



King's Research Portal

DOI:
[10.3233/SW-233467](https://doi.org/10.3233/SW-233467)

Document Version
Peer reviewed version

[Link to publication record in King's Research Portal](#)

Citation for published version (APA):
Maia Rocha Amaral, G., Rodrigues, O., & Simperl, E. (2023). ProVe: A Pipeline for Automated Provenance Verification of Knowledge Graphs Against Textual Sources. *Semantic Web*. <https://doi.org/10.3233/SW-233467>

Citing this paper

Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

General rights

Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Research Portal

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

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

Gabriel Amaral^{a,*}, Odinaldo Rodrigues^a and Elena Simperl^a

^a *Informatics Department, King's College London, United Kingdom*

E-mails: gabriel.amaral@kcl.ac.uk, odinaldo.rodrigues@kcl.ac.uk, elena.simperl@kcl.ac.uk

Editors: Lucie-Aimée Kaffee, Hasso-Plattner-Institut, Germany; Simon Razniewski, Max Planck Institute for Informatics, Germany; Pavlos Vougiouklis, Huawei Technologies, United Kingdom

Solicited reviews: Christos Christodoulopoulos, Amazon, United Kingdom; Michael Röder, Paderborn University, Germany; Houcemeddine Turki, University of Sfax, Tunisia; Anonymous Reviewer

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 large network of interconnected entities, representing their semantic types, properties, and relationships to one another [1–3]. The information stored in most KGs can be seen as a set of semantic triples, formed by a subject, a predicate, and an object [3, 4]. They represent both concrete 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

*Corresponding author. E-mail: gabriel.amaral@kcl.ac.uk.

relationships, coded via uniquely identified predicates [2, 4, 5]. KGs serve as sources of both human and machine-readable semantically structured data for various crucial applications in the modern web landscape. This includes Wikipedia infoboxes, search engine results, voice-activated assistants, and information gathering projects [5, 6].

Developed and maintained by ontology experts, data curators, and even anonymous volunteers, KGs have massively grown in size and adoption in the last decade [2, 3, 7], including as secondary sources of information [8]. 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 [9].

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 [9–11]. However, such processes are currently performed mostly manually [12], with little automation options, and do not scale with size. Manually ensuring high verifiability on vital KGs such as Wikidata and DBpedia is prohibitive due to their sheer size [10]. On Wikidata, for instance, the creation of a provenance verification framework is highly desired by its product management, contributors, and users, and is actively tracked in its Phabricator page.¹ Wikidata recognizes their editors do not have the time, resources, or motivation to manually check the over 1 billion claims with documented provenance [13] that exist in the platform.

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 of KG provenance verification is a specific application of Automated Fact Checking (AFC). AFC is a currently well-explored topic of research with several published papers, surveys, and datasets [14–26]. It is 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 its main outputs. We define AFC on KGs as AFC where verified claims are KG triples. While AFC in general, and also AFC on KGs, mostly take a claim and a searchable evidence base as inputs [27–31], KG provenance verification is further defined by us as AFC on KGs where the evidence source is the triple’s documented textual provenance.

Approaches tackling AFC on KGs through textual evidence are very few. The only works of note in a similar direction, as far we know, are DeFacto [27, 32] and its successors FactCheck [29] and HybridFC [31]. Different to ProVe, however, they all rely on a searchable document base instead of a given reference and judge triples on a true-false spectrum instead of verifiability. ProVe is also amongst the first approaches to tackle AFC on KG triples with large pre-trained Language Models (LMs), which can be expanded to work in languages other than English and can benefit from Active Learning scenarios.

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 (79.4% accuracy, 57.4% F1-macro, 0.753 AUC) 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 Provenance Verification on KGs based on large LMs. This is our main contribution, as the usage of LMs to tackle AFC on KGs as provenance verification, as well as fine-tuning with adjacent datasets and tasks, is novel despite relying on existing models.
2. A benchmarking dataset of Wikidata triples and references for AFC on KGs, covering a variety of information domains as well as a balanced sample of diverse web domains.

¹<https://phabricator.wikimedia.org/T90881>

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 in 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. In this paper, ProVe’s applicability to the first use case is tested, with the remaining two tackled in future work.

The remainder of this paper is structured as follows. Section 2 explores related work on KG data quality, mainly verifiability, as well as related approaches to AFC. 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 Figshare² and GitHub.^{3,4}

2. Related Work

ProVe attempts to solve the task of AFC on KGs, more specifically KG provenance verification, 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. We will then explore how state of the art approaches to AFC in general and AFC on KGs tackle these tasks and how ProVe learns or differs from them.

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 [9]. It is an essential aspect of trustworthiness [4, 9, 33], yet is amongst the least explored quality dimensions [9, 33], with most measures carried superficially, unlike correctness or consistency [33–37].

For instance, Farber et al. [4] measure verifiability only by considering whether any provenance is provided at all. Flouris et al. [38] 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 [10, 13] 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 [39, 40].

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 [10, 13], multiple quality assurance techniques are implemented to ensure collected annotations are trustworthy [41]. To the best of our knowledge, this is the first work to use crowdsourcing as a tool to measure the relevance and stance of references in regard to KG triples at levels varying from whole references to individual text passages.

²<https://figshare.com/s/df0ec1c233ebd50817f4>

³<https://github.com/gabrielmaia7/RSP>

⁴<https://github.com/gabrielmaia7/ClaimVerificationHIT>

2.2. Automated Fact Checking on Knowledge Graphs

2.2.1. AFC in General

AFC is a topic of several works of research, datasets, and surveys [14–26]. AFC is commonly defined in the NLP domain as a broader category of tasks and subtasks [14–16] whose goal is to, given a claim, verify its veracity or support by automatically collecting and reasoning over pieces of evidence. Such claims are often textual, but can also be subject-predicate-object triples [14, 16] while evidence is extracted from the searchable document or data corpora, such as collections of web pages or KGs. The collected evidence constitutes AFC’s output alongside the claim’s verdict [14–16]. 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 [15, 16], and can be seen in Figure 1. Zeng et al. [15] 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 (also referred to as evidence selection) then identifies and extracts from retrieved documents a set of few individual text passages that are deemed the most relevant. Such passages will serve as evidence. In the case of KGs as evidence sources, paths or embeddings are retrieved instead. Based on this evidence, a *claim verification* step provides the final verdict. Guo et al. [16] 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.

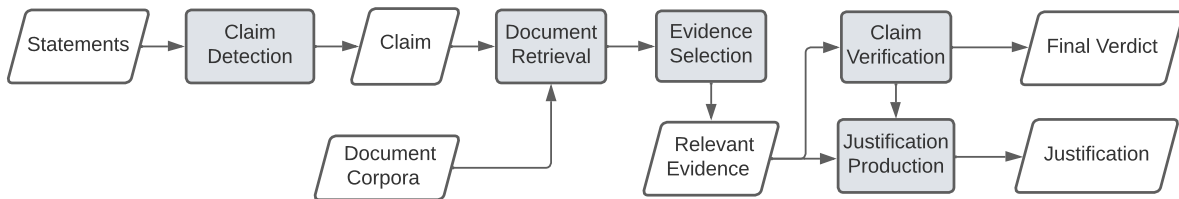


Fig. 1. Overview of a general AFC pipeline. White diamond blocks are documents and objects, and grey square blocks are AFC subtasks. Specific formulations and implementation of course might differ.

2.2.2. AFC on KGs

AFC mostly 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 [24] and its associated large-scale benchmark FEVER dataset [25]. Still, some works in AFC take semantic triples as verifiable claims, either from KGs [42, 43] or by extracting them from text. Some also utilise KGs as reasoning structures from where to draw evidence [16, 44–47]. For instance, Thorne and Vlachos [45] directly map claims found in text to triples in a KG to verify numerical values. Both Ciampaglia et al. [46] and Shiralkar et al. [47] 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 [48, 49].

KGCleaner [28] 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 [30] validates KG triples for three specific organisation properties, using specifically designed extractions from HTML content. Both approaches require manual work, cover a limited amount of predicates, and do not provide human-readable evidence.

DeFacto [27] and its successors FactCheck [29] and HybridFC [31] represent the current lineage of state of the art systems on AFC on KG given a wide searchable textual evidence base. They verbalise KG triples using text patterns and use it to retrieve web pages with related content. HybridFC converts retrieved evidence sentences into numeric vectors using a sentence embedding model. It additionally relies on graph embeddings and paths. These approaches score sentences based on relevance to the claim and use a supervised classifier to classify the entire web page as evidence. Despite their good performance, the first two approaches depend on string matching, which might miss

1 verbalisations that are more nuanced and also entail considerable overhead for unseen predicates. Additionally, all 1
2 three depend on large searchable document bases from where to retrieve evidence and rely on training on the KG 2
3 predicates the user wants them to verify. ProVe, on the other hand, covers any non-ontological predicate by using 3
4 pre-trained LMs that leverage context and meaning to infer verbalisations. We define ontological predicates as those 4
5 whose meaning serves to structure or describe the ontology itself, such as *subclass of* and *main category of*. 5

6 Due to its specific application scenario, approaches tackling AFC on KGs differ from the general AFC frame- 6
7 work [15, 16] seen in Figure 1. A *claim detection* step is not deemed necessary, as triples are trivial to extract and 7
8 it is commonly assumed they all need verifying. Alternatively, one can easily select only triples with particular 8
9 predicates or properties. The existence of a *document retrieval* step depends on whether explicit provenance exists 9
10 or needs to be searched from a repository, with the former scenario dismissing the need for the step. This is the case 10
11 for ProVe, but not for the DeFacto line, which searches for web documents. 11

12 Additionally, KG triples are often not understood by the components' main labels alone. Descriptions, alterna- 12
13 tive labels, editor conventions, and discussion boards help define their proper usage and interpretation, rendering 13
14 their meaning not trivial. As such, approaches tackling AFC on KGs rely on transforming KG triples into natural 14
15 sentences [27, 29, 32] through an additional *claim verbalisation* step. While both DeFacto [27] and FactCheck [29] 15
16 rely on sentence patterns that are completed with the components' labels, and HybridFC assumes sentences will be 16
17 retrieved by either of those systems, ProVe relies on state of the art LMs for data-to-text conversion. 17

18 Lastly, evidence document corpora normally used in general AFC tend to have a standard structure or come 18
19 from a specific source. Both FEVER [25] and VitaminC [50] take their evidence sets from Wikipedia, which has 19
20 a text-centered and text-rich layout, with FEVER's even coming pre-segmented as clean individual sentences. Vo 20
21 and Lee [51] use web articles from snopes.com and politifact.com only. KGs, however, accept provenance from 21
22 potentially any website domains. As such, unlike general AFC approaches, ProVe employs a *text extraction* step 22
23 in order to retrieve and segment text from triples' references. While previous approaches simply remove HTML 23
24 markup, ProVe employs a rule-based approach that allows for more flexibility. 24
25

26 2.2.3. Large Pre-trained Language Models on AFC 26

27 Advances towards textual evidence-based AFC, particularly the *sentence selection* and *claim verification* sub- 27
28 tasks, have been facilitated by resources like the FEVER [24, 25] shared task and its benchmarking dataset. The 28
29 FEVER dataset consists of a large set of claims annotated with one of three classes: supported, refuted, and not 29
30 enough information to determine. The dataset also provides pre-extracted and segmented passages from Wikipedia 30
31 as evidence for each claim. 31

32 Tackling FEVER through pre-trained LMs [17–19] and graph networks [21–23] represents the current state of 32
33 the art. While approaches using graph networks (such as KGAT [22], GEAR [23], and DREAM [21]) for *claim* 33
34 *verification* slightly outperform those based mainly on sequence-to-sequence LMs, they still massively depend on 34
35 the latter for *sentence selection*. Additionally, explainability for task-specific graph architectures, like those of KGAT 35
36 and DREAM, is harder to tackle [52, 53] than for generalist textual sequence-to-sequence LM architectures which 36
37 are shared across the research community [54–56]. Slightly decreasing potential performance in favour of a simpler 37
38 and more explainable pipeline, ProVe employs LMs for both *sentence selection* and *claim verification*. 38
39

40 On *sentence selection*, the common strategy is to assign relevance scores to text passages based on their contextual 40
41 proximity to a verifiable claim. GEAR [23] does so with LSTMs, but use a BERT model to acquire text encodings. 41
42 Soleimani et al. [17], KGAT [22], and current state of the art DREAM [21] outperform GEAR by directly using 42
43 BERT for the rankings, an approach ProVe also follows. Graph networks are employed at the *claim verification* 43
44 subtask [21–23]. Soleimani et al. [17] are among the few to achieve near state of the art results using an LM and 44
45 a rule-based aggregation instead of graph networks. While ProVe handles the subtask similarly, it uses a weak 45
46 classifier as its final aggregation. 46

47 As a subtask in AFC on KGs, *claim verbalisation* is normally done through text patterns [27, 29] and by filling 47
48 templates [28], both of which are either distantly learned or manually crafted to address specific predicates. ProVe 48
49 is the first approach to utilise an LM for this subtask in the AFC context. Amaral et al. [57] show that a T5 model 49
50 fine-tuned on WebNLG achieves very good results when verbalising triples from Wikidata across many domains, 50
51 covering more than just the learned predicates. ProVe follows suit by also using a T5. 51

Table 1 shows a comparison of ProVe to other AFC approaches mentioned in this section grouped by specific tasks. It showcases the particular subtasks each approach targets, as well as the datasets used as a basis for their evaluation. AFC on KGs through textual evidence is amongst the least researched topics within AFC, and this paper is the first to do so through a KG provenance verification approach. ProVe tackles it through fine-tuned LMs that adapt to unseen KG predicates and is evaluated on a dataset consisting of Wikidata triples with multiple non-ontological predicates.

Description	Input Type	Evidence Source	Evidence Returned	Subtasks	Evaluation Dataset	Approaches
AFC on text through text	Textual claims	Text	Yes	DR, SS, CV	FEVER	[17, 18, 21–23],
AFC on text claims through KGs	Textual claims	KG	Yes	SS, CV	Freebase	[44, 45]
AFC on KGs through KGs	KG triples	KG paths	Yes	PS, CV	DBpedia, SemMedDB, Wikipedia	[42, 43, 46, 47],
		KGE	No	RE, EL, CV	Kaggle, news articles, DBpedia, Freebase	[48, 49]
AFC on KGs through text	KG triples	Text	Yes	CVb, DR, TA, CV	DBpedia, FactBench	[27, 29, 31, 32]
			No	SS, CV	Wikidata (48 predicates), SWC 2017	[28, 30]
			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, PS = Path 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 AFC on KGs. It takes as inputs a KG triple that is non-ontological in nature and its documented provenance in the form of a web page or text document. It then automatically verifies whether the provenance 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 LMs.

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 $p_i \in P$, 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 $\sigma_i^k \in [0, 1]$ for $k \in \mathcal{K}$, and $\sum_{k \in \mathcal{K}} \sigma_i^k = 1$ with $\mathcal{K} = \{SUPP, REF, NEI\}$. 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 .

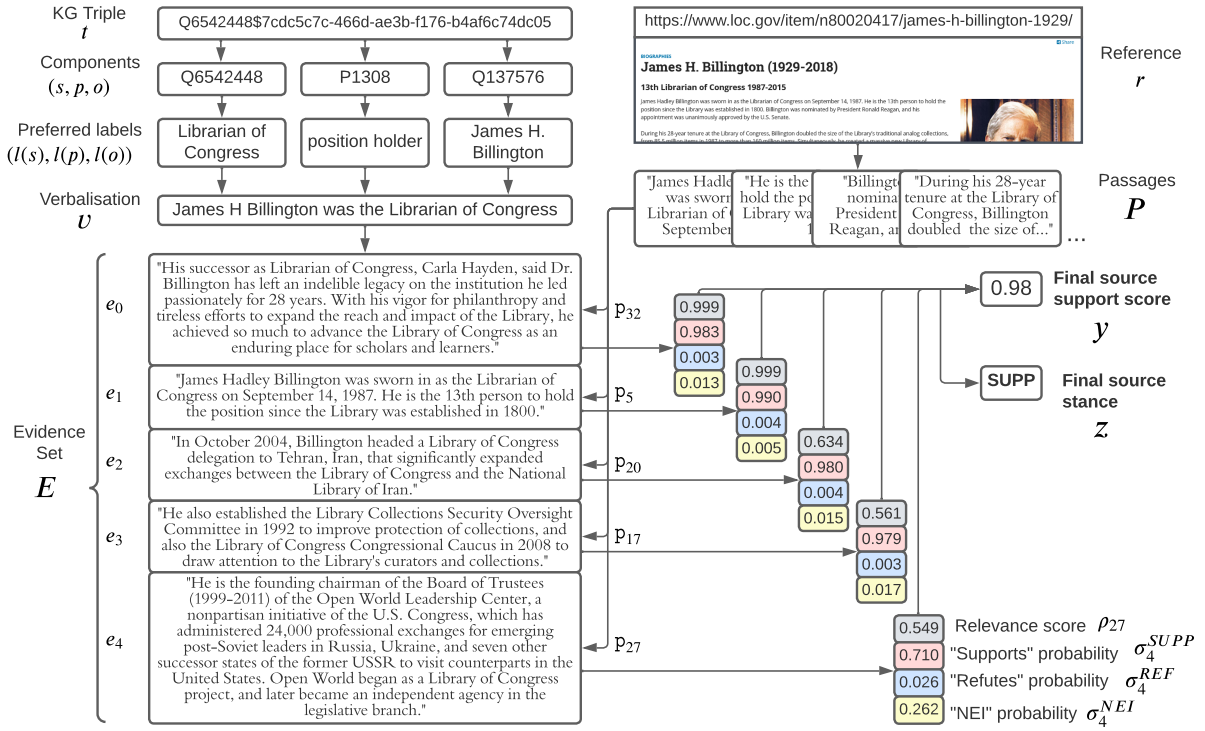


Fig. 2. An example of the inputs and outputs of ProVe when applied to a Wikidata triple and its provenance. A triple’s (t) subject, predicate, and object elements (s, p, o) have their labels extracted and verbalised (v). Reference r has its passages extracted (P). The 5 most relevant passages are compiled 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 27th extracted passage is the 4th 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 (v) that expresses the same information. As KG entities and predicates might contain multiple labels, multiple possible verbalisations can be generated. Users might choose those labels that best portray the triple’s meaning, which we

define as preferred labels, and rules within the KG can help 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.

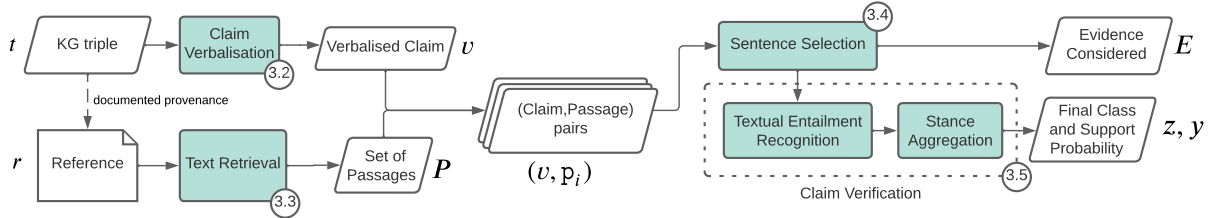


Fig. 3. Overview of ProVe’s pipeline. The white blocks are artefacts while the green are modules, further detailed in the subsections indicated in the circles.

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 Recognizing Textual Entailment (RTE) step assigns each extracted passage with probabilities of having each of the three evidence stances. 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.

On the other hand, ProVe handles both KG triples and unstructured text with the same model architectures and does not make use of KG paths for evidence. Thus, it needs to convert the triples into text through a claim verbalisation module, akin to most other approaches in this task [27–29, 32]. 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 [25]), ProVe needs a non-trivial text retrieval module so that it can identify informative passages. Finally, such KGs are often secondary sources of information and triples should not include conclusions or interpretations from editors. Hence, 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 on which something was founded, written, formed, painted, or created, and applies to any entity with a beginning. Its description even 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 [27, 29, 32]. 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.

$$v = \phi(l(s), l(p), l(o)) \quad (1)$$

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 [57, 58], a pre-trained transformer is used for this subtask. The function $\phi(\cdot)$ is carried out by a T5-base [59] model fine-tuned on the WebNLG 2017 dataset [60]. 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. [57] use this exact same model to produce the WDV dataset, which consists of verbalised Wikidata triples. They 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 a 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 scraper, 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 $\langle p \rangle$ 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 is fed into spaCy’s sentence segmenter using the *en_core_web_lg* model [61], producing a set of text segments S . If r is a text document, it is fed directly into sentence segmentation.

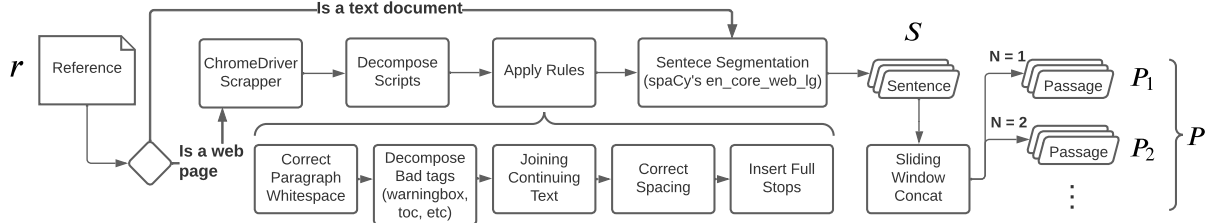


Fig. 4. Illustration of the text extraction module’s workflow, taking a reference r as input, dividing its text into a set of segments S , and concatenating them into a set of extracted passages P .

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 an n -sized sliding window and the union P of all such passages, respectively.

$$P_n = \{\odot(S_{i,i+n-1}) \mid 0 \leq i \leq |S| - n\} \quad (2)$$

$$P = \bigcup_{n \in \mathcal{N}} P_n \quad (3)$$

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 concatenates 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 [62, 63]. As no classification is perfect, these 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. We define relevance as contextual proximity e.g. similar entities or overlapping information. Sentence selection is an integral part of most recent AFC approaches [15, 16], including AFC on KGs [27, 29], as discussed in Section 2.2. It is a subtask of the FEVER fact-checking shared

task [24], directly supported as a supervised task by the FEVER dataset [25], and is explored by a large body of work [17, 18, 21–23] to excellent results.

Following on KGAT’s [22] and DREAM’s [21] 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 . The loss is a pairwise margin ranking loss with the margin set to 1 . Fine-tuning is achieved by feeding the model pairs of inputs. Each pair’s first element is a concatenation of a claim and a relevant sentence. The second element is the same claim, but with an irrelevant sentence instead. The model is trained assign higher scores to the first element, such that the difference in scores 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 [17, 22, 23] 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 $p_i \in P$ 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).

$$\rho_i = \psi(v, p_i) \quad (4)$$

$$E = \underset{P' \subseteq P^*, |P'|=5}{\operatorname{argmax}} \sum_{p_i \in P'} \rho_i \quad (5)$$

3.5. Claim Verification

As discussed in Section 2.2, claim verification is a crucial subtask in AFC, central to various approaches [14, 16], 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 model fine-tuned on data from FEVER [25] performs RTE 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. [17]: two fine-tuned BERT models, one for sentence selection, another for claim verification. Although task-specific graph-based approaches [21, 22] outperform Soleimani, it is by less than a percentage point (on FEVER score), while explainability for such generalist pre-trained LMs is increasingly researched [54–56] by the NLP community.

3.5.1. Recognizing Textual Entailment

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 RTE 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 (‘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 [17, 18].

ProVe’s RTE step is a BERT model fine-tuned on a multiclass classification RTE task. It consists of identifying a piece of evidence’s stance towards a claim using the three classes from FEVER. 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 NEI, 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 [22]) 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 [64] optimizer with 0.01 weight decay. Population Based Training was used to tune learning rate (1.25e-05), batch size (14), and warmup ratio (0.1).

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 RTE 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 RTE BERT model, which takes the concatenation of v and e_i as input (truncated to fit the model’s maximum token length by cutting at the end of e_i) 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 $\sum_{k \in \mathcal{K}} \sigma_i^k = 1$.

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

3.5.2. 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 weighted sum μ of the RTE probability arrays σ_i using the relevance scores ρ_i as weights, where negative relevance scores are dismissed (Equation 7). A final RTE class z is 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 probabilities of the ‘SUPPORTS’ class (Equation 9).

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

$$z = \operatorname{argmax}_{k \in \mathcal{K}} \mu^k \quad (8)$$

$$y = \mu^{SUPP} \quad (9)$$

2. The rule-based strategy adopted by Malon et al. [18] and Soleimani et al. [17]. A triple-reference pair (t, r) is assigned a final RTE 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 if the triple-reference pair is classified as ‘SUPPORTS’ and 0 otherwise (Equation 11). This strategy does not allow editors to vary the classification threshold.

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

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

3. A classifier, trained on an annotated set of triple-reference pairs (t, r) . The classifier takes as features all relevance scores ρ_i and all RTE 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 RTE 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 is defined as the class with the highest probability (Equation 13), and y as the probability θ^{SUPP} assigned to the ‘SUPPORTS’ class (Equation 14).

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

$$z = \operatorname{argmax}_{k \in \mathcal{K}} \theta^k \quad (13)$$

$$y = \theta^{SUPP} \quad (14)$$

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 [65]. In Wikidata, triples should, except for rare exceptions, have one or more references to their provenance [65]. Over 70% of Wikidata triples satisfy this requirement [13]. This is crucial as Wikidata does not care for truthfulness of information, but rather supportive provenance. There are no false triples, but Wikidata suffers from unsupported information due to erroneous provenance, both intentionally provided or not. 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 over a billion 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 shown by Amaral et al. [13], 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.

4.1.1. Selecting References

WTR is constructed from the Wikidata dumps from March 2022. First, all nearly 90M reference identifiers 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 randomly sampled from the resulting set due to storage size. Based on previous research [13], this maps to an estimated 220M claims.

Next, each extracted reference has its initial web URL 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. Three criteria were used:

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. [13], a substantial number of references fall in the first criteria, with an estimated over 70% of English references being automatically verifiable through API calls. Additionally, references with URLs to websites not in English (according to FastText's language identification models [66, 67]), 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 out using web domains as strata. The 30 most populous web domains are defined as 30 separate groups, with two additional groups formed from the remaining web domains: *RARE*, for web domains that appear only once, and *OTHER*, for all others.

4.1.2. 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 on capturing the population's proportion of supportive references. 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.

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

10 4.2. Dataset Annotation

11
12 As described in Section 3 (see Figure 2), ProVe tackles its AFC task as a sequence of text extraction, ranking,
13 classification, and aggregation subtasks. Given a triple-reference pair, ProVe extracts text passages from the ref-
14 erence, ranks them according to relevance to the triple, and individually classifies them according to their stance
15 towards the triple. Then, triple-reference pairs are classified according to the overall stance of the reference towards
16 the triple.

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

24 4.2.1. Crowdsourcing Evidence-level Annotations

25
26 Collecting evidence-level annotations for all retrievable sentences of each triple-reference pair in WTR, in order to
27 account for different rankings that can be outputted by ProVe, would be prohibitively expensive and inefficient. Thus,
28 evidence-level annotations are only provided for the five most-relevant passages in each reference, i.e. the collected
29 evidence. First, ProVe’s text retrieval (Section 3.3) and sentence selection (Section 3.4) modules are applied to each
30 reference and the five pieces of evidence for each collected. Since often only a couple of passages among the five
31 tend to be relevant, this does not severely bias the annotation towards highly-relevant text passages. It actually allows
32 for a more even collection of both relevant and irrelevant text passages. Then, evidence-level annotations for each
33 individual piece of evidence, as well as for the whole evidence set, are collected through crowdsourcing, totalling 6
34 annotation tasks per triple-reference pair.

35
36 4.2.1.1. *Task Design* Two crowdsourcing task designs have been created to carry out this evidence-level anno-
37 tation process. Task design 1 (T1) asks workers to assess the stance of an individual evidence towards the triple,
38 which can also be used as an indication of relevance. Each task in T1 is a bundle of 6 subtasks, each providing the
39 worker with a Wikidata triple, a piece of evidence extracted from its paired reference, the reference’s URL. Each
40 task then asks the worker the stance of that evidence: either it **supports** the claim, **refutes** it, has **not enough infor-**
41 **mation** for either, or the worker is **not sure**. This bundling is done in order to get more annotations out of a single
42 crowdsourcing task assignment. Task design 2 (T2) asks workers to assess the collective stance of the evidence set
43 (all five individual pieces of evidence) towards the triple. Similarly to tasks in T1, tasks in T2 are made from 6 sub-
44 tasks. Each T2 task provides the worker with a triple, five text passages extracted from the reference (the evidence
45 set), shown in a random order, and the reference’s URL. It then asks the *collective* stance of the five passages, with
46 similar response options to T1. The designs for both T1 and T2 tasks can be seen in Appendix A.

47
48 4.2.1.2. *Recruitment and Quality Assurance* The crowdsourcing campaign received ethical approval by King’s
49 College London on 7th of April, 2022, with registration number MRA-21/22-29584. All tasks were carried out
50 through Amazon Mechanical Turk (AMT). A pilot was run to collect feedback on instructions, design, and compen-
51 sation. After proper adjustments, the main data annotation tasks were carried out. Several quality control tech-
niques [41] were applied, following similar tasks by Amaral et al. [13, 57]. 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 [68], these results show fair and moderate agreement, respectively. Several factors that contribute to lower inter-annotator agreement [69] 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%) were 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’.

4.2.2. 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 sentence-level annotations, the mental load and task complexity of interacting with the page to inspect all information (e.g. in text, infoboxes, images, charts) 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):

- 1.A. Support explicitly stated as text as natural language sentences
- 1.B. Support explicitly stated as text, but not as natural language sentences
- 1.C. Support explicitly stated, but not as text
- 1.D. Support implicitly stated

2. Non-supporting References

- 2.A. Reference refutes claim (directly maps to the ‘**refutes**’ class)
- 2.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. They also allow us 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 structure 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. There, we also compare ProVe to other AFC on KG approaches on WTR.

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 [57], 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.

5.1.1. Model Validation

ProVe’s verbalisation module consists of a pre-trained T5 model [59] fine-tuned on the WebNLG dataset [60]. To confirm that fine-tuning was properly carried out, the authors measure the BLEU [70] scores of its verbalisations on WebNLG data. The scores are 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 [58].

Amaral et al. [57] 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 [57] contains a more detailed evaluation.

5.1.2. 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

1 by KG editors and curators. One example is the predicate `child` (P40), whose alias ‘has child’ is used to remedy 1
2 the main label’s lack of explicit direction. 2

3 Out of the 416 verbalisations generated by ProVe through main labels, 62.6% were adequate and had good to 3
4 excellent fluency. Another 29.8% followed suit after predicate alias replacement, totalling 92.4%. The remaining 4
5 7.6% either had no aliases or could not be improved by them and had to be manually corrected before being passed 5
6 down the pipeline. While manual corrections are more demanding than alias selection, those were not frequent 6
7 and mainly affected specific properties such as `isomeric SMILES` (P2017) and `designation` (P1435). 7
8 Corrections were also not extensive; the normalised Levenshtein distance before and after corrections was under 8
9 0.25 (a quarter total length) for 80% of them. 9

10 Finally, 7 of the claims verbalised ended up having both identical URLs and verbalisations, and were thus dropped 10
11 from the evaluation downstream as they would have the exact same results. This results from ProVe not taking claim 11
12 qualifiers into consideration, which is further discussed in Section 6.3. 12

13 5.2. Text Extraction 13

14 Due to the complexity of defining metrics that measure success in text extraction, this section instead first defines 14
15 metrics that can be used for an indirect evaluation of the text extraction module. It then explores insightful descriptive 15
16 metrics obtained from executing the module on WTR. 16

17 5.2.1. Indirect Evaluation 17

18 ProVe’s text extraction module essentially performs a full segmentation of the referenced web-page’s textual 18
19 content without excluding any text. The model can not be directly evaluated through annotations due to the sheer 19
20 quantity of ways in which one can segment all references contained in the evaluation dataset. Annotating such text 20
21 extractions would require manually analysing entire web pages to find all textual content relevant to the claim, 21
22 inspecting all the text extractable from such page, and segmenting it such that boundaries are properly placed in 22
23 terms of syntax and relevant passages are kept unbroken. One would then need to compare one or multiple ideal 23
24 extractions to the extraction performed by ProVe. It is neither trivial to simplify this process for crowdworkers, nor 24
25 efficient for the authors to carry it out by hand. It is also not trivial to define what constitutes ‘well-placed’ sentence 25
26 boundaries nor how and if one can break relevant passages of text. 26
27 27

28 Thus, instead of a direct evaluation, the performances of the subsequent sentence selection and final aggregation 28
29 steps are used as indirect indicators of ProVe’s text extraction. A correlation between ProVe’s relevance scores 29
30 (ρ) and evidence-level crowd annotations, as defined in Section 5.3.2.3, can be used to measure how much ProVe 30
31 captures human-perceived relevance. A high value of such metric indicates ProVe extracts text passages such that 31
32 relevant and irrelevant text segments are well divided. Otherwise, there would be a dissonance between humans and 32
33 models in rating relevance, as while humans are good in judging badly divided passages, models would struggle 33
34 significantly. 34
35 35

36 Likewise, a good final classification performance, measured against WTR’s reference-level annotations, indicates 36
37 ProVe’s capacity of extracting useful sentences. Classification metrics for ternary and binary classification tasks, 37
38 such as accuracy and f-scores, are shown in Section 5.4. Still, the direct evaluation of ProVe’s text extraction module, 38
39 encompassing sentence segmentation and meaning extraction from unstructured and semi-structured textual content, 39
40 is intended as future work. 40
41 41

42 5.2.2. Execution on WTR 42

43 The text extraction module was applied to each of the 416 triple-reference pairs (t, r) in WTR, each yielding a 43
44 set of passages P , as defined in Section 3.3. The total number of extracted passages was of nearly 64K, an average 44
45 of 154 passages per reference. This average number of extracted passages varied heavily according to web domain, 45
46 ranging from 1 to 804, with relatively low inter-domain standard deviation (29.51 median standard deviation). 46
47 The average size, in characters, of individual extracted passages behaved similarly, ranging from 24.20 to 4059.23 47
48 depending on web domain. Finally, the number of passages $|P|$ extracted from a reference r and their average size 48
49 $(\sum_{i=0}^{|P|-1} |p_i|)/|P|$ have a slightly moderate negative correlation (-0.2513 Pearson’s r), with a few domains, such as 49
50 `bioguide.congress.gov` returning very few but very large passages. 50
51 51

These metrics confirm that extracted textual content mainly varies 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 `bioguide.congress.gov`, due to their textual content being contained in a single paragraph (`<p>` HTML tag) without periods or any whitespaces to serve as 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 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 is measured by relying on the crowdsourced annotations.

5.3.1. 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 Scorer⁵, the model scores 0.945 and 0.870 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 [22] and DREAM [21], current state of the art, on the test partition.⁶ These indicate the model has properly fine-tuned to FEVER.

5.3.2. 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.

5.3.2.1. Distribution of Relevance Score Relevance scores (ρ) varied heavily across web domains. For each triple-reference pair (t, r) in WTR, its passages P were scored by the sentence selector module, and the scores of the resulting 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 was extensive variation across web domains. This was 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.

5.3.2.2. Proportion of Irrelevant Passages Given the distributions seen in Figure 5, we 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.

⁵<https://github.com/sheffieldnlp/fever-scorer>

⁶<https://competitions.codalab.org/competitions/18814#results>

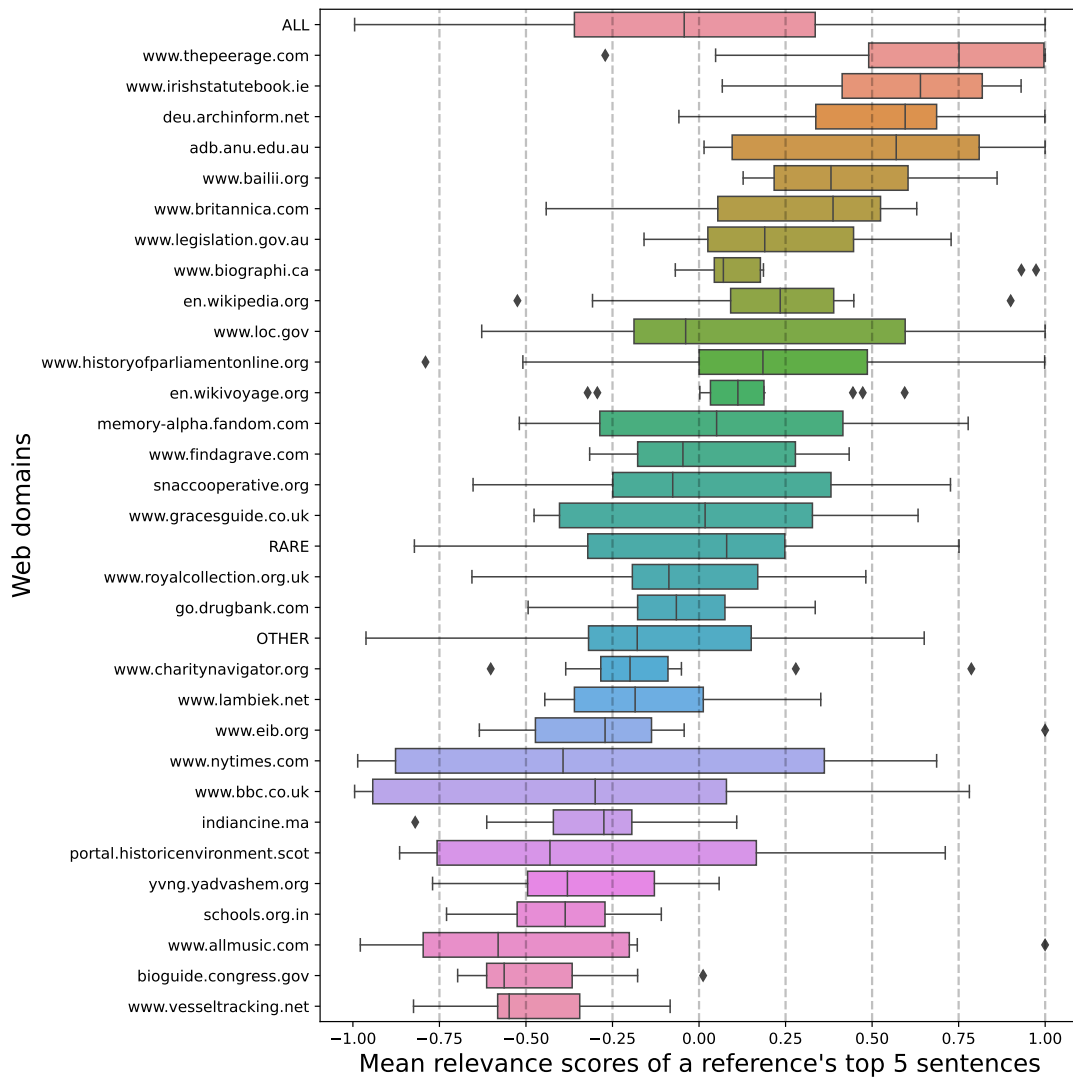


Fig. 5. Relevance scores distributions across and within different web domains. 'ALL' stands for the combined distribution of the subsequent 32 groups. Data values here are the averages of the top 5 passages' relevance scores for each reference.

5.3.2.3. Correlation Between Modeled and Crowdsourced Relevance 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.

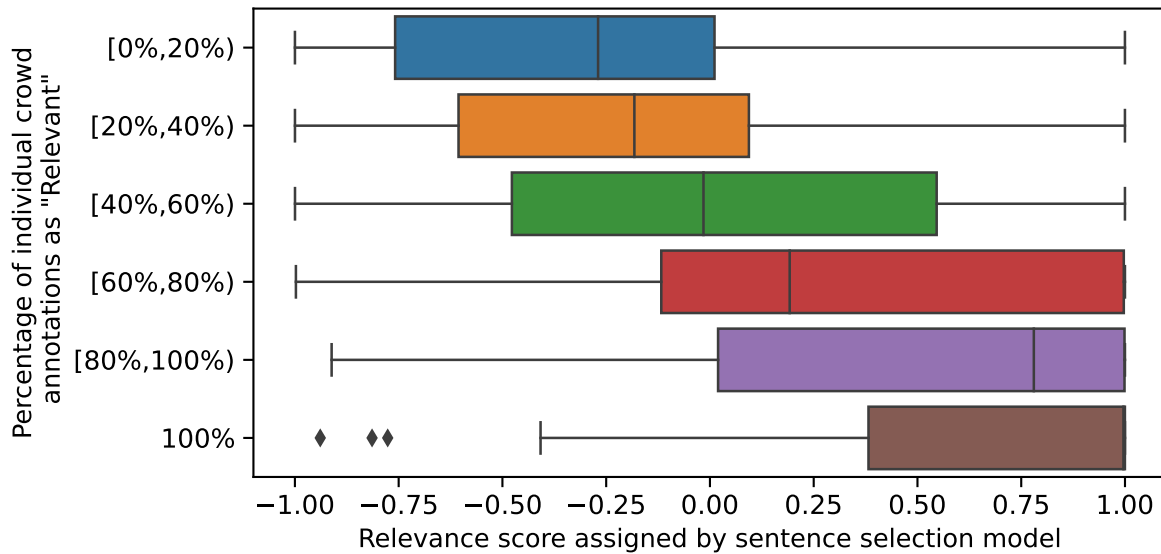


Fig. 6. Distributions of relevance scores given by the module divided by the percentage of crowd annotations deeming that passage as ‘relevant’ (either ‘supports’ or ‘refutes’).

5.4. Claim Verification

The first step of ProVe’s claim verification module consists of a RTE classification task, resolved by a fine-tuned BERT model, as described in Section 3.5. Such RTE 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 RTE 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 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.

5.4.1. 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 RTE 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 out 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. That is due to neither requiring training a new classifier and thus have similar complexity to other approaches used to tackle FEVER [17, 18], 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 RTE classes. FEVER score is an accuracy 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 0.696 label accuracy and 0.695 FEVER score on the validation set, and 0.650 and 0.617 respectively on the testing set. Method #2, the rule-based aggregation, scores 0.762 label accuracy and 0.761 FEVER score on the validation set, and 0.703 and 0.673 respectively on the testing set. Method #2 puts ProVe 3.2 percentage points below state of the art (DREAM [21]) on FEVER score, which the authors consider sufficient to validate the module’s fine-tuning.

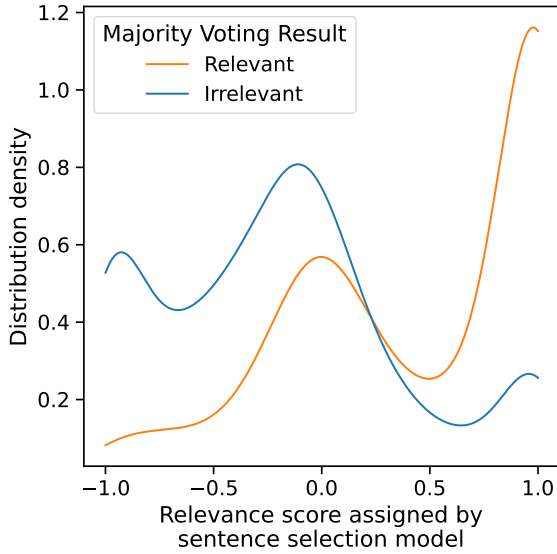


Fig. 7. Relevance score distributions of passages majority-voted as ‘relevant’ and of those voted ‘irrelevant’.



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

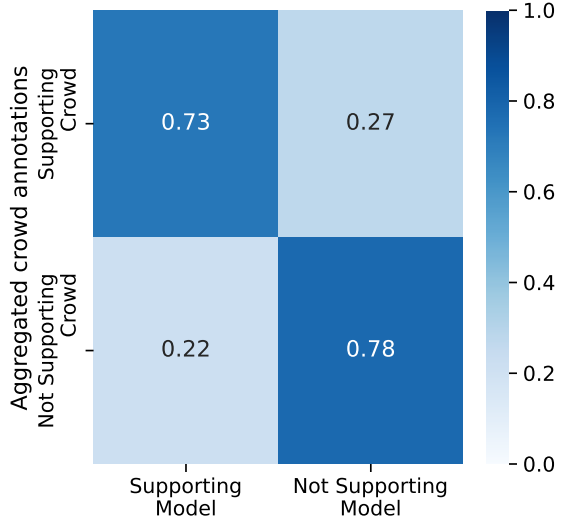
5.4.2. Execution on WTR

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.

5.4.2.1. Distributions of RTE Class Probabilities Similarly to the relevance scores outputted by the sentence selection module, the RTE class probabilities outputted by the RTE 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 RTE class for that individual evidence z_i , where $z_i = \operatorname{argmax}_{k \in \mathcal{K}} \sigma_i^k$. Web domains such as *indiancine.ma* have close to 95% of its evidence classified as ‘not enough information’, while domains 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.

5.4.2.2. Individual RTE Classification Metrics Using the argmax approach described above 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, one can measure classification metrics for ProVe’s individual RTE 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 RTE 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.

ProVe’s pipeline focuses on the classification of entire references rather than individual sentences, and the RTE class probabilities are merely features for the final aggregation. Still, RTE classifications for individual pieces of evidence can be greatly improved by, rather than argmax, using a classifier with the three RTE 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.



TER model class predictions (simple binary classifier)

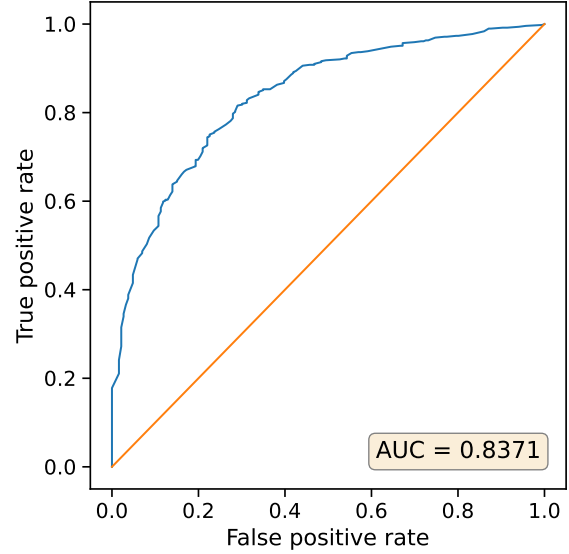


Fig. 10. ROC curve for the simplified binary stance classification performed by a RFC.

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

Furthermore, by grouping the ‘refutes’ and ‘neither’ 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 0.83, as seen in Figures 9 and 10.

5.4.2.3. Collective RTE Classification Metrics The annotations obtained through T2 tasks describe human judgments 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 classifier considerably outperforms the other two aggregation approaches, especially in the binary classification scenario.

Method	Classes	Accuracy	Macro Averaged			Weighted Averaged		
			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 RTE classification formulations. Majority-voted annotations obtained in T2 are used as true labels. Results from method 3 were cross-validated with $k = 5$.

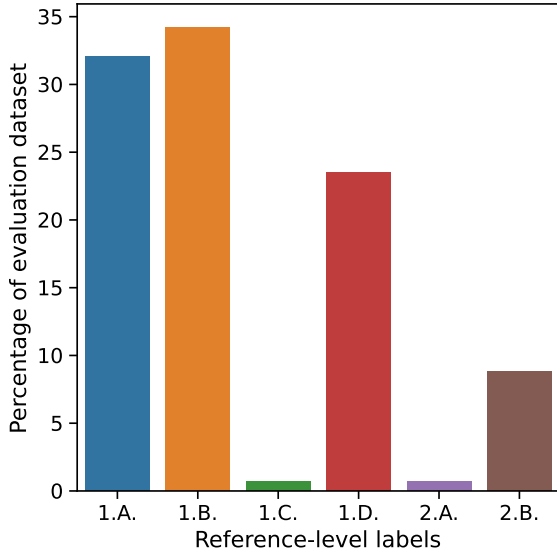


Fig. 11. Distribution of reference-level author labels of the evaluation dataset.

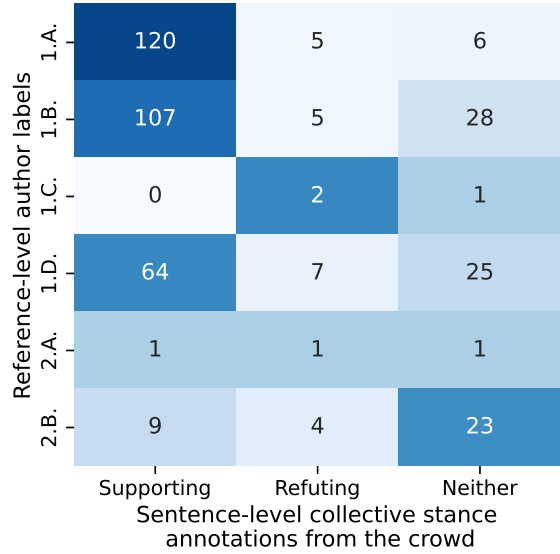


Fig. 12. Comparison between reference-level author label and sentence-level collective stance annotations from the crowd.

5.4.2.4. 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 RTE 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’. 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.

5.4.2.5. Full Pipeline Performance To evaluate the full pipeline, we investigate ProVe’s performance on a ‘supported’ vs. ‘not supported’ binary classification scenario 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 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 result 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.).

Support type	Accuracy	Macro Averaged			Weighted Averaged			AUC
		Precision	Recall	F1-Score	Precision	Recall	F1-Score	
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 classifier as aggregation method. Values obtained through cross-validation ($k = 5$).

5.4.2.6. Comparison Against Other AFC on KG Systems As explained in Section 2.2.2, the systems that come closest to ProVe in task formulation and approach within AFC on KGs are DeFacto [27], FactCheck [29], and HybridFC [31]. There are, however, crucial differences between how the DeFacto lineage and ProVe formulate this task. DeFacto, FactCheck, and HybridFC all evaluate the truthfulness of KG triples, meaning verifying whether the triple’s information is true or not. For that, they assume a collective body of knowledge that is cohesive and aligned with the truth, in the form of indexed Wikipedia dumps searchable through patterns or web documents retrievable by search engines. ProVe formulates AFC on KGs as provenance verification where support is verified, rather than truthfulness, i.e. it cares only whether a given piece of provenance supports a given KG triple, regardless if it is true.

As ProVe requires the provenance document as input, it is not possible to test it on FactBench, the benchmark dataset of the DeFacto lineage. However, if we assume that all retrievable information for a KG triple’s fact checking is a single pre-selected provenance document associated to it, and that its support is the same as a declaration of truthfulness, then DeFacto, FactCheck, and HybridFC can all be applied to the WTR dataset.

DeFacto relies on BOA patterns which are no longer available or maintained, thus only FactCheck and HybridFC can be evaluated. HybridFC consists of a mix of fact-checking methods, two of which rely on KG structures surrounding the input triple rather than solely on the provenance document. Since we do not consider the KG to hold a singular truth and are limiting ourselves to the provenance, we will not employ those two methods of HybridFC.

All these approaches rely on phrase patterns for document retrieval. For both FactCheck and HybridFC, documents and sentences are retrieved through an Elasticsearch⁷ query whose keywords are the concatenation of the triple’s subject, predicate, and object labels. FactCheck was executed on WTR using only the predicates’ preferred labels (as defined in Section 3.2), and then again after including all labels and aliases as provided by Wikidata. HybridFC was executed using the sentences retrieved by FactCheck’s second run. Both approaches also need to be trained on newly introduced predicates. Training and testing both systems was carried through a 50/50 stratified split of WTR. The resulting metrics for comparison can be seen in Table 4. It is important to note that WTR has a large class imbalance, with 158 supportive references and 25 non-supportive. This has greatly affected both approaches.

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.

⁷<https://www.elastic.co/>

Support type	Macro Averaged				Weighted Averaged			
	Accuracy	Precision	Recall	F1-Score	Precision	Recall	F1-Score	AUC
ProVe	0.794	0.565	0.635	0.574	0.864	0.794	0.821	0.753
FactCheck	0.328	0.544	0.594	0.312	0.889	0.328	0.396	0.572
FactCheck + Aliases	0.369	0.549	0.617	0.345	0.893	0.369	0.447	0.615
HybridFC (Textual Component)	0.863	0.432	0.500	0.462	0.745	0.863	0.800	0.618

Table 4

ProVe’s results compared to FactCheck and HybridFC. Metrics for ProVe are based on the entirety of WTR. Metrics for FactCheck and HybridFC are based on the test partition.

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 [71]. Its effectiveness depends 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 [71] 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 [27], FactCheck [29], and HybridFC [31]. ProVe clearly outperforms this line of systems on the provenance verification task presented by the WTR dataset. FactCheck and HybridFC suffered due to being limited to a single document as a source of evidence, as well as by the lack of semantic capabilities during sentence retrieval. For 71% of triples in WTR, using all labels and aliases, no evidence sentence was collected at all by both, with 15% of triples limited to a single extracted sentence.

ProVe’s use as a tool can greatly benefit from an active learning scenario which would further enhance the models and techniques it employs. At 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 show where to improve text extraction. Be it through descriptive statistics or comparison against golden data, this is 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

1 as ‘start time’ and ‘end time’. If a person were Librarian of Congress twice, this qualifier would differentiate two 1
2 triples that would otherwise be identical if expressed only with main components. ProVe does not take qualifiers 2
3 into consideration. As such, two triples with distinct IDs and meanings can have exactly the same verbalisations 3
4 which, while adequate, do not contain all information. 4

5 Ribeiro et al. [58] show transformers can verbalise multiple triples into a single sentence. Hence, adding qualifiers 5
6 as secondary triples is possible, generating more detailed verbalisations. However, ProVe’s sentence selection and 6
7 claim verification modules contain models fine-tuned on FEVER, whose vast majority of sentences contain only 7
8 a main piece of information with little to no additional details, e.g. ‘Adrienne Bailon is an accountant’ and ‘The 8
9 Levant was ruled by the House of Lusignan’. In order to make proper use of qualifiers during verbalisation, there 9
10 needs to be an assurance that downstream modules can properly handle more complex sentences by either different 10
11 or augmented training data. 11

12 6.4. Detecting Refuting Sources with FEVER 12

13 13
14 14
15 The FEVER dataset presents claims that are normally short and direct in nature, from multiple domains, and 15
16 associated evidence extracted directly from Wikipedia. ProVe shows it is possible to use FEVER to train pipeline 16
17 modules to detect supportive and non-supportive sources of evidence. However, as seen in Section 5.4, detecting 17
18 refuting sources is hard for ProVe and is believed to be due to how FEVER generates refuted claims through ar- 18
19 tificial alterations. Claims labelled by FEVER as ‘REFUTES’ are those generated by annotators who alter claims 19
20 that would otherwise be supported by its associated evidence. Alterations follow six types: paraphrasing, negation, 20
21 entity/relationship substitution, and making the claim more general/specific. This leads to claims that, while mean- 21
22 ingful and properly annotated, would never be encoded in a KG triple, such as “As the Vietnam War raged in 1969, 22
23 Yoko Ono and her husband John Lennon **did not** have two week-long Bed-Ins for Peace” or “Ruth Negga **only** 23
24 acts in Irish cinema”. Additionally, associated evidence often rely on common sense in order to refute these claims, 24
25 such as “Kingdom Hearts III is owned by Boyz II Men”, whose relevant evidence at no point elaborate on Kingdom 25
26 Hearts III’s ownership, only describing it as a Japanese video-game. We supposedly know Boyz II Men is a music 26
27 group, rendering the claim implausible. 27

28 While useful for other tasks, these refuted claims are very different from refutable triples occurring naturally in 28
29 KGs, which mainly consist of triples whose objects have different values in the provenance. One such example 29
30 is “Robert Brunton was born on 23/03/1796”, whose reference actually mentions the “10th of February 1796”. In 30
31 order to better detect KG provenance that refute their triples, ProVe’s claim verification module needs re-training on 31
32 a fitting subset of FEVER, or on a new dataset containing non-artificial refuted claims. 32

33 6.5. Conclusions 33

34 34
35 35
36 KGs are widespread secondary sources of information. Their data is extremely useful and available in a semantic 36
37 format that covers a myriad of domains. Ensuring the verifiability of this data through documented provenance is a 37
38 task that is crucial to the upkeep of their usability. This provenance verification task should be actively supported by 38
39 automated and semi-automated tools to help data curators and editors cope with the sheer volume of information. 39
40 However, as of now, there are no such tools deployed at large scale KGs and only a very small family of approaches 40
41 tackle this task from a research standpoint. 41

42 This paper proposes, describes, and evaluates ProVe, a pipelined approach to support the upkeep of KG triple 42
43 verifiability through their documented provenance. ProVe leverages large pre-trained LMs, rule-based methods, and 43
44 classifiers, to provide automated assistance to the activity of creating and maintaining references in a KG. ProVe’s 44
45 pipeline aims at extracting relevant textual information from references and evaluating whether or not they support 45
46 an associated KG triple. This provides its users with support classification, support probability, as well as relevance 46
47 and textual entailment metrics for the evidence used. Deployed correctly, ProVe can help detect verifiability issues 47
48 in existing references, as well as improve the reuse of good sources. Additionally, the approach can be expanded to 48
49 work in a multilingual setting. 49

50 ProVe has been evaluated on WTR, a dataset of triple-reference pairs extracted directly from Wikidata and an- 50
51 notated by both crowdworkers and the authors. ProVe achieves 79.4% accuracy, 0.574 F1-macro, and 0.753 AUC 51

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). ProVe was also compared to other similar AFC on KG systems, such as FactCheck [29] and HybridFC [31], on the WTR dataset. ProVe outperforms such systems on this provenance verification task by a considerable margin.

In addition to the application of ProVe on reference re-usability and recommendation, 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] M. Krötzsch and G. Weikum, JWS special issue on knowledge graphs, Journal of Web Semantics. https://iccl.inf.tu-dresden.de/web/JWS_special_issue_on_Knowledge_Graphs/en.
- [2] P. Cimiano and H. Paulheim, Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods, *Semant. Web* **8**(3) (2017), 489–508–. doi:10.3233/SW-160218.
- [3] A. Hogan, E. Blomqvist, M. Cochez, C. D’Amato, G.D. Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier, A.-C.N. Ngomo, A. Polleres, S.M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab and A. Zimmermann, Knowledge Graphs, *ACM Comput. Surv.* **54**(4) (2021). doi:10.1145/3447772.
- [4] A. Zaveri, D. Kontokostas, S. Hellmann, J. Umbrich, M. Färber, F. Bartscherer, C. Menne, A. Rettinger, A. Zaveri, D. Kontokostas, S. Hellmann and J. Umbrich, Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO, *Semant. Web* **9**(1) (2018), 77–129–. doi:10.3233/SW-170275.
- [5] S. Ji, S. Pan, E. Cambria, P. Marttinen and P.S. Yu, A Survey on Knowledge Graphs: Representation, Acquisition, and Applications, *IEEE Transactions on Neural Networks and Learning Systems* **33**(2) (2022), 494–514. doi:10.1109/TNNLS.2021.3070843.
- [6] S. Malyshev, M. Krötzsch, L. González, J. Gonsior and A. Bielefeldt, Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia’s Knowledge Graph, in: *The Semantic Web – ISWC 2018*, D. Vrandečić, K. Bontcheva, M.C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee and E. Simperl, eds, Springer International Publishing, Cham, 2018, pp. 376–394. ISBN 978-3-030-00668-6.
- [7] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao and Z. Duan, Knowledge Graph Completion: A Review, *IEEE Access* **8** (2020), 192435–192456. doi:10.1109/ACCESS.2020.3030076.
- [8] D. Vrandečić, The Rise of Wikidata, *IEEE Intelligent Systems* **28**(4) (2013), 90–95–. doi:10.1109/MIS.2013.119.
- [9] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann and S. Auer, Quality assessment for Linked Data: A Survey, *Semantic Web* **7** (2015), 63–93. doi:10.3233/SW-150175.
- [10] A. Piscopo, L.-A. Kaffee, C. Phethean and E. Simperl, Provenance Information in a Collaborative Knowledge Graph: An Evaluation of Wikidata External References, in: *The Semantic Web – ISWC 2017*, C. d’Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange and J. Hefflin, eds, Springer International Publishing, Cham, 2017, pp. 542–558. ISBN 978-3-319-68288-4.
- [11] X. Wang, L. Chen, T. Ban, M. Usman, Y. Guan, S. Liu, T. Wu and H. Chen, Knowledge graph quality control: A survey, *Fundamental Research* **1**(5) (2021), 607–626. doi:<https://doi.org/10.1016/j.fmre.2021.09.003>. <https://www.sciencedirect.com/science/article/pii/S2667325821001655>.
- [12] E. McAndrew and C. Strathmann, Quality Assurance and reliability, The University of Edinburgh, 2021. <https://www.ed.ac.uk/information-services/help-consultancy/is-skills/wikimedia/wikidata/quality-assurance-and-reliability>.
- [13] G. Amaral, A. Piscopo, L.-a. Kaffee, O. Rodrigues and E. Simperl, Assessing the Quality of Sources in Wikidata Across Languages: A Hybrid Approach, *J. Data and Information Quality* **13**(4) (2021). doi:10.1145/3484828.
- [14] J. Thorne and A. Vlachos, Automated Fact Checking: Task Formulations, Methods and Future Directions, in: *Proceedings of the 27th International Conference on Computational Linguistics*, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 3346–3359. <https://aclanthology.org/C18-1283>.
- [15] X. Zeng, A.S. Abumansour and A. Zubiaga, Automated Fact-Checking: A Survey, *Language and Linguistics Compass* **15**(10) (2021), e12438.
- [16] Z. Guo, M. Schlichtkrull and A. Vlachos, A Survey on Automated Fact-Checking, *Transactions of the Association for Computational Linguistics* **10** (2022), 178–206. doi:10.1162/tacl_a_00454. <https://aclanthology.org/2022.tacl-1.11>.
- [17] A. Soleimani, C. Monz and M. Worring, BERT for Evidence Retrieval and Claim Verification, in: *Advances in Information Retrieval*, J.M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M.J. Silva and F. Martins, eds, Springer International Publishing, Cham, 2020, pp. 359–366. ISBN 978-3-030-45442-5.
- [18] C. Malon, Team Papeló: Transformer Networks at FEVER, in: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 109–113. doi:10.18653/v1/W18-5517. <https://aclanthology.org/W18-5517>.

- [19] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel and D. Kiela, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, in: *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20*, Curran Associates Inc., Red Hook, NY, USA, 2020. ISBN 9781713829546.
- [20] A. Hanselowski, H. Zhang, Z. Li, D. Sorokin, B. Schiller, C. Schulz and I. Gurevych, UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification, in: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 103–108. doi:10.18653/v1/W18-5516. <https://aclanthology.org/W18-5516>.
- [21] W. Zhong, J. Xu, D. Tang, Z. Xu, N. Duan, M. Zhou, J. Wang and J. Yin, Reasoning Over Semantic-Level Graph for Fact Checking, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 2020, pp. 6170–6180. doi:10.18653/v1/2020.acl-main.549. <https://aclanthology.org/2020.acl-main.549>.
- [22] Z. Liu, C. Xiong, M. Sun and Z. Liu, Fine-grained Fact Verification with Kernel Graph Attention Network, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 2020, pp. 7342–7351. doi:10.18653/v1/2020.acl-main.655. <https://aclanthology.org/2020.acl-main.655>.
- [23] J. Zhou, X. Han, C. Yang, Z. Liu, L. Wang, C. Li and M. Sun, GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification, in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Florence, Italy, 2019, pp. 892–901. doi:10.18653/v1/P19-1085. <https://aclanthology.org/P19-1085>.
- [24] J. Thorne, A. Vlachos, O. Cocarascu, C. Christodoulopoulos and A. Mittal, The Fact Extraction and VERification (FEVER) Shared Task, in: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 1–9. doi:10.18653/v1/W18-5501. <https://aclanthology.org/W18-5501>.
- [25] J. Thorne, A. Vlachos, C. Christodoulopoulos and A. Mittal, FEVER: a Large-scale Dataset for Fact Extraction and VERification, in: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 809–819. doi:10.18653/v1/N18-1074. <https://aclanthology.org/N18-1074>.
- [26] A. Sathe, S. Ather, T.M. Le, N. Perry and J. Park, Automated Fact-Checking of Claims from Wikipedia, in: *Proceedings of the Twelfth Language Resources and Evaluation Conference*, European Language Resources Association, Marseille, France, 2020, pp. 6874–6882. ISBN 979-10-95546-34-4. <https://aclanthology.org/2020.lrec-1.849>.
- [27] D. Gerber, D. Esteves, J. Lehmann, L. Bühhmann, R. Usbeck, A.-C. Ngonga Ngomo and R. Speck, DeFacto—Temporal and multilingual Deep Fact Validation, *Journal of Web Semantics* **35** (2015), 85–101, Machine Learning and Data Mining for the Semantic Web (MLDMSW). doi:<https://doi.org/10.1016/j.websem.2015.08.001>. <https://www.sciencedirect.com/science/article/pii/S1570826815000645>.
- [28] A. Padia, F. Ferraro and T. Finin, KGcleaner : Identifying and Correcting Errors Produced by Information Extraction Systems, *CoRR abs/1808.04816* (2018). <http://arxiv.org/abs/1808.04816>.
- [29] Z.H. Syed, M. Röder and A.-C. Ngonga Ngomo, FactCheck: Validating RDF Triples Using Textual Evidence, in: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18*, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1599–1602. ISBN 9781450360142. doi:10.1145/3269206.3269308.
- [30] R. Speck and A.-C. Ngonga Ngomo, Leopard — A Baseline Approach to Attribute Prediction and Validation for Knowledge Graph Population, *Web Semant.* **55**(C) (2019), 102–107. doi:10.1016/j.websem.2018.12.006.
- [31] U. Qudus, M. Röder, M. Saleem and A.-C. Ngonga Ngomo, HybridFC: A Hybrid Fact-Checking Approach For Knowledge Graphs, in: *The Semantic Web – ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings*, Springer-Verlag, Berlin, Heidelberg, 2022, pp. 462–480. ISBN 978-3-031-19432-0. doi:10.1007/978-3-031-19433-7_27.
- [32] J. Lehmann, D. Gerber, M. Morsey and A.-C. Ngonga Ngomo, DeFacto - Deep Fact Validation, in: *The Semantic Web – ISWC 2012*, P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J.X. Parreira, J. Hendler, G. Schreiber, A. Bernstein and E. Blomqvist, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 312–327. ISBN 978-3-642-35176-1.
- [33] A. Piscopo and E. Simperl, What We Talk about When We Talk about Wikidata Quality: A Literature Survey, in: *Proceedings of the 15th International Symposium on Open Collaboration, OpenSym '19*, Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450363198. doi:10.1145/3306446.3340822.
- [34] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe and P. Szekely, A study of the quality of Wikidata, *Journal of Web Semantics* **72** (2022), 100679. doi:<https://doi.org/10.1016/j.websem.2021.100679>. <https://www.sciencedirect.com/science/article/pii/S1570826821000536>.
- [35] M. Sabou, L. Aroyo, K. Bontcheva, A. Bozzon, M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, F. Flöck and J. Lehmann, Detecting Linked Data Quality Issues via Crowdsourcing: A DBpedia Study, *Semant. Web* **9**(3) (2018), 303–335. doi:10.3233/SW-160239.
- [36] D. Kontokostas, A. Zaveri, S. Auer and J. Lehmann, TripleCheckMate: A Tool for Crowdsourcing the Quality Assessment of Linked Data, in: *Knowledge Engineering and the Semantic Web*, P. Klinov and D. Mourmoultsev, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 265–272. ISBN 978-3-642-41360-5.
- [37] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer and J. Lehmann, Crowdsourcing Linked Data Quality Assessment, in: *The Semantic Web – ISWC 2013*, H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J.X. Parreira, L. Aroyo, N. Noy, C. Welty and K. Janowicz, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 260–276. ISBN 978-3-642-41338-4.
- [38] G. Flouris, Y. Roussakis, M. Poveda-Villalón, P.N. Mendes and I. Fundulaki, Using Provenance for Quality Assessment and Repair in Linked Open Data, in: *Joint Workshop on Knowledge Evolution and Ontology Dynamics*, 2012, oeg. <https://oa.upm.es/14477/>.
- [39] B. Xue and L. Zou, Knowledge Graph Quality Management: A Comprehensive Survey, *IEEE Transactions on Knowledge and Data Engineering* **35**(5) (2023), 4969–4988. doi:10.1109/TKDE.2022.3150080.
- [40] M. Cao, J. Zhang, S. Xu and Z. Ying, Knowledge Graphs Meet Crowdsourcing: A Brief Survey, in: *Cloud Computing*, L. Qi, M.R. Khosravi, X. Xu, Y. Zhang and V.G. Menon, eds, Springer International Publishing, Cham, 2021, pp. 3–17. ISBN 978-3-030-69992-5.

- [41] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah and M. Allahbakhsh, Quality Control in Crowdsourcing: A Survey of Quality Attributes, Assessment Techniques, and Assurance Actions, *ACM Comput. Surv.* **51**(1) (2018). doi:10.1145/3148148.
- [42] B. Shi and T. Weninger, Discriminative predicate path mining for fact checking in knowledge graphs, *Knowledge-Based Systems* **104** (2016), 123–133. doi:https://doi.org/10.1016/j.knosys.2016.04.015. https://www.sciencedirect.com/science/article/pii/S0950705116300570.
- [43] J. Kim and K.-s. Choi, 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*, International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 1677–1686. doi:10.18653/v1/2020.coling-main.147. https://aclanthology.org/2020.coling-main.147.
- [44] A. Vlachos and S. Riedel, Identification and Verification of Simple Claims about Statistical Properties, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 2596–2601. doi:10.18653/v1/D15-1312. https://aclanthology.org/D15-1312.
- [45] J. Thorne and A. Vlachos, 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*, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 37–40. https://aclanthology.org/E17-3010.
- [46] G.L. Ciampaglia, P. Shiralkar, L.M. Rocha, J. Bollen, F. Menczer and A. Flammini, Computational Fact Checking from Knowledge Networks, *PLoS one* **10**(6) (2015), e0128193. doi:https://doi.org/10.1371/journal.pone.0128193.
- [47] P. Shiralkar, A. Flammini, F. Menczer and G.L. Ciampaglia, Finding Streams in Knowledge Graphs to Support Fact Checking, in: *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 859–864. doi:10.1109/ICDM.2017.105.
- [48] A. Ammar and R. Celebi, Fact validation with knowledge graph embeddings, CEUR-WScourws@sunsite.informatik.rwth-aachen.de, 2019, Fact validation in a knowledge graph is a task to determine whether a given fact (subject, predicate, object) should appear in the knowledge graph. In this paper, we have described our approach for the fact validation task in the context of the Semantic Web Challenge 2019. We used embedding features with machine learning to predict facts that were missing from the knowledge graph. The embedding features were generated applying a knowledge graph method known as the RDF2Vec method on the knowledge graph with only positive statements. To improve our machine learning model, we added the test facts that we could validate via the public sources into the positive knowledge graph. We trained a Random Forest classifier on the training data (positive and negative statements) plus the verified test statements and made predictions for test data. ISSN 1613-0073. http://hdl.handle.net/10754/679467.
- [49] U. Joshi and J. Urbani, Ensemble-Based Fact Classification with Knowledge Graph Embeddings, in: *The Semantic Web*, P. Groth, M.-E. Vidal, F. Suchanek, P. Szekley, P. Kapanipathi, C. Pesquita, H. Skaf-Molli and M. Tamper, eds, Springer International Publishing, Cham, 2022, pp. 147–164. ISBN 978-3-031-06981-9.
- [50] T. Schuster, A. Fisch and R. Barzilay, Get Your Vitamin C! Robust Fact Verification with Contrastive Evidence, in: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Online, 2021, pp. 624–643. doi:10.18653/v1/2021.naacl-main.52. https://aclanthology.org/2021.naacl-main.52.
- [51] N. Vo and K. Lee, Where Are the Facts? Searching for Fact-checked Information to Alleviate the Spread of Fake News, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Online, 2020, pp. 7717–7731. doi:10.18653/v1/2020.emnlp-main.621. https://aclanthology.org/2020.emnlp-main.621.
- [52] A. Krajna, M. Kovac, M. Brcic and A. Šarčević, Explainable Artificial Intelligence: An Updated Perspective, in: *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, 2022, pp. 859–864. doi:10.23919/MIPRO55190.2022.9803681.
- [53] H. Yuan, H. Yu, S. Gui and S. Ji, Explainability in Graph Neural Networks: A Taxonomic Survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**(5) (2023), 5782–5799. doi:10.1109/TPAMI.2022.3204236.
- [54] A.M.P. Braşoveanu and R. Andonie, Visualizing and Explaining Language Models, in: *Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery*, B. Kovalerchuk, K. Nazemi, R. Andonie, N. Datia and E. Banissi, eds, Springer International Publishing, Cham, 2022, pp. 213–237. ISBN 978-3-030-93119-3. doi:10.1007/978-3-030-93119-3_8.
- [55] I. Rastas, Y. Ciarán Ryan, I. Tiihonen, M. Qaraei, L. Repo, R. Babbar, E. Mäkelä, M. Tolonen and F. Ginter, 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*, Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 68–77. doi:10.18653/v1/2022.lchange-1.7. https://aclanthology.org/2022.lchange-1.7.
- [56] P. Kumar, A. Singh, P. Kumar and C. Kumar, An Explainable Machine Learning Approach for Definition Extraction, in: *Machine Learning, Image Processing, Network Security and Data Sciences*, A. Bhattacharjee, S.K. Borgohain, B. Soni, G. Verma and X.-Z. Gao, eds, Springer Singapore, Singapore, 2020, pp. 145–155. ISBN 978-981-15-6318-8.
- [57] G. Amaral, O. Rodrigues and E. Simperl, WDV: A Broad Data Verbalisation Dataset Built from Wikidata, in: *The Semantic Web – ISWC 2022*, U. Sattler, A. Hogan, M. Keet, V. Presutti, J.P.A. Almeida, H. Takeda, P. Monnin, G. Pirrò and C. d’Amato, eds, Springer International Publishing, Cham, 2022, pp. 556–574. ISBN 978-3-031-19433-7.
- [58] L.F.R. Ribeiro, M. Schmitt, H. Schütze and I. Gurevych, Investigating Pretrained Language Models for Graph-to-Text Generation, in: *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, Association for Computational Linguistics, Online, 2021, pp. 211–227. doi:10.18653/v1/2021.nlp4convai-1.20. https://aclanthology.org/2021.nlp4convai-1.20.
- [59] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P.J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, *J. Mach. Learn. Res.* **21**(1) (2020).
- [60] C. Gardent, A. Shimorina, S. Narayan and L. Perez-Beltrachini, The WebNLG Challenge: Generating Text from RDF Data, in: *Proceedings of the 10th International Conference on Natural Language Generation*, Association for Computational Linguistics, Santiago de Compostela, Spain, 2017, pp. 124–133. doi:10.18653/v1/W17-3518. https://aclanthology.org/W17-3518.

- [61] I. Montani, M. Honnibal, M. Honnibal, A. Boyd, S.V. Landeghem, H. Peters, P.O. McCann, jim geovedi, J. O'Regan, M. Samsonov, D. de Kok, G. Orosz, M. Blättermann, D. Altinok, M. Kannan, R. Mitsch, S.L. Kristiansen, Edward, L. Miranda, P. Baumgartner, R. Bournhonesque, R. Hudson, E. Bot, Roman, L. Fiedler, R. Daniels, kdarakos, W. Phatthiyaphaibun and Schero1994, explosion/spaCy: v3.6.1: Support for Pydantic v2, find-function CLI and more, Zenodo, 2023. doi:10.5281/zenodo.8225292.
- [62] T. Vogels, O.-E. Ganea and C. Eickhoff, Web2Text: Deep Structured Boilerplate Removal, in: *Advances in Information Retrieval*, G. Pasi, B. Piwowarski, L. Azzopardi and A. Hanbury, eds, Springer International Publishing, Cham, 2018, pp. 167–179. ISBN 978-3-319-76941-7.
- [63] J. Leonhardt, A. Anand and M. Khosla, Boilerplate Removal Using a Neural Sequence Labeling Model, in: *Companion Proceedings of the Web Conference 2020*, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 226–229. ISBN 9781450370240. doi:10.1145/3366424.3383547.
- [64] I. Loshchilov and F. Hutter, Decoupled Weight Decay Regularization, in: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [65] D. Vrandečić, Wikidata: A new platform for collaborative data collection, *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion* (2012), 1063. ISBN 9781450312301. doi:10.1145/2187980.2188242.
- [66] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou and T. Mikolov, FastText.zip: Compressing text classification models, *ArXiv abs/1612.03651* (2016). <https://api.semanticscholar.org/CorpusID:16196524>.
- [67] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, Bag of Tricks for Efficient Text Classification, in: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 427–431. <https://aclanthology.org/E17-2068>.
- [68] J.R. Landis and G.G. Koch, The Measurement of Observer Agreement for Categorical Data, *Biometrics* **33**(1) (1977), 159–174. <http://www.jstor.org/stable/2529310>.
- [69] P.S. Bayerl and K.I. Paul, What Determines Inter-Coder Agreement in Manual Annotations? A Meta-Analytic Investigation, *Computational Linguistics* **37**(4) (2011), 699–725. doi:10.1162/COLI_a_00074. <https://aclanthology.org/J11-4004>.
- [70] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, BLEU: A Method for Automatic Evaluation of Machine Translation, in: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, Association for Computational Linguistics, USA, 2002, pp. 311–318. doi:10.3115/1073083.1073135.
- [71] N. Walter, J. Cohen, R.L. Holbert and Y. Morag, Fact-Checking: A Meta-Analysis of What Works and for Whom, *Political Communication* **37**(3) (2020), 350–375. doi:10.1080/10584609.2019.1668894.

Appendix A. Crowdsourcing Task Designs

Show Instructions

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

Read the following **affirmation**:

G V Raja died in Kullu Valley.

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](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.

Appendix B. WTR Dataset Format

WTR is available at Figshare⁸ 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:

⁸<https://figshare.com/s/df0ec1c233ebd50817f4>

1	* Evidence: The entire textual evidence set collected and being collectively annotated;	1
2	* MTurk IDs: A list of anonymous worker IDs and assignment IDs denoting the crowd workers who provided annotations/votes;	2
3		3
4	* TER Relation: The list of TER stances voted by the workers, where 0 = SUPP, 1 = REF, 2 = NEI, 3 = Not Sure;	4
5		5
6	* 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
7		7
8	* Times: The times in seconds taken by a worker to provide their full annotations;	8
9	* Aggregated TER Relation: The majority-voting aggregation of each individual worker's TER stance annotation;	9
10		10
11	* Sentence-level author annotations: The sentence-level annotation representing the stance of the entire reference towards the triple.	11
12		12
13		13
14		14
15		15
16		16
17		17
18		18
19		19
20		20
21		21
22		22
23		23
24		24
25		25
26		26
27		27
28		28
29		29
30		30
31		31
32		32
33		33
34		34
35		35
36		36
37		37
38		38
39		39
40		40
41		41
42		42
43		43
44		44
45		45
46		46
47		47
48		48
49		49
50		50
51		51