, 38], and generally defined as the verification of a natural language claim by collecting and reasoning over evidence extracted from text documents or structured data sources. Both the verification verdict and the collected evidence are the main outputs. While general AFC takes a textual claim and a searchable evidence base as inputs, AFC on KGs takes a single KG triple and its documented provenance in the form of an external reference.

Approaches tackling AFC on KGs are very few, with the two only works of note in a similar direction, as far as is known by the authors, being DeFacto [25, 14] and its successor FactCheck [45]. While they tackle this task mostly as defined above, they rely on a searchable document base instead of a given reference and judge triples on a true-false spectrum instead of verifiability. Like these few approaches, ProVe diverges from the general AFC framework and introduces a few different sub-tasks. Still, it makes use of the current state-of-the-art on those subtasks in common, being the first approach to tackle AFC on KGs with large pre-trained Language Models (LMs), which can be expanded to work in languages other than English and benefits from an Active Learning scenario.

ProVe is evaluated on an annotated dataset of Wikidata triples and their references, combining multiple types of properties and web domains. ProVe achieves promising results overall (75% accuracy and 68.1% F1-macro) on classifying references as either supporting their triples or not, with an excellent performance on explicit and naturally written references (87.5% accuracy, 82.9% F1-macro, 0.908 AUC). Additionally, ProVe assesses passage relevance with a strong positive correlation (0.5058 Pearson's r) to human judgements.

In summary, this paper's main contributions are:

1. A novel pipelined approach to evidence-based Automated Fact Checking on Knowledge Graphs based on large Language Models;
2. A benchmarking dataset of Wikidata triples and references for Automated Fact Checking on Knowledge Graphs, covering a variety of information domains as well a balanced sample of diverse web domains;
3. Novel crowdsourcing task designs that facilitate repeatable, quick, and large-scale collection of human annotations on passage relevance and textual entailment at good agreement levels.

These contributions directly aid KG curators, editors, and researchers in improving KG provenance. Properly deployed, ProVe can do so in multiple ways. Firstly, by assisting the detection of verifiability issues in existing references, bringing them to the attention of humans. Secondly, given a triple and its reference, it can promote re-usability of the reference by verifying it against neighbouring triples. Finally, given a new KG triple entered by editors or suggested by KG completion processes, it can analyse and suggest references. The remainder of this paper is structured as follows. Section 2 explores related work on KG data quality, mainly verifiability, as well as approaches to AFC on KGs. Section 3 presents ProVe's formulation and covers each of its modules in detail. Section 4 presents an evaluation dataset consisting of triple-reference pairs, including its generation and its annotation. Section 5 details the results of ProVe's evaluation. Finally, Section 6 delivers discussions around this work and final conclusions. All code and data used in this paper are available on Figshare [1] and GitHub. [2,3]

2 Related Work

ProVe attempts to solve the task of AFC on KGs, with the purpose of assisting data curators in improving the verifiability of KGs. Thus, to understand how ProVe approaches this task, it is important to first understand how the data quality dimension of verifiability is currently defined and measured in KGs, as well as how state-of-the-art approaches to general AFC and AFC on KGs tackle these tasks and how ProVe learns or differs from them.

[1] https://figshare.com/s/df0ec1c233ebd50817f4
[2] https://anonymous.4open.science/r/RSP-F367/
[3] https://anonymous.4open.science/r/ClaimVerificationHIT-A04D

2.1 Verifiability in KGs

In order to properly evaluate the degree to which ProVe adequately predicts verifiability, this dimension first needs to be well defined and a strategy needs to be established to measure it given an evaluation dataset. Verifiability in the context of KGs, whose information is mainly secondary, is defined as the degree to which consumers of KG triples can attest these are truly supported by their sources [56]. It is an essential aspect of trustworthiness [56, 11, 34], yet is amongst the least explored quality dimensions [56, 34], with most measures carried superficially, unlike correctness or consistency [34, 40, 2, 22, 1].

For instance, Farber et al. [11] measure verifiability only by considering whether any provenance is provided at all. Flouris et al. [12] look deeper into sources' contents, but only verify specific and handcrafted irrationalities, such as a city being founded before it had citizens. Algorithmic indicators are not suited to directly measure verifiability, as sources are varied and natural language understanding is needed. As such, recent works [33, 3] measure KG verifiability through crowdsourced manual verification, giving crowdworkers direct access to triples and references. Crowdsourcing allows for more subjective and nuanced metrics to be implemented, as well as for natural text comprehension [55, 8].

Thus, this paper employs crowdsourcing in order to measure verifiability metrics of individual triple-reference pairs. By comparing a pair's metrics with ProVe's outputs given said pair as input, ProVe and its components can be evaluated. Like similar crowdsourcing studies [33, 3], multiple quality assurance techniques are implemented to ensure collected annotations are trustworthy [10]. To the best of the authors' knowledge, this is the first work to use crowdsourcing as a tool to measure the relevance and stance of references in regards to KG triples at levels varying from whole references to individual text passages.

2.2 Automated Fact Checking on Knowledge Graphs

General AFC

Automated Fact Checking (AFC) is a topic of several works of research, datasets, and surveys [47, 57, 15, 43, 29, 27, 16, 58, 28, 59, 49, 48, 38]. AFC is commonly defined in the Natural Language Processing (NLP) domain as a broader category of tasks and subtasks [47, 57, 15] whose goal is to, given a textual claim and searchable document corpora as inputs, verify said claim's veracity or support by collecting and reasoning over evidence. Such evidence is extracted from the input document corpora and constitutes AFC's output alongside the claim's verdict. While a detailed exploration of individual AFC state-of-the-art approaches is out of this paper's scope, it is crucial to define their general framework in order to properly cover ProVe's architecture.

A general framework for AFC has been identified by recent surveys [57, 15], and can be seen in Figure 1. Zeng et al. [57] define it as a multi-step process where each step can be tackled as a subtask. Firstly, a *claim detection* step identifies which claims need to be verified. Based on such claims, a *document retrieval* step gathers documents that might contain information relevant to verifying the claim. A *sentence selection* step then identifies and extracts from retrieved documents a set of few individual text passages that are deemed the most relevant. Based on these passages, a *claim verification* step provides the final verdict. Guo et al. [15] add that a final *justification production* step is crucial for explainability. Given the framework's nature, it is no wonder pipelined approaches are extremely popular and compose the current state-of-the-art.

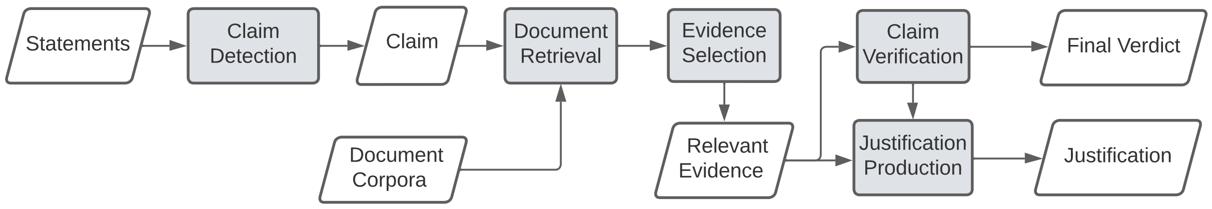


Fig. 1. Overview of a general AFC pipeline. White diamond blocks are documents and objects, and grey square blocks are AFC subtasks. Specific formulations and implementation of course might differ.

AFC mainly deals with text, both as claims to be verified and as evidence documents, due to recent advances in this direction being greatly facilitated by textual resources like the FEVER shared task [49] and its associated large-scale benchmark FEVER dataset [48]. Still, some tasks in AFC take semantic triples as verifiable claims, either from KGs [41, 21] or by extracting them from text. Some also utilise KGs as reasoning structures from where to draw evidence [15, 50, 46, 9, 42]. For instance, Thorne and Vlachos [46] directly map claims found in text to triples in a KG to verify numerical values. Both Ciampaglia et al. [9] and Shiralkar et al. [42] use entity paths in DBpedia to verify triples extracted from text. Other approaches based on KG embeddings associate the likelihood of a claim being true to that of it belonging as a new triple to a KG [5, 18].

These tasks, while incorporating semantic triples and KGs, can not be exactly defined as AFC on KG; either the verified triples do not come from full and consistent KGs, or the evidence used for reasoning is not taken from sources that could serve as provenance, but inferred from the graph itself.

AFC on KGs

AFC on KGs is a more specific task within AFC, explored by a handful of approaches, the most prominent of which are DeFacto [14] and its successor FactCheck [45]. Its main purpose is to ensure KGs are fit for use by asserting whether their information is verifiable by trustworthy evidence. Given a KG triple and either its documented external provenance or searchable external document corpora whose items could be used as provenance, AFC on KGs can be defined as the automated verification of said triple's veracity or support by collecting and reasoning over evidence extracted from such actual or potential provenance. Its outputs are the verdict and the evidence used.

KGCleaner [31] uses string matching and manual mappings to retrieve sentences relevant to a KG triple from a document, using embeddings and handcrafted features to predict the triple's credibility. Leopard [44] validates KG triples for three specific organisation properties, using specifically designed extractions from HTML content. Both approaches entail manual work overhead, cover a limited amount of predicates, and do provide human-readable evidence.

DeFacto [14] and its successor FactCheck [45] represent the current state-of-the-art on this task. They verbalise KG triples using text patterns and use it to retrieve web pages with related content. They then score sentences based on relevance to the claim and use a supervised classifier to classify the entire web page. Despite their good performance, both approaches depend on string matching, which might miss verbalisations that are more nuanced and also entail considerable overhead for unseen predicates. ProVe, on the other hand, covers any non-ontological predicate (such as *subclass of* and *main category of*) by using pre-trained LMs that leverage context and meaning to infer verbalisations.

Due to its specific application scenario, approaches tackling AFC on KGs differ from the general framework [57, 15] seen in Figure 1. A *claim detection* step is not deemed necessary, as triples are trivial to extract and it is commonly assumed they all need verifying. Alternatively, triples with particular predicates can be easily selected. The existence of a *document retrieval* step depends on whether provenance exists or needs to be searched from a repository, with the former scenario dismissing the need for the step. This is the case for ProVe, but not for DeFacto [14] and FactCheck [45], which search for web documents.

Additionally, KG triples are often not understood by the components' main labels alone. Descriptions, alternative labels, editor conventions, and discussion boards help define their proper usage and interpretation, rendering their meaning not trivial, in contrast to the natural language sentences tackled by general AFC. As such, approaches tackling AFC on KGs rely on transforming KG triples into natural sentences [25, 14, 45] through an additional *claim verbalisation* step. While both DeFacto [14] and FactCheck [45] rely on sentence patterns that are completed with the components' labels, ProVe relies on state-of-the-art Language Models (LMs) for data-to-text conversion.

Lastly, evidence document corpora normally used in general AFC tend to have a standard structure or come from a specific source. Both FEVER [48] and VitaminC [39] take their evidence sets from Wikipedia, with FEVER's even coming pre-segmented as individual and clean sentences. Vo and Lee [51] use web articles from snopes.com and politifact.com only. KGs, however, accept provenance from potentially any website domains. As such, unlike general AFC approaches, ProVe employs a *text extraction* step in order to retrieve and segment text from triples' references. While previous approaches simply remove HTML markup, ProVe employs a rule-based approach that allows for more flexibility.

Large Pre-trained Language Models on AFC

Advances towards textual evidence-based AFC, particularly the *sentence selection* and *claim verification* subtasks, have been facilitated by resources like the FEVER [49, 48] shared task and its benchmarking dataset. The FEVER dataset consists of a large set of claims annotated with one of three classes: supported, refuted, and not enough information to determine (neither). The dataset also provides pre-extracted and segmented passages from Wikipedia as evidence for each claim.

Tackling FEVER through pre-trained LMs [43, 27, 29] and graph networks [28, 59, 58] represents the current state-of-the-art. While approaches using graph networks (such as KGAT [28], GEAR [59], and DREAM [58]) for *claim verification* slightly outperform those based mainly on sequence-to-sequence LMs, they still massively depend on the latter for *sentence selection*. Additionally, explainability for task-specific graph architectures, like those of KGAT and DREAM, is harder to tackle than for generalist sequence-to-sequence LM architectures which are shared across the research community [7, 36, 23]. Slightly decreasing potential performance in favour of a simpler and more explainable pipeline, ProVe employs LMs for both *sentence selection* and *claim verification*.

On *sentence selection*, the common strategy is to assign relevance scores to text passages based on their contextual proximity to a verifiable claim. GEAR [59] does so with LSTMs, but use a BERT model to acquire text encodings. Soleimani et al. [43], KGAT [28], and current state-of-the-art DREAM [58] outperform GEAR by directly using BERT for the rankings, an approach ProVe also follows. Graph networks are employed at the *claim verification* subtask [28, 59, 58]. Soleimani et al. [43] are among the few to achieve near state-of-the-art results using an LM and a rule-based aggregation instead of graph networks. While ProVe handles the subtask similarly, it uses a weak classifier as its final aggregation.

As a subtask of AFC on KGs, *claim verbalisation* is normally done through text patterns [14, 45] and by filling templates [31], both of which can either be distantly learned or manually crafted. ProVe is the first approach to utilise an LM for this subtask. Amaral et al. [4] shows that a T5 model fine-tuned on WebNLG achieves very good results when verbalising triples from Wikidata across many domains. ProVe follows suit by also using a T5.

Table 1 shows a comparison of ProVe to other AFC approaches mentioned in this section grouped by specific task, showcasing the particular subtasks each targets, as well as the datasets used as a basis for their evaluation. AFC on KG is amongst the least researched tasks within AFC. ProVe is the first to tackle it through fine-tuned LMs that adapt to unseen KG predicates and to be evaluated on a Wikidata dataset consisting of multiple non-ontological predicates.

Task	Input Type	Evidence Source	Evidence Returned	Subtasks	Evaluation Dataset	Approaches
General text-based AFC	Textual claims	Text	Yes	DR, SS, CV	FEVER	[43, 29, 28, 59, 58],
Graph-based AFC	Textual claims	KG	Yes	SS, CV	Freebase	[50, 46]
KG triple prediction	KG triples	KG paths	Yes	SS, CV	DBpedia, SemMedDB, Wikipedia	[41, 21, 9, 42],
		KGE	No	RE, EL, CV	Kaggle, news articles DBpedia, Freebase	[5, 18]
AFC on KGs	KG triples	Text	Yes	CVb, DR, TA, CV	DBpedia, FactBench	[25, 14, 45]
			No	SS, CV	Wikidata (48 predicates), SWC 2017	[31, 44]
			Yes	CVb, TR, SS, CV	Wikidata (any non-ontological predicate)	ProVe

Table 1. Comparison between ProVe and others within AFC. KGE = KG Embeddings, DR = Document Retrieval, SS = Sentence Selection, CV = Claim Verification, RE = Relation Extraction, EL = Embedding Learning, CVb = Claim Verbalisation, TA = Trustworthiness Analysis, TR = Text Retrieval.

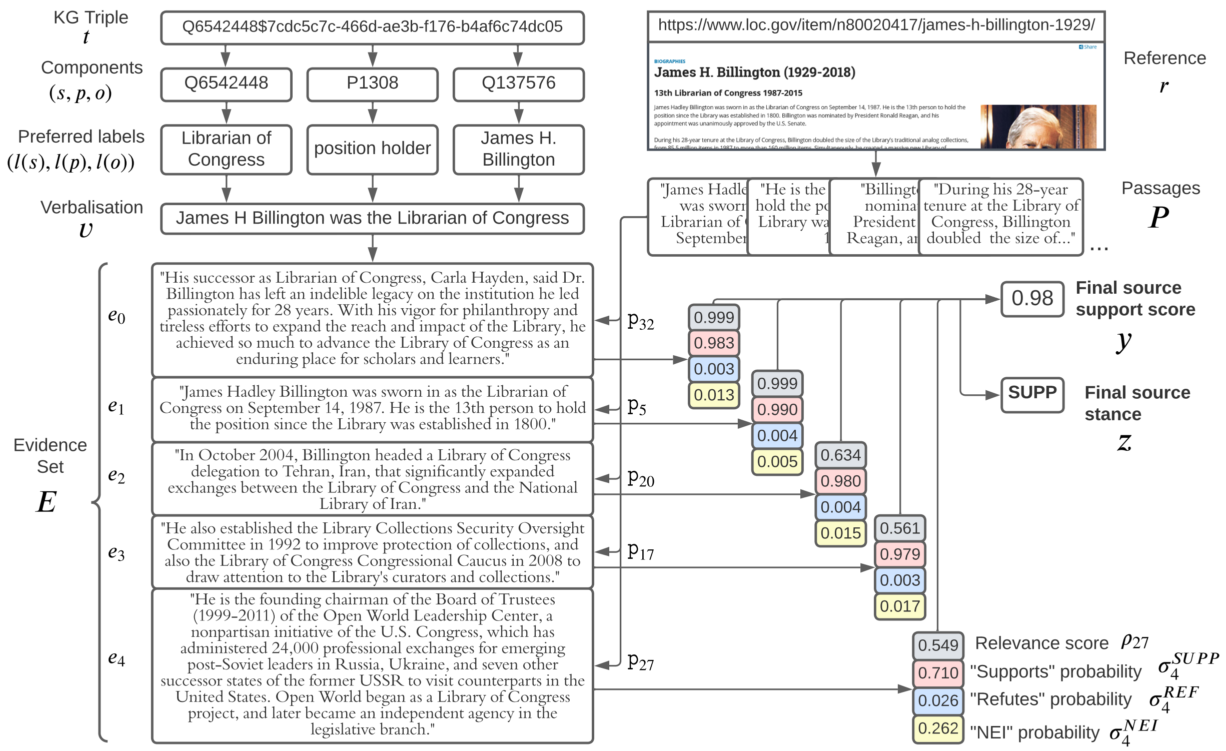
3 Approach

ProVe consists of a pipeline for Automated Fact Checking (AFC) on Knowledge Graphs (KG) that, provided with a KG triple that is not ontological in nature (e.g. denoting subclasses, categories, lists, etc) and its documented provenance in the form of a web page or text document, automatically verifies whether the page textually supports the triple, retrieving from it relevant text passages that can be used as evidence. This section presents an overview of ProVe and its task, as well as detailed descriptions of its modules and the subtasks they target.

3.1 Overview

From a KG's set of non-ontological triples T, consider a KG triple $t \in T$, where t is composed by a subject, a predicate, and an object, i.e. $t = (s, p, o)$. Consider also a reference r to a web page or text document, acting as the documented provenance of t. ProVe assesses whether r textually supports t through a pipeline of rule-based methods and language models.

Figure 2 shows a KG triple (taken from Wikidata), its reference to a web page, and ProVe's processing according to the definitions provided in this section. ProVe extracts the text from r and divides it into a set of passages P. Each passage $\mathrm{p}_i \in P$, with i in the closed integer interval $[0..|P|-1]$, receives a relevance score $\rho_i \in [-1, 1]$ indicating how relevant they are to the triple t. The five highest-ranking passages from P are selected as evidence and assembled into the evidence set E. Each evidence $e_i \in E$ receives three stance probabilities $(\sigma_i^{SUPP}, \sigma_i^{REF}, \sigma_i^{NEI})$, with i in the integer interval $[0..4]$, such that each probability $\sigma_i^k \in [0, 1]$ and $\sum \sigma_i^k = 1$ for $k \in \mathcal{K}$ and $\mathcal{K} = \{SUPP, REF, NEI\}$. These probabilities denote the individual stance of evidence e_i towards the triple t, which can either support the triple ($SUPP$), refute it (REF), or not have enough information to do either (NEI). Finally, ProVe uses the relevance scores ρ_i and the stance probabilities $\sigma_i^k \; \forall \; k \in \mathcal{K}$ of each $e_i \in E$ to define a final stance $z \in \mathcal{K}$, as well as to calculate a support probability $y \in [0, 1]$ indicating how much the triple t is supported by its reference r.


KG Triple t
Q6542448$7cdc5c7c-466d-ae3b-f176-b4af6c74dc05
Components (s, p, o)
Q6542448
P1308
Q137576
Preferred labels $(l(s), l(p), l(o))$
Librarian of Congress
position holder
James H. Billington
Verbalisation v
James H Billington was the Librarian of Congress
https://www.loc.gov/item/n80020417/james-h-billington-1929/
James H. Billington (1929-2018)
13th Librarian of Congress 1987-2015
Reference r
Passages P
"James Hadl... was sworn... Librarian of... September...
"He is the... hold the p... Library wa...
"Billingt... nomina... President... Reagan, a...
"During his 28-year tenure at the Library of Congress, Billington doubled the size of..."
...
Evidence Set E
e_0
e_1
e_2
e_3
e_4
p_{32}
p_5
p_{20}
p_{17}
p_{27}
0.999
0.983
0.003
0.013
0.999
0.990
0.004
0.005
0.634
0.980
0.004
0.015
0.561
0.979
0.003
0.017
0.549
0.710
0.026
0.262
Relevance score ρ_{27}
"Supports" probability σ_4^{SUPP}
"Refutes" probability σ_4^{REF}
"NEI" probability σ_4^{NEI}
0.98
Final source support score y
SUPP
Final source stance z

"His successor as Librarian of Congress, Carla Hayden, said Dr. Billington has left an indelible legacy on the institution he led passionately for 28 years. With his vigor for philanthropy and tireless efforts to expand the reach and impact of the Library, he achieved so much to advance the Library of Congress as an enduring place for scholars and learners."

"James Hadley Billington was sworn in as the Librarian of Congress on September 14, 1987. He is the 13th person to hold the position since the Library was established in 1800."

"In October 2004, Billington headed a Library of Congress delegation to Tehran, Iran, that significantly expanded exchanges between the Library of Congress and the National Library of Iran."

"He also established the Library Collections Security Oversight Committee in 1992 to improve protection of collections, and also the Library of Congress Congressional Caucus in 2008 to draw attention to the Library's curators and collections."

"He is the founding chairman of the Board of Trustees (1999-2011) of the Open World Leadership Center, a nonpartisan initiative of the U.S. Congress, which has administered 24,000 professional exchanges for emerging post-Soviet leaders in Russia, Ukraine, and seven other successor states of the former USSR to visit counterparts in the United States. Open World began as a Library of Congress project, and later became an independent agency in the legislative branch."

Fig. 2. An example of the inputs and outputs of ProVe when applied to a Wikidata triple and its provenance. A triple's (t) subject, predicate, and object elements (s, p, o) have their labels extracted and verbalised (v). Reference r has its passages extracted (P). The 5 most relevant passages are compiled as the evidence set (E), with their respective relevance scores (ρ_i) and stance probabilities (σ_i^k) used to calculate a final class (z) and a support probability (y). Note that i indices between passages in P and evidence in E are different, i.e. the 27^{th} extracted passage is the 4^{th} evidence.

A modular view of ProVe's pipeline can be seen in Figure 3. Its inputs, as previously stated, are a KG triple (t) and a referenced web page or text document (r), while its outputs are a final stance class (z), a support probability (y), and a set of textual evidence used to calculate it (E). ProVe takes any non-ontological KG triple as long as its components are accompanied by labels in natural language. The claim verbalisation module takes the preferred labels of each of the triple's components (i.e. subject, predicate, and object) as its inputs and produces a natural language sentence that expresses the same information (v). As KG entities and predicates might contain multiple labels, multiple possible verbalisations can be generated; users might chose those labels that best portray the triple's meaning, which are here defined as preferred labels, and even rules within the KG can determine them. The reference (r) can consist of any HTML page containing natural language text, but also plain text documents. As ProVe makes no assumptions as to the page's layout in order to optimise recall, its text retrieval module extracts all identified passages (P) from the page, even if they contain boilerplate text or have poor syntax due to layout-dependant contents, e.g. tables, headers, etc.

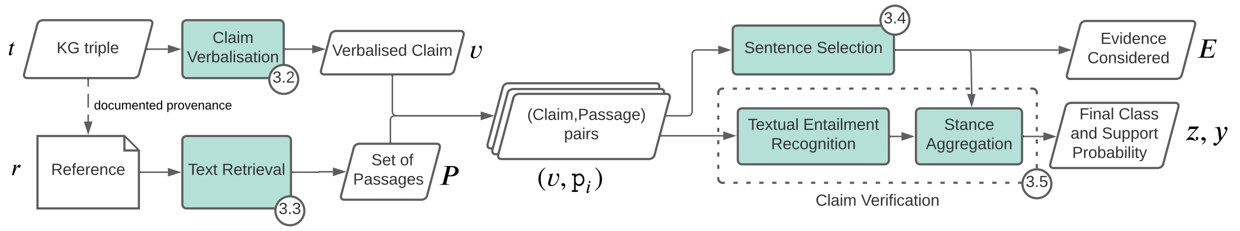


Fig. 3. Overview of ProVe's pipeline. The white blocks are artefacts while the green are modules, further detailed in the subsections indicated in the circles.

Pairs consisting of the verbalised claim (v) alongside each extracted passage (p_i) are given to the sentence selection and the claim verification modules. At the former, extracted passages are given a relevance score from -1 to 1, indicating how contextually relevant to the claim they are, regardless of stance. The five highest scoring passages are selected as evidence (E). The claim verification module has two steps. First, a Textual Entailment Recognition (TER) step assigns each extracted passage with probabilities of having the three following stances: supports the claim, refutes the claim, or does not hold enough information for either (NEI). Finally, a stance aggregation step takes the relevance scores plus stance probabilities for the five passages in the evidence set and outputs the final class (z) and support probability (y), indicating how supportive of the triple is the reference.

ProVe's workflow differs from the AFC framework seen in Figure 1. This is due to the particular task ProVe tackles, i.e. the verification of KG triples using text from documented provenance, where triples can have any non-ontological predicate and such provenance can come from varied sources. As detailed in Section 2 and evidenced in Table 1, this is currently a little studied task compared to others in AFC, posing distinct problems and requiring specific subtasks to be solved. For instance, ProVe does not need to perform either claim detection or document retrieval, as both the claims and the sources are given to it as inputs, although such modules can be easily plugged in from other pipelines.

On the other hand, as ProVe handles both KG triples and unstructured text with the same model architectures and does not make use of KG paths for evidence, it needs to convert the triples into text through a claim verbalisation module, akin to most other approaches in this task [25, 14, 45, 31]. As its input references can lead to web pages having any HTML layout and their text does not come pre-segmented (such as with FEVER [48]), ProVe needs a non-trivial text retrieval module so that it can identify informative passages. Finally, as such KGs are often secondary sources of information and triples should not include conclusions or interpretations from editors, ProVe needs to consider pieces of evidence in isolation; this is to lower ProVe's reliance on multi-sentence reasoning, as concluding a triple from multiple text passages should not configure support in this task. Thus, ProVe first identifies stances of retrieved evidence individually, aggregating them into a final verdict afterwards. Each of ProVe's modules is further detailed in the remainder of this section.

3.2 Claim Verbalisation

KG entities and properties have natural language labels that help clarify their meanings, with KGs like Wikidata and Yago also containing multiple aliases and alternative names. However, these entities and predicates are often not created by prioritising human understanding, but rather data organisation, and thus rely heavily on descriptions in order to set out proper usage rules. Many serve as abstract concepts that unite other related but not identical concepts, using a very broad main label and more specific aliases. One example of such is Wikidata's *inception* property (*P571*), which indicates the date in which something was founded, written, formed, painted, or created, and applies to any entity with a beginning; its description clearly points out that for dates of official opening, *P1619* should be used instead. Many also depend heavily on context (e.g. subject and object types) or editor conventions to have a clear meaning. One example of such is the *child* property (*P40*), which follows the convention that the subject has the object as its child, but should not be used for stepchildren. However, the triple (*John, child, Paul*) alone makes it unclear which is the parent. As such, merely concatenating labels does not convey the full meaning of the triple [14, 25, 45]. Thus, ProVe relies on a claim verbalisation component that, based on similar triples and their verbalisations, is able to fill format and meaning gaps.

ProVe's claim verbalisation module takes as input a KG triple t, made by its subject, predicate, and object components such that $t = (s, p, o)$. A component's preferred label is assumed to be its main (and often only) label, but alternative labels or aliases can be manually chosen by editors and curators employing ProVe. Denoting by the function $l(\cdot)$ the process of defining a component's preferred label, ProVe's claim verbalisation module outputs a natural language formulation v, called the verbalisation, as defined by Equation 1.

$$v = \phi(l(s), l(p), l(o)) \tag{1}$$

The function $\phi(\cdot)$ represents the generation of a fluent and adequate verbalisation from the components' preferred labels. A fluent verbalisation is defined as one written in good grammar, resembling natural text, and an adequate verbalisation as one that carries the same meaning as the original triple intended. Like recent works in data verbalisation [37, 4], a pre-trained transformer is used for this subtask. The function $\phi(\cdot)$ is carried out by a T5-base [35] model fine-tuned on the WebNLG 2017 dataset [13]. The WebNLG 2017 dataset consists of DBpedia triples belonging to 15 categories and their corresponding verbalisations in English; 10 categories (the *SEEN* partition) were used for training, validation, and testing, and the remaining 5 (the *UNSEEN* partition) for testing only. Fine-tuning was carried out with a constant 3e-5 learning rate, an Adam optimiser, and a cross-entropy loss for 100 epochs with early stopping patience of 15 epochs. Its text generation is done via beam search with 3 beams.

Amaral et al. [4] use this exact same model to produce the WDV dataset, which consists of verbalised Wikidata triples, and then evaluate its quality with human annotations on both fluency and adequacy. Such evaluations are covered in more detail in Section 5. In WDV, main labels are used as preferred labels for all triple components, despite aliases often representing better choices. ProVe allows editors to manually define the behaviour of $l(\cdot)$ to replace contextually-dependent and vague main labels with alternate labels in order to address some of the fluency and adequacy issues observed in WDV. While this entails the extra effort by ProVe's users of choosing proper labels, it is still a much more scalable alternative to manually generating verbalisations.

3.3 Text Retrieval

In KGs, provenance is documented per individual triples and is often presented as URLs to web pages or text documents. Such references form the basis of KG verifiability and should point to sources of evidence that support their associated KG triples. Additionally, they can come from a huge variety of domains as long as they adhere to verifiability criteria, that is, humans can understand the information they contain. As humans are excellent in making sense of structured and unstructured data combined, KG editors do not need to worry much about how references express their information. Images, charts, tables, headers, infoboxes, and unstructured text, can all serve as evidence to the information contained in KG triple. However, this complicates the automated extraction of such evidence in a standard format so that LMs can understand it.

Rather than only free-flowing text, referenced web pages can have multiple sections, layouts, and elements, making it non-trivial to automatically segment its textual contents into passages. Thus, ProVe employs a combination of rule-based methods and pre-trained sentence segmenters to extract passages. Figure 4 details this process. The module takes as input a reference r, which can either be a URL to a web page or a text document. If an URL, the module extracts all HTML contents via a web scrapper, assuring that all contents accessible by users are rendered and processed. The module then removes scripts and code from the HTML, leaving only markup and text. A list of rule-based cleaning steps are then applied. Whitespaces in continuous text elements are corrected by ensuring text within tags such as $<p>$ do not have separations that could be breaking sentences. Tags that are sure to be boilerplate are removed, such as tables of contents and navigation bars. Text that is broken across sequential similar tags is joined. Lastly, spacing and punctuation (full stops) are corrected. Following this process addresses the most severe cases of improper sentence segmentation. Leftover HTML markup is removed and the text text is fed into spaCy's sentence segmenter using the $en\_core\_web\_lg$ model [17], producing a set of text segments S. If r is a text document, it is fed directly into sentence segmentation.

As a last step, multiple n-sized sliding window concatenations are used to create the final set of passages P. Given a positive integer $n \in \mathcal{N}$, where $\mathcal{N}$ is the set of window sizes to be applied, an n-sized window slides through the sequential set S of text segments produced by the sentence segmentation step. Let $S^i_j = [s_i, \dots, s_j]$, for $j \geq i$, be the window including all segments from s_i to s_j and $\odot(\cdot)$ the function that concatenates a sequence of segments interleaving them with a blank space as a separator. Equations 2 and 3 define the set of all passages P_n produced by a n-sized sliding window and the union P of all such passages, respectively.

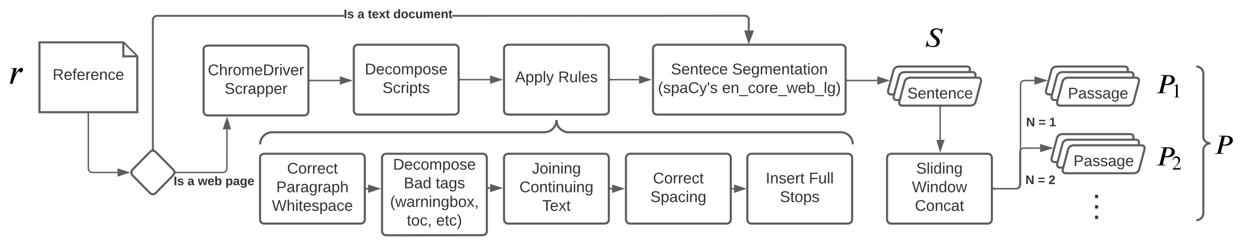


Fig. 4. Illustration of the text extraction module's workflow, taking a reference r as input, dividing its text into a set of segments S, and concatenating them into a set of extracted passages P.

$$P_n = \{\odot(S^i_{i+n-1}) \mid 0 \leq i \leq |S| - n\} \tag{2}$$

$$P = \bigcup_{n \in \mathcal{N}} P_n \tag{3}$$

ProVe concatenates text segments in this fashion for two reasons. Firstly, meaning can often be spread between sequential segments, e.g. in the case of anaphora. Secondly, HTML layouts might separate text in ways ProVe's general rules were not able to join, e.g. a paragraph describing an entity in the header. For a trade-off between coverage and sentence length, ProVe defines $\mathcal{N} = \{1, 2\}$, i.e. it concatenate all segments by themselves (P_1) and all sequential pairs (P_2).

Current best approaches to extracting textual content from web pages, often called boilerplate removal, are based on supervised classification [52, 26] and, as no classification is perfect, might miss relevant text. ProVe's text retrieval module aims at maximizing recall by retrieving all reachable text from the web page and arranging it into separate passages by following a set of rules based on the HTML structure. The sentence selection module is later responsible for performing relevance-based filtering. ProVe's rule-based method can easily be updated with ad hoc cleaning techniques to help treat difficult cases, such as linearisation or summarisation of tables, automated image and chart descriptions, converting hierarchical header-paragraph structures and infoboxes into linear text, etc.

3.4 Sentence Selection

As ProVe's text extraction module extracts all the text in a web page in the form of passages P, it needs a way to filter those based on their relevance to the triple t. Sentence selection consists of, given a claim, rank a set of sentences based on how relevant each is to the claim, where relevance is defined as contextual proximity e.g. similar entities or overlapping information. Sentence selection is an integral part of most recent AFC approaches [15, 57], incuding AFC on KGs [45, 14], as discussed in Section 2.2. It is a subtask of the FEVER fact-checking shared task [49], directly supported as a supervised task by the FEVER dataset [48], and is explored by a large body of work [43, 29, 28, 59, 58] to excellent results.

Following on KGAT's [28] and DREAM's [58] approach to FEVER's sentence selection subtask, ProVe employs a large pre-trained BERT transformer. ProVe's sentence selection BERT is fine-tuned on the FEVER dataset by adding to it a dropout and a linear layer, as well as a final hyperbolic tangent activation, making outputted scores range from -1 to 1, and training it on a pairwise margin ranking loss with the margin set to 1. Fine-tuning is achieved by feeding the model pairs of inputs, where the first element is a concatenation of a claim and a relevant sentence, while the second element is the same but with an irrelevant sentence instead, and training it to assign higher scores to the first element, such that the difference in scores between the pair is 1 (the margin). FEVER is used for training and validation. For each FEVER claim in the training and validation partition, relevant sentences are provided as annotations. Irrelevant sentences were retrieved by applying the same document retrieval process used by other works [43, 28, 59] to define relevant Wikipedia articles, which FEVER breaks into pre-segmented sentences. All sentences from such retrieved documents that were not already annotated as relevant were taken as irrelevant.

This fine-tuned module is thus used to assign scores ranging from -1 to 1 to passages from P, expressing how relevant they are to the given triple t. Taking the triple t's verbalisation v as the claim, and a passage p_i as the sentence, the BERT model takes the concatenation of v and p_i as input and outputs a relevance score $\rho_i \in [-1, 1]$. This is defined in Equation 4, where $\psi(\cdot)$ represents the execution of the sentence selection BERT on the concatenated input.

ProVe ranks all passages $\mathrm{p}_i \in P$ based on their relevance scores ρ_i. As passages are generated with sliding windows of different sizes by the text extraction module, they might have overlapping content. To avoid unnecessary repetition of information, a passage p_i is removed from P whenever there is another passage p_j that overlaps with it and is more relevant (i.e, $\rho_j > \rho_i$), yielding the set of passages P^*, whose five highest scoring passages constitute the evidence set E (see Equation 5).

$$\rho_i = \psi(v, \mathrm{p}_i) \tag{4}$$

$$E = \underset{P' \subseteq P^*, |P'|=5}{\operatorname{argmax}} \sum_{\mathrm{p}_i \in P'} \rho_i \tag{5}$$

3.5 Claim Verification

As discussed in Section 2.2, claim verification is a crucial subtask in AFC, central to various approaches [15, 47], and consists on assigning a final verdict to a claim, be it on its veracity or support, given retrieved relevant evidence. ProVe's claim verification relies on two steps: first, a pre-trained BERT fine-tuned on data from FEVER [48] performs Textual Entailment Recognition (TER) to detect stances of individual pieces of evidence, and then an aggregation considers the stances and relevance scores from all evidence to define a final verdict. As ProVe also uses a BERT model for sentence selection, its approach is similar to that of Soleimani et al. [43], which uses two fine-tuned BERT models, one for sentence selection, another for claim verification. Although task-specific graph-based approaches [28, 58] outperform Soleimani, it is by less than a percentage point (on FEVER score), while explainability for such generalist pre-trained LMs is increasingly researched [7, 36, 23] by the NLP community.

Textual Entailment Recognition

Like sentence selection, claim verification is a well defined subtask of AFC, supported by both the FEVER shared task and the FEVER dataset. FEVER annotates claims as belonging to one of three TER classes: those supported by their associated evidence ('SUPPORTS'), those refuted by it ('REFUTES'), and those wherein the evidence is not enough to reach a conclusion ('NOT ENOUGH INFO', abbreviated as 'NEI'). As previously mentioned, ProVe is meant to handle KGs as secondary sources of information. Thus, it assesses evidence first in isolation, aggregating assessments afterwards in a similar fashion to other works [43, 29].

ProVe's TER step is a BERT model fine-tuned on a multiclass classification TER task. It consists of identifying a piece of evidence's stance towards a claim using the three classes from FEVER ('SUPPORTS','REFUTES','NEI'). To fine-tune such model, a labelled training dataset of claims-evidence pairs is built out of FEVER. For each claim in FEVER labeled as 'SUPPORTS', all sentences annotated as relevant to it are paired with the claim; such pairs are labelled as 'SUPPORTS'. The same is done for all claims in FEVER labeled as 'REFUTES', generating pairs classified as 'REFUTES'. For claims labeled as 'NOT ENOUGH INFO', FEVER does not annotate any sentence as relevant to them. Thus, ProVe's sentence selection module is applied to documents deemed relevant to such claims (retrieved in a similar fashion to KGAT [28]) and each claim is paired with all sentences that have relevance scores greater than 0 in regards to them. All such pairings are labelled 'NEI'. Fine-tuning was carried for 2 epochs with an AdamW optimizer with 0.01 weight decay. Population Based Training was used to tune learning rate, batch sizes, and warmup ratio.

Thus, for a verbalisation v obtained from a triple t and a piece of evidence $e_i \in E$, retrieved as a passage from the t's reference r, ProVe's TER step returns a probability array σ_i that describes e_i's stance towards v. Given that v is fluent and adequate, σ_i also describes e_i's stance towards t. Equation 6 formulates this, where $\tau(\cdot)$ is the function representing ProVe's TER BERT model, which takes the concatenation of v and e_i as input and outputs σ_i, an array that is normalised through a softmax layer. The array σ_i consists of the probabilities of each FEVER class $k \in \mathcal{K} = \{SUPP, REF, NEI\}$. Notice that $\sum_{k \in \mathcal{K}} \sigma_i^k = 1$.

$$\sigma_i = (\sigma_i^{SUPP}, \sigma_i^{REF}, \sigma_i^{NEI}) = \tau(e_i, v) \tag{6}$$

Stance Aggregation

After classifying the stance and relevance of each individual piece of evidence $e_i \in E$ towards the triple t, ProVe aggregates these scores (ρ_i and σ_i) into a final stance class $z \in \mathcal{K}$ and a probability y denoting the level of support shown by the triple-reference pair (t, r). Multiple aggregation strategies can be adopted, with ProVe proposing three:

1. A simple weighted sum σ of the TER probability arrays σ_i using the relevance scores ρ_i as weights, where negative relevance scores are dismissed (Equation 7). A final TER class z can be defined as the class $k \in$

$\mathcal{K}$ with the highest weighted sum (Equation 8). The support probability y is then defined as the weighted summed probability of the 'SUPPORTS' class (Equation 9).

$$\sigma^k = \sum_{0 \leq i \leq |E|} max(\rho_i, 0) * \sigma_i^k \tag{7}$$

$$z = \operatorname*{argmax}_{k \in \mathcal{K}} \sigma^k \tag{8}$$

$$y = \sigma^{SUPP} \tag{9}$$

2. The rule-based strategy adopted by Malon et al. [29] and Soleimani et al. [43]. A triple-reference pair (t, r) is assigned a final TER class z (Equation 10) of 'SUPPORTS' if any individual evidence $e_i \in E$ is most likely to support the triple. If that is not the case, z is set as 'REFUTES' if any individual evidence $e_i \in E$ is most likely to refute the triple. If that is also not the case, z is set to 'NEI'. The final probability y is 1 is the triple-reference is classified as 'SUPPORTS' and 0 otherwise (Equation 11). This strategy does not allow editors to vary the classification threshold.

$$z = \begin{cases} SUPP, & \text{if there exists } e_i \in E \text{ s.t. } \operatorname{argmax}_{k \in \mathcal{K}} \sigma_i^k = SUPP \\ REF, & \text{else if there exists } e_i \in E \text{ s.t. } \operatorname{argmax}_{k \in \mathcal{K}} \sigma_i^k = REF \\ NEI, & \text{otherwise} \end{cases} \tag{10}$$

$$y = \begin{cases} 1, & \text{if } z = SUPP \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

3. A simple classifier, trained on an annotated set of triple-reference pairs (t, r). The classifier takes as features all relevance scores ρ_i and all TER probability arrays σ_i calculated from every piece of evidence $e_i \in E$, as well as their sizes in characters. It is trained on a multiclass classification task to predict the annotated final TER class z of the pair (t, r) by outputting a probability array θ, as defined by Equation 12, where $\omega(\cdot)$ represents the classifier. Thus, z can be defined as the class with the highest probability (Equation 13), and y as the probability θ^{SUPP} assigned to the 'SUPPORTS' class (Equation 14).

$$\theta = (\theta^{SUPP}, \theta^{REF}, \theta^{NEI}) = \omega(\{(\rho_i, \sigma_i, |e_i|) \mid e_i \in E\}) \tag{12}$$

$$z = \operatorname*{argmax}_{k \in \mathcal{K}} \theta^k \tag{13}$$

$$y = \theta^{SUPP} \tag{14}$$

4 Reference Evaluation Dataset

This section presents and describes the dataset used to evaluate ProVe: Wikidata Textual References (WTR). WTR is mined from Wikidata, a large scale multilingual and collaborative KG, produced by voluntary anonymous editors and bots, and maintained by the Wikimedia Foundation [53]. WTR consists of a series of detailed and annotated triple-reference pairs, each consisting of a non-ontological Wikidata triple paired with a reference to a web page, documented as its provenance. Unlike other benchmarking datasets used in AFC, WTR contains no artificial data and reflects only naturally occurring information. WTR's triples are detailed with all three main components (subject, predicate, and object), as well as their unique Wikidata identifiers, main labels, aliases (alternative labels), and textual descriptions. WTR's references are detailed with the URLs they resolve to, as well as the HTML contents within. WTR is balanced in terms of web domains contemplated, meaning the web domains represented by its references have mostly an equal number of triple-reference pairs.

Each triple-reference pairing in WTR is annotated both at evidence-level and at reference-level. Evidence-level annotations are provided by crowd-workers and describe the stance that specific text passages from the reference display towards the triple. Reference-level annotations are provided by the authors and describe the stance the whole referenced web page displays towards the triple. Evaluation, described in Section 5, consists of comparing ProVe's final class (z) and support probability (y) outputs to such annotations. Section 4.1 covers WTR's construction, while Section 4.2 details its annotation process.

4.1 Dataset Construction

Wikidata has been chosen as the source for ProVe's evaluation dataset, as it contains vast amounts of triples that explicitly state their provenance, pertain to various domains, and are accompanied by aliases and descriptions that greatly aid annotators. Since many references in Wikidata can be automatically verified through API calls, as showcased by Amaral et al. [3], WTR is built focusing on those that can not. Furthermore, to prevent biases towards frequently used web domains, such as Wikipedia and The Peerage, WTR is built to represent a variety of web domains with equal amounts of samples from each.

Selecting References

WTR is constructed from the Wikidata dumps from March 2022. First, all references are extracted. Those associated to at least one triple and which lead to a web page by either an external URL (through the `reference URL (P854)` property), or by an external identifier property that has formattable URLs, e.g. `VIAF ID (P214)`, are kept. These two types of references constitute 91.52% of all references. The remaining portion consists of references to Wikidata items, inferred from other specific Wikidata claims, imported from Wikipedia (but no page specified), and those without any provenance property. These types of references are avoided due to potential issues with circular or vague provenance. Close to 20M unique references are extracted.

Next, each extracted reference has their initial web URLs defined. For references with direct URLs (`reference URL (P854)`), such URLs are used. For references with external ID properties, the property's `formatter URL (P1630)` is combined with the linked external ID to establish the URL. For instance, a reference might use the `IMDb ID (P345)` property, which has the formatter URL "https://wikidata-externalid-url.toolforge.org/?p=345&url_prefix=https://www.imdb.com/&id=$1". By replacing the '$1' with the ID linked by the property, one establishes the IMDB URL represented by the reference.

The extracted set of references and their respective initial URLs is then filtered. References that are inadequate to the scenario in which ProVe will be used are removed. These consist of three groups:

1. References with URLs to domains that have APIs or content available as structured data (e.g. JSON or XML), as these can be automatically checked through APIs, e.g. PubMed, VIAF, UniProt, etc.
2. References with URLs linking to files such as CSV, ZIP, PDF, etc., as parsing these file formats is outside of ProVe's scope.
3. References with URLs to pages that have very little to no information in textual format, such as images, document scans, slides, and those consisting only of infoboxes, e.g. nga.gov, expasy.org, Wikimedia commons, etc.

As shown by Amaral et al. [3], this first group is very substantial, with an estimated over 70% of English references being automatically verifiable through API calls. Additionally, references with URLs to websites not in English (according according to FastText's language identification models [19, 20]), posing security risks, or unavailable (e.g. 404 and 502 HTML response codes) are removed.

Close to 7M references are left after these removals, wherein English Wikipedia alone represents over 40% of URLs. To avoid biasing evaluation towards populous web domains, a stratified sampling is carried using web domains as strata. The 30 most populous web domains are defined as 30 separate groups, with two additional groups formed from remaining web domains: *RARE*, for web domains that appear only once, and *OTHER*, for all others.

Pairing References with Triples

Given the total number of references contemplated (7M), a sample size of 385 represents a 95% confidence interval and a 5% margin of error. An equal amount of references from each of the 32 groups is sampled, totalling over 400 references. Samples for a group are retrieved one by one through the following method. A reference is randomly sampled without replacement and checked as to whether it is associated to at least one triple fitting to be used for evaluation; if so, it is kept. A triple fitting for evaluation is defined as one which carries non-ontological and reliable meaning, and which can be expressed concisely through natural language. They are identified by the following criteria:

– Is not deprecated by Wikidata;
– Has an object that is not the "novalue" or "somevalue" special nodes;
– Has an object that is not of an unverbalisable type, i.e. URLs, globe coordinates, external IDs, and images.

- Has a predicate that is not ontological, e.g. `instance of (P31)`, `merged into (P7888)`, `entry in abbreviations table (P8703)`, etc.

These steps produce a stratified representative sample of references, including their unique identifiers, resolved URLs, web domains, and HTTP response attributes. Finally, for each sampled reference, a triple-reference pair is formed by extracting from Wikidata a random triple associated to the reference and fitting for evaluation, alongside its unique identifiers, object data types, main labels, aliases, and descriptions. This construction process creates WTR, ensuring it is composed of triples carrying non-ontological meaning and verifiable through their associated references, thus useful for evaluating ProVe. It also ensures meaning and context understanding is evaluated, rather than merely string matching, e.g. in the case of URLs, globe coordinates, and IDs. As for image data, tackling such multimodal scenarios is outside ProVe's scope.

4.2 Dataset Annotation

As described in Section 3 (see Figure 2), ProVe tackles its AFC task as a sequence of text extraction, ranking, classification, and aggregation subtasks. Given a triple-reference pair, ProVe extracts text passages from the reference, ranks them according to relevance to the triple, and individually classifies them according to their stance towards the triple. Then, triple-reference pairs are classified according to the overall stance of the reference towards the triple.

To allow for a fine-grained evaluation of ProVe's subtasks, WTR receives three sets of annotations: (1) on the stance of individual pieces of evidence towards the triple, (2) on the collective stance of all evidence, and (3) on the overall stance of the entire reference. The two first sets of annotations are deemed evidence-level annotations, while the last is reference-level. Crowdsourcing is used to collect evidence-level annotations, due to the large number of annotations needed (six per triple-reference pair) in combination with the simplicity of the task, which requires workers only to read short passages of text. Reference-level annotations are less numerous (one per triple-reference pair), much more complex, and hence manually annotated by the authors.

Crowdsourcing Evidence-level Annotations

Collecting evidence-level annotations for all retrievable sentences of each triple-reference pair in WTR, in order to account for different rankings that can be outputted by ProVe, would be prohibitively expensive and inefficient. Thus, evidence-level annotations are only provided for the five most-relevant passages in each reference, i.e. the collected evidence. First, ProVe's text retrieval (Section 3.3) and sentence selection (Section 3.4) modules are applied to each reference and the five pieces of evidence for each collected. This does not severely bias the annotation towards highly-relevant text passages, actually allowing for a more even collection of both relevant and irrelevant text passages, as often only a couple of passages amidst the five tend to be relevant. Then, evidence-level annotations for each individual piece of evidence, as well as for the whole evidence set, are collected through crowdsourcing, totalling 6 annotation tasks per triple-reference pair.

Task Design Two crowdsourcing task designs have been created to carry out this evidence-level annotation process. Task design 1 (T1) asks workers to assess the stance of an individual evidence towards the triple, which can also be used as an indication of relevance. Each task in T1 is a bundle of 6 subtasks, each providing the worker with a Wikidata triple, a piece of evidence extracted from its paired reference, the reference's URL, and asking the worker the stance of that evidence: either it **supports** the claim, **refutes** it, has **not enough information** for either, or the worker is **not sure**. This bundling is done in order to get more annotations out of a single crowdsourcing task assignment. Task design 2 (T2) asks workers to assess the collective stance of the evidence set (all five individual pieces of evidence) towards the triple. Similarly to tasks in T1, tasks in T2 are made from 6 subtasks. Each T2 task providing the worker with a triple, five text passages extracted from the reference (the evidence set), shown in a random order, and the reference's URL. It then asks the *collective* stance of the five passages, with similar response options to T1. The designs for both T1 and T2 tasks can be seen in Appendix A.

Recruitment and Quality Assurance The crowdsourcing campaign received ethical approval by King's College London on 7th of April, 2022, with registration number MRA-21/22-29584. All tasks were carried through Amazon Mechanical Turk (AMT). A pilot was run to collect feedback on instructions, design, and compensation. After proper adjustments, the main data annotation tasks were carried out. Several quality control techniques [10] were applied, following similar tasks by Amaral et al. [4, 3]. A number of subtasks in both T1 and T2 were manually created and annotated by the authors and used as golden-standard subtasks to reject low-quality workers. A randomised attention test was put at the start of each task to discourage spammers. Finally, detailed instructions and examples were available at all times to workers.

Execution times for each task in the pilot were measured and used to define payment in USD proportionally to double the US minimum hourly wage (7.25): USD 0.50 for tasks in T1 and USD 1.00 for tasks in T2, calculated based on the highest between mean and median execution time. 500 tasks were generated for T1 and 91 tasks for T2, assigned to about 200 and 140 unique workers, respectively. Workers needed to have finished at least 1000 tasks in AMT with at least 80% approval. Each task was resolved 5 times and annotation aggregation was done via majority voting to reduce worker bias. In the event of a tie (4% of cases for T1 and 13.4% for T2), authors served as tie-breakers.

To assure annotation aggregation was trustworthy, inter-annotator agreement was measured through kappa values, achieving 0.56 for tasks in T1 and 0.33 for tasks in T2. According to Landis and Koch [24], these results show fair and moderate agreement, respectively. Several factors that contribute to lower inter-annotator agreement [6] are present in this crowdsourcing setting: subjectivity inherent to natural language interpretation, a high number of annotators which also lack domain expertise, and class imbalance. On individual annotations for T1, the majority of passages (68.7%) was deemed as neither supporting nor refuting, followed by passages annotated as supporting (27.5%), and a small portion refuting (3.3%). Only 0.4% of annotations were 'not sure'. Aggregated by majority-voting, these values are, respectively, 70.5%, 28.5%, 1.0%, and 0.0%. For T2, the proportions of individual (and aggregated) annotations were 65.5% (73.6%) for supporting sets of evidence, 9.3% (5.9%) for refuting, 24.6% (20.5%) for neither, and 0.6% (0.0%) for 'note sure'.

Gathering Reference-level Annotations

WTR has reference-level annotations for each triple-reference pair. They define a reference's overall stance towards its associated triple, and are manually provided by the authors. These annotations are crucial in order to provide a ground truth for an evaluation of the entire pipeline's performance when taking the whole web page into consideration. They consider a reference's full meaning and context, and not only what was captured and processed by the modules as evidence. Differently from the other annotation level, covered by crowdsourcing tasks that could be simplified to directly comparing extracted text passages, the mental load and task complexity of interacting with the page to inspect all information (e.g. in text, infoboxes, images, charts), on top of cases where the information is nonexistent, is too high for cost-effective crowdsourcing. Thankfully, with one annotation per triple-reference pair, it is feasible for manual annotations to be created by the authors. The authors have thus annotated the over 400 references into different categories and sub-categories, which are a more detailed version of the three stance classes used at evidence-level annotations (and by FEVER):

1. Supporting References (directly maps to the '**supports**' class):
   (a) Support explicitly stated as text as natural language sentences
   (b) Support explicitly stated as text, but not as natural language sentences
   (c) Support explicitly stated, but not as text
   (d) Support implicitly stated
2. Non-supporting References
   (a) Reference refutes claim (directly maps to the '**refutes**' class)
   (b) Reference neither supports nor refutes the claim (directly maps to the '**not enough information**' class)

These six subclasses allow WTR to aid in evaluating the overall performance of ProVe in both ternary (the three sentence-level stances) and binary (supporting vs. not supporting) classifications, as well as to investigate which presentations of supporting information are better captured by the pipeline.

WTR contains 416 Wikidata triple-reference pairs, representing 32 groups of text-rich web domains commonly used as sources, as well as 76 distinct Wikidata properties. 43% of references were obtained through external IDs and 57% through direct URLs. Its structured is shown in Appendix B.

5 Evaluation

This section covers the evaluation of ProVe's performance by applying it to the evaluation dataset WTR, described in Section 4, and comparing ProVe's outputs with WTR's annotations. These inspections and comparisons provide insights into the pipelines' execution and results at its different stages and modules. Each module in ProVe is covered in a following subsection. The overall classification performance of ProVe is indicated by the outputs of the claim verification module's aggregation step and is covered at the end of the section.

5.1 Claim Verbalisation

Given that ProVe's verbalisation module is the exact same model used to create the Wikidata triple verbalisations found in the WDV dataset [4], this section first reports the relevant evaluation results obtained by WDV's authors. It then analyses the claim verbalisation module's execution on the WTR evaluation dataset, looking at the quality of its outputs.

Model Validation

ProVe's verbalisation module consists of a pre-trained T5 model [35] fine-tuned on the WebNLG dataset [13]. To confirm that fine-tuning was properly carried, the authors measure the BLEU [32] scores of its verbalisations on WebNLG data. They measure 65.51, 51.71, and 59.41 on the testing portion of the *SEEN* partition, on the *UNSEEN* partition, and on their combination, respectively. These are all within a percentage point from current state-of-the-art [37].

Amaral et al. [4] use this exact same fine-tuned model to create multiple Wikidata triple verbalisations, which compose the WDV dataset, and evaluate them with human annotators. WDV consists of a large set of Wikidata triples, alongside their verbalisations, whose subject entities come from three distinct groups (partitions) of Wikidata entity classes. The first partition consists of 10 classes that thematically map to the 10 categories in WebNLG's *SEEN* partition. The second, of 5 classes that map to the 5 categories in WebNLG's *UNSEEN* partition. The third consists of 5 new classes not covered in WebNLG but populous in Wikidata.

WDV's verbalisations were evaluated by Amaral et al. through aggregated crowdsourced human annotations of fluency and adequacy dimensions, as defined in Section 3.2. Fluency scores range from 0, i.e. very poor grammar and unintelligible sentences, to 5, i.e. perfect fluency and natural text. Adequacy scores consisted of *0/No* for inadequate verbalisations and *1/Yes* for adequate ones. WDV's authors observed 96% of annotated verbalisations having a median fluency score of 3 or higher, where 3 denotes "Comprehensible text with minor grammatical errors", and around 93% being voted by the majority of annotators as adequate. These results did not vary considerably between WDV partitions, indicating model stability regardless if classes are seen in training or have mappings to WebNLG (DBpedia) classes. The WDV paper [4] contains a more detailed evaluation.

Execution on WTR

The verbalisation module was applied to all 416 triple-reference pairs in WTR. For each triple t, its three components were retrieved from Wikidata: subject (s), predicate (p), and object (o). Subjects and predicates are all Wikidata entities and, thus, have associated main labels and aliases. Objects have multiple possible data types, including Wikidata entity, from which labels were retrieved like with subjects. Strings and quantities without units are used as-is as single labels, otherwise the unit's main label and aliases are added. Date-time values are formatted into multiple string patterns based on their granularity.

The process of defining preferred labels for verbalisation, represented in Section 3.2 through function $l(\cdot)$, consisted of using the main labels of all three components for verbalisation, and changing the predicate's label for an alias in case the verbalisation (v) was not fluent or adequate. First, for a triple t, a concatenation of its components' main labels is fed to the verbalisation model to generate a verbalisation v. Then, these verbalisations were manually inspected by the authors in order to assess fluency and adequacy as previously defined. In case a verbalisation v scored lower than a 4 on fluency or was inadequate, it was replaced by an alternate verbalisation v' generated by using a predicate alias rather than the predicate's main label. As mentioned in Section 3.5, contextually-dependant and vague predicate main labels hinder verbalisations and choosing proper aliases for it can be quickly carried out by KG editors and curators. One example is the predicate `child (P40)`, whose alias 'has child' is used to remedy the main label's lack of explicit direction.

Out of the 416 verbalisations generated by ProVe through main labels, 62.6% were adequate and had good to excellent fluency. Another 29.8% followed suit after predicate alias replacement, totalling 92.4%. The remaining 7.6% either had no aliases or could not be improved by them and had to be manually corrected before being passed down the pipeline. While manual corrections are more demanding than alias selection, those were not frequent and mainly affected specific properties such as `isomeric SMILES (P2017)` and `designation (P1435)`. Corrections were also not extensive; the normalised Levenshtein distance before and after corrections was under 0.25 (a quarter total length) for 80% of them.

Finally, 7 of the claims verbalised ended up having both identical URLs and verbalisations, and were thus dropped from the evaluation downstream as they would have the exact same results. This results from ProVe not taking claim qualifiers into consideration, which is further discussed in Section 6.3.

5.2 Text Extraction

Due to the complexity of defining metrics that measure success in text extraction, this section instead first defines metrics that can be used for an indirect evaluation of the text extraction module. It then explores insightful descriptive metrics obtained from executing the module on WTR.

Indirect Evaluation

ProVe's text extraction module essentially performs a full segmentation of the referenced web-page's textual content without excluding any text. The model can not be directly evaluated through annotations due to the sheer quantity of ways in which one can segment all references contained in the evaluation dataset. Annotating such text extractions would require manually analysing entire web pages to find all textual content relevant to the claim, inspecting all the text extractable from such page, and segmenting it such that boundaries are properly placed in terms of syntax and that relevant passages are kept unbroken. One would then need to compare one or multiple ideal extractions to the extraction performed by ProVe. It is neither trivial to simplify this process for crowdworkers, nor efficient for the authors to carry it out by hand. It is also not trivial to define what constitute 'well-placed' sentence boundaries nor how and if one can break relevant passages of text.

Thus, instead of a direct evaluation, the performances of the subsequent sentence selection and final aggregation steps are used as indirect indicators of ProVe's text extraction. A correlation between ProVe's relevance scores (ρ) and evidence-level crowd annotations, as defined in Section 5.3, can be used to measure how much ProVe captures human-perceived relevance. A high value of such metric indicates ProVe extracts text passages such that relevant and irrelevant text segments are well divided. Otherwise, there would be a dissonance between humans and models in rating relevance, as while humans are good in judging badly divided passages, models would struggle significantly.

Likewise, a good final classification performance, measured against WTR's reference-level annotations, indicates ProVe's capacity of extracting useful sentences. Classification metrics for ternary and binary classification tasks, such as accuracy and f-scores, are shown in Section 5.4. Still, the direct evaluation of ProVe's text extraction module, encompassing sentence segmentation and meaning extraction from unstructured and semi-structured textual content, is intended as future work.

Execution on WTR

The text extraction module was applied to each of the 416 triple-reference pairs (t, r) in WTR, each yielding a set of passages P, as defined in Section 3.3. The total number of extracted passages was of nearly 64K, an average of 154 passages per reference. This average number of extracted passages varied heavily according to web domain, ranging from as low as 1 to as high as 804, with relatively low inter-domain standard deviation (29.51 median standard deviation). The average size, in characters, of individual extracted passages behaved similarly, ranging from 24.20 to 4059.23 depending on web domain. Finally, the number of passages $|P|$ extracted from a reference r and their average size $(\sum_{i=0}^{|P|-1} |\mathtt{p_i}|)/|P|$ have a slightly moderate negative correlation (-0.2513 Pearson's r), with a few domains, such as bioguides.congress.gov returning very few but very large passages.

These metrics confirm that extracted textual content mainly vary based on web domain. It indicates ProVe's extraction depends heavily on particular web layouts, e.g. having difficulty segmenting the contents of specific domains like bioguides.congress.gov, due to their textual content being contained in a single paragraph (<p> HTML tag) without periods or any other sentence breaks.

5.3 Sentence Selection

ProVe's sentence selection module contains a BERT model fine-tuned on FEVER's training partition, as described in Section 3.4. This section first performs a sanity check of by evaluating the model on FEVER's validation and testing partitions, measuring standard classification metrics to ensure the model has properly fine-tuned to FEVER. Afterwards, the entire module is applied to WTR and its performance measured by relying on the crowdsourced annotations.

Model Validation

For each claim-sentence pair in FEVER's validation and testing partitions, the sentence selection module's BERT model outputs a relevance score between -1 and 1. A sentence is classified as relevant to a claim if it figures within the top 5 highest scoring sentences for that claim, and as irrelevant otherwise. According to FEVER Scorer [4], the

[4] https://github.com/sheffieldnlp/fever-scorer

model scores 0.945 and 0.87 recall on validation and testing, respectively. Similarly, F1-scores were 0.421 and 0.386, which are less than a percentage point away from KGAT [28] and DREAM [58], current state-of-the-art, on the test partition.[5] These indicate the model has properly fine-tuned to FEVER.

Execution on WTR

Inputs to the sentence selection module consist of the verbalisations v outputted by the claim verbalisation module, as described in Section 5.1, and the extracted passages P taken from the reference, as described in Section 5.2. For each passage p $\in P$, a relevance score $\rho \in [-1, 1]$ is calculated.

Distribution of Relevance Score Relevance scores (ρ) varied heavily across web domains. For each triple-reference pair (t, r) in WRT, its passages P were scored by the sentence selector module, and the top 5 highest scores (which correspond to the evidence set E) were averaged, with Figure 5 showing the distribution of these averages across and within domains. Overall, relevance scores spanned the whole range of values (-1 to 1), with a median close to zero. Variation within web domains was not wide, denoted by a prevalence of small to medium interquartile ranges. Web domains with large variations were those that cover a large range of information domains and page layouts, such as bbc.co.uk. In contrast, there is extensive variation across web domains. This is due, firstly, to how each domain conveys its information, where long and continuous textual contents greatly favour reliable scores, as opposed to spread-out information which pushes scores down. Secondly, due to the amount of content actually related to the triple. A website like thepeerage.com mentions the triples's subject many times, leading to many positively-scored sentences, as opposed to vesseltracking.net, which provides fewer, shorter, and more direct information.

Proportion of Irrelevant Passages Given the distributions seen in Figure 5, one can define the value zero as the threshold between likely relevant and likely irrelevant passages. By using only the passages extracted by the $n = 1$ sliding window (P_1), this threshold leaves 24.7% of triple-reference pairs as containing only passages that are likely irrelevant. This decreases to 17.6% when combining the $n = 1$ and $n = 2$ sliding windows, clearly indicating the method described in Section 3.3 achieves its desired objective of generating more relevant passages by combining multiple text segments. This percentage of triple-reference pairs without likely relevant passages also varies heavily across web domains, and is a problem mostly affecting those same domains at the lower end of the distribution seen in Figure 5.

Correlation Between Modeled and Crowdsourced Relevance For each triple-reference pair (t, r) in WTR, crowd-workers annotated each of its 5 most relevant passages (its Evidence set E) as ranked by the sentence selection module, described in Section 4.2 as the evidence-level annotations obtained through task design T1. Pieces of evidence were annotated according to their individual stance towards the reference's associated triple, using one of four choices: 'supports', 'refutes', 'neither' (not enough information), and 'not sure'. In order to reduce crowd bias, multiple individual annotations were collected per evidence and aggregated through majority voting. Authors served as tie-breakers.

In order to compare the model's relevance scores (ρ) with human-perceived values of relevance (T1 crowd annotations), both 'supports' and 'refutes' annotations are grouped as 'relevant', with 'neither' relabelled as 'irrelevant'. Relevance score distributions are then analysed based on this binary class annotation. Figure 6, shows annotated passages divided into groups based on the percentage of 'relevant' individual annotations (or votes) each has received, as well as each group's relevance score distribution. There is a clear pattern of association between the model's relevance scores and real humans' opinions of relevance. This conclusion is reinforced by the strong correlation between relevance scores and percentages of individual annotations as 'relevant' (0.5058 Pearson's r).

Such strong correlation can also be seen with aggregated annotations, as shown in Figure 7, where the relevance score distributions for the group of passages majority-voted as 'relevant' vs. those voted as 'irrelevant'. These metrics and distributions indicate that ProVe's sentence selection module produces scores that are well-related to human judgements of relevance.

5.4 Claim Verification

The first step of ProVe's claim verification module consists of a TER classification task, resolved by a fine-tuned BERT model, as described in Section 3.5. Such TER model is used to classify the stances of individual pieces of evidence ($e \in E$) towards a KG triple (t) by calculating three class probabilities (σ) corresponding to the three TER classes found in FEVER: 'supports', 'refutes', and 'not enough information'. The module's second step is an aggregation which uses these classification probabilities to calculate a final verdict for the whole reference (r). This section first

[5] https://competitions.codalab.org/competitions/18814#results

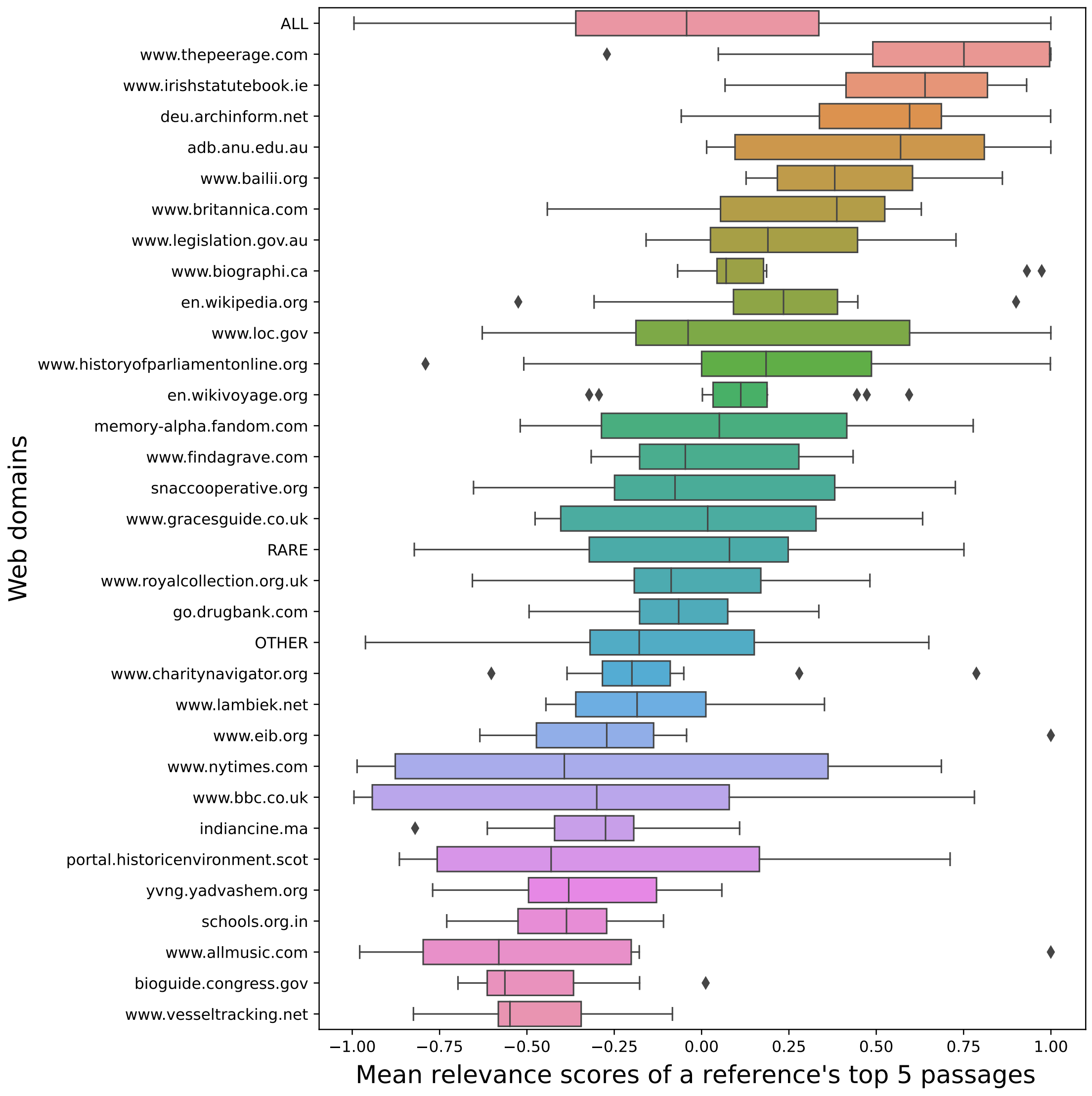


Fig. 5. Relevance scores distributions across and within different web domains. 'ALL' stands for the combined distribution of the subsequent 32 groups. Data values here are the averages of the top 5 passages' relevance scores for each reference.

describes a sanity-check of both steps by evaluating them in conjunction on FEVER's validation and testing partitions. It then assesses their performance on WTR.

Model Validation

ProVe's claim verbalisation module has been applied to FEVER's validation and testing partitions. For the validation partition, the sentences pre-annotated as relevant to the claim being judged were used as evidence. The first TER step is performed to calculate the individual stance probabilities of each piece of evidence. The second aggregation step is then carried out to define the claim's final verdict. The same process is carried for the testing partition, however, since its sentences do not come pre-annotated, ProVe's sentence selection module was used instead.

For the final aggregation step, only methods #1 (a weighted sum) and #2 (Malon's strategy) were used, as neither require training a new classifier and thus have similar complexity to other approaches used to tackle FEVER [29, 43], yielding a more direct comparison. Label accuracy and FEVER score were calculated as evaluation metrics. Label accuracy is a normal classification accuracy calculated over the three TER classes. FEVER score is an accuracy

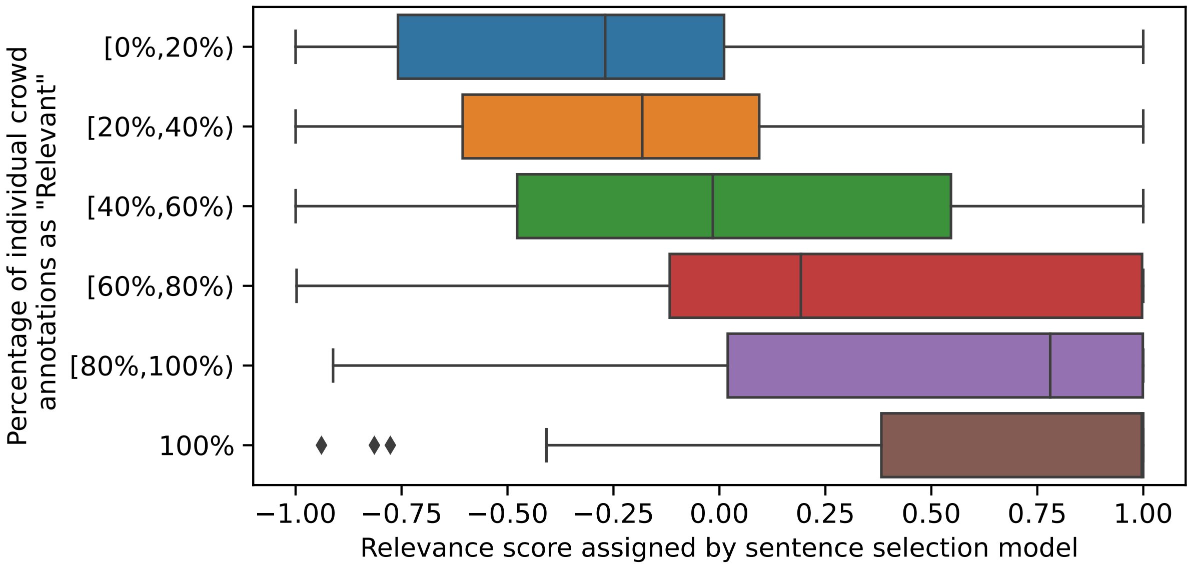


Fig. 6. Distributions of relevance scores given by the module divided by the percentage of crowd annotations deeming that passage as 'relevant' (either 'supports' or 'refutes'.

calculated by also taking collected evidence into consideration: a prediction is correct if the label is correct **and** the predicted evidence set contains all correct evidence.

Aggregation method $\#1$, the weighted sum, scores 0.6964 label accuracy and 0.6952 FEVER score on the validation set, and 0.6508 and 0.617 respectively on the testing set. Method $\#2$, the rule-based aggregation, scores 0.7624 label accuracy and 0.76110 FEVER score on the validation set, and 0.7037 and 0.6739 respectively on the testing set. Method $\#2$ puts ProVe 3.2 percentage points below state-of-the-art (DREAM [58]) on FEVER score, which the authors consider sufficient to validate the module's fine-tuning.

Execution on WTR

ProVe's claim verification module is tested on WTR by comparing its outputs to WTR's annotations, both at evidence level and reference level. Such annotations, as described in Section 4.2, consist of: crowd annotations denoting the stances of individual pieces of evidence towards KG triples (from crowdsourcing tasks T1), crowd annotations denoting the collective stances of sets of evidence towards KG triples (from crowdsourcing tasks T2), and author annotations denoting the stance of the entire reference towards its associated KG triple. Crowdsourced annotations are collected multiple times and aggregated through majority voting, with authors serving as tie-breakers.

Distributions of TER Class Probabilities Similarly to the relevance scores outputted by the sentence selection module, the TER class probabilities outputted by the TER model for individual pieces of evidence varied greatly across web domains. Consider a triple-reference pair (t, r) and its i-th piece of evidence e_i, whose stance probabilities assigned by the model are $(\sigma_i^{SUPP}, \sigma_i^{REF}, \sigma_i^{NEI})$. Consider also the highest of the three probabilities to denote the predicted TER class for that individual evidence z_i, where $z_i = \text{argmax}_{k \in \mathcal{K}}\, \sigma_i^k$. Web domains such as indiancine.ma have close to 95% of its evidence classified as 'not enough information', while domain such as deu.archinform.net classify close to 98% as 'supports'. Likewise to sentence selection, such variations can be attributed to web layouts. For instance, infoboxes, implicit subjects, and large amounts of boilerplate clearly inflate the number of unrelated sentences that might get highly ranked for relevance, but are neither supportive nor refutative.

Individual TER Classification Metrics Using this argmax approach to define the predicted stance z_i of an individual piece of evidence e_i, and using the aggregated annotations provided by the crowd through T1 tasks as ground truth, once can measure classification metrics for ProVe's individual TER classification. Figure 8 shows the resulting confusion matrix. Accuracy was 0.56 and Macro F1-score was 0.43. While predicting supporting sentences with moderate precision, ProVe's TER model has difficulty in disentangling refuting sentences from those that neither support nor refute. This is believed to be due to the differences between refuting evidence data found in FEVER and refuting references that occur naturally in Wikidata's sources, discussed in more detail in Section 6.4.

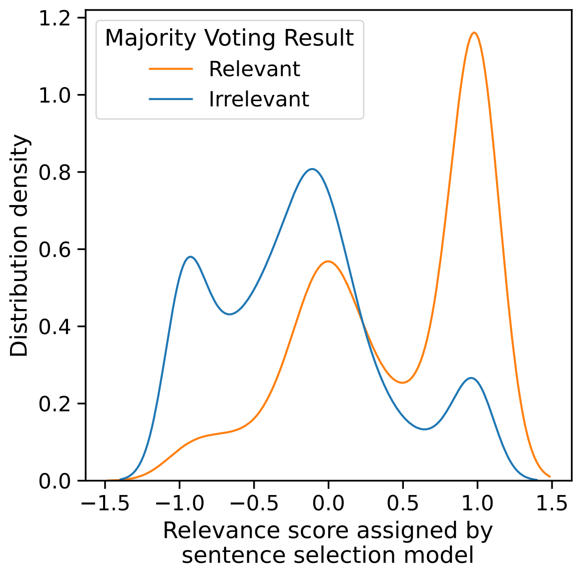


Fig. 7. Relevance score distributions of passages majority-voted as 'relevant' and of those voted 'irrelevant'.

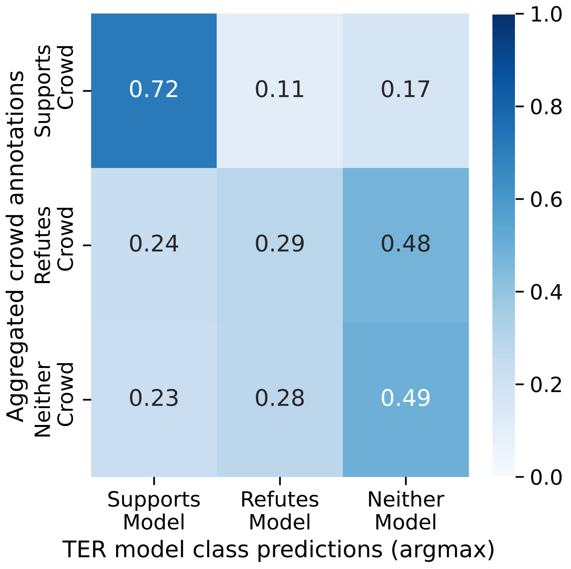


Fig. 8. Stance classes predicted for single claim-reference pairs (obtained through argmax) vs. the crowd's aggregated annotations.

ProVe's pipeline focuses on the classification of entire references rather than individual sentences, and the TER class probabilities are merely features for the final aggregation. Still, TER classifications for individual pieces of evidence can be greatly improved by, rather than argmax, using a simple classifier with the three TER class probabilities (σ_i), the evidence's relevance score (ρ_i), and the evidence's length ($|e_i|$) as features. The best scoring classifier was a Random Forests Classifier (RFC) with cross-validated (k=5) scores of 0.77 accuracy and 0.50 F1-score. Furthermore, by grouping the 'refutes' and 'neither' (NEI) classes into one 'not supporting' class, turning this into a binary classification task, cross-validated (k=5) scores reach 0.79 accuracy and 0.76 F1-score, as well as an Area Under the ROC Curve (AUC) of 0.85, as seen in Figures 9 and 10.

Collective TER Classification Metrics The annotations obtained through T2 tasks describe human judgements on the collective stance of the sets of evidence extracted from a reference towards its associated triple in WTR. After majority-voting aggregation, there are 301 evidence sets collectively supporting the triple, 24 refuting, and 84 that neither support nor refute it. Taking these annotations as ground truth labels, one can measure the classification performance of ProVe. Table 2 showcases and compares these results with different aggregation methods. Due to class imbalance, both macro averaged and weighted averaged results are reported. Not only the original ternary classification problem but also to a simplified 'supporting' vs. 'not supporting' binary classification problem is explored. Although a binary formulation is easier to solve, having a pipeline that provides assistance on differentiating between supporting and non-supporting references is of huge benefit to any KG's curation and editing. Results show how a simple classifier considerably outperforms the other two aggregation approaches, especially in the binary classification scenario.

Reference Representation through Retrieved Evidence Verifying whether ProVe properly represents entire references through the evidence set it retrieves from them is possible by comparing the evidence-level crowd annotations on the collective stance of retrieved evidence (T2 tasks) with the reference-level author annotations, which represent the stance of the reference as a whole.

As detailed in Section 4.2, reference-level annotations consist of six categories which directly map to the three TER labels used in crowd annotations. Figure 11 shows WTR's reference-level annotation distribution. On the ternary classification task, labels $1.A.$ through $1.D.$ map to 'supporting', label $2.A.$ maps to 'refuting', and label $2.B.$ to 'neither'. As for the binary classification task, labels $1.X.$ map to 'supporting' and labels $2.X.$ map to 'not supporting'.

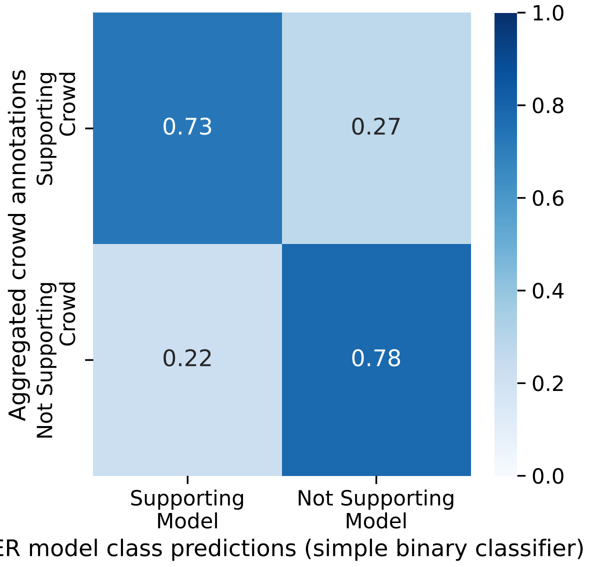


Fig. 9. Binary stance classes of single claim-reference pairs predicted by a RFC vs. the crowd's aggregated annotations.

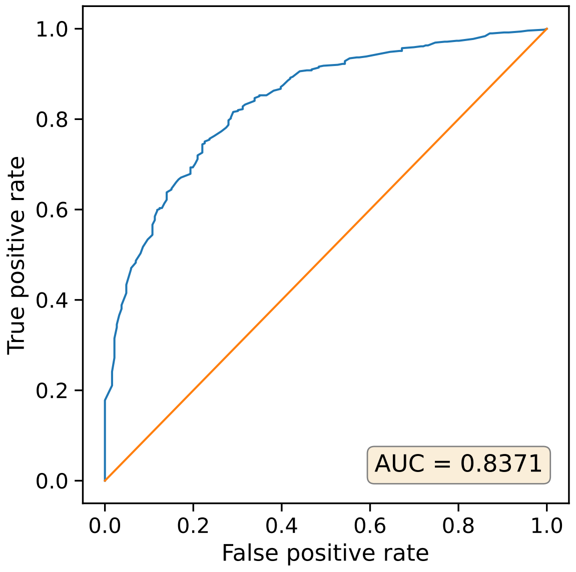


Fig. 10. ROC curve for the simplified binary stance classification performed by a RFC.

			Macro Averaged			Weighted Averaged		
Method	Classes	Accuracy	Precision	Recall	F1	Precision	Recall	F1
1: Weighted Sum	3	0.592	**0.439**	**0.484**	0.439	0.700	0.592	0.626
2: Malon's	3	0.641	0.430	0.456	0.436	0.691	0.641	0.659
3: RFC	3	**0.726**	0.433	0.468	**0.446**	**0.709**	**0.726**	**0.714**
1: Weighted Sum	2	0.626	0.609	0.639	0.596	0.716	0.626	0.648
2: Malon's	2	0.667	0.614	0.638	0.617	0.712	0.667	0.683
3: RFC	2	**0.750**	**0.681**	**0.664**	**0.667**	**0.747**	**0.750**	**0.745**

Table 2. Classification performance of each of the three aggregation methods on both ternary and binary collective stance TER classification formulations. Majority-voted annotations obtained in T2 are used as true labels. Results from method 3 were cross-validated with $k = 5$.

There is a high class imbalance, with authors deeming only 2 references as 'refuting'. This number is much lower than the 24 majority-voted by the crowd as 'refuting'.

Figure 12 shows a complete comparison between evidence-level collective stance annotations from the crowd and reference-level author labels. Crowd annotators very successfully judge references in $1.A.$ (explicit textual support in natural language), and obtain moderate to high performance on all other groups, except for $1.C.$ (non-textual support) and $2.A.$ (refuting reference). It is trivial to see why ProVe, a text-based approach, fails at $1.C.$ The failure at $2.A.$ is partially due to the disparity between refuting text seen in training and in the evaluation dataset, as explained in Section 6.4. The low amount of refuting references compromises a deeper look. Still, where supportive textual information is the most available, explicit, and naturally written, the better ProVe performs.

Full Pipeline Performance Lastly, ProVe's performance on a 'supported' vs. 'not supported' binary classification scenario is investigated by comparing the outputs obtained with the WTR evaluation dataset to its reference-level annotations provided by the authors. The variation of such performance based on how supportive information is expressed on text is also analysed.

ProVe is evaluated on the entire WTR by using the reference-level annotations as ground-truth and adopting ProVe's best performing aggregation method, the simple classifier (method #3). Additionally, for each of the supporting reference-level labels $1.A.$ through $1.D.$ (except for $1.C$, as it is non-textual), WTR is modified by keeping only those triple-reference pairs with either that specific supporting label or labelled as 'not supporting'. ProVe is then also evaluated on these modified datasets. Due to the very low amount of $2.A.$ labels, only the binary 'supporting' vs. 'not supporting' classification task is evaluated. Table 3 showcases the results obtained. It shows that ProVe has a good re-

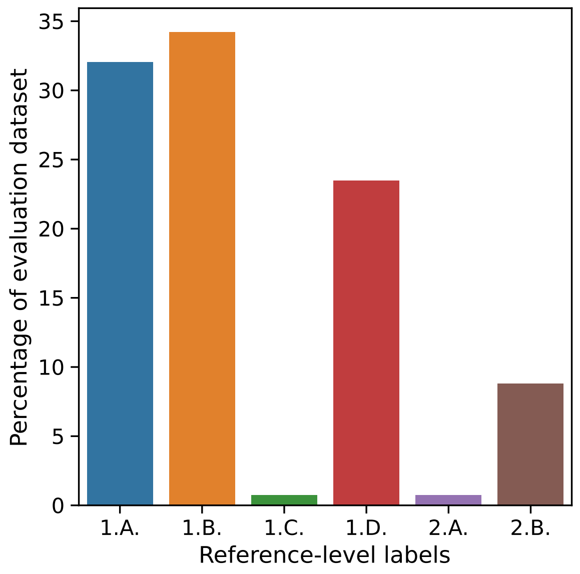


Fig. 11. Distribution of reference-level author labels of the evaluation dataset.

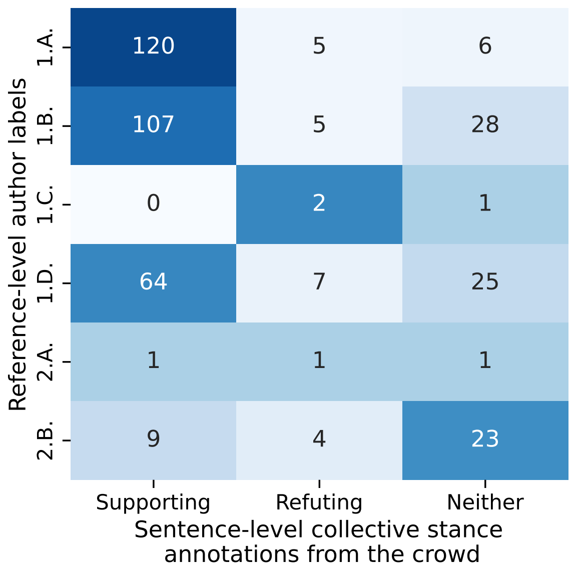


Fig. 12. Comparison between reference-level author label and sentence-level collective stance annotations from the crowd.

sult on the evaluation dataset overall, with close to 80% accuracy, and an excellent performance on identifying support from references that showcase said support through explicit and naturally written textual information ($1.A.$). It also shows good results on references where support is not naturally written ($1.B.$).

		Macro Averaged			Weighted Averaged			
Support type	Accuracy	Precision	Recall	F1-Score	Precision	Recall	F1-Score	AUC
1.A.	**0.875**	**0.821**	**0.838**	**0.829**	**0.878**	**0.875**	**0.876**	**0.908**
1.B.	0.779	0.668	0.637	0.648	0.762	0.779	0.768	0.745
1.D.	0.666	0.662	0.669	0.649	0.749	0.666	0.682	0.754
ALL	0.794	0.565	0.635	0.574	0.864	0.794	0.821	0.753

Table 3. ProVe's binary classification performance on all WTR and per type of textual support. Reference-level annotations were used as ground-truth, and a simple classifier as aggregation method. Values obtained through cross-validation ($k = 5$).

6 Discussions and Conclusions

In this section, aspects and limitations of the implementation and evaluation results of ProVe are further discussed. Additionally, future directions of research are pointed out and final conclusions are drawn.

6.1 ProVe for Fact Verification

Fact checking as a tool to assist users in discerning between factual and non-factual information has a myriad of applications, formulations, approaches, and, overall, considerably ambiguous results. The effects of fact-checking interventions, while significant, are substantially weakened by its targets' preexisting beliefs and knowledge [54]. Its effectiveness depend heavily on many variables, such as the type of scale used and whether facts can be partially checked, as well as different impacts if they go along or against a person's ideology. This further motivates ProVe's standpoint of judging support instead of veracity. Triples are evaluated not as factual or non-factual, but based on their documented provenance, passing the onus of providing trustworthy and authoritative sources to the graph's curators. This keeps ProVe's judgements from clashing with the ideologies of its users, as the pipeline does not pass factual, but linguistic judgement. Additionally, by using only two to three levels of verdict and not including graphical elements, the presence of elements that compromise fact-checking [54] is hampered. The authors' focus for future research lies in increasing ProVe's explainability in order to increase trust and understanding.

The results achieved by ProVe, especially on text-rich references, are considered by the authors as more than satisfactory, representing an excellent addition to a family of approaches that currently includes very few, e.g. DeFacto [14] and FactCheck [45]. Still, there is a need for a dataset specialised in AFC on KGs in order to tie in these approaches and make benchmarking them and future works possible. While WTR (ProVe's evaluation dataset) can serve this purpose, it can definitely be improved in size and predicate coverage.

ProVe's use as a tool can greatly benefit from an active learning scenario which would further enhance the models and techniques it employs. As the same time, users of the tool inherently introduce a bias based on their demographics, with the same being valid for the crowdsourced evaluation of ProVe's pipeline. Being aware of this bias is crucial to the proper deployment of such approaches.

6.2 Text Extraction Add-ons

ProVe's text extraction module has three essential steps: rendering web pages with a web crawler, using rule-based methods to convert content inside HTML tags into text, and sentence segmentation. While the rule-based methods presented in this paper are simple, they are quite effective, as shown in Section 5. Better and more specialized rules and methods to detect and extract text from specific HTML layouts, such as turning tabular structures or sparse infoboxes into sequential and syntactically correct sentences, can be seamlessly integrated into ProVe. Both supervised and unsupervised approaches can also be applied.

In order to properly assess such added methods, as well as to provide more insight into the text extraction module in general, a direct evaluation of its performance would be extremely helpful. Although good performance on downstream tasks is a good indicator, it does not indicate where to improve text extraction. Be it through descriptive statistics or comparison against golden data, this as a focus of future research alongside model explainability

6.3 Usage of Qualifiers

Triples in KGs such as Wikidata often are accompanied by qualifiers that further detail it. A triple such as the one seen in Figure 2 (*<Librarian of Congress, position holder, James H. Billington>*) has several qualifiers, such as 'start time' and 'end time'. If a person were Librarian of Congress twice, this qualifier would differentiate two triples that would otherwise be identical if expressed only with main components. ProVe does not take qualifiers into consideration. As such, two triples with distinct IDs and meanings can have exactly the same verbalisations which, while adequate, do not contain all information.

Ribeiro et al. [37] show transformers can verbalise multiple triples into a single sentence. Hence, adding qualifiers as secondary triples is possible, generating more detailed verbalisations. However, ProVe's sentence selection and claim verification modules contain models fine-tuned on FEVER, whose vast majority of sentences contain only a main piece of information with little to no additional details, e.g. 'Adrienne Bailon is an accountant' and 'The Levant was ruled by the House of Lusignan'. In order to make proper use of qualifiers during verbalisation, there needs to be an assurance that downstream modules can properly handle more complex sentences by either different or augmented training data.

6.4 Detecting Refuting Sources with FEVER

The FEVER dataset presents claims that are normally short and direct in nature, from multiple domains, and associated evidence extracted directly from Wikipedia. ProVe shows it is possible to use FEVER to train pipeline modules to detect supportive and non-supportive sources evidence. However, as seen in Section 5.4, detecting refuting sources is hard for ProVe and believed to be due to how FEVER generates refuted claims through artificial alterations. Claims labelled by FEVER as 'REFUTES' are those generated by annotators who alter claims that would otherwise be supported by its associated evidence. Alterations follow six types: paraphrasing, negation, entity/relationship substitution, and making the claim more general/specific. This leads to claims that, while meaningful and properly annotated, would never be encoded in a KG triple, such as "As the Vietnam War raged in 1969, Yoko Ono and her husband John Lennon **did not** have two week-long Bed-Ins for Peace" or "Ruth Negga **only** acts in Irish cinema". Additionally, associated evidence often rely on common sense in order to refute these claims, such as "Kingdom Hearts III is owned by Boyz II Men", whose relevant evidence at no point elaborate on Kingdom Hearts III's ownership, only describing it as a Japanese video-game. We supposedly know Boyz II Men is a music group, rendering the claim implausible.

While useful for other tasks, these refuted claims are very different from refutable triples occurring naturally in KGs, which mainly consist of triples whose objects have different values in the provenance. One such example is "Robert

Brunton was born on 23/03/1796", whose reference actually mentions the "10th of February 1796". In order to properly detect KG provenance that refute their triples, ProVe's claim verification module needs re-training on a fitting subset of FEVER, or on a new dataset containing non-artificial refuted claims.

6.5 Conclusions

Knowledge graphs are widespread secondary sources of information. Their data is extremely useful and available in a semantic format that covers a myriad of domains. Ensuring the verifiability of this data through documented provenance is a task that is crucial to the upkeep of their usability and one that should be actively supported by automated and semi-automated tools to help data curators and editors cope with the sheer volume of information. However, as of now, there are no such tools deployed at large scale KGs and only a very small family of approaches tackle this task from a research standpoint.

This paper proposes, describes, and evaluates ProVe, a pipelined approach to support the upkeep of KG triple verifiability through their documented provenance. ProVe leverages large pre-trained LMs, rule-based methods, and simple classifiers, to provide automated assistance to the activity of creating and maintaining references in a KG. ProVe's pipeline aims at extracting relevant textual information from references and evaluating whether or not they support an associated KG triple, providing its users with a support classification, a support probability, as well as relevance and textual entailment metrics for the evidence used. Deployed correctly, ProVe can help detect verifiability issues in existing references, as well as improve the reuse of good sources. Additionally, the approach can be expanded to work in a multilingual setting.

ProVe has been evaluated with WTR, a dataset of triple-reference pairs extracted directly from Wikidata, a large KG, and annotated by both crowdworkers and the authors. ProVe achieves 75% accuracy, 0.681 F1-macro, and 0.667 AUC on the full evaluation dataset, which includes references to many different web domains. On references where support is stated explicitly and in natural text, ProVe achieves an excellent 87.5% accuracy (0.829 F1-macro and 0.908 AUC).

Future work mainly lies in exploring techniques to improve ProVe's explainability, with a focus on its sentence selection and claim verification steps. Other directions can include expanding the size and the distinct predicate coverage of the benchmarking dataset WTR, as well as a direct evaluation of text extraction and segmentation techniques.

Acknowledgements This research received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 812997.

References

1. Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Auer, S., Lehmann, J.: Crowdsourcing linked data quality assessment. In: International semantic web conference. pp. 260–276. Springer (2013)
2. Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Flöck, F., Lehmann, J.: Detecting linked data quality issues via crowdsourcing: A dbpedia study. Semantic web **9**(3), 303–335 (2018)
3. Amaral, G., Piscopo, A., Kaffee, L.A., Rodrigues, O., Simperl, E.: Assessing the quality of sources in wikidata across languages: a hybrid approach. Journal of Data and Information Quality (JDIQ) **13**(4), 1–35 (2021)
4. Amaral, G., Rodrigues, O., Simperl, E.: WDV: A broad data verbalisation dataset built from wikidata. arXiv preprint arXiv:2205.02627 (2022)
5. Ammar, A., Celebi, R.: Fact validation with knowledge graph embeddings. In: ISWC (Satellites). pp. 125–128 (2019)
6. Bayerl, P.S., Paul, K.I.: What determines inter-coder agreement in manual annotations? a meta-analytic investigation. Computational Linguistics **37**(4), 699–725 (2011)
7. Braşoveanu, A.M., Andonie, R.: Visualizing and explaining language models. In: Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery, pp. 213–237. Springer (2022)
8. Cao, M., Zhang, J., Xu, S., Ying, Z.: Knowledge graphs meet crowdsourcing: A brief survey. In: International Conference on Cloud Computing. pp. 3–17. Springer (2020)
9. 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)
10. Daniel, F., Kucherbaev, P., Cappiello, C., Benatallah, B., Allahbakhsh, M.: Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. ACM Computing Surveys (CSUR) **51**(1), 1–40 (2018)
11. Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. Semantic Web **9**(1), 77–129 (2018)
12. Flouris, G., Roussakis, Y., Poveda-Villalon, M., Mendes, P.N., Fundulaki, I.: Using provenance for quality assessment and repair in linked open data (2012)

13. Gardent, C., Shimorina, A., Narayan, S., Perez-Beltrachini, L.: The webnlg challenge: Generating text from rdf data. In: Proceedings of the 10th International Conference on Natural Language Generation. pp. 124–133 (2017)

14. Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngomo, A.C.N., Speck, R.: Defacto—temporal and multilingual deep fact validation. Journal of Web Semantics **35**, 85–101 (2015)

15. Guo, Z., Schlichtkrull, M., Vlachos, A.: A survey on automated fact-checking. Transactions of the Association for Computational Linguistics **10**, 178–206 (2022)

16. Hanselowski, A., Zhang, H., Li, Z., Sorokin, D., Schiller, B., Schulz, C., Gurevych, I.: UKP-athene: Multi-sentence textual entailment for claim verification. In: Proceedings of the First Workshop on Fact Extraction and VERification (FEVER). pp. 103–108. Association for Computational Linguistics, Brussels, Belgium (Nov 2018). https://doi.org/10.18653/v1/W18-5516, https://aclanthology.org/W18-5516

17. Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python (2020). https://doi.org/10.5281/zenodo.1212303

18. Joshi, U., Urbani, J.: Ensemble-based fact classification with knowledge graph embeddings. In: European Semantic Web Conference. pp. 147–164. Springer (2022)

19. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext.zip: Compressing text classification models. arXiv preprint arXiv:1612.03651 (2016)

20. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)

21. Kim, J., Choi, K.S.: Unsupervised fact checking by counter-weighted positive and negative evidential paths in a knowledge graph. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 1677–1686 (2020)

22. Kontokostas, D., Zaveri, A., Auer, S., Lehmann, J.: Triplecheckmate: A tool for crowdsourcing the quality assessment of linked data. In: International Conference on Knowledge Engineering and the Semantic Web. pp. 265–272. Springer (2013)

23. Kumar, P., Singh, A., Kumar, P., Kumar, C.: An explainable machine learning approach for definition extraction. In: International Conference on Machine Learning, Image Processing, Network Security and Data Sciences. pp. 145–155. Springer (2020)

24. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. biometrics pp. 159–174 (1977)

25. Lehmann, J., Gerber, D., Morsey, M., Ngonga Ngomo, A.C.: Defacto-deep fact validation. In: International semantic web conference. pp. 312–327. Springer (2012)

26. Leonhardt, J., Anand, A., Khosla, M.: Boilerplate removal using a neural sequence labeling model. In: Companion Proceedings of the Web Conference 2020. pp. 226–229 (2020)

27. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)

28. Liu, Z., Xiong, C., Sun, M., Liu, Z.: Fine-grained fact verification with kernel graph attention network. arXiv preprint arXiv:1910.09796 (2019)

29. Malon, C.: Team papelo: Transformer networks at fever. arXiv preprint arXiv:1901.02534 (2019)

30. Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the most out of wikidata: semantic technology usage in wikipedia's knowledge graph. In: International Semantic Web Conference. pp. 376–394. Springer (2018)

31. Padia, A., Ferraro, F., Finin, T.: Kgcleaner: Identifying and correcting errors produced by information extraction systems. arXiv preprint arXiv:1808.04816 (2018)

32. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics. pp. 311–318 (2002)

33. Piscopo, A., Kaffee, L.A., Phethean, C., Simperl, E.: Provenance information in a collaborative knowledge graph: an evaluation of wikidata external references. In: International semantic web conference. pp. 542–558. Springer (2017)

34. Piscopo, A., Simperl, E.: What we talk about when we talk about wikidata quality: a literature survey. In: Proceedings of the 15th International Symposium on Open Collaboration. pp. 1–11 (2019)

35. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)

36. Rastas, I., Ryan, Y.C., Tiihonen, I.L.I., Qaraei, M., Repo, L., Babbar, R., Mäkelä, E., Tolonen, M., Ginter, F.: Explainable publication year prediction of eighteenth century texts with the bert model. In: Proceedings of the 3rd Workshop on Computational Approaches to Historical Language Change. The Association for Computational Linguistics (2022)

37. Ribeiro, L.F., Schmitt, M., Schütze, H., Gurevych, I.: Investigating pretrained language models for graph-to-text generation. arXiv preprint arXiv:2007.08426 (2020)

38. Sathe, A., Ather, S., Le, T.M., Perry, N., Park, J.: Automated fact-checking of claims from wikipedia. In: Proceedings of the 12th Language Resources and Evaluation Conference. pp. 6874–6882 (2020)

39. Schuster, T., Fisch, A., Barzilay, R.: Get your vitamin c! robust fact verification with contrastive evidence. arXiv preprint arXiv:2103.08541 (2021)

40. Shenoy, K., Ilievski, F., Garijo, D., Schwabe, D., Szekely, P.: A study of the quality of wikidata. Journal of Web Semantics **72**, 100679 (2022)

41. Shi, B., Weninger, T.: Discriminative predicate path mining for fact checking in knowledge graphs. Knowledge-based systems **104**, 123–133 (2016)

42. Shiralkar, P., Flammini, A., Menczer, F., Ciampaglia, G.L.: Finding streams in knowledge graphs to support fact checking. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 859–864. IEEE (2017)

43. Soleimani, A., Monz, C., Worring, M.: Bert for evidence retrieval and claim verification. In: European Conference on Information Retrieval. pp. 359–366. Springer (2020)

44. Speck, R., Ngomo, A.C.N.: Leopard—a baseline approach to attribute prediction and validation for knowledge graph population. Journal of Web Semantics **55**, 102–107 (2019)

45. Syed, Z.H., Röder, M., Ngonga Ngomo, A.C.: Factcheck: Validating rdf triples using textual evidence. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 1599–1602 (2018)

46. Thorne, J., Vlachos, A.: An extensible framework for verification of numerical claims. In: Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics. pp. 37–40. Association for Computational Linguistics (2017)

47. Thorne, J., Vlachos, A.: Automated fact checking: Task formulations, methods and future directions. arXiv preprint arXiv:1806.07687 (2018)

48. Thorne, J., Vlachos, A., Christodoulopoulos, C., Mittal, A.: Fever: a large-scale dataset for fact extraction and verification. arXiv preprint arXiv:1803.05355 (2018)

49. Thorne, J., Vlachos, A., Cocarascu, O., Christodoulopoulos, C., Mittal, A.: The fact extraction and verification (fever) shared task. arXiv preprint arXiv:1811.10971 (2018)

50. Vlachos, A., Riedel, S.: Identification and verification of simple claims about statistical properties. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 2596–2601. Association for Computational Linguistics (2015)

51. Vo, N., Lee, K.: Where are the facts? searching for fact-checked information to alleviate the spread of fake news. arXiv preprint arXiv:2010.03159 (2020)

52. Vogels, T., Ganea, O.E., Eickhoff, C.: Web2text: Deep structured boilerplate removal. In: European Conference on Information Retrieval. pp. 167–179. Springer (2018)

53. Vrandečić, D.: Wikidata: A new platform for collaborative data collection. WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion p. 1063 (2012). https://doi.org/10.1145/2187980.2188242

54. Walter, N., Cohen, J., Holbert, R.L., Morag, Y.: Fact-checking: A meta-analysis of what works and for whom. Political Communication **37**(3), 350–375 (2020)

55. Xue, B., Zou, L.: Knowledge graph quality management: a comprehensive survey. IEEE Transactions on Knowledge and Data Engineering (2022)

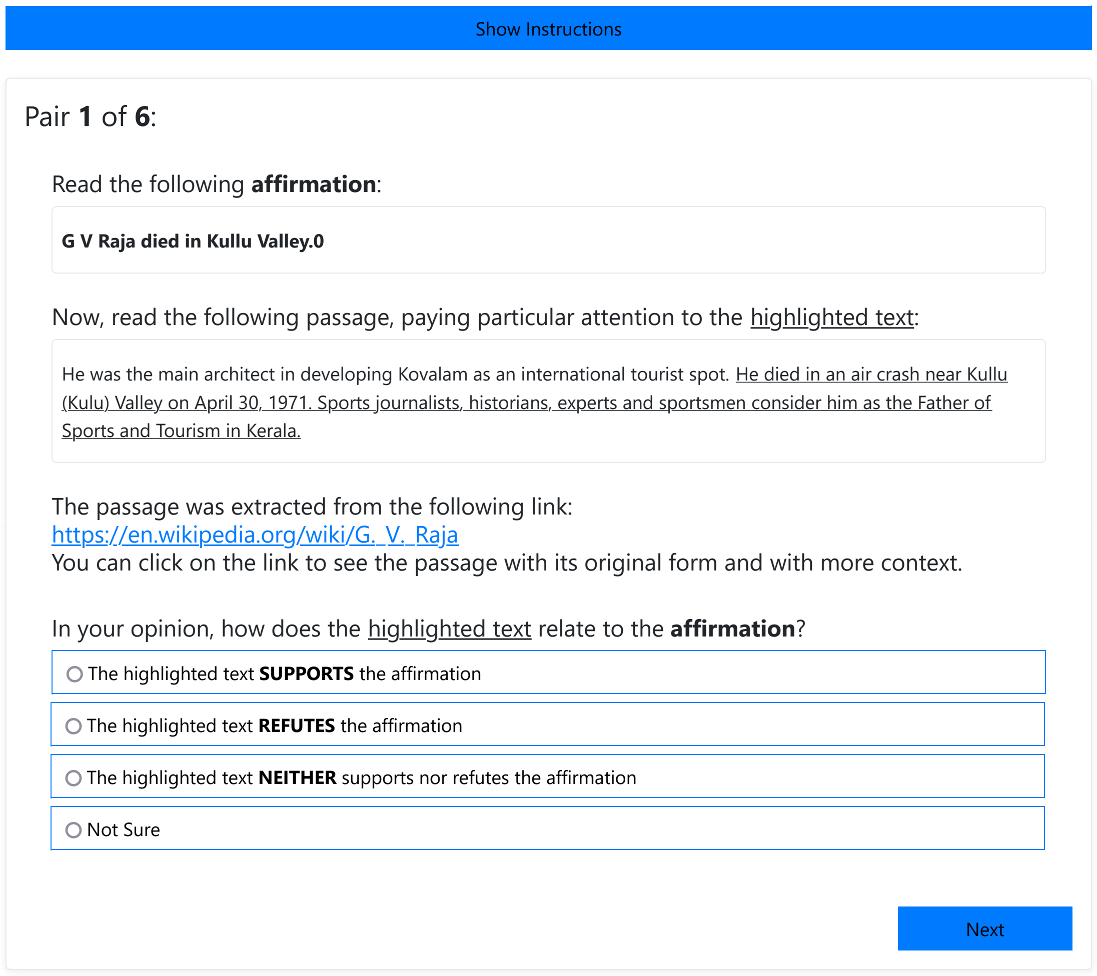
56. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for linked data: A survey. Semantic Web **7**(1), 63–93 (2016)

57. Zeng, X., Abumansour, A.S., Zubiaga, A.: Automated fact-checking: A survey. Language and Linguistics Compass **15**(10), e12438 (2021)

58. Zhong, W., Xu, J., Tang, D., Xu, Z., Duan, N., Zhou, M., Wang, J., Yin, J.: Reasoning over semantic-level graph for fact checking. arXiv preprint arXiv:1909.03745 (2019)

59. Zhou, J., Han, X., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Gear: Graph-based evidence aggregating and reasoning for fact verification. arXiv preprint arXiv:1908.01843 (2019)

A Crowdsourcing Task Designs


Show Instructions

Pair **1** of **6**:

Read the following **affirmation**:

G V Raja died in Kullu Valley.0

Now, read the following passage, paying particular attention to the highlighted text:

He was the main architect in developing Kovalam as an international tourist spot. He died in an air crash near Kullu (Kulu) Valley on April 30, 1971. Sports journalists, historians, experts and sportsmen consider him as the Father of Sports and Tourism in Kerala.

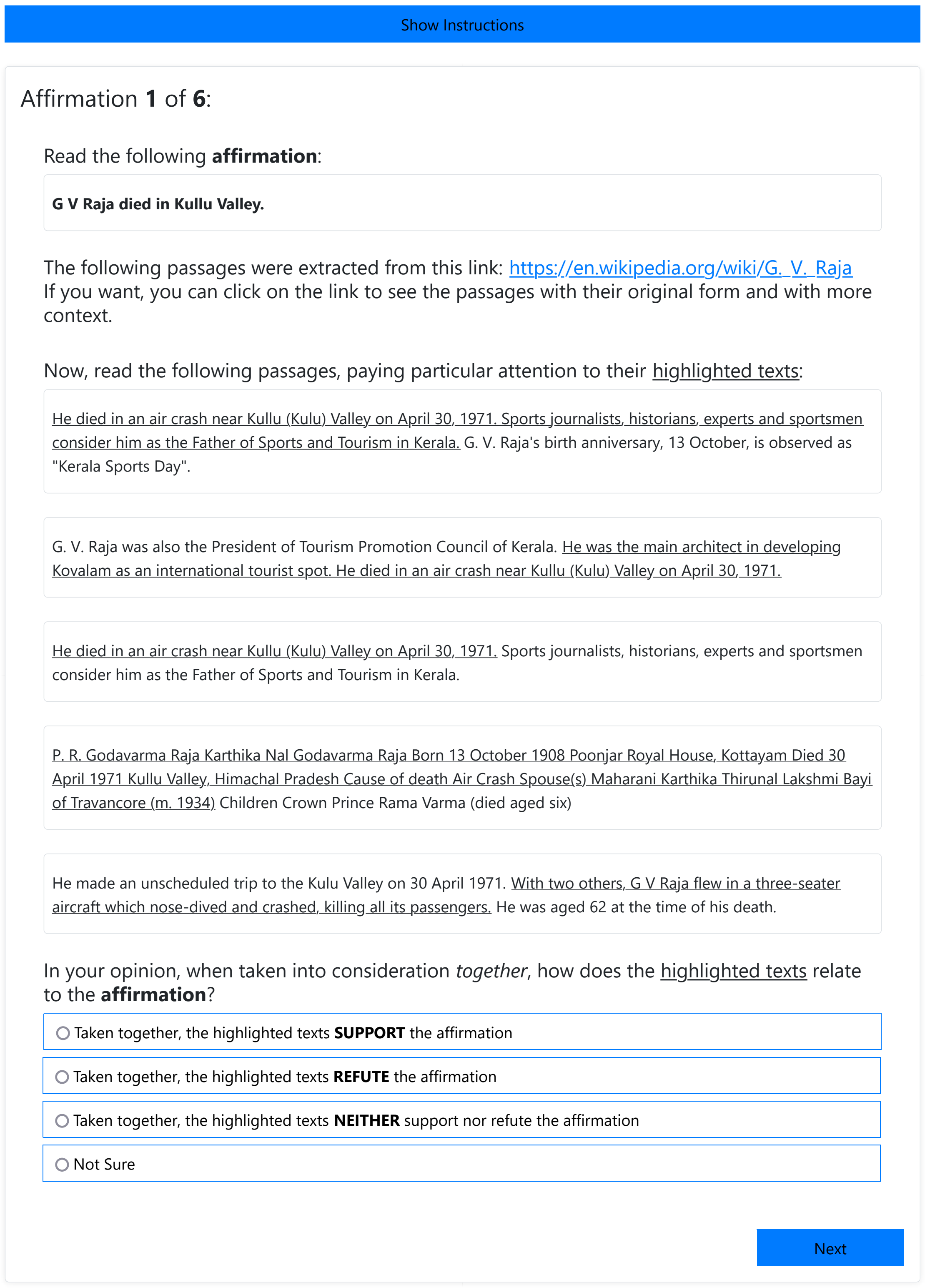
The passage was extracted from the following link:
https://en.wikipedia.org/wiki/G._V._Raja
You can click on the link to see the passage with its original form and with more context.

In your opinion, how does the highlighted text relate to the **affirmation**?

- ○ The highlighted text **SUPPORTS** the affirmation
- ○ The highlighted text **REFUTES** the affirmation
- ○ The highlighted text **NEITHER** supports nor refutes the affirmation
- ○ Not Sure

Next

Fig. 13. The task design T1, which collects evidence-level annotations of the individual stances of pieces of evidence towards a Wikidata triple.


Show Instructions

Affirmation **1** of **6**:

Read the following **affirmation**:

G V Raja died in Kullu Valley.

The following passages were extracted from this link: https://en.wikipedia.org/wiki/G._V._Raja
If you want, you can click on the link to see the passages with their original form and with more context.

Now, read the following passages, paying particular attention to their highlighted texts:

He died in an air crash near Kullu (Kulu) Valley on April 30, 1971. Sports journalists, historians, experts and sportsmen consider him as the Father of Sports and Tourism in Kerala. G. V. Raja's birth anniversary, 13 October, is observed as "Kerala Sports Day".

G. V. Raja was also the President of Tourism Promotion Council of Kerala. He was the main architect in developing Kovalam as an international tourist spot. He died in an air crash near Kullu (Kulu) Valley on April 30, 1971.

He died in an air crash near Kullu (Kulu) Valley on April 30, 1971. Sports journalists, historians, experts and sportsmen consider him as the Father of Sports and Tourism in Kerala.

P. R. Godavarma Raja Karthika Nal Godavarma Raja Born 13 October 1908 Poonjar Royal House, Kottayam Died 30 April 1971 Kullu Valley, Himachal Pradesh Cause of death Air Crash Spouse(s) Maharani Karthika Thirunal Lakshmi Bayi of Travancore (m. 1934) Children Crown Prince Rama Varma (died aged six)

He made an unscheduled trip to the Kulu Valley on 30 April 1971. With two others, G V Raja flew in a three-seater aircraft which nose-dived and crashed, killing all its passengers. He was aged 62 at the time of his death.

In your opinion, when taken into consideration *together*, how does the highlighted texts relate to the **affirmation**?

- ○ Taken together, the highlighted texts **SUPPORT** the affirmation
- ○ Taken together, the highlighted texts **REFUTE** the affirmation
- ○ Taken together, the highlighted texts **NEITHER** support nor refute the affirmation
- ○ Not Sure

Next

Fig. 14. The task design T2, which collects evidence-level annotations of the collective stances of evidence sets towards a Wikidata triple.

B WTR Dataset Format

WTR is available at Figshare [6] and contains 409 Wikidata triple-reference pairs, representing 32 groups of text-rich web domains commonly used as sources, as well as 76 distinct Wikidata properties. Out of the 416 sampled triple-reference pairs, as described in Section 4.1, 7 were left out due to having exactly the same triple components and referenced URL as other triple-reference pair, as explained in Section 6.3. 43% of references were obtained through external IDs and 57% through direct URLs. Each entry has the following attributes:

- Reference attributes:
  - Reference ID: An unique identifier issued to the reference by Wikidata;
  - Reference property ID: The unique identifier of the Wikidata property used by the reference to encode the URL we retrieve for it;
  - Reference datatype: Whether the reference's URL was retrieved by a direct URL or an URL formatted with an external identifier;
  - URL: The URL retrieved for this reference;
  - Netloc: The actual web domain of this URL;
  - Netloc group: The web domain of this URL after grouping references under the RARE and OTHER groups;
  - Final URL: The URL reached after redirects and whose HTML and text were extracted by ProVe into sentences for annotation;
  - HTML: The HTML code extracted from the reference's final URL.
- Claim attributes:
  - Claim ID: An unique identifier issued to the claim by Wikidata;
  - Rank: The claims's rank, either normal or preferred;
  - Datatype: The datatype of the claim's object, e.g. quantity, string, etc;
  - Component IDs: The Wikidata IDs of the claim's subject and property;
  - Component labels: Main labels for subject, property, and object;
  - Component aliases: Aliases lists for subject, property, and object;
  - Component descriptions: Wikidata descriptions for subject, property, and object (if it is a Wikidata item).
- Annotations for evaluation:
  - Evidence-level annotations (T1): The evidence-level annotations that describe the individual TER stances of each piece of evidence towards the claim, in which the evidence set is the five most relevant passages collected from the URL. This consists the following attributes for each piece of evidence:
    - ∗ Evidence: The individual textual evidence collected and being annotated;
    - ∗ MTurk IDs: A list of anonymous worker IDs and assignment IDs denoting the crowd workers who provided annotations/votes;
    - ∗ TER Relation: The list of TER stances voted by the workers, where 0 = SUPP, 1 = REF, 2 = NEI, 3 = Not Sure;
    - ∗ Reason for 'Not Sure;: If a voter gave 'not sure' as their relation, this denotes the reason why, out of the list of options seen in this paper's appendix or a free-text reason;
    - ∗ Times: The times in seconds taken by a worker to provide their full annotations;
    - ∗ Aggregated TER Relation: The majority-voting aggregation of each individual worker's TER stance annotation;
  - Evidence-level annotations (T2): The evidence-level annotations that describe the collective TER stances of the entire evidence set towards the claim, in which the evidence set is the five most relevant passages collected from the URL. This consists of the following attributes:
    - ∗ Evidence: The entire textual evidence set collected and being collectively annotated;
    - ∗ MTurk IDs: A list of anonymous worker IDs and assignment IDs denoting the crowd workers who provided annotations/votes;
    - ∗ TER Relation: The list of TER stances voted by the workers, where 0 = SUPP, 1 = REF, 2 = NEI, 3 = Not Sure;
    - ∗ Reason for 'Not Sure;: If a voter gave 'not sure' as their relation, this denotes the reason why, out of the list of options seen in this paper's appendix or a free-text reason;

[6] https://figshare.com/s/df0ec1c233ebd50817f4

  * Times: The times in seconds taken by a worker to provide their full annotations;
  * Aggregated TER Relation: The majority-voting aggregation of each individual worker's TER stance annotation;
- Sentence-level author annotations: The sentence-level annotation representing the stance of the entire reference towards the triple.