Joshua Cowan

# Automated multi-hop fact checking with NLP

Computer Science Tripos - Part II
Clare College
2021

# Declaration

I, Joshua Cowan of Clare College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

I, Joshua Cowan of Clare College, am content for my dissertation to be made available to the students and staff of the University.

Signed: Joshua Cowan
Date: May, 2021

# Proforma

| | |
|---|---|
| Candidate Number: | **2425D** |
| Project Title: | **Automated multi-hop fact checking with NLP** |
| Examination: | **Computer Science Tripos – Part II, May 2021** |
| Word Count: | **11978 words** [1] |
| Code Line Count: | **902** |
| Project Originator: | **The author** |
| Supervisors: | **Dr Andrew Caines & Zheng Yuan** |

## Original Aims of the Project

I originally set out to devise an end-to-end automated fact-checking system which uses natural language processing, first order logic and information from online news sources to verify unstructured textual claims. This required converting input claims into an abstract meaning representation, converting this into logical formulae, searching for relevant evidence, and deriving a final judgement using an automated theorem prover. I sought to evaluate both classification performance on the labelling of a claim's veracity and the quality of the chains of reasoning which resulted in the given classification label, and to compare these with existing approaches for the selected datasets.

## Work Completed

All success criteria were met or exceeded, with the devised system fully implementing the envisaged pipeline. I achieved performance on the classification of True and False claims competitive with that of existing approaches for the chosen datasets. I demonstrate that explanations derived using formal logical reasoning rival those derived from opaque machine learning features commonplace in existing solutions, and that logical reasoning over the relations expressed in text is a feasible approach which requires neither the tight evidence domain constraints imposed in some existing solutions nor the rich set of input features or existing fact-checking articles required by others.

## Special Difficulties

None.

[1] This word count was computed by `texcount -1 -sum -merge -q -inc -template="hword" main.tex output.bbl > main-words.sum` counting just the five main chapters.

# Contents

# Chapter 1: Introduction

## 1.1 Motivation

The 2013 World Economic Forum (WEF) *Global Risks Landscape* report [1] described the proliferation of misinformation (information which is false) as one of society's 50 greatest challenges. Since then, misinformation has surged [2], with the ensuing damages manifesting in lynchings at the behest of WhatsApp rumours [3], the storming of the United States Capitol following the 2020 Presidential Election [4], and what the World Health Organisation describe as an 'infodemic' inhibiting the fight against COVID-19 [5, 6, 7]. The WEF characterise misinformation as 'digital wildfires' to reflect how it spreads rapidly through hyper-connected media, with efforts to extinguish them often lagging behind. Indeed, the median time between a rumour's propagation and its debunking has been placed at 14 hours [8], with extensive damage having occurred in the interstitial. There is a consequent need for immediate, automatic fact-checking to tackle misinformation before it can proliferate, a task which I have chosen to explore.

## 1.2 Existing Work

Vlachos and Thorne define the task of fact-checking [9] as the assignment of a truth value to a textual claim, and later survey the array of approaches to automated fact checking to define the space from which I propose my approach [10]. Considering inputs to the task, they note that misinformation occurs naturally as unstructured text and that datasets already exist with this form of input, and so I adopt individual textual claims drawn from PolitiFact[1] (a professional fact checking service) as the inputs on which to base my approach.

Turning to methodology, Sharma et al. [11] survey techniques for identifying misinformation, focussing on linguistic cues which Machine Learning (ML) classifiers can use in determining veracity. However, they conclude that whilst linguistic clues can aid classification, "the truthfulness is still ultimately defined by the statements it makes"[11, p 34]. This suggested research direction concurs with the motivations of Ahmadi et al [12], who explain that ML based systems often do not provide evidence to support their judgements, and those that do fail to explain *how* a decision was made from these sources. For a fact-check to be trustworthy, users must understand the rationale behind the reported veracity, and to this end they propose reasoning with formal logic. They subsequently implemented a system which mines a fixed set of rules relating to a claim, before obtaining information from knowledge graphs and the wider web and using Probabilistic Answer Set Programming to

[1] https://www.PolitiFact.com/

derive both a veracity label and a logical derivation. Whilst I have embraced their use of first-order logic as a reasoning tool, the relations they extract focus on detecting logical inconsistencies or verifying ontological relations common to knowledge bases (e.g. date Of Birth, book genre). The claims in PolitiFact are, conversely, not well served by knowledge bases; claims such as 'Democrats spotted stuffing ballot boxes' and its operative predicate *stuffing* are of a different domain to those explored by Ahmadi et al.

## 1.3 Devising an approach

Real-time fact checking thus requires an approach which relies on neither a well-populated knowledge graph nor an existing fact check. Instead, I seek to emulate the behaviour recommended of skeptical individuals: to seek references to the claim in a more reliable source.[2] A reasonable, if imperfect, assumption to draw from this is that **claims which are true are more likely to be reported by reputable sources**. This assumption fails on claims which are true but for which no discussion in reputable media exists (as in the case of many emerging stories), however this is an issue which human fact-checkers also face in the absence of reputable information, and for which there is no immediate solution. Facts which require more complex reasoning to determine their veracity also violate this assumption; the claim that 'Obama had his feet up in the oval office on 9/11' requires knowledge that Obama was not the president in 2001 to disprove it, which I deem out of scope here.

An advantage of working with this assumption is that the *meaning* of claims requires minimal consideration. To verify that 'Hillary supports pro-choice', meaning must only be understood such that on receipt of the contextualised evidence 'Clinton donated to pro-choice causes', 'Clinton' can be co-resolved with 'Hillary' and 'supports' can be understood as expressing a similar intent to 'donates'. This is the key insight which facilitates my use of reputable news sources as an evidence source.

Under this assumption, and in light of this body of existing work, I have devised an automated fact-checking system which takes a candidate unstructured textual claim and outputs both a veracity label and a supporting chain of reasoning. In doing so, I have explored whether logical reasoning based systems are well-suited to the task of automated fact-checking by evaluating if they can effectively:

- Classify a claim by its veracity?
- Produce human-intelligible chains of reasoning in support of the assigned veracity?

I consider these with respect to existing approaches which exploit linguistic or social media based features, and conclude that a logical-inference driven approach to automated fact-checking is both feasible and performs competitively in classification and explanation generation, whilst also requiring far fewer input features than existing systems.

[2] https://abcnews.go.com/US/ways-spot-disinformation-social-media-feeds/story?id=67784438

# Chapter 2: Preparation

## 2.1 Starting Point

Previous courses in 'Logic & Proof' and 'Prolog' have familiarised me with automated proof systems and first-order logic, with 'Formal Models of Language' providing a baseline understanding in Natural Languages. I have prior experience with SpaCy[1] and Information Extraction with Python, however no previous work has been included in this project. All code written is consequently my own and written during the project, with the standard exception of imported packages and models being used according to their respective licences.

## 2.2 Project Refinement

Preparation following the proposal submission resulted in some minor changes to the project's structure. Entity linking functionality was pared back after exploratory analysis indicated that the news-oriented domain of the data meant few of the claims would benefit from online knowledge graph information. The extension tasks designed to alleviate the expected performance bottleneck (in the semantic role labeller, proposed extension criteria C3 & C4) were replaced with optimisation of argument coreference resolution metrics after it became apparent that this was where the performance bottleneck actually resided.

[1] https://spacy.io/

## 2.3 Requirements Analysis & Evaluation Metrics

As a result of the described adjustments, the evaluation criteria were finalised as follows:

**C1a** System successfully extracts logical formulae from the claim and derives a judgement.
**C1b** Given the novel approach, better than 50% accuracy on classification of the veracity of claims in a label-balanced subset of the PolitiHop dataset.
**C2** The quality of natural language chains of reasoning attached to the veracity claims are compared with those made in gold-standard (PolitiHop), with quantitative summarisation of comparative performance.

Criterion C1b refers to the arithmetic mean of the accuracies for the True and False classes, i.e. accuracy were the number of ground True and ground False claims to be equal in number. A third class (half-true) is dropped from this metric for reasons detailed below. Whilst criterion **C1a** specifies the overall functionality of the project, I also establish more granular functionality requirements by adapting from the milestones offered in the proposal's work-plan:

**R1** Natural Language relations are extracted from claims.
**R2** Claims are converted to an abstract meaning representation (AMR).
**R3** AMRs are converted to logical formulae and placed in a knowledge base.
**R4** Relevant evidence is collected and processed into the knowledge base.
**R5** A veracity label is determined and returned alongside corresponding evidence.

### 2.3.1 Datasets & Evaluation metrics

Both PolitiHop [13] and FakeNewsNet [14, 15, 16], the datasets I have chosen to evaluate performance on, derive from PolitiFact. Both provide candidate claims plus an attribution (to any natural or legal person, or anonymous/social media claims), alongside a veracity judgement mapped from PolitiFact's five-label scale to a coarser set of labels (Table 2.2). PolitiHop is a new dataset which focuses specifically on multi-'hop' reasoning, where more than a single piece of evidence is required to verify a claim, whereas FakeNewsNet claims mostly require a single piece of evidence. FakeNewsNet is a more widely used dataset, with existing approaches covering a range of methods. Evaluating performance on this dataset thus contextualises my system's performance within the wider space of fact-checking systems and enables comparison with these other approaches, as set out in the research questions.

| | **True** | **False** | **Half-True** |
|---|---|---|---|
| **Polithop-train** | 42 | 179 | 17 |
| **PolitiHop-test** | 29 | 157 | 14 |
| **FakeNewsNet** | 432 | 524 | 0 |

Table 2.1: Dataset Splits for counts of PolitiFact claims.

| **PolitiFact label** | **PolitiHop label** | **FakeNewsNet label** |
|---|---|---|
| True, Mostly True | True | True |
| Pants-on-fire, False, Barely True/Mostly False | False | False |
| Half-True | Half-True | |

Table 2.2: Label Mapping

The PolitiHop baseline model constrains the domain of evidence sources to solely the PolitiFact fact-check article associated with the claim in question. To enable comparative evaluation, I offer performance under this constraint (named 'closed-domain'), but also offer a separate evaluation with an evidence domain constrained to 'reputable sources' (named 'open-domain'), as this more closely reflects the demands of a real-time fact-checking system. The success criterion **C2** requires a consideration of the quality of derived chains of reasoning with respect to those in PolitiHop, and so I will also evaluate whether the derived chains of reasoning correctly entail the veracity classification label before informally comparing the quality of explanation provided with that in the PolitiHop baseline.

Given the imbalance in the datasets, I initially proposed the use of balanced accuracy as a metric. I will now additionally offer Macro-F1 for each dataset and domain, for these are the reported statistics for the models which I seek to compare performance with. I also offer weighted-F1 as a more representative metric in light of the infrequency of the half-true class. These are defined as follows:

$$F1\text{-}Score = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

$$\text{and Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

$$Macro\text{-}F1 = \frac{1}{N} \sum_{i=0}^{N} F1\text{-}Score_i$$

Where with N classes, class i = 'Positive', and all other classes collectively are 'Negative'.

$$Weighted\text{-}F1 = \sum_{i=0}^{N} F1\text{-}Score_i \; P(i)$$

As with Macro-F1, and with $P(i)$ being the proportion of all examples $\in i$.

PolitiFact's classification system introduces further complexity by using the 'Half-True' label to refer to claims whose overall veracity is more complex than a true or false judgement, or as Thorne and Vlachos [10] remark, have a veracity dependent on the how the claim is interpreted. Considering this alongside the difficulty of distinguishing an undiscussed claim from a false one, I decided not to address half-true labels within the project, and these are included only to enable comparative evaluation. Table 2.3 summarises the evaluation metrics.

| Evaluation Criteria | Existing approaches | Input Source | Evidence Domain | Evaluation Metrics |
|---|---|---|---|---|
| PolitiHop (Open-domain) | N/A | PolitiFact Claim | Reputable News Sources | F1 (Macro & per class), Accuracy on class-balanced subset |
| PolitiHop (Closed-domain) | PolitiHop (model) | | PolitiFact article | |
| FakeNewsNet (Open-domain) | SVM baseline, RST, SAF/A | | Reputable News Sources | F1 (Macro & Weighted) |

Table 2.3: Summary of evaluation metrics

## 2.4 Relevant Theory

### 2.4.1 Semantic Role Labelling & PropBank

To exploit formal methods of reasoning with natural language, unstructured information embedded within text must first be transformed into a structured form. This is the definition of *Information Extraction* [17, p.759], of which relation extraction —the task of identifying and classifying semantic relations between entities —is a key subtask. Traditional (*closed*) information extraction entails extracting relations which fit a defined schema of relations, such as specifying genealogical relations and attempting to extract a family tree from the text. A more general form is *Open* Information Extraction (OIE) [18], where the relation itself must also be extracted. This obviates the need to preemptively extract or manually specify the sought relations, thus facilitating more generalisable fact-checking systems.

A simple form of OIE entails extracting subject-verb-object triples, such as converting 'The boy sells lemonade every Tuesday' to `(boy, sells, lemonade)`. The expressiveness of such triples is limited, however; the temporal modifier 'every Tuesday' prevents it from being naturally captured by a triple. More generally, *Semantic Role Labelling* is the task of automatically finding the constituent arguments for each predicate (verb) in a sentence [17, p.705], and to determine their semantic relation (role) to the predicate. The definition of 'semantic role' varies based on the schema used, and can include thematic roles [17, p.655], or alternatively can refer to one of a set of predicate-specific roles, which is the method used by *PropBank* [19, 20, 21]. In the PropBank scheme, extracted relations consist of a central predicate (the verb) plus a set of numbered arguments, plus as any modifiers (phrases which qualify the sense of another word).

| | |
|---|---|
| ARG0 | agent |
| ARG1 | patient |
| ARG2 | instrument, benefactive, attribute |
| ARG3 | starting point, benefactive, attribute |
| ARG4 | ending point |
| ARGM | modifier |

Figure 2.1: Generic semantic roles in Propbank [22]

```
Sense:  Purchase.01
Roles:
Arg0:  purchaser
Arg1:  thing purchased
Arg2:  seller
Arg3:  price paid
Arg4:  benefactive
```

Figure 2.2: Exemplar sense-specific semantic roles in Propbank [22]

The set of semantic roles are generic (Figure 2.1) but their interpretations (and how they are assigned) vary with the *verb sense* which they are associated with. A (word) sense is a representation of one aspect of the meaning of a word [17, p.646], and disambiguates individual meanings from an overloaded orthographic representation. For example, two senses of the word *pen* are 'a writing instrument', and 'a fenced area where animals live'. The canonical example with verbs is 'they can fish', in which 'can' has the sense 'are permitted to' and the sense 'put item in a can'. Each verb sense has an associated *frame* in PropBank, which states the specific role that the constituent arguments play (Figure 2.2). Using verb frames reduces *lexical* ambiguity, which is especially chronic in some natural language processing applications where there is a loss of the context which usually disambiguates a word's meaning —*'I'll meet you by the bank'* has a different default

interpretation for city-dwellers than for those living in a rural riverside community.

*The Principle of Compositionality* asserts that an expression's meaning is composed of its constituents and the rules relating them. Taking noun phrases as constituents and semantic roles as relational rules produces an *Abstract Meaning representation* (AMR), a directed acyclic graph in which multiple relations are *composed* into a graphical representation of canonical semantics. In this regard, claims which express the same meaning with differing syntax should map to the same AMR [23].

A further consideration is whether modifiers are *logically conjunctive*. For example, 'The Democrats tried to impeach the president illegally' has the `manner` modifier which is integral to the overall meaning and is thus conjunctive. To establish the claim's veracity, the 'illegal' manner must also be established. Conversely, 'John likes strawberries too' has the less meaningful `discourse` modifier ('too'). Treating this as conjunctive would require incoming evidence to also fulfill this modifier, and so 'John likes strawberries' would be deemed insufficient evidence.

#### 2.4.1.1 Coreference resolution & Anaphora

*Anaphora* is the use of an expression (*anaphor*) which refers to a prior expression (*the antecedent*). In 'Bob went shopping. He bought some eggs', 'Bob' is the antecedent and 'He' the anaphor. In an information extraction sense, understanding that it was Bob who purchased the eggs is critical, but without an understanding of anaphora the egg-purchasing would be unhelpfully attributed to the (nonexistent) 'He'. More generally, coreference resolution (abbreviated as co-resolution) is the task of finding all expressions that refer to the same entity. This can include less explicit cases such as 'Coronavirus hit Europe in March. The virus has so far....' where 'the virus' is the anaphor.

### 2.4.2 Similarity metrics

#### 2.4.2.1 Wordnet & Verb relations

*Wordnet*[24, 25] is a lexical database of English words which groups word senses into *synsets*, with each representing a single concept. It also details hierarchical semantic *relations* between noun senses, and between verb senses. For verbs, these relations are:

| | |
|---|---|
| Hypernym | X is a hypernym of Y if Y is a form of X<br>e.g. cook is a hypernym of grill |
| Entailment | X entails Y<br>e.g. snoring entails sleeping |

| | |
|---|---|
| Troponym / Hyponym | X is a Troponym of Y if Y is a form of X<br>e.g. grill is a troponym of cook |
| Verb Group | Similar terms e.g. gobble and picnic; both are troponym's of eat, but the first is related to a manner of eating, and the second the eating of a specific meal. |

Table 2.4: Sourced from [24], [26]

This hierarchy facilitates the distance between any two senses to be calculated [17, p.687], with $\frac{1}{1+\text{distance}}$ then forming a similarity metric.

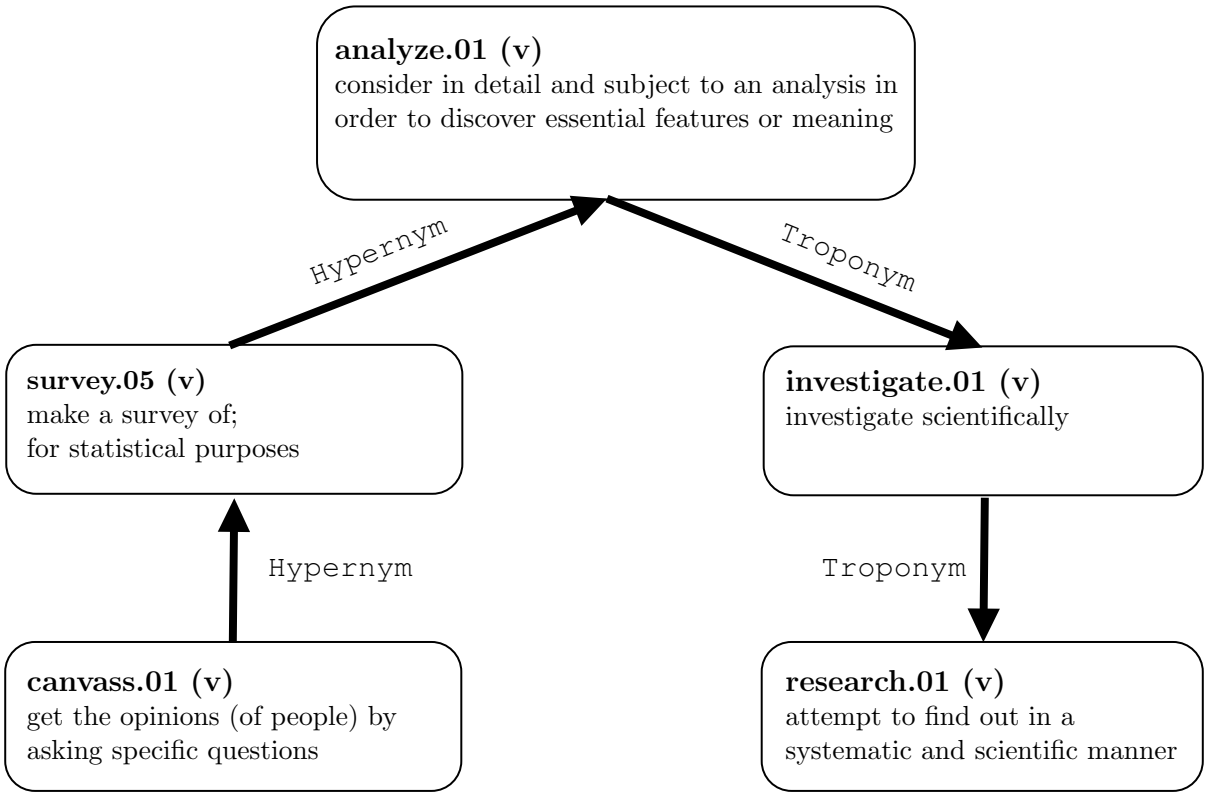



Figure 2.3: Path through hierarchy between two similar verbs. Path similarity = $\frac{1}{1+4}$

#### 2.4.2.2 Levenshtein distance

Levenshtein/edit distance captures how similar two strings are as a count of how many 'edits' are required to change one string into the other, with an 'edit' defined as either the insertion of a character, the deletion of a character, or the replacement of a character[27].

With two strings a and b, provided both are not empty:

$$\mathsf{levenshtein\_distance}(a,b) = \mathsf{lev}_{a,b}(|a|,|b|)$$

$$\mathsf{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(\mathsf{i},\mathsf{j}) = 0, \\ \min \begin{cases} \mathsf{lev}_{a,b}(i-1,j)+1 \\ \mathsf{lev}_{a,b}(i,j-1)+1 \\ \mathsf{lev}_{a,b}(i-1,j-1)+1 \end{cases} & \text{otherwise} \end{cases}$$

The above approach is defined *recursively*, however large speedups can be achieved with an iterative dynamic-programming approach [28], resulting in an $\mathcal{O}(mn)$ time complexity for two strings of length $m$ and $n$. A normalised similarity metric can then be formed as:

$$1 \; - \; \frac{levenshtein\_distance(a,b)}{max(length(a), length(b))}$$

#### 2.4.2.3 Longest Common Substring (LCS)

The longest common substring problem is defined as finding the longest string which is common to both input strings; for example, the LCS of 'elephants' and 'phantoms' is 'phant'. [2] By using a suffix tree, the length of the LCS can be calculated in $\Theta(m+n)$ time [29], and it can be converted to a similarity metric as with Levenshtein distance.

#### 2.4.2.4 Cosine Similarity between Word Vectors

Individual words can be represented by their properties using *word vectors*. GloVe [30] is based on learning multi-dimensional vectors which represent words by their co-occurences (how often words appear in the same context), as trained over large corpora. These vectors can be spatially interpreted, with the cosine of the angle between them reflecting their proximity in the vector space and thus their similarity based on the trained features.
This is *cosine similarity*, which is formally defined between two vectors **A** and **B** as:

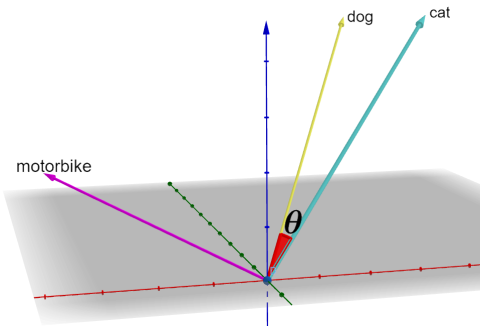



Figure 2.4: simplified spatial representation of word vectors

$$\text{Cosine similarity} = cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|A\| \, \|B\|}$$

A cosine similarity of 1 indicates perfect similarity, 0 indicates no relation, and -1 indicates perfect dissimilarity. Figure 2.4 depicts an example where **A** represents 'dog' and **B** represents 'cat'. Notably, using co-occurences means that 'cat' and 'dog' will be marked as similar, as the two tend to appear in similar contexts despite being distinct, or in some contexts opposing, concepts. The adjectives 'slow' and 'fast' are similarly affected.

[2] This is distinct from the longest common sub*sequence* problem, where constituent characters need not be consecutive (and so 'phant**s**' would instead be found).

## 2.5 Technical preparation & Tool selection

Before starting the main body of the project, some preparatory work was required.

### 2.5.1 Tool selection

Python was the obvious choice for an implementation language given the wealth of NLP tools and models implemented in/compatible with it. I also chose Python to exploit its object-oriented paradigm to promote information organisation and utilise the provided encapsulation. I have had prior experience with SpaCy, an open-source Natural Language Processing framework designed for mass processing. SpaCy's `en_core_web_lg` model is their large English model which has been pre-trained on web data and has superlative performance in the fundamental tasks of tokenization, vectorization (using GloVE), dependency parsing, Part-of-speech tagging & Named Entity Recognition. This allows me to focus on the more novel aspects of the project. SpaCy also has a rich ecosystem of further tools and models, such as neuralcoref [3], which adds coreference resolution capabilities to the SpaCy pipeline. I also made use of existing implementations for calculating Levenshtein distance & longest common substrings, choosing `textdistance`[4] and `pylcs`[5] after informally profiling various solutions' speed. I also utilised the extensive real-world knowledge provided by Wordnet[24] and Wikidata[6], in order to handle entity coreference resolution.

The current state-of-the-art model for Semantic Role Labelling (for English PropBank on Newswire sentences) is Shi & Lin [31], who present a BiLSTM built on top of BERT. This model also implemented verb sense disambiguation —the task of assigning a verb sense to the predicate. I initially used AllenNLP's[7] implementation, however as this did not implement the verb sense disambiguation aspect I switched to the *transformers-srl*[8] implementation which quoted similar performance and *also* performed verb sense disambiguation. As this model was still within the AllenNLP model framework, I could harness GPU compute via PyTorch and CUDA.

Similar research was conducted on potential automatic theorem provers. Whilst I considered Vampire[9] and Isabelle[10], I chose to use Prover9[11] as it was the only one natively supported by the framework in which I had undertaken preparatory work studying natural language understanding (NLTK). First order logic was chosen as it strikes a suitable balance between expressiveness and simplicity. Natural language predicates are conveniently captured by logical predicates, and semantic roles are captured as naturally from the semantic roles attributed to predicate's frame.

[3] https://github.com/huggingface/neuralcoref
[4] https://pypi.org/project/textdistance/
[5] https://pypi.org/project/pylcs/
[6] https://www.wikidata.org/
[7] https://allennlp.org/
[8] https://github.com/Riccorl/transformer-srl
[9] https://vprover.github.io/
[10] https://www.cl.cam.ac.uk/research/hvg/Isabelle/
[11] https://www.cs.unm.edu/ mccune/prover9

To develop abstract meaning representations, I considered existing solutions such as `amrlib`[12], however this provided too low a level of abstraction for the applications I required, with this superfluous decomposition introducing errors which were not present in the abstract meaning representation system I then developed. Creating this formed a significant portion of the project, and required extensive use of networkX[13] and Graphviz (for visualisation) [14].

For the web-scraping aspect, I researched potential search engines to power the evidence collection queries. Bing allowed searches to be constrained to an unlimited number of domains (i.e. the 'reputable news sources')[15] whereas Google enforced a maximum of 10 domains[16]. Both provided sufficient free usage provided I cached results appropriately, but to distribute usage I opted for a mix of APIs. I had also considered using news-aggregation APIs, however these had a significantly lower coverage at a higher cost.

General data management was conducted with `pandas`[17], with `newspaper`[18] and `BeautifulSoup`[19] used for news article parsing. `Pickle`[20] was used to cache data from the web and `NLTK`'s interface with WordNet [24] was used to implement verb sense distance.

### 2.5.2 Skills preparation

Prior to commencing programming, I spent time studying abstract meaning representation and the methods to treat natural language verbs as predicates, as well as PropBank and semantic role labelling generally. Usually this began with reading *Speech and Language Processing* [17], before moving on to more specific sources such as *Automatic labeling of semantic roles* [32]. I also obtained working knowledge of Graphviz & dot in order to produce visual representations of abstract meaning representations.

## 2.6 Professional Practice & Engineering approach

### 2.6.1 Data management, Licensing & Ethics approval

Due to potential processing of personally identifiable (albeit public) data, I sought and obtained Ethics Committee clearance from the department. All data ultimately used was publicly available, and so fewer measures were required to guard the security of the input data. Numerous packages and models were utilised to great effect, with all licensed under one of the permissive BSD, Apache 2.0 or MIT licences, thus facilitating any further open source development or adoption by organisations averse to copyleft licences.

[12] https://github.com/bjascob/amrlib
[13] https://networkx.org/
[14] https://github.com/xflr6/graphviz
[15] https://docs.microsoft.com/en-us/bing/search-apis/bing-custom-search/overview
[16] https://developers.google.com/custom-search/v1/overview
[17] https://pandas.pydata.org/
[18] https://github.com/codelucas/newspaper
[19] https://www.crummy.com/software/BeautifulSoup/
[20] https://docs.python.org/3/library/pickle.html

Whilst the project makes use of datasets which have been chiefly used to devise Machine Learning models, and the project itself makes use of ML models, the project is not devising such a model. However, the standard practice of separating test and development/training data was observed given that much tuning occurred with respect to the PolitiHop training set and there was thus a risk of 'overfitting'. The PolitiHop test set was consequently kept separate for use only in the final evaluation. FakeNewsNet was not used for any training/tuning, but given its size (and my limited search API credit) I evaluated performance on a randomly stratified 25% subset of the overall dataset (obtained with `pandas.Series.sample`, using a fixed seed for reproducibility). Whilst PolitiHop and FakeNewsNet are both derived from PolitiFact, there was no overlap between claims in the PolitiHop training set and the FakeNewsNet sample.

### 2.6.2 Reputable Sources & Ethical Web Crawling

Defining a 'reputable' source is an ill-posed task beyond the scope of this project. In the narrowest definition, that which I initially proposed, only newswire sources would be classed as 'reputable'. To facilitate a higher chance of proving a claim, however, I elected to use sources listed as 'reliable' by Wikipedia's *Perennial sources* index[21]. Whilst Wikipedia's facility for user-edited content can render it unreliable, its 'editor community' adhere to stricter journalistic standards [22] such that I am confident in trusting the listed websites, at least at the level of this project. To promote simplicity I removed some of the more infrequent/niche sources from this list[23], leaving the final 'reputable source' list which is listed in the appendix.

There exist both ethical and legal issues around web-scraping [33, 34]. I have consequently sought to observe 'scraping etiquette' by using an undisguised user-agent string and caching all incoming data (where permitted) to prevent any undue burden being placed on the scraped services.

### 2.6.3 Developmental approach, Source Control and Hardware

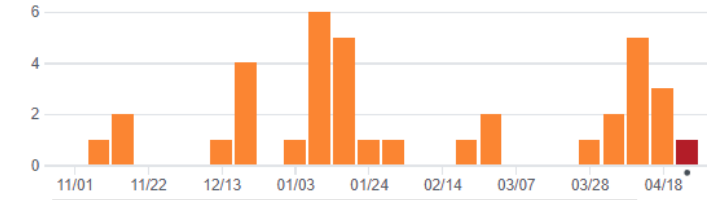
The modular nature of the project prompted the use of an iterative model of development, which facilitated each module to be developed, be subjected to unit testing, and then be integrated with the existing modules. When a module was tentatively completed, development would begin on the next module, with the expectation that past iterations tackling previous modules may be revisited should requirements of later modules demand such. This did indeed occur, when it became apparent that the NLP pipeline required a filter to prevent expensive components being run on evidence which was already discardable.

My focus throughout was optimising in line with my research questions, prioritising classification performance and quality of yielded explanations over time and space efficiency. However, I still sought to maximise efficiency where not at a significant expense of performance. For example, I chose a non-ensemble model for Semantic Role Labelling despite the best ensemble option [35] potentially having delivered a marginally better performance.

[21] https://en.wikipedia.org/wiki/Wikipedia:Reliable_sources/Perennial_sources
[22] https://en.wikipedia.org/wiki/Wikipedia:Editorial_oversight_and_control
[23] Such as Burke's Peerage, which covers the genealogy of British nobility.

Turning to source control, Git integrates well with my IDE of choice (PyCharm [24]), and its branch management was used to great effect to develop two simultaneous approaches when a design decision wasn't possible until further along in the development process. I used GitHub for remote repository management, and made hourly backups to my personal OneDrive account as well as weekly backups to an external hard-disk.

I spent a significant period of the project working from home with a constrained internet connection, and so remotely accessing the departmental cluster was impractical; I thus worked entirely from my own laptop.



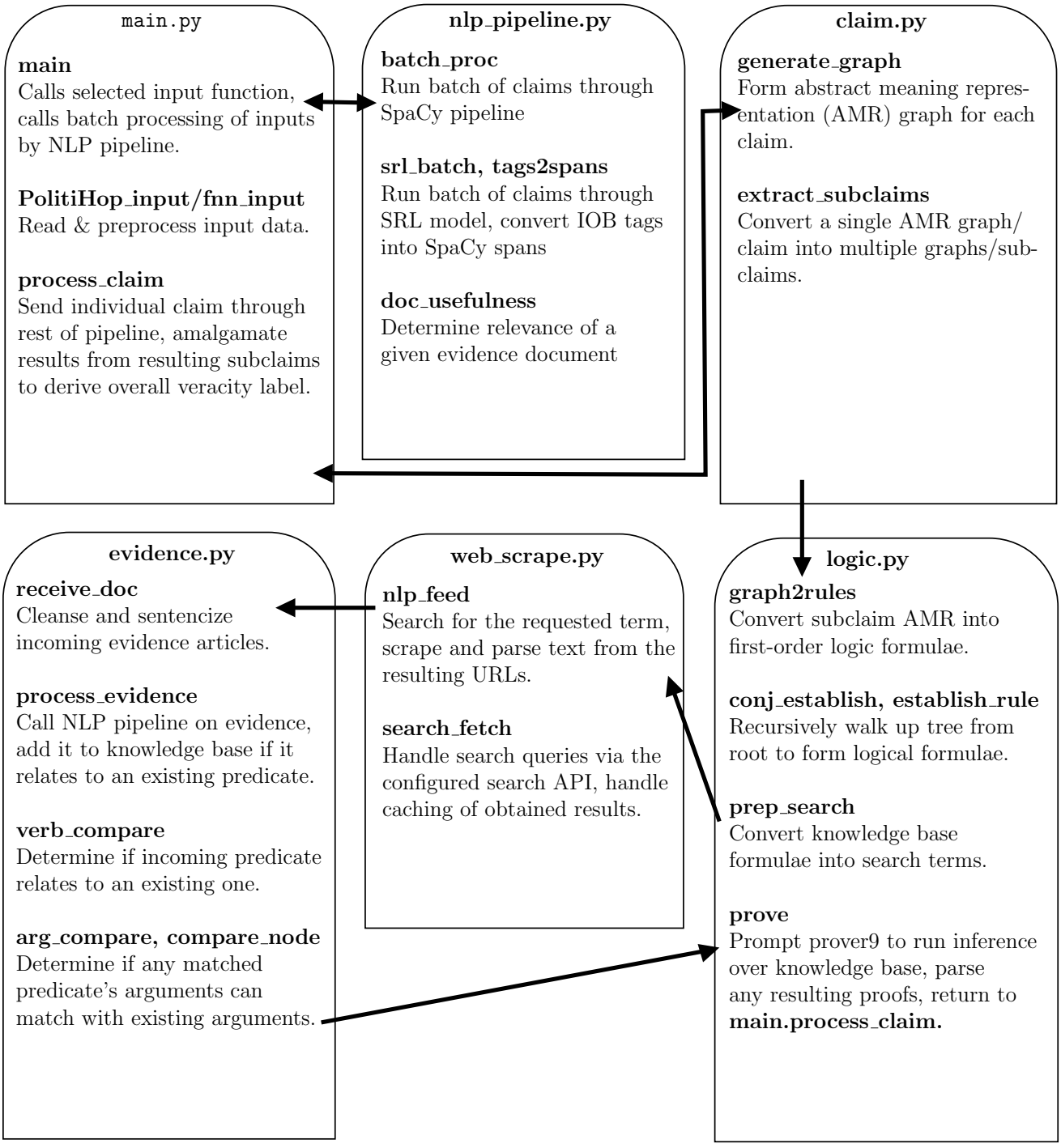

| | |
|---|---|
| Switched SRL to batching with allenNLP only (runs faster accordin | 22/04/2021 16:20 |
| Final tuning for testing, still need to cleanup logic.py and evidence | 21/04/2021 12:26 |
| Cleaning up 3 | 18/04/2021 15:30 |
| Cleaning up 2 | 13/04/2021 21:58 |
| Cleaning up | 13/04/2021 14:35 |
| Mostly improved performance | 12/04/2021 16:34 |
| Version A deviation - newer form of compareNoun which has low | 12/04/2021 13:25 |
| Further tuned noun comparison and migrat origin & Version_B | 12/04/2021 13:21 |
| Further tuned noun comparison and migrated to a batch system t | 09/04/2021 19:54 |
| Improved performance by tweaking noun matching parameters, i | 07/04/2021 01:19 |
| Chain of reasoning functionality added. Added LiarLiar support and | 03/04/2021 00:45 |
| Rewritten nounmatch | 02/03/2021 00:04 |
| rewriting nounmatch | 28/02/2021 16:28 |
| 90% completing the loop - handled UVI equivalence & Noun resol | 22/02/2021 23:21 |
| Intermediate commit to redact API keys. | 31/01/2021 00:28 |
| Devising evidence handling part of pipeline - developed two alter | 27/01/2021 16:45 |
| Update gitignore, cleanup imports on web origin & refactoring | 21/01/2021 23:51 |
| Switched coreference model to neuralcoref, added bing search su | 21/01/2021 23:45 |
| Caching on webcrawl.py implemented, Flair+AllenNLP SRL model | 21/01/2021 14:33 |
| V1 - first attempt at complete run, - latest use of AllenNLP for SRL. | 17/01/2021 23:04 |
| V1 - first attempt at complete run, - latest use of AllenNLP for SRL. | 17/01/2021 23:04 |
| Implemented web-crawling functionality, added documentation. | 16/01/2021 17:55 |
| Implemented web-crawling functionality, added documentation. | 16/01/2021 17:53 |
| Merge logic.formatArg into the sanitization done within claim so that | 13/01/2021 17:38 |
| Refactoring for conceptual simplicity. | 13/01/2021 14:52 |
| Documentation | 13/01/2021 13:40 |
| Added GPU support for AllenNLP, coreference handling for knowledg | 13/01/2021 12:04 |
| Implemented tree -> Predicate extraction and created knowledg | 05/01/2021 23:22 |
| Implemented networkx foundations - splitting up big document-l | 24/12/2020 00:36 |
| Connectives attached to graph - post-handling duplicate subsume | 22/12/2020 14:27 |
| Connectives attached to graph - pre-attempt to handle duplicate | 22/12/2020 14:07 |
| Implemented connective detection - still need to conect to main | 21/12/2020 23:44 |
| Post AMR-enriched representation. | 14/12/2020 01:11 |
| Added coreference resolution and implemented graph-output to | 21/11/2020 17:35 |
| First Commit. Shell of 'claim' and 'Top level claim' classes written, | 17/11/2020 14:04 |
| Initial commit | 08/11/2020 17:47 |

Figure 2.5: Git commit log and commit activity summary.

[24] https://www.jetbrains.com/pycharm/

# Chapter 3: Implementation

## 3.1 Repository Overview



### 3.1.1 Pipeline Structure

The entirety of the refined work-plan was completed, barring some ancillary features detailed later. Excluding imported packages and models, all 900 LOC are written by myself, of which a significant proportion tackles deriving abstract meaning representations, converting these to logical forms, and working on coreference resolution for incoming evidence.
This chapter will detail the functionality of the project's constituent modules, however the overall procedure for checking a batch of input claims is as follows:

1. The input claims are read in and pre-processed.
2. The claims are passed as a batch through the extended SpaCy pipeline. The semantic role labeller then extracts relations from the text.
3. The extracted relations are converted to the custom abstract meaning representation, forming a 'claim graph'.
4. Each claim/claim graph is split into subclaims/subgraphs.

   Then, for each subclaim:

   (a) The devised meaning representation is collapsed into first order logic rules which are used to populate a subclaim-specific knowledge base.

   (b) The search engine API is called with search terms extracted from the input claim, and the returned URLs/articles are scraped and parsed into raw text.

   (c) The raw text is pre-processed and passed through a modified version of the customised SpaCy pipeline.

   (d) The predicates and arguments forming the incoming evidence documents are compared with the existing formulae in the subgraph knowledge base. Where incoming predicates are found to be be similar to an existing predicate, they are mapped to the existing predicate.

   (e) The arguments associated with those incoming relations with mapped predicates are similarly resolved with arguments of the corresponding formula in the knowledge base. If the incoming arguments have resolved with at least 2/3 of the existing formula's arguments, then a copy of the antecedent for the rule on which the match occurred is added to the knowledge base.

   (f) The prover attempts to establish the subclaim from the knowledge-base, returning a parsed derivation if it has (otherwise returning False).

5. After repeating for all subclaims, if any one subclaim has been verified, then the overall claim is deemed to have been verified and is thus classified as True.

### 3.1.2 Modified functionality

Some features were initially planned, and in some cases implemented, before being dropped for performance reasons. These included:

- **Entity Linking**. After being pared back during the preparation/refinement stage, it was later reintroduced for use in entity co-resolution, as to understand that 'the U.S.' and 'the United States of America' are synonymous, a source of external knowledge is required to find the common canonical form —I opted for WikiData owing to its expansive coverage.
- **Handling logical connectives.** Logical connectives (such as 'or', 'xor', 'if-then'/'implies') allow meanings to be composed in more complex manners [36], and so I devised functionality to represent natural language connectives accordingly. This was later dropped after it became apparent that the infrequency with which it provided benefit was outweighed by the severe overheads it added to the pipeline.
- **Aspects of multi-hop functionality.** I initially planned to iterate between collecting evidence and attempting to derive a proof, until one was found or until no new evidence was collected. However, I determined that very little evidence arrived after the first evidence collection phase, even after refining the search parameters based on any advancement of the proof following the previous round of evidence. The multi-hop aspect in PolitiHop is still achievable, however, by composing evidence to match the composition of the meaning representation.

## 3.2 Input and Pre-processing

Data input and pre-processing is handled on a per-dataset basis. PolitiFact report their claims as 'Said that 80% of prisoners in America are from Latin America', and list 'Donald Trump' in the 'author' field. Pre-processing for PolitiHop-derived datasets consequently entails pushing in the author to form the statement 'Donald Trump said that 80%...' as to capture the claim in its entirety. This does not apply in cases where the author is a 'Facebook post' or 'Viral image' due to the difficulty associated with attributing quotations to them. Ground truth is stored for evaluation, and the pre-processed claims are delivered to the NLP-extraction module.

## 3.3 NLP-Extraction Module

Sequestered to `nlpPipeline`, this module receives a batch of raw claims and outputs a corresponding batch of enriched SpaCy documents. SpaCy works around a central pipeline (`Language`) instantiated from a pre-trained model, which I then extended with further functionality to furnish the documents with the requisite information. The final pipeline, addressed in order, is as follows:

**Tokenizer:** SpaCy's English model contains a non-destructive tokenizer which forms a `Doc` object to store the derived `Token` list. This forms the central representation of the textual form of the claim. Slices of `Doc`s are called `Span`s, which in turn can have their own slice-specific properties, such as an named entity type label.

**Part-of-speech (POS) tagging, Dependency Parsing & Named Entity Recognition:** SpaCy's model handles these fundamental tasks, with their outputs required for downstream tasks such as extracting search terms, detecting semantically empty phrases, and resolving coreferences.

**Coreference Resolution:** Neuralcoref's pre-trained model extracts clusters of coreferences where the key is the canonical reference, and values are `Span`s which corefer to them, including anaphora. Initially, I used AllenNLP's coreference resolution model due to environment incompatibilities with my development setup, however a small work-around overcame this and I was able to switch to neuralcoref, which was better integrated with SpaCy's pipelining. Whilst revisiting the coreference resolution component during a later development cycle in order to make the switch, I took the opportunity to greatly simplify the way that coreferences are registered on their corresponding spans by better utilising native SpaCy functionality.

**Semantic Role Labelling:** Transformer-SRL is built within AllenNLP's architecture, and consequently isn't natively supported by SpaCy's. Whilst SpaCy does allow the creation of custom pipeline components, I ascertained that collating documents at the end of the SpaCy pipeline and then batch processing them with AllenNLP improved efficiency. According to cProfile, this was a roughly 10% speed increase in processing the hundreds of input claims (using a batch size of 100), but a 50% speed increase in processing the thousands of evidence documents, due to the overheads of marshalling the batches being increasingly neglible compared to the speed increase from the batching itself. I also wrote functionality to handle this marshalling and to parse the SRL model's output (IOB tags) into `Span`s (via `nlp_pipeline.tags2spans`).

## 3.4 Natural Language Understanding (NLU) Module

### 3.4.1 Building an Abstract Meaning Representation

After passing through the NLP pipeline, each input claim is assigned a newly instantiated `Doc_claim` object which relates the `Doc` with its abstract meaning representation graph.

To retain maximum control over how claims are represented, I implemented a custom abstract meaning representation system in which verbs, arguments and modifiers become nodes, with the relations (i.e. their semantic roles) forming links. Figure 3.1 is an example of the meaning representation output by the NLU module, with more complex examples presented in the appendix.

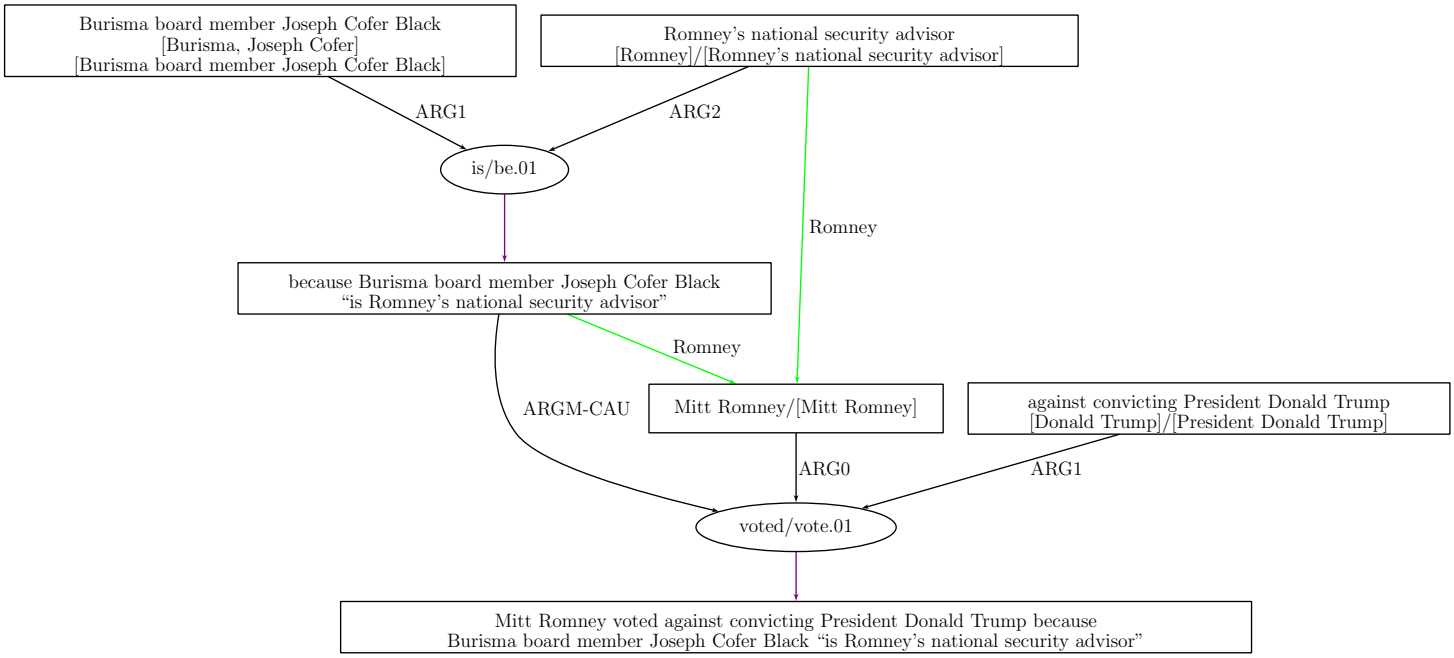



Figure 3.1: Example AMR. Purple links have *interior nodes* as their sinks, green edges represent coreferences. Text in brackets is [Entities][Noun Chunks].

Further to this, I define an *interior* argument as one which itself contains further relations and can thus be decomposed (such as 'because the dog hated cats'), with the converse being a *leaf* argument (such as 'the dog'). At this stage, modifiers are treated as arguments. Purple/implication edges represent relations to these interior arguments from the predicates of the relations they encompass, and are interpreted as 'the fulfilment of every encompassed relation implies the encompassing interior argument'. The practical purpose of this is to allow for recursive decomposition of a claim, such as in Figure 3.1 where there are two levels to the claim, one tackling Romney's vote, and another describing Joseph Black's occupation. The semantic role labeller extracts different relations for these two verbs, which are then connected with an implication edge.

Each individual 'node' (i.e. predicate or argument) consists of both a unique ID for use by networkx and logical reasoning modules, as well as its existing textual representation (`span`),

which are both properties of an `argNode` object instantiated upon the first calculation of its ID. Algorithm 1 presents the overall procedure for forming abstract meaning representations, with the corresponding python code reproduced in the appendix:

**Algorithm 1** Form abstract meaning representation

**Inputs:**
    $SRL\_rels$ = list of extracted relations,
    entire_doc = node representing entire claim
**Initialize:**
    $graph \leftarrow$ Graph(), $list\_of\_verbs \leftarrow []$, $list\_of\_args \leftarrow []$

**for** $rel$ **in** $SRL\_rels$ **do**
    $root \leftarrow rel$['verb']
    $list\_of\_verbs$.append(root)

    **for** $arg\_type, arg\_value$ **in** $rel$ **do**
        **if** $arg\_type \neq$ 'verb' **then**
            $graph$.add_node(ID = hash($arg\_value$))
            ▷ Add the black is-an-argument edge:
            $graph$.add_edge(from=$arg\_value$, to=$root$, label=$arg\_type$)
            $list\_of\_args$.append($arg\_val$)
        **end if**
    **end for**
**end for**

▷ Join up components by adding implication edges from their roots to their corresponding interior arguments.
**for** $verb$ **in** $list\_of\_verbs$ **do**
    $shortest\_encl\_span \leftarrow entire\_doc$
    **for** $arg$ **in** $list\_of\_args$ **do**
        **if** $verb$ is contained within $arg$ and $|arg| < |shortest\_encl\_span|$ **then**
            $shortest\_encl\_span \leftarrow arg$
        **end if**
    **end for**
    $graph$.add_node(ID = hash($shortest\_encl\_span$))
    $graph$.add_edge(from=$verb$, to=$shortest\_encl\_span$, label=$arg\_type$)
**end for**

### 3.4.2 Splitting into subclaims (`extract_subclaims`)

Each claim (such as that in Figure 3.2) is typically comprised of multiple subclaims with each expressing a different proposition. Splitting a claim into subclaims therefore allows each aspect of the claim to be focussed on separately. A subclaim is graphically defined as a tree rooted at a parent of the overall claim's root, as establishing the conjunction of these subclaims implies the overall claim is proven. This splitting process is depicted in Figure 3.2. Inaccuracies in the Semantic Role Labeller output sometimes introduced errant edges to the meaning representation —this was mostly edges outgoing from verbs (as opposed to arguments, which should be incoming edges), which were easy to systematically remove.

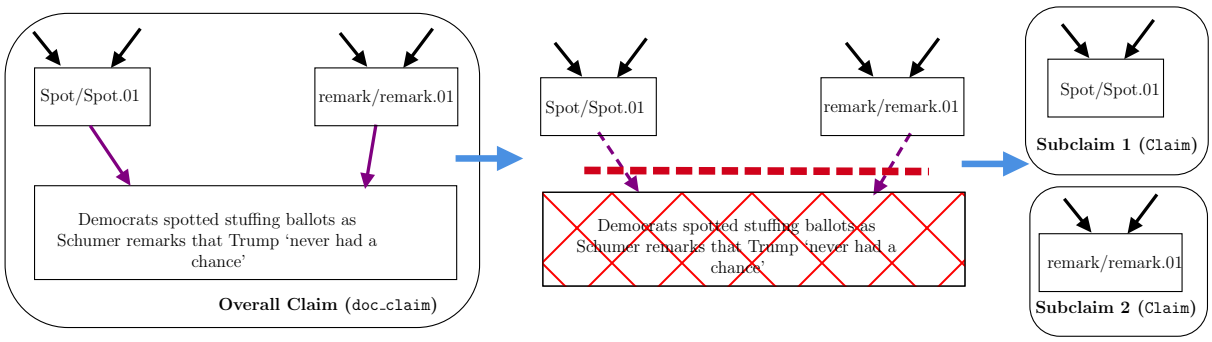



Figure 3.2: `extract_subclaims` converts one claim graph into multiple subclaims by removing the root and instantiating a subclaim for each remaining component. Co-reference links are ignored here as they could wrongly connect two otherwise disconnected components.

## 3.5 Logical Inference Module I —Devising Rules

Having obtained a meaning representation, the pipeline then sets up the knowledge base for each subclaim and populates it with rules extracted from the meaning representation. Adding rules to the knowledge base entails walking up the subclaim tree from the obtained root, parsing natural language predicates (verbs), their arguments, and their modifiers into first-order logic formulae, as shown in Algorithm 2. `IS_MEANINGFUL` filters out semantically weak arguments by label (e.g. `discourse`) or by value (e.g. `adverbs` such as 'just') to ensure that they do not wrongly become a conjunctive argument, as discussed in section 2.4.1. Once the rules are derived and processed into the knowledge base, search terms are extracted and passed back to `process_claim`, which prompts the web-scraper to search for evidence.

**Algorithm 2** Convert abstract meaning representation to FOL formulae.

▷ Starting at the subclaim's root, call conj_establish(root.incoming_neighbours)

**function** CONJ_ESTABLISH(nodes) ▷ input the group of nodes to establish
  **return** $\bigcap_{n \in nodes}$ ESTABLISH_RULE(n)
**end function**

**function** ESTABLISH_RULE(node)
  **Initialize:**
  $negate\_predicate \leftarrow$ False, $formula\_args \leftarrow []$, $modifiers \leftarrow []$

  ▷ Distinguish between leaf/interior and modifier/numbered argument:
  **for** $edge(x \rightarrow node)$ **in** $node.incoming\_edges$ **do** ▷ For all edges from 'x' to 'node'
    **if** NUMBEREDARGUMENT(x) **then** ▷ If x's label is ARG1, ARG2 ...
      **if** INTERIORARGUMENT(x) **then** ▷ If x roots a subtree
        $placeholder\_var$ = GET_PLACEHOLDER( )
        $knowledge\_base$.new_rule(antecedent=ESTABLISH_RULE(x),
                                   consequent= (x = $placeholder\_var$))
        $formula\_args$.add($placeholder\_var$)
      **else**
        $formula\_args$.add($x$)
      **end if**

    **else**
      **if** $modifier.semantic\_role$ = 'NEG' **then**
        $negate\_predicate$ =True
      **else**
        **if** IS_MEANINGFUL($x.value$) **then**
          $modifiers$.add($x$)
        **end if**
      **end if**
    **end if**
  **end for**

  ▷ After sorting through the arguments, can form the formula
  $formula$ =FORM_FORMULA(predicate=$verb$, arguments=$formula\_args$)

  ▷ Then register negations and modifiers
  **if** $negate\_formula$ = True **then**
    $formula = \neg formula$
  **end if**
  **for** $modifier$ **in** $modifiers$ **do**
    $formula = formula \cap modifier$
  **end for**
  **return** $formula$
**end function**

## 3.6 Web-scraper & Evidence selection

### 3.6.1 Web-scraper: Searching and Scraping (`web_scrape.py`)

Having established the rules which need to be verified, the web-scraper seeks relevant evidence by passing the extracted search terms to the chosen search API. The returned queries are cached and then passed to a scraper (`newspaper` and/or `BeautifulSoup`) which extracts the article's body of text and passes it on to `evidence.py` for processing.

The parameters of the search conducted correspond to a configuration identifier in the search API; in the closed-domain case only the corresponding PolitiFact article is returned, and in the open-domain case the top $n$ relevant articles from the 'reliable news source' domains are returned (with $n = 5$ as default). I primarily used the Bing Custom Search API, which was configured with an online console and interfaced with by a RESTful web service, and I made extensive use of caching as to abide by good scraping practices and minimize incurred search API costs.

The main issue faced here, under open-domain constraints, was encountering claim-specific fact-checking articles from non-PolitiFact sources, which I then filtered out by dropping incoming URLs which mention fact-checks. The theoretical justification for this is ambiguous, as arguably such a system should have the option of utilising all available information. However, as operation under the open-domain constraints is proposed as a more real-time solution, It would be best for the open-domain task to be evaluated under conditions where claims have *not* already been fact-checked.

### 3.6.2 Processing the encountered evidence (`evidence.py`)

The evidence obtained from the web-scraper is validated before being partitioned by a standalone SpaCy sentencizer. These sentences each become a separate document and then pass through the first (and coarsest) of evidence filters (`evidence.receive_doc`, which rejects any documents so long that they would prompt excessive overheads during downstream processing, or which simply repeat the claim in question. The remaining documents then pass through the SpaCy pipeline. At this stage, the majority of remaining documents will still be irrelevant, and so a finer layer of filtration drops document with no entities resembling those in the knowledge base[1] prior to entering the relatively slow semantic role labeller.

Evidence which is deemed relevant is then passed through the semantic role labeller before being compared with the existing knowledge base population in an attempt to equate incoming evidence with existing formulae. This resolving of similar predicates/arguments together is motivated by the concept of multiple notions mapping to one canonical form/meaning [17, p.583]. Essentially, without this resolution then the knowledge base would be filled with relations which *effectively* entail those that constitute the subclaim's rules, but cannot be logically resolved together due to the blindness of the inference system to semantics. For example, if a rule states that selling(bob, apples) $\implies$ greengrocer(bob), then without similarity resolution the evidence vending(bob, apples) $\not\Longrightarrow$ $greengrocer(bob)$;

[1] 'resemblance' is defined here as clearing a low threshold of syntactic similarity.

the antecedent cannot be fulfilled as 'selling' and 'vending' are different predicates and thus cannot be resolved together under the rules of FOL binary resolution [37, p.381]. The aspect of the codebase with access to *semantics* must determine that vending *equals* selling, and inform the theorem prover of this. Two possible approaches to this are to add a rule that says vending(bob,apples) $\implies$ sells(bob,apples), but it's simpler to just determine vending = selling and add sells(bob,apples) as evidence when seeing vending(bob,apples) as evidence. A significant amount of time was required to tune the method used to determine whether two predicates/arguments should be resolved together, and this formed the largest amount of unexpected work in the project overall.

### 3.6.3 Matching predicates (`evidence.verb_match`)

I initially planned to determine predicate (verb) similarity solely by Wordnet path similarity, but it became apparent that this resulted in low sensitivity. Investigations initially pointed to failure to handle auxiliary verbs (as in 'he is sleeping) and *light verbs* such that 'Steve presented a...' would erroneously fail to co-resolve with that of 'Steve gave a presentation about..'. The semantic role labeller does (in theory) label such verbs appropriately, but appears to do so with low sensitivity which in turn explains the low sensitivity for verb resolution. To tackle this, I used `SentiWordnet` [38] to assign a sentiment score to Wordnet senses, with a match now also being recorded if two verbs each expressed either a positive sentiment *or* one of the most frequent verbs which formed light or auxillary verbs —'do', 'be' and 'have'. This resulted in the intended increase in sensitivity and overcame the described issues, albeit to the detriment of specificity; this is counteracted somewhat by the noun matching system which exhibits the reverse behaviour. As the annotation scheme used by Propbank differs from that of Wordnet, I compiled a dictionary (`pb2wn.json`) to convert between the two via joining Wordnet to ontonotes and ontonotes to Propbank using the mappings from the Unified Verb Index[2].

### 3.6.4 *Matching* arguments (`evidence.noun_match`)

Those relations which have verbs/predicates that have matched those already in the knowledge base are passed to the argument matcher, which iterates through the arguments belonging to the matched evidence predicates and attempts to resolve them with the arguments of the corresponding knowledge base formula (`evidence.node_compare`). If more than 2/3 of the corresponding knowledge base formula's arguments[3] have been resolved with arguments of the incoming evidence predicate, then the evidence is converted to a logical formula and added to the knowledge base (thus satisfying the antecedent of the knowledge base rule whose formula was matched with).

My initial argument matcher implementation used a combination of edit distance and cosine similarity, however this yielded ostensibly poor performance with seemingly unrelated arguments being co-resolved. I consequently took a classification problem style approach to attempt to distinguish similar pairs from dissimilar ones, and manually labelled a set of 1000 pairs of entities/noun chunks before running them through a range of syntactic text-similarity metrics. I obtained the optimal threshold for each metric by using sklearn

[2]https://uvi.colorado.edu/

[3]The optimal threshold according to experimental testing.

to plot ROC, before comparing both the accuracies they achieved on this mini dataset and their AUC. These experiments indicated that longest common substring would most strongly discriminate (syntactically) between the two classes, with the results of these experiments listed in the appendix.

This difficulty of this task is best illustrated with examples. 'Face masks' and 'face coverings' are not identical, but are deemed sufficiently similar in most cases to be co-resolved. Conversely, Donald Trump and Donald Knuth are both entities with a syntactic similarity resembling that of the masks/coverings example, but are entirely different people/nouns and so should not be co-resolved. 'He' and 'Castro' will not be equated by any metric naturally, but with the context of 'He' being an anaphor for 'Fidel Castro' then the two are in fact similar. There were also issues related to the longer arguments associated with predicates nearer the root of the meaning representation tree. "The democrats" would match with one argument in "Trump said 'The democrats have stolen the election from us'", but the overall argument should not co-resolve as there is no reference to election-stealing in the incoming evidence. Separately, *stop words* —very common words which add little *lexical* meaning to a sentence, such as 'was' and 'the' in 'the blue car was fast' —can interfere with vector-based similarity metrics by adding noise to the average of the `span`'s word vectors [39, p.27].
These individual issues were tackled with an ensemble of approaches to classifying argument similarity:

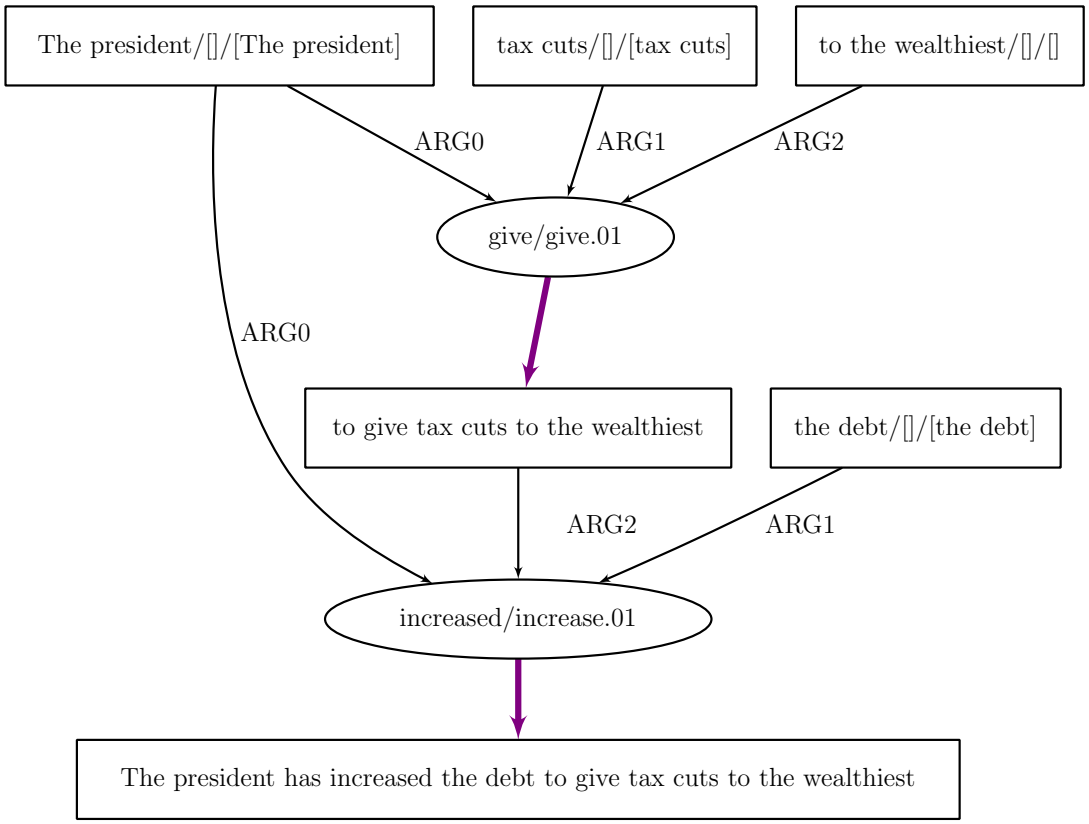
- Cosine similarity: To compare word embeddings, I initially used SpaCy's `similarity` function which finds the average word embedding for each input span by averaging over the embeddings of **every** constituent token in the span, and then returning the cosine similarity between each span's average. However, it became evident that stop-words were adding noise to this averaging process, and so I wrote a custom `Span` similarity function (`compute_similarity`) which averages only over non-stop words, before returning the cosine distance between the two average vectors.
- Number comparison (`num_compare`): To compare numbers only, any textual components (if present) are converted to numerical form (e.g. £4 million → £4,000,000), before returning True if $0.7 * \text{existing} \geq \text{incoming} \geq 1.3 * \text{existing}$. As this function is required in conjunction with other comparisons (i.e. it isn't enough to pass on this alone), it returns True if any non-numerical entities are passed in.
- Entity Linking: World knowledge is sometimes required to co-resolve two nouns, such as 'the USA' and 'the United States'. Syntactic measures fail due to the two being syntactically dissimilar, and to lower the word embedding similarity threshold to the point where these would be deemed equivalent would also cause 'Japan' to be deemed equivalent as it appears in similar contexts to 'the USA' (but clearly is not equivalent entity).
- Longest Common Substring: The strings to be compared are lower-cased and the longest common substring is then found. Initially this was done with difflib, but pylcs exhibited a consistent performance improvement which was critical given the frequency with which it is called. The length of this substring is then normalized, with True returned if this normalised similarity clears the threshold that I set after tuning with scikit-learn ( 0.37).

These measures were combined by taking every incoming entity and existing entity pair and evaluating on a subset of the above metrics. Where the number of **existing** entities

which an incoming entity has matched with exceeds the number of existing entities which have not been matched with by the incoming entities, then the two spans are deemed to have matched. The same is also repeated with noun chunks. Furthermore, a match can be derived where two overall spans are very similar syntactically (via a very high longest common substring) or semantically (via word embeddings, and only if neither span contains named entities as to avoid 'Japan' and 'USA' being equated). The python code implementing this is listed in the appendix.

## 3.7 Logical Inference Module II —Proving

Having added evidence to the knowledge base (or having found none), inference is run by prover9 on the knowledge base to see if truth can be determined. The rules added to the knowledge base consist only of ground formulae to ensure that only the specified values can unify with the rule's antecedent (rather than any variables eligible for unification). Placeholder variables are used in the case where a predicate contains an interior argument (see 3.4.1), such that if the encompassed relations are established then the encompassing argument is assigned its true value which then allows it to resolve with another piece of evidence fulfilling the rest of the predicate's required arguments. This is illustrated by the following example:




Logical Formulae:
1. Give(president, tax cuts, wealthiest few) $\implies$ increase_arg2 = to give tax cuts to wealthiest)
2. Increase(president, debt, to give tax cuts to wealthiest) $\implies$ argF(root)

Figure 3.3: Exemplar generated meaning representation & corresponding logical formulae. argF(root) is the special symbol representing the overall subclaim

In Figure 3.3, formula 1 depicts how the interior argument 'to give tax cuts to the wealthiest' is implied by the encompassed 'give' relation. Evidence establishing the 'give' relation would cause the placeholder increase_arg2 to be equated with its true value. Then, on receipt of the evidence Increase(president, debt, increase_arg2), the antecedent of formula 2 can be fulfilled. This is because increase_arg2 can now unify with "to give tax cuts to wealthiest" due to the equality implied by fulfilment of the 'give' relation.

If a derivation has been found, prover9's output is parsed into a readable derivation via a backtracking function. If no evidence has been added during the evidence collection phase, no proof can be found, or a contradiction is found, then a verdict of False will be returned. The result and derivation are then returned to `main.process_doc` where the results for each constituent `Claim` in the `Doc_claim` are collated and an overall truth value is determined. These final steps entailed some decisions which appear counter-intuitive when contrasted with human-lead fact-checking approaches:

Firstly, that where a claim cannot be proven, it is assumed to be False. I initially attempted to prove the claim in question and then attempt to prove its negation if no proof could be found, returning 'half-true' if neither could be proven. As established in the preparation chapter, however, this would be an erroneous interpretation of the 'half-true' label. Furthermore, it would mean that claims which are made but not discussed elsewhere (*because* they are baseless) would be marked as half-true. Switching to assuming all documents are false unless the claim is proven resulted in a significant boost in performance owing to a sharp drop in the number of False claims labelled as Half-True.

Second is the function mapping the judgement for the $n$ subclaims to the overall claim's judgement. A human-lead fact-check would require that for an overall claim to hold, every constituent subclaim must also hold. Taking this approach, however, would mean that issues of low specificity would have a multiplicative effect on overall performance. Experimentation indicated that performance is maximised when declaring the overall claim to be true when *any* of the subclaims are true, in direct opposition to the theoretical approach. There are real world consequences here; the first subclaim in 'Pelosi wants to remove Trump and install a serial killer', is true, but the second is false in a way that should prohibit an overall True verdict being returned. The infrequency of such claims meant this had little impact on performance, but it could be explored in future work.

### 3.7.1 Optimisation

Extensive use of cProfile enabled quick identification of bottlenecks, with the aim of reducing runtimes as far as possible without significantly reducing classification performance. This identified three culprits causing excessive run-times: the SRL model (which took about half of the total runtime), queries to Wikidata, and then edit distance calculations. I addressed these by introducing batching to the SRL model evaluation, more aggressive caching for queries made to Wikidata, and running trials to establish more efficient implementations of edit distance, although this was eventually partially replaced with optical string alignment. Further optimisation was achieved by introducing increasingly fine filtering of incoming evidence, thus cutting the amount of overall work required by minimising processing done on irrelevant documents.

# Chapter 4: Evaluation

## 4.1 Review of Project Requirements

✓**C1a** System successfully extracts logical formulae from the claim and derives a judgement.
✓**C1b** Given the novel approach, better than 50% accuracy on classification of veracity of claims in a label-balanced subset of the PolitiHop dataset.
✓**C2** The quality of natural language chains of reasoning attached to the veracity claims are compared with those made in gold-standard (PolitiHop), with quantitative summarisation of comparative performance.

Returning to the two research questions, I sought to establish how logical-inference based automated fact checking performed in terms of the classification performance in establishing veracity (criterion C1) and in producing useful explanations to support the assigned veracity class (criterion C2). Criterion C1 is split further into C1a, which specifies a general requirement for functionality (as elaborated upon in the requirements analysis), with C1b specifying the metrics by which it is *formally* evaluated. All five requirements listed in the requirements analysis were fulfilled, with this success being evidenced by overall classification/explanation performance. I also provide more informal evidence to indicate *how* the individual requirements were successfully met.

#### 4.1.0.1 R1 Natural Language relations are extracted from claims & R2 Claims are converted to an abstract meaning representation.

The abstract meaning representations are the central achievement of the project. The balance required between expressiveness and simplicity allows for claims to be fully represented but not *over-specified* as to discard evidence which approximately verifies it. To further improve this balance, a claim can be verified by satisfying the meaning representation at any level. In Figure 4.1, the graph upwards from the 'stop' predicate need not be verified should the argument 'to stop him from buying NBC' be established atomically. However, for more complex claims which do require decomposition then individual components can be proven and their results composed to derive a final judgement.

These meaning representations capture co-references (green edges), the nested nature of predicate-argument structures (purple edges), and the various modifiers —whether logically conjunctive (the purpose modifier in 4.1), semantically empty, or interior nodes. Negation is handled, as are claims which are composed of numerous subclaims, as well as claims which are more complex than a simple subject-verb-object representation would allow for and claims whose predicate's orthographic form is overloaded. The meaning

representation thus serves as a good point from which further work with logical inference could be developed.

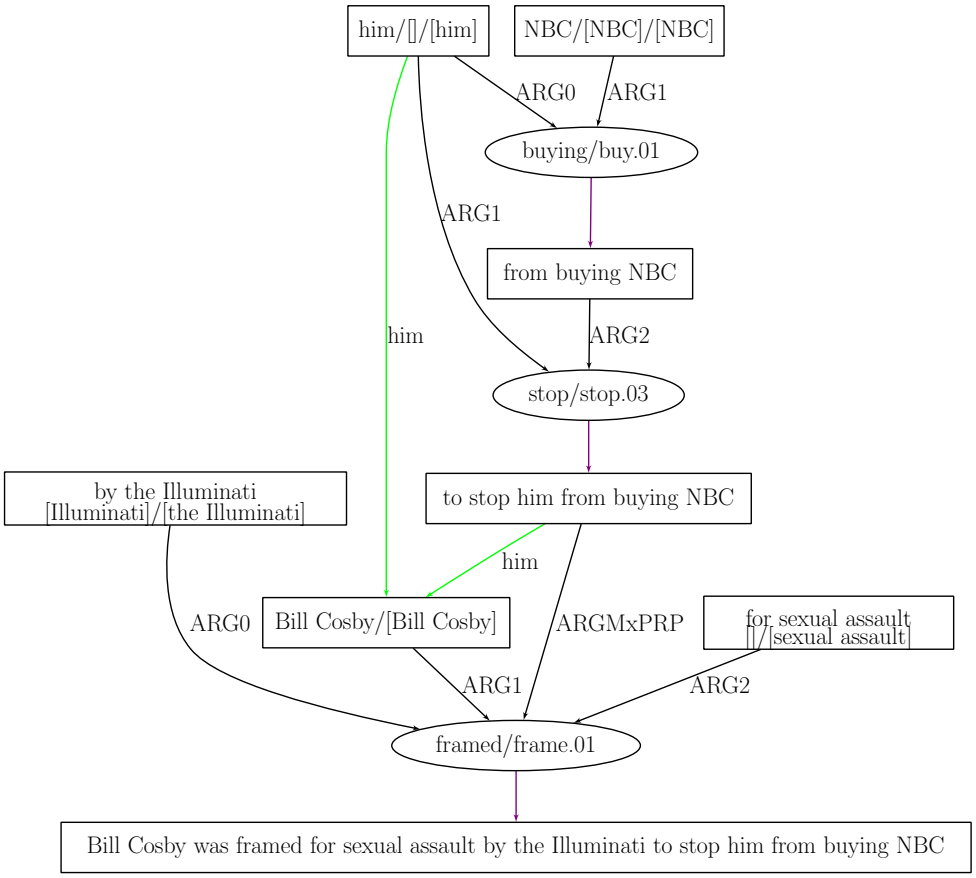



Figure 4.1: a

#### 4.1.0.2 R3: Meaning representations are converted to logical formulae and put in a knowledge base.

This was also successful, with logical formulae being produced ready to be handled by prover9. The above example would map to:

```
buying(him,NBC) ⟹ (stop_arg1 = frombuyingNBC)
stop(him,frombuyingNBC) ⟹ PRPframed(tostophimfrombuyingNBC)
framed(bytheIlluminati,BillCosby,forsexualassault) &
PRPframed(tostophimfrombuyingNBC)) ⟹ argF(root)
```

#### 4.1.0.3 R4 Relevant evidence is collected and processed into the knowledge base. & R5 A judgement is determined and the evidence associated with the decision made is returned.

The received evidence is output as part of a derivation should the claim be verified, and logical proofs (i.e. prover9 output) can easily be output (an example is in the appendix).

## 4.2 Classification results

| Evaluation Criteria | Existing approaches | Input Source | Evidence Domain | Evaluation Metrics |
|---|---|---|---|---|
| PolitiHop (Open-domain) | N/A | PolitiFact Claim | Reputable News Sources | F1 (Macro & per class), Accuracy on class-balanced subset |
| PolitiHop (Closed-domain) | PolitiHop (model) | | PolitiFact article | |
| FakeNewsNet (Open-domain) | SVM baseline, RST, SAF/A | | Reputable News Sources | F1 (Macro & Weighted) |

Table 4.1: Summary of evaluation metrics (repeated from page 5)

### 4.2.1 PolitiHop: Closed-domain

I first compare performance with the baseline PolitiHop models [13]. All models in Table 4.2 were 'closed-domain', and so only received the claim, the author of the claim, and the PolitiFact fact-checking article as input; for this project, that meant the claim and author were inputs and the fact-checking article was used as the sole source of evidence. The PolitiHop baseline models were: a naive bayes model with 2-gram and 3-gram TF-IDF scores (word vectors based on a word's relevance); a Transformer model with article sentences encoded by base-weighted BERT (PolitiHop/BERT), and the same but with a Transformer-XH model which permits information sharing between the article's sentences in the hope of providing 'multi-hop' functionality.

| 3-Class | Validation | | Test | |
|---|---|---|---|---|
| | F1 (Macro) | Accuracy (Unweighted) | F1 (Macro) | Accuracy (Unweighted) |
| *Random* | *34.1* | *38.5* | *24.2* | *27.7* |
| *TF-IDF* | *34.4* | *69.5* | *34* | *76* |
| PolitiHop/ Transformer-XH | 61.1 | 76.6 | 43.3 | 75.5 |
| PolitiHop/ BERT | 54.7 | 69.5 | 44.8 | 76 |
| This project (PolitiHop closed domain) | 39.4 | 66.1 | 32.2 | 58.0 |

Table 4.2: Performance on the 3-class label veracity classification problem under the 'closed-domain' parameters, compared with the baseline models in PolitiHop [13].

The F1-macro score of 39.4 achieved by this project is evidently unremarkable compared with either of the two BERT based models; however this is qualified with a key caveat. For the reasons discussed in Section 2.2, I decided against supporting predicting the half-true class, and so only ever predicted one of the True or False classes. This directly explains the significantly worse macro-F1 performance, as whilst the low proportion of half-true classes has very little impact on the accuracy, the presence of a third class means that the denominator in the macro-F1 class calculation is 3 rather than 2, which leads to significantly lower F1 scores compared with the PolitiHop baselines which were designed for three classes. It follows that as weighted-F1 accounts for the low prevalence of

half-true, it is hence a more representative —if less comparable —measure of performance for this project specifically. This is a natural consequence of the method I've chosen; models which are powered by natural language inference can introduce this half-true class without necessarily having to consider the subjective or partial veracity of the actual claims being made, and this is why the BERT-based PolitiHop baselines perform better overall. However, whilst this serves as useful comparison to existing approaches, it is not where this project excels.

### 4.2.2 PolitiHop: Open-domain

The open-domain parameters specified that the claim and author would be provided as input, with the evidence domain being any URL from one of the 'reliable news source' domains. This disparity in input variables renders comparison with PolitiHop's baselines inappropriate, however I still provide comparison with this project's performance under the closed-domain parameters for reference (Table 4.3).

| 3 Class | | **F1 (Macro)** | **F1 (Weighted)** | **Accuracy (Unweighted)** | **Accuracy (C1b) (50/50 T/F only)** |
|---|---|---|---|---|---|
| Mine (PolitiHop Open-domain) | **Train** | 40.2 | 68.6 | 71.8 | 60.5 |
| | **Validation** | 34.4 | 61.7 | 67.8 | 54.3 |
| | **Test** | 38.6 | 70.8 | 73.0 | 60.6 |
| Mine (PolitiHop Closed-domain) | **Train** | 34.2 | 63.0 | 64.3 | 53.7 |
| | **Validation** | 39.4 | 63.1 | 66.1 | 63.2 |
| | **Test** | 32.2 | 60.6 | 58.0 | 53.8 |

Table 4.3: Comparison of open-domain and closed-domain performance

Given the aforementioned dropping of the half-true class, the evaluation criterion C1b is concerned only with accuracy from the 'True' and 'False' classes, and specifies that the arithmetic mean of the two classes accuracies must exceed 50%. As evident in Table 4.3, this is attained by divisions of the data under both sets of evaluation parameters, and thus the criterion is fulfilled. The successful derivation of a (meaningful) judgement in conjunction with the intermediary data representations in section 4.1 means that criterion C1a has also been attained. The F1 scores presented for the test and training splits of the open-domain parameters exceed those achieved on the closed-domain parameters, and I explore some reasons for this in the error analysis section.
As PolitiHop is a less established dataset, I also evaluated performance on *FakeNewsNet* [16], which provides PolitiFact claims alongside their authors and a rich set of social media interactions with the claim in question.

### 4.2.3 FakeNewsNet: Open-domain

Evaluation on the FakeNewsNet dataset facilitates the comparison of differing approaches to misinformation classification. This includes the baseline models provided alongside FakeNewsNet [16]. Both SAF/A (social article fusion; features which capture temporal patterns of user engagement with the claim's source) and RST (which used rhetorical structure theory to capture the style of discourse) perform similarly to the logical reasoning system I proposed. dEFEND [40] achieves the current state-of-the-art results for

| 2-class | Features/Method | Test | | | | |
|---|---|---|---|---|---|---|
| | | F1 (False claims) | F1 (Macro) | F1 (Weighted) | Accuracy (Unweighted) | Accuracy 50/50 T/F |
| SVM [16] | Text contents only (vectorised) | 65.9 | N/A | N/A | 58.0 | N/A |
| SAF/A [16] | User engagements, Social Context | 61.9 | | | 66.7 | |
| RST[40] | Stylistic cues, Discourse Structure | 56.9 | | | 60.7 | |
| dEFEND[40] | All of the above | 60.7 | | | 90.4 | |
| This project (Open domain) | Logical Reasoning | 61.3 | 55.2 | 54.1 | 56.1 | 60.5 |

Table 4.4: Performance on the PolitiFact set of FakeNewsNet. Results for different approaches to FakeNewsNet classification are shown to enable comparison of methods. SVM and SAF/A results sourced from [16], RST & dEFEND from [40].

classification performance, and uses a recurrent neural network to encode the PolitiFact article sentences and user comments responding to the source of the claim in question on the source's website or on social media. dEFEND performed significantly better than all other systems, but required a full spectrum of rich input features. It also performed evidence selection by using a co-attention mechanism to capture the 'explainability' of news sentences. Conversely, the system I have devised requires only the claim itself to be explicitly entered (providing the author is also helpful but not strictly required). The domains of evidence differ also, with this project (on the open domain) not requiring an existing fact-check but benefiting from the wider realm of information available on the web. The explanations formed by this project also offer a citation as to enable manual verification, in accordance with the demand from fact-checking communities to provide *helpful tools* rather than a simple veracity label based on unreliable evidence or opaque linguistic features [41].

The general patterns of performance, with respect to my initial research questions, is similar across all experiments. That is that logical inference is less adept at deriving a classification label than those based on linguistic features, social context, or natural language inference, due mostly to the lower sensitivity across all tasks. This is fundamentally rooted in the mantra that if evidence cannot be found to support a claim, then the claim is marked as false, which in turn comes from the focus on providing adequate explanations. Whereas the other presented solutions can label claims as true without having supporting evidence, the reverse is true of my system, which requires the presence of supporting evidence to label a claim as True. Error analysis, therefore, explores some of the potential reasons why claims were difficult to empirically prove, and suggest ways these issues could be ameliorated.

## 4.3 Error analysis

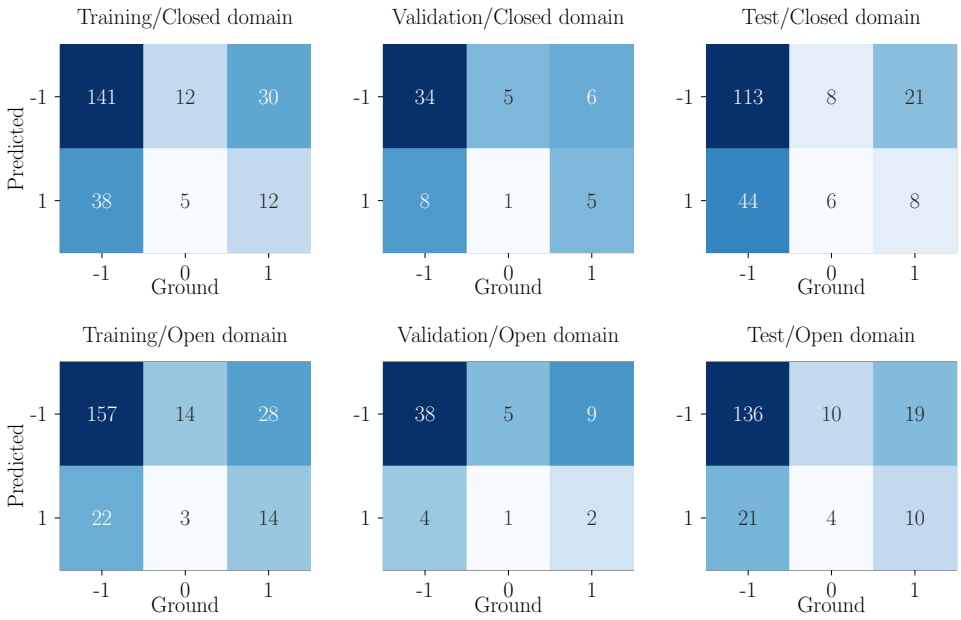



Figure 4.1: Confusion matrices for classification on PolitiHop across all three sets and under both evidence parameters. -1 is false, 1 is true, and 0 is half-true.

### 4.3.1 Evidence Selection

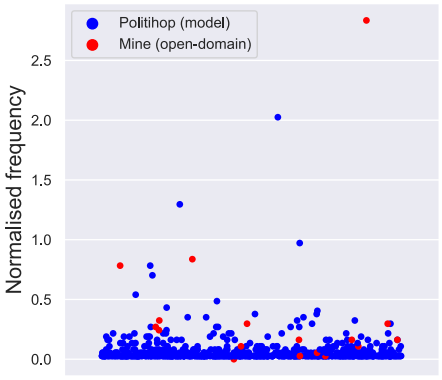



Figure 4.2: Frequency of web-domains used as evidence by PolitiHop versus this model. Normalised by total number of unique domains used by each model.

Considering those 'true' claims wrongly labelled as 'false' in the PolitiHop open-domain training set, roughly 73% can be attributed to a failure to collect relevant evidence. That is, that the evidence collected did not contain the information required to verify the claim. This does not mean that the claim *would* have been verified were the correct data collected (as similar performance on the closed-domain training set shows), but that this possibility couldn't be evaluated as the evidence was not present in the data. This group can be subdivided into those caused by an improperly guided search —where the evidence was available but was not found due to ineffectual searching —and those caused by evidence being beyond the reach of the web-crawler. This includes sources outside those of the defined 'reliable sources' as well as in media not amenable to information extraction tools, a difficulty also noted in [10].

An example of this is the claim "Over the past two years, climate...damage has cost the U.S. over $400 billion.", which was incorrectly labelled as False under open-domain parameters, but correctly verified under closed-domain parameters. This was because the justification

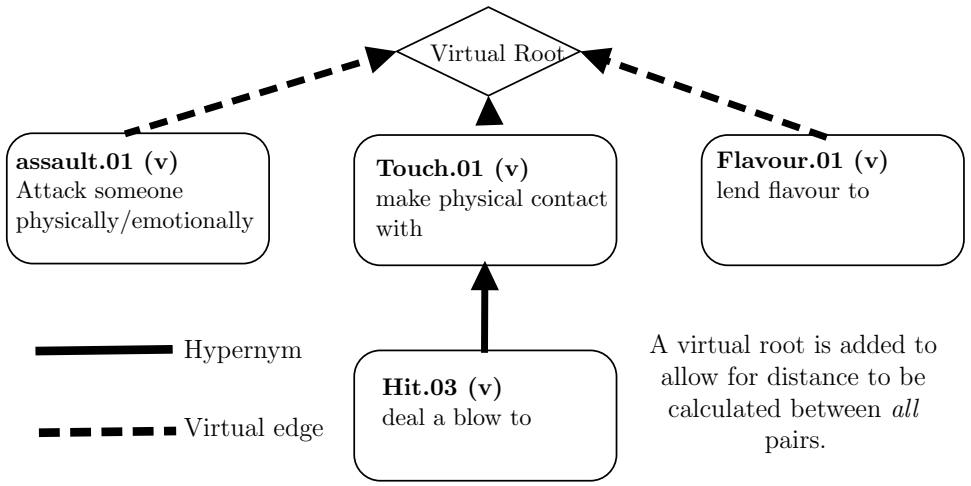



Figure 4.3: Example of wordnet bushiness causing inappropriate co-resolution

cited by PolitiFact originates in a technical report (pdf), so whilst a human journalist had extracted the required information for the PolitiFact article, which could then be used under closed-domain parameters, the source report could not be accessed directly and thus the open-domain task could not obtain that evidence. PolitiFact articles also include sources specific to the claim; for example, verifying a claim regarding Arizonan law requires the state legislature website rather than media sources. Figure 4.2 depicts this disparity in the breadth of sources used.

This would suggest that performance should be better on the closed-domain, however this is not the case in practice. The issues contributing to low sensitivity on the closed-domain training set are those which affect the remainder of open domain ground-True predicted-False cases, but are simply more prevalent as there is a smaller amount of evidence collected. Whilst the issues affecting tasks under either constraint are the same, collecting five times as many articles dampens the effect for the open-domain setting as there is a lower chance of *all* relevant documents being affected.

### 4.3.2 Verb similarity

The previously detailed low sensitivity of Wordnet verb matching is due to the structure of the verb hierarchy, which is much bushier than was anticipated. NLTK adds a virtual root to allow similarity between all pairs of verbs to be found, however as the average distance from a sought verb to the (virtual) root was only $1.873 \pm 0.03$ hops, allowing paths of hop-length 3 would permit unconnected verbs to be co-resolved via the 2-hop link between the (real) roots of unrelated trees. Figure 4.3 depicts an example of this, where *assault* is shown to be closer to *flavour* than it is *hit*, despite *flavour* being unrelated. To counter this co-resolution of unrelated verbs I allowed a maximum of 2 hops between verbs judged as similar in order to improve specificity, with the consequent severe loss in sensitivity a further indication of this being a poor method to compare verbs. Disabling this virtual root would not have solved the issue of *hit* and *assault* being presented as unrelated. Whilst this was ameliorated with the introduction of sentiment-based matching, moving to a different (and more bespoke) hierarchy such as Verbnet would be advantageous.

The introduction of sentiment to improve verb matching sensitivity was ultimately responsible for around 63% of ground-false labelled-true cases in the training/closed-domain set, with much of the remainder being due to accepting a rephrasing of the initial claim as evidence. Whilst I made efforts to exclude recycled re-statements of the claim, this was difficult to remove especially given PolitiFact journalists' propensity for using constructs such as "The Moon is made of cheese. That's what a Facebook post would have you believe."

### 4.3.3 Coreferences

The devised custom abstract meaning representation was not a major contributor of error, with a failure to resolve some coreferences the only significant problem. As most NLP models run in superlinear-time with respect to text length, a standard step for receiving evidence was to sentencize text before processing it. However, this can mean in cases of co-references that the antecedent and referent end up in separate sentences, consequently reducing the usefulness of the sentence containing the referent.

A more common issue arises in the closed-domain case where protracted discussion of the claim in question means that one subclaim's required information is scattered across multiple sentences. In these cases, whilst the referent is initially referred to in full detail, subsequent referring noun phrases are unlikely to restate any such detail. For example, the claim 'West Virginia reported over 10,000 children and youth identified as homeless for the 2018-2019 school year' cannot be resolved with the evidence 'The number of homeless students has risen to 10,522 in the 2018-19 school year' obtained from the PolitiFact article, because the 'West Virginia' aspect is missing.

### 4.3.4 Argument Comparison

Whilst this posed a significant performance issue during performance, the measures taken during development largely addressed them. Figure 4.4 demonstrates how similar and dissimilar nouns were eventually distinguished, with table 4.5 showing some edge cases and whether they were or were not correctly handled. Introducing dimensions which better distinguish similar argument pairs from dissimilar ones would serve as relevant further work, as would considering non-linear classifiers.

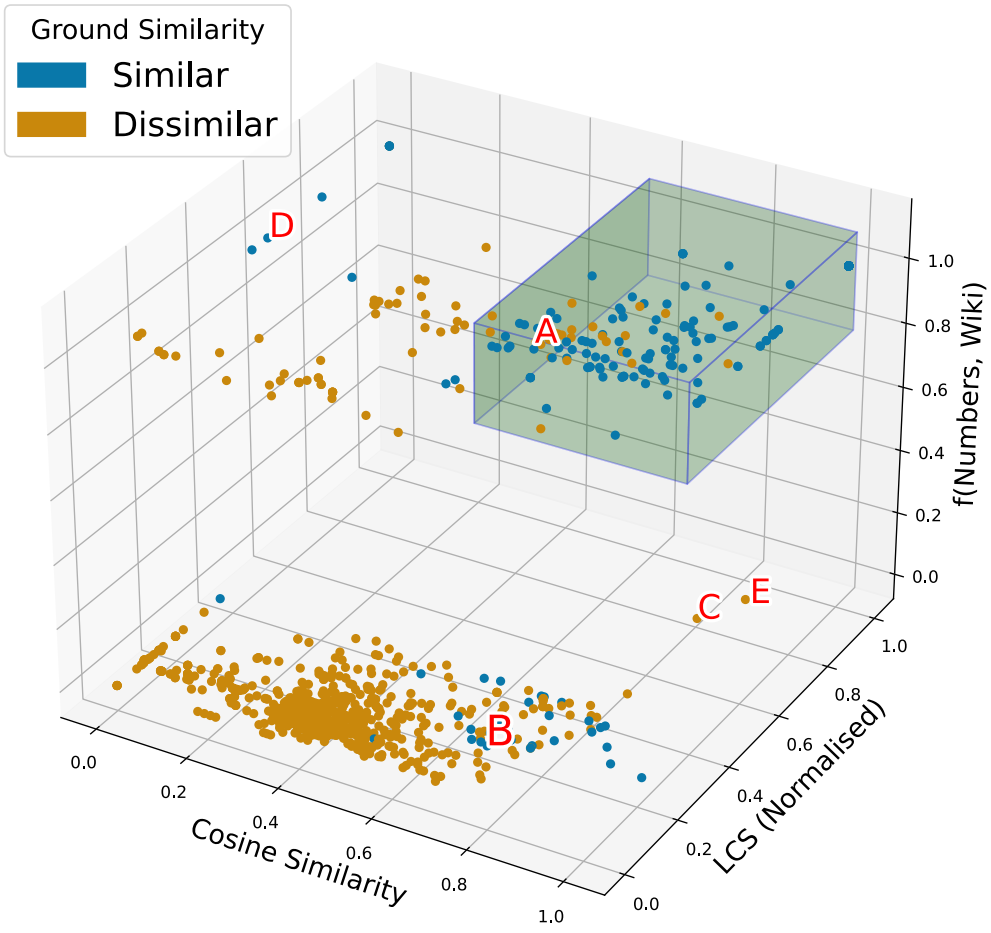



Figure 4.4: Plotting of pairs of similar and dissimilar arguments by their values for three similarity metrics.

Table 4.5: Selected pairs of arguments and why they were/were not marked as similar

| **Noun 1** | **Noun 2** | **Fig xx point** | **Ground** | **Correct** | **Explanation** |
|---|---|---|---|---|---|
| Trump | Trump's son | A | Dissimilar | No | Mediawiki/wikipedia cannot distinguish between these two, and they're both syntactically similar and share similar word embeddings. |
| crime | criminal offence | B | Similar | No | Whilst the word embeddings are similar, the two are syntactically different. Noun chunks like this also tend not to be understood by the wikipedia search. |
| Medicare | Medicaid | C | Dissimilar | Yes | These are both syntactically and semantically very similar, however, disambiguation via the mediawiki/wikipedia API correctly distinguishes between the two. |
| the call | that call | D | Similar | No | SpaCy treats both 'the' and 'call' as stop-words, and so doesn't assign them a word vector. Thus the cosine similarity is returned as 0. |
| 8.5 million americans | 36 million americans | E | Dissimilar | Yes | Both are syntactically similar, and have similar word embeddings. However, the numerical reasoner determines that these two numbers are too far apart and so f(numbers, wiki) $\leftarrow 0$ |

## 4.4 Chains of reasoning and criterion C2

### 4.4.1 Comparison of approaches

The key advantage of using logical inference is that judgements are derived by reasoning which approximates human reasoning. This differs from the PolitiHop baselines and dEFEND, which solely select sentences from the input PolitiFact article (plus social comments for dEFEND) based on machine learning features. As such, neither of these models provide any guarantee of logical soundness in their derived 'evidence', only that the selected sentences were deemed sufficiently 'evidential' by inference with their respective models.
At this level, the evidence extracted by my system and the PolitiHop baseline models are largely indistinguishable; whilst I provided functionality to chain together sources of evidence to match the derivation returned by the theorem prover, this did not occur in practice. This was largely because most claims which were successfully verified were done so in a single step, however there remained some claims which were not successfully verified but could have been with multi-hop logic which consequently an area for further investigation.

### 4.4.2 Summary of chains of reasoning results

Of the claims correctly labelled as True, 87.5% of explanations were correct for the FakeNewsNet dataset, with 65% correct for PolitiHop with open-domain, and 72% with closed-domain (for which 'correct' is defined as matching the PolitiHop dataset gold standard, as stated in criterion C2). The evidence output by dEFEND is similar to that of the PolitiHop baselines, so I now offer a comparison between the reasoning output by this project and the evidence extracted by PolitiHop-BERT.

| **Example 1** | |
|---|---|
| **Claim** | When Fidel Castro came into office, you know what he did? He had a massive literacy program. |
| **This project** | Castro set about improving literacy with a program that launched in April 1961 and ended about nine months later, according to Bill Leogrande, a professor of government at American University. (cnn.com/....) |
| **PolitiHop** | The academic consensus seems to be that Fidel Castro's government did increase the literacy rate of the island's population, at the same time that it clearly used the literacy campaign for propaganda purposes," said Jorge Duany of the Cuban Research Institute at Florida International University. |

Here, the explanation extracted by this project provides both a verification of the source *and* the individual to which the verification or quote is attributed to. In this example, the reasoning provided is correctly provided in the form of a contextualised answer, rather than a simple restatement of the quoted claim. The granular label for this claim was 'mostly-true', which is aptly captured in the PolitiHop gold-standard. However, the quote used by PolitiHop is from an email exchange conducted by the author of the fact-checking

article[1], and so the evidence selected by this project is of a high quality in light of this information being unavailable under the open-domain parameters.
The second example is lifted from FakeNewsNet, however as I lack access to the API required to run dEFEND, I cannot generate explanations to compare mine with.

| **Example 2** | |
|---|---|
| **Claim** | 'McCain opposed a requirement that the government buy American-made motorcycles. And he said all buy-American provisions were quote "disgraceful."' |
| **This project** | 1. Back in Washington, McCain opposed the requirement that the government buy American-made motorcycles.' (politico.com/...)<br>2. 'he said all buy-American provisions were, quote, disgraceful.' (politico.com/....) |

In this case, the evidence is simply derived from a news article which discusses the validity of the claim that was made. This example also showcases how a single claim can be split into multiple subclaims, with each having their own derivation returned.

Whilst no explanation is provided for claims which cannot be proven, there is actually functionality to provide a derivation for claims which were *proven* to be false (as opposed to neither being able to prove to disprove the close). Prover9 can discriminate between *exhaustion* and *contradiction*, and so the proof leading to the latter could feasibly have been output. Experimenting with this feature suggested the two cases couldn't be distinguished often enough to be useful, but the functionality does exist and could serve as a good focus for future work.

Overall, criterion C2 has been fully met, for the majority of the explanations are indeed useful and of sufficient quality. This consequently means that all success criteria have been fully met.

[1]https://www.PolitiFact.com/factchecks/2020/feb/24/bernie-sanders/sanders-correct-cuba-literacy-campaign-skimps-prop/

# Chapter 5: Conclusions

The result of this project is an end-to-end fact-checking system which approximates the human approach to fact-checking by applying logical reasoning to information extracted from raw textual sources. I have built upon existing NLP tools to devise a suitably expressive meaning representation which captures a canonical representation of a textual claim. This is then faithfully converted into logical formulae which can be used to derive a formal proof with first order logic, using evidence collected from reputable online sources.

In light of my established research questions, I then compare performance with that of the PolitiHop and FakeNewsNet machine learning based models, which use features such as linguistic features or social interaction context. My approach achieves competitive performance on the classification of True vs False claims and, uniquely, requires just an input claim and access to online news sites rather than a curated set of rich input features (dEFEND) or an existing fact-check article (PolitiHop baselines). Furthermore, I return chains of reasoning associated with the claims which can be verified, and contrast this with the more simplistic evidence-selection approach favoured by the PolitiHop baselines and dEFEND. Whilst the multi-hop aspect of these explanations could not be fully evaluated, the quality of explanations largely matched that of existing approaches. Considering these achievements, I conclude that all success criteria were fulfilled and indeed surpassed.

My interest in this area meant I came into the project with a strong mental model as to the overall functionality, but the complexity of natural language is not so easily shoehorned into methods which approximate human reasoning. In future projects I would take a more longitudinal approach to development with the aim of establishing incompatibilities between the envisaged functionality and what is materially possible earlier in development. As for this project, aspects related to matching incoming evidence required more attention than I could give them, which resulted in the imperfect solution for comparing verbs. However, given the mostly novel approach taken, the end-performance is still deemed a success, and the factors holding performance back are largely errors in ancillary components. This indicates the promise in the general approach I have proposed and the potential for drastic performance improvements upon resolution of some of the described limiting factors. I was surprised to discover how narrowly scoped the evidence domains of existing systems were; this project suggests promise in considering a wider range of online sources rather than requiring a human-devised fact-check to have already been written.

Full Fact, an industry leader in fact-checking, note a key design principle is to "build tools that do something useful right now"[42], and to this effect, the meaning representation and evidence retrieval aspects of this project could aid human fact-checkers by seeking and pre-filtering potential evidence. Further work could also consider whether a chain of

reasoning is required over simply listing evidence sentences and allowing users to 'join up the dots' themselves. Approaches to fact-checking which attempt proofs with formal systems could benefit from building upon the designed meaning representation, and work in paraphrase detection and recognising textual entailment could be incorporated to enhance the matching of evidence to extracted meaning representations.

As vaccinations for COVID-19 roll out amidst a sea of vaccine skepticism and misinformation, the need for dispelling misinformation before it can proliferate has perhaps never been more pertinent. The results of this project suggests the promise of an approach that favours human-like reasoning over the sometimes opaque results of machine learning inference systems.

# Bibliograpy


[1] "Global Risks 2013 (Eighth Edition): An Initiative of the Risk Response Network Insight Report," pp. 23–27, 2013. [Online]. Available: http://www3.weforum.org/docs/WEF_GlobalRisks_Report_2013.pdf

[2] "Global Risks 2018 (13th Edition): An Initiative of the Risk Response Network Insight Report," pp. 48–49, 2018. [Online]. Available: http://www3.weforum.org/docs/WEF_GRR18_Report.pdf

[3] S. Banaji and R. Bhat, "WhatsApp Vigilantes: An exploration of citizen reception and circulation of WhatsApp misinformation linked to mob violence in India," 2019. [Online]. Available: https://www.lse.ac.uk/media-and-communications/assets/documents/research/projects/WhatsApp-Misinformation-Report.pdf

[4] Center for an Informed Public and Digital Forensic Research Lab, Graphika and Stanford Internet Observatory, "The long fuse: Misinformation and the 2020 election," 2021. [Online]. Available: https://purl.stanford.edu/tr171zs0069

[5] World Health Organization, "Call for Action: Managing the Infodemic," Who.int, 12 2020. [Online]. Available: https://www.who.int/news/item/11-12-2020-call-for-action-managing-the-infodemic

[6] M. S. Islam, T. Sarkar, S. H. Khan, A.-H. M. Kamal, S. M. M. Hasan, A. Kabir, D. Yeasmin, M. A. Islam, K. I. A. Chowdhury, K. S. Anwar, A. A. Chughtai, and H. Seale, "COVID-19-Related Infodemic and Its Impact on Public Health: A Global Social Media Analysis," *The American Journal of Tropical Medicine and Hygiene*, vol. 103, no. 4, pp. 1621 – 1629, 07 Oct. 2020. [Online]. Available: https://www.ajtmh.org/view/journals/tpmd/103/4/article-p1621.xml

[7] Z. Barua, S. Barua, S. Aktar, N. Kabir, and M. Li, "Effects of misinformation on COVID-19 individual responses and recommendations for resilience of disastrous consequences of misinformation," *Progress in Disaster Science*, vol. 8, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590061720300569

[8] A. Zubiaga, M. Liakata, R. Procter, G. Wong Sak Hoi, and P. Tolmie, "Analysing How People Orient to and Spread Rumours in Social Media by Looking at Conversational Threads," *PLOS ONE*, vol. 11, no. 3, p. e0150989, 03 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0150989

[9] A. Vlachos and S. Riedel, "Fact Checking: Task definition and dataset construction," Proceedings of the ACL 2014 Workshop on Language Technologies and Computational

Social Science. Association for Computational Linguistics, 06 2014, p. 18–22. [Online]. Available: https://www.aclweb.org/anthology/W14-2508

[10] J. Thorne and A. Vlachos, "Automated fact checking: Task formulations, methods and future directions," Proceedings of the 27th International Conference on Computational Linguistics. Association for Computational Linguistics, 08 2018, p. 3346–3359. [Online]. Available: https://www.aclweb.org/anthology/C18-1283

[11] K. Sharma, F. Qian, H. Jiang, N. Ruchansky, M. Zhang, and Y. Liu, "Combating Fake News: A Survey on Identification and Mitigation Techniques." *ACM Trans. Intell. Syst. Technol.*, vol. 10, pp. 21:1–21:42, 2019. [Online]. Available: https://doi.org/10.1145/3305260

[12] N. Ahmadi, J. Lee, P. Papotti, and M. Saeed, "Explainable fact checking with probabilistic answer set programming." Truth and Trust Online, TTO, 2019. [Online]. Available: https://truthandtrustonline.com/wpcontent/uploads/2019/09/paper_15.pdf

[13] W. Ostrowski, A. Arora, P. Atanasova, and I. Augenstein, "Multi-hop fact checking of political claims," *arXiv preprint arXiv:2009.06401*, 2020. [Online]. Available: https://arxiv.org/abs/2009.06401

[14] K. Shu, S. Wang, and H. Liu, "Exploiting tri-relationship for fake news detection," *arXiv preprint arXiv:1712.07709*, 2017. [Online]. Available: https://arxiv.org/abs/1712.07709

[15] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake News Detection on Social Media: A Data Mining Perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[16] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media," *Big Data*, vol. 8, no. 3, pp. 171–188, 2020.

[17] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, ser. Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009.

[18] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, "Open information extraction from the web," *Commun. ACM*, vol. 51, no. 12, p. 68–74, 12 2008. [Online]. Available: https://doi.org/10.1145/1409360.1409378

[19] P. Kingsbury and M. Palmer, "From TreeBank to PropBank," in *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. European Language Resources Association (ELRA), May 2002. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2002/pdf/283.pdf

[20] M. Palmer, D. Gildea, and P. Kingsbury, "The Proposition Bank: An annotated corpus of semantic roles," *Computational Linguistics*, vol. 31, no. 1, pp. 71–106, 2005. [Online]. Available: https://www.aclweb.org/anthology/J05-1004

[21] D. Gildea and M. Palmer, "The Necessity of Parsing for Predicate Argument Recognition," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Jul. 2002, pp. 239–246. [Online]. Available: https://www.aclweb.org/anthology/P02-1031

[22] C. Bonial, J. Hwang, J. Bonn, K. Conger, O. Babko-Malaya, and M. Palmer, "English propbank annotation guidelines," *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*, vol. 48, 2012.

[23] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, "Abstract Meaning Representation for sembanking," in *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Aug. 2013, pp. 178–186. [Online]. Available: https://www.aclweb.org/anthology/W13-2322

[24] C. Fellbaum, *WordNet(s)*, 2nd ed. Elsevier, 2006, pp. 665–670. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B0080448542009469

[25] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, p. 39–41, Nov. 1995. [Online]. Available: https://doi.org/10.1145/219717.219748

[26] Princeton University, "About wordnet." WordNet, 2010. [Online]. Available: https://wordnet.princeton.edu/

[27] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966.

[28] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, p. 31–88, Mar. 2001. [Online]. Available: https://doi.org/10.1145/375360.375365

[29] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, ser. EBL-Schweitzer. Cambridge University Press, 1997.

[30] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[31] P. Shi and J. Lin, "Simple bert models for relation extraction and semantic role labeling," *CoRR*, vol. abs/1904.05255, 2019. [Online]. Available: http://arxiv.org/abs/1904.05255

[32] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ser. ACL '00. USA: Association for Computational Linguistics, 2000, p. 512–520. [Online]. Available: https://doi.org/10.3115/1075218.1075283

[33] V. Krotov, L. Johnson, and L. Silva, "Tutorial: Legality and ethics of web scraping," *Communications of the Association for Information Systems*, vol. 47, p. 22, 2020.

[34] Intellectual Property Office, "Exceptions to copyright," GOV.UK, 06 2014. [Online]. Available: https://www.gov.uk/guidance/exceptions-to-copyright

[35] H. Ouchi, H. Shindo, and Y. Matsumoto, "A span selection model for semantic role labeling," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Oct.-Nov. 2018, pp. 1630–1642. [Online]. Available: https://www.aclweb.org/anthology/D18-1191

[36] L. Humberstone, *The connectives*. Cambridge, Mass.: MIT Press, 2011.

[37] J. H. Gallier, *Logic for computer science : foundations of automatic theorem proving* , ser. Harper & Row computer science and technology series ; 5. New York ; London: Harper and Row, 1986.

[38] A. Esuli and F. Sebastiani, "SENTIWORDNET: A publicly available lexical resource for opinion mining," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA), May 2006. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2006/pdf/384_pdf.pdf

[39] C. D. Manning, P. Raghavan, and H. Schutz, *Introduction to information retrieval*. Cambridge University Press, 2008.

[40] K. Shu, L. Cui, S. Wang, D. Lee, and H. Liu, "Defend: Explainable fake news detection," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 395–405. [Online]. Available: https://doi.org/10.1145/3292500.3330935

[41] F. Fact, "Automated fact checking," Full Fact, 2015. [Online]. Available: https://fullfact.org/about/automated/

[42] M. Babakar, W. Moy, and Full Fact, "The State of Automated Factchecking," fullfact.org, 08 2016. [Online]. Available: https://fullfact.org/media/uploads/full_fact-the_state_of_automated_factchecking_aug_2016.pdf

# Appendix A: Appendices

## A.1 Comparing textual similarity metrics

Table A.1: Textual similarity metrics compared; Euclidian distance is from point on ROC curve nearest optimal.

**Ents**

| Name | Optimal Threshold | Min. Euclidean dist from optimal | AUC |
|---|---|---|---|
| overlap | 0.8333 | 0.3059 | 0.9638 |
| lcsstr | 0.2157 | 0.3955 | 0.9119 |
| ratcliff_obershelp | 0.3548 | 0.4049 | 0.9219 |
| levenshtein | 0.2273 | 0.4474 | 0.9093 |
| lev-damerau | 0.2273 | 0.4474 | 0.9093 |
| smith-waterman | 0.3529 | 0.4625 | 0.9128 |
| cosine | 0.4767 | 0.4783 | 0.9130 |
| zlib_ncd | 0.4762 | 0.5137 | 0.8693 |
| lcsseq | 0.2941 | 0.5166 | 0.8842 |
| gotoh | 0.5400 | 0.5379 | 0.8614 |
| jaccard | 0.3333 | 0.5433 | 0.8970 |
| sorenson-dice | 0.5000 | 0.5433 | 0.8970 |
| tversky | 0.3333 | 0.5433 | 0.8970 |
| editex | 0.4091 | 0.5583 | 0.8674 |
| monge-elkan | 0.0313 | 0.5739 | 0.8869 |
| sqrt_ncd | 0.2441 | 0.6088 | 0.8609 |
| bag | 0.4167 | 0.6211 | 0.8612 |
| jaro-winkler | 0.5870 | 0.6227 | 0.8539 |
| strcmp95 | 0.6279 | 0.6406 | 0.8380 |
| needleman-wunsch | 0.5500 | 0.6518 | 0.7983 |
| entropy_ncd | 0.8470 | 0.6726 | 0.8173 |
| bz2_ncd | 0.6750 | 0.6766 | 0.7804 |
| hamming | 0.0196 | 0.6925 | 0.8149 |
| prefix | 0.0556 | 0.7403 | 0.8087 |
| postfix | 0.2273 | 0.7603 | 0.7773 |
| length | 0.7273 | 0.8312 | 0.6237 |
| bwtrle_ncd | 0.0769 | 0.9079 | 0.6030 |
| *identity* | *1.0000* | *0.9270* | *0.6875* |
| matrix | 1.0000 | 0.9270 | 0.6875 |
| lzma_ncd | 0.6774 | 0.9639 | 0.4961 |

**Nouns**

| Name | Optimal Threshold | Min. Euclidean dist from optimal | AUC |
|---|---|---|---|
| lcsstr | 0.2000 | 0.3429 | 0.9405 |
| smith-waterman | 0.3333 | 0.3649 | 0.9362 |
| ratcliff_obershelp | 0.3548 | 0.4084 | 0.9211 |
| levenshtein | 0.2727 | 0.4425 | 0.9084 |
| lev-damerau | 0.2727 | 0.4425 | 0.9084 |
| gotoh | 0.5100 | 0.4944 | 0.8970 |
| zlib_ncd | 0.3846 | 0.4977 | 0.9148 |
| lcsseq | 0.3000 | 0.5009 | 0.8952 |
| cosine | 0.5222 | 0.5033 | 0.9004 |
| jaccard | 0.3030 | 0.5202 | 0.8848 |
| sorenson-dice | 0.4651 | 0.5202 | 0.8848 |
| tversky | 0.3030 | 0.5202 | 0.8848 |
| editex | 0.3889 | 0.5257 | 0.8838 |
| overlap | 0.7273 | 0.5551 | 0.8626 |
| bag | 0.4000 | 0.5665 | 0.8557 |
| sqrt_ncd | 0.2565 | 0.5748 | 0.8562 |
| needleman-wunsch | 0.3971 | 0.6146 | 0.8333 |
| postfix | 0.0769 | 0.6182 | 0.8567 |
| entropy_ncd | 0.7788 | 0.6942 | 0.8146 |
| strcmp95 | 0.6274 | 0.7078 | 0.7644 |
| jaro-winkler | 0.6132 | 0.7190 | 0.7619 |
| bwtrle_ncd | 0.0571 | 0.7535 | 0.6907 |
| bz2_ncd | 0.6383 | 0.7540 | 0.7421 |
| length | 0.4167 | 0.7577 | 0.7219 |
| mra | 0.3333 | 0.7662 | 0.7515 |
| lzma_ncd | 0.6923 | 0.8435 | 0.6094 |
| hamming | 0.0417 | 0.8648 | 0.6701 |
| prefix | 0.0476 | 0.9302 | 0.6390 |
| monge-elkan | 0.0612 | 0.9317 | 0.5464 |
| *identity* | *0.0000* | *0.9798* | 0.5000 |

## A.2 Distribution and final list of reputable sources

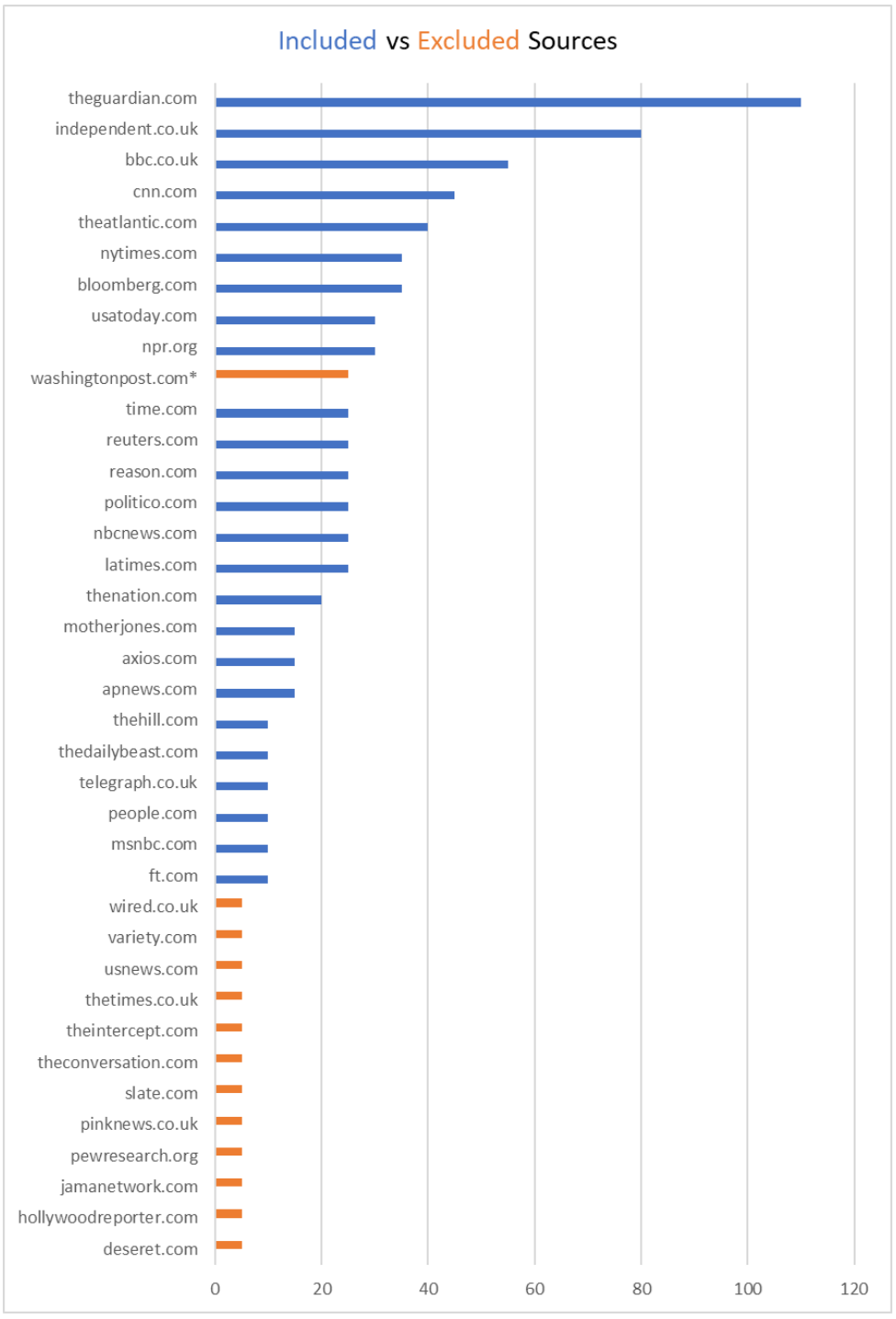



Figure A.1: Distribution of reputable source selected, with frequencies for experimentation shown.

## A.3 Python code snippets

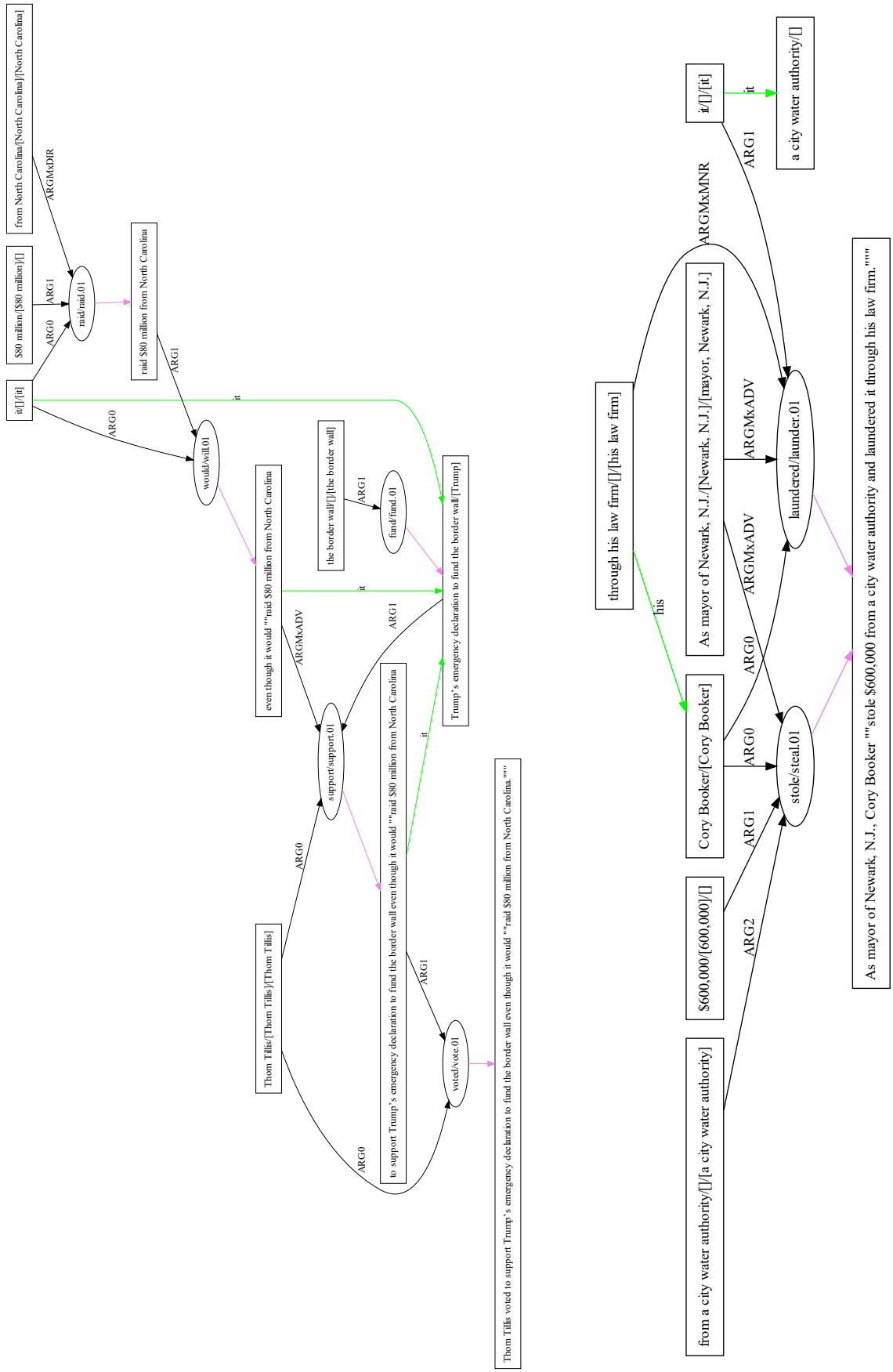
```
# Create graph relating extracted arguments, outputting it if requested. Populates disconnected subgraphs based on
# sets of verb frames and arguments (OIESubclaims), and then adds shared co-references and implication where a verb
# is a substring of the argument of another verb.
def generate_graph(self, oie_subclaims, output=True):
    # The graph is directed as all edges are either implication or property-of relations. Use strict Digraphs to
    # prevent any duplicate edges.
    G = Digraph(strict=True, format='pdf')
    arg_set = set()
    verb_set = set()
    coref_nodes, coref_edges = [], []

    # Create the overall leaf, for which the conjunction of the root verbs will imply:
    G.node(self.argID(self.doc[:]), self.doc.text, shape='box')

    # Iterate through all extracted relations
    seenRoots = []
    for claim in oie_subclaims:

        # Plot the verb after having converted it to a node.
        root = claim['V']
        seenRoots.append(root)
        check = ''.join(filter(str.isalnum, str(root))).replace(' ', '')
        if check == "":  # Sometimes some nonsense can be attributed as a verb by oie.
            continue

        # Add arg to graph, with a helpful label for optics.
        G.node(self.argID(root), root.text + '/' + self.argBaseC[self.argID(root)].get_uvi_output())

        for arg_type, argV in claim.items():
            if arg_type != 'V' and argV not in seenRoots:
                # Create a node for each argument, and a link to its respective verb labelled by its arg type.
                G.node(self.argID(argV), argV.text + "/" + str(argV.ents) + "/" + str(list(argV.noun_chunks)),
                       shape='box')
                G.edge(self.argID(argV), self.argID(root), label=arg_type.replace('-', 'x'),
                       style=get_edge_style(arg_type, argV))
                # Replace any '-' with 'x' as '-' is a keyword for networkx, but is output by SRL.

                # Add the argument to the list of arguments eligible to be implied by another subtree.
                arg_set.add(argV)

                # Add coreference edges to the graph from the initial text to the entity being coreferenced, only
                # for the version of the graph that is displayed. Co-references must be omitted from the true
                # graph as they interfere with splitting into subclaims as the edges becomes bridges. The
                # coreferences themselves are not lost as they're properties of the doc/spans. They are useful for
                # illustratory and debugging purposes, and so can be output when requested with output=True.
                # This is sound to omit from the networkx graph as a coreference does not result in two claims
                # being co-dependent e.g. 'My son is 11. He likes to eat cake.' - the coreference bridges the two
                # otherwise separate components when there should be no co-dependence implied. Both are about the
                # son, but there is not an iff relation between them.
                arg_corefs = get_span_coref(argV)
                if output and len(arg_corefs):
                    for cluster in arg_corefs:
                        canonical_reference = cluster.main
                        for inst in cluster.mentions:
                            if inst.start >= argV.start and inst.end <= argV.end and inst != canonical_reference:
                                coref_nodes.append((self.argID(canonical_reference), canonical_reference.text + "/"
                                                    + str(canonical_reference.ents)))
                                coref_edges.append((self.argID(argV), self.argID(canonical_reference), inst.text))

            # Add all verbs to the list of eligible roots for subtrees.
            else:
                verb_set.add(argV)

    # Create 'purple' edges - i.e. edges that link a verb rooting a subtree (of size >= 1) to the argument that they
    # imply. Only one verb can imply any one argument in order to preserve tree-like structure.
    for argV in verb_set:
        shortest_span = self.doc[:]
        for parent in arg_set:
            if argV != parent and argV.start >= parent.start and argV.end <= parent.end and (
                    parent.end - parent.start) < (shortest_span.end - shortest_span.start):
                shortest_span = parent
        G.node(self.argID(shortest_span), shortest_span.text, shape='box')
        G.edge(self.argID(argV), self.argID(shortest_span), color='violet')

    return G
```

```python
# Compare two nouns on a mix of metrics.
def node_compare(inc_node_label, inc_node_span, exst_rich_arg, wiki_cache):
    # Baseline embedding similarity required, and cannot include nodes with an argument type that's dotted (excluded)
    if inc_node_label != 'V' and claim.get_edge_style(inc_node_label, inc_node_span) != 'dotted':
        icorefs, ecorefs = coref_collect(inc_node_span), coref_collect(exst_rich_arg.span)
        # If No entities in either noun, then compare if embeddings are close (by cosine similarity).
        if (lcs(inc_node_span, exst_rich_arg.span) > 0.7 and tda.overlap.normalized_similarity(inc_node_span.text, exst_rich_arg.span.text) > 0.8) or
            return True
        else:
            # Check if more noun-chunks are alike than are dissimilar, based on levenshtein.
            similar_count, dissimilar_count = 0, 0

            # Existing MUST come first here, as an incoming can provide more information than the existing, but not less.
            # E.g. existing: the democrat lead campaign of domestic terrorism, incoming: democrats.
            # If iterating through incoming first, democrats will match and the iteration completed with 100% matches.
            # If iterating through existing first, democrats will match but noun of the other nouns will -> ~30%.
            for x in exst_rich_arg.span.noun_chunks:
                for y in inc_node_span.noun_chunks:
                    if lcs(x.root,y.root) > 0.35 and compute_similarity(x,y) > 0.57:
                        similar_count += 1
                        break
                else:
                    dissimilar_count += 1

            if similar_count > dissimilar_count:
                return True
            # If entities are present in both, require sane length and for numbers to be sufficiently similar (or the
            # comparison return True as the entities passed are not elligible), and one of levenshtein or wikidata
            # coresolution to exhibit sufficient similarity.
            similar_ents = 0
            dissimilar_ents = 0
            for x in set(exst_rich_arg.span.ents+ecorefs):
                for y in inc_node_span.ents+icorefs:
                    if num_compare(x, y) and lcs(x,y) > 0.39 or wikidata_compare(x, y, wiki_cache):
                        similar_ents += 1
                        break
                else:
                    dissimilar_ents += 1

            if similar_ents > dissimilar_ents:
                return True

    # If not matched on any criteria, then return False.
    return False
```

## A.4 Exemplar prover9 proof

```
 b'============================== Prover9 ===============================
Prover9 (32) version Dec-2007, Dec 2007.
============================== end of head ===========================

============================== INPUT =================================
assign(max_seconds,60).
clear(auto_denials).

formulas(assumptions).
a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales) -> a6X7Xare2ARG1 = a7X17Xforexpandingbackgroundcheckstoallgunsales.
a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales) -> argF(root).
a6X7Xare2(a2X6X90ofpolicemen,a6X7Xare2ARG1).
a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales).
a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales).
end_of_list.

formulas(goals).
argF(root).
end_of_list.

============================== end of input ==========================

============================== PROCESS NON-CLAUSAL FORMULAS ==========

% Formulas that are not ordinary clauses:
1 a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales) -> a6X7Xare2ARG1 = a7X17Xforexpandingbackgroundcheckstoallgunsales # label(non_claus
2 a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales) -> argF(root) # label(non_clause).  [assumption].
3 argF(root) # label(non_clause) # label(goal).  [goal].

============================== end of process non-clausal formulas ===

============================== PROCESS INITIAL CLAUSES ===============

% Clauses before input processing:

formulas(usable).
end_of_list.

formulas(sos).
-a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales) | a7X17Xforexpandingbackgroundcheckstoallgunsales = a6X7Xare2ARG1.  [clausify(1)].
-a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales) | argF(root).  [clausify(2)].
a6X7Xare2(a2X6X90ofpolicemen,a6X7Xare2ARG1).  [assumption].
a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales).  [assumption].
a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales).  [assumption].
-argF(root).  [deny(3)].
end_of_list.

formulas(demodulators).
end_of_list.


============================== CLAUSES FOR SEARCH ====================

% Clauses after input processing:

formulas(usable).
end_of_list.

formulas(sos).
5 -a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales) | a6X7Xare2ARG1 = a7X17Xforexpandingbackgroundcheckstoallgunsales.  [copy(4),flip(1
6 -a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales) | argF(root).  [clausify(2)].
7 a6X7Xare2(a2X6X90ofpolicemen,a6X7Xare2ARG1).  [assumption].
8 a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales).  [assumption].
9 a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales).  [assumption].
10 -argF(root).  [deny(3)].
end_of_list.

formulas(demodulators).
end_of_list.

============================== end of clauses for search =============

============================== SEARCH ================================

% Starting search at 0.01 seconds.

given #1 (I,wt=6): 5 -a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales) | a6X7Xare2ARG1 = a7X17Xforexpandingbackgroundcheckstoallgunsale

given #2 (I,wt=5): 6 -a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales) | argF(root).  [clausify(2)].

given #3 (I,wt=3): 7 a6X7Xare2(a2X6X90ofpolicemen,a6X7Xare2ARG1).  [assumption].

given #4 (I,wt=3): 8 a10X11Xexpanding2(a11X13Xbackgroundchecks,a13X17Xtoallgunsales).  [assumption].

given #5 (I,wt=3): 9 a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales).  [assumption].
\x07-------- Proof 1 --------

============================== PROOF =================================

% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 7.
% Level of proof is 3.
% Maximum clause weight is 5.
% Given clauses 5.

2 a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales) -> argF(root) # label(non_clause).  [assumption].
3 argF(root) # label(non_clause) # label(goal).  [goal].
6 -a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales) | argF(root).  [clausify(2)].
9 a6X7Xare2(a2X6X90ofpolicemen,a7X17Xforexpandingbackgroundcheckstoallgunsales).  [assumption].
10 -argF(root).  [deny(3)].
12 argF(root).  [hyper(6,a,9,a)].
13 $F.  [resolve(12,a,10,a)].

============================== end of proof ==========================
```

## A.5 Sample AMRs



*Joshua Cowan*
*Clare College*
*jgc46*

Computer Science Tripos Part II Project Proposal

# Automated multi-hop fact checking with NLP

22nd October 2020

**Project Originator:** Joshua Cowan

**Project Supervisors:** Andrew Caines and Zheng Yuan

**Signatures:**

**Director of Studies:** Larry Paulson

**Signature:**

**Overseers:** Andreas Vlachos and Andrew Moore

**Signatures:**

# Introduction & Description of the Work

The rapid spread of misinformation by humans [10] has been posited to have influenced elections [4], hampered efforts to tackle COVID-19 [3], and driven instances of criminality [5]. As such, there is demand for an automated fact-checking system [2] to limit the spread of misinformation in real-time rather than retroactively. Crucially, it should provide a chain of reasoning intelligible by an unskilled human given that "the truthfulness [of a news article] is still ultimately defined by the statements it makes"[8].

I seek to institute a fact-checking framework built around an information extraction system, with an attached reasoning module and web-crawler, broadly following the framework set out in [1] but varying/replacing the constituent parts. The claim to be checked is processed into natural language relations and then into logical formulae, which then prompts a web-crawler to obtain relevant information from newswire sources to be processed similarly. A reasoning module then uses the derived formulae to establish a chain of reasoning leading to a proof/contradiction, or if inconclusive then advances the available knowledge which in turn guides the web crawler as part of a new fact-finding iteration. The overall task is that described in [6], and so by evaluating on the same dataset I am able to evaluate performance comparatively.

Due to potential processing of personally identifiable (albeit public) data, I sought and have received Ethics Committee clearance (#1758).

# Starting Point

I have prior experience working with SpaCy and employing off-the-shelf information extraction models, as well as using Wikidata - all as part of a summer internship during which I worked in Python, however no code from this will be employed or adapted. The 1B course in 'Logic & Proof' underpins my knowledge of the logical reasoning module, and the 1A 'Machine Learning and Real World Data' course informs training and testing a model, with 'Formal Models of Language' aiding some NLP steps. AllenNLP's existing SRL model will likely be used, as will standard NLP packages such as NLTK and SpaCy as well as packages to handle logical systems as this is not the core focus of the project. I have also read a number of papers related to information extraction & differing linguistic features of misinformation as well as the approaches entered to the FEVER 1st shared task [9]. In addition, I shall make use of both the Politihop [6] dataset provided by CopeNLU, and a subset of claims from Politifact consisting solely of direct quotes ('Politifact-speech') [7]. The claims in the Politihop dataset are designed for logical decomposition and 'multi-hop' reasoning, and so are well suited to this task. I have also applied for access to the BOLT dataset[1] for use in a potential extension and for which I await a response.

[1] https://catalog.ldc.upenn.edu/LDC2020T21

# Substance and Structure of the Project

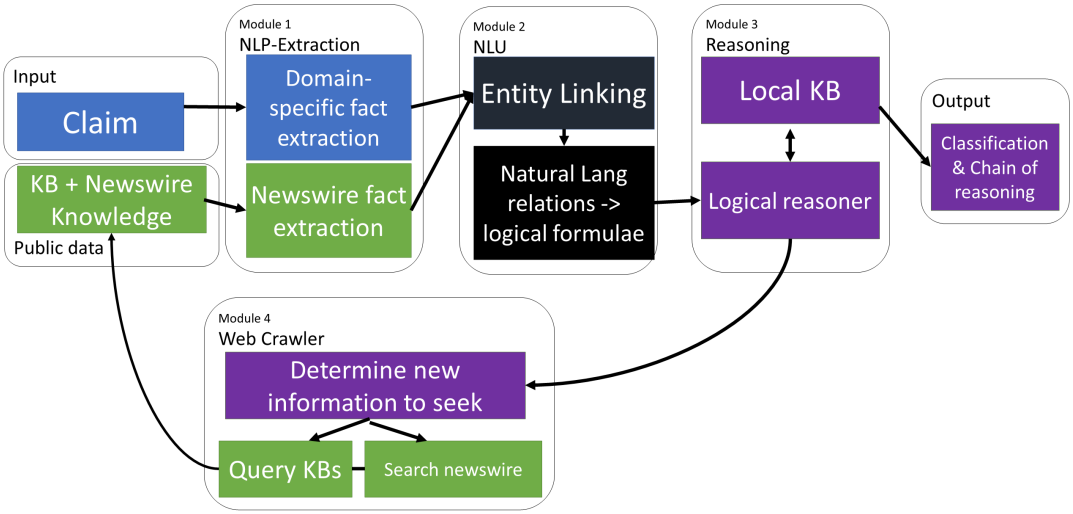



Figure 1: Proposed Structure

## Main body

Python will be the primary language used due to the wealth of NLP tools available for it. Java may be used to enable use of StanfordNLP. The project will be constructed from modules as shown in *Figure 1.*

NLP-Extraction module: Performs *open* information extraction using a Semantic Role Labelling model, initially using that packaged with AllenNLP with a view to train a custom model, conditional on time permitting and obtaining the BOLT dataset or similarly annotated data. The information extraction is *open* as a fixed set of rules is insufficiently flexible to capture the variation in content in the claims. SpaCy will structure data pertaining to each source 'doc(ument)', as well as performing Named Entity Recognition. NLTK, AllenNLP and textacy may be used where required to handle sub-tasks such as coreference resolution or temporal reasoning.

NLU (Natural Language Understanding) module: Links entities with an online knowledge base, likely Wikidata, and then converts these Natural Language relations to logical formulae. A key issue here is aliased predicates e.g. 'donated to' is similar to 'supported', which will be resolved using tools such as the Unified Verb Index[2].

Reasoning & Web crawler modules: Includes the local knowledge base in which derived relations are stored, likely using NLTK's support for logic systems. Also contains an off-the-shelf automated proof system, the choice of which is pending research. The web crawler will have shallow searching capabilities for keywords/named

[2]https://uvi.colorado.edu/

entities to extract relevant information from newswire sources, and will also query knowledge bases. These two modules are not the focus of the project, however, and so existing solutions will be sought wherever possible.

### Extensions

Possible extensions include:

- Training a custom Semantic Role Labelling model to handle directly quoted claims specifically (i.e. spoken claims and those typed directly by the individual). The stylistically distinct nature of spoken quotes may be better captured with an exclusively domain-relevant training set i.e. A subset of the BOLT dataset. 'Politifact-speech' would then be used for end-to-end system evaluation as this contains a more raw representation of typical argument making than the paraphrased and optimised claims that make up the majority of Politifact's data.
- Adapting the pipeline to handle FEVER [9] claims as to enable comparative evaluation.
- Using a soft-logic system within the reasoner module.

## Success Criteria

### Initial Criteria

C1a System successfully extracts logical formulae from the claim and derives a judgement.

C1b Given the novel approach, better than 50% accuracy on classification of veracity of claims in a label-balanced subset of the Politihop dataset.

C2 Evaluate quality of natural language chains of reasoning attached to the veracity claims by comparison with those made in gold-standard (Politihop). Perform quantitative summarisation of comparative performance.

### Potential Extension Criteria

Conditional on the relevant extensions having been completed.

C3 Evaluate performance of custom Semantic Role Labeller on held-back test data.

C4 Evaluate classification veracity & qualitative quality of reasoning on the Politifact-speech dataset using the new model configuration[3].

C5 Evaluate modified pipeline's FEVER Score [9] (via codalab), success judged as approximately matching or exceeding score of baseline system.

[3]With the assistance of https://github.com/gabrielStanovsky/oie-benchmark

# Timetable and Milestones

2 week slots are detailed, permitting for truly unseen circumstances and enforcing steady progress.

## Preparation

**Epoch 0** - *17th October – 31st October*

Complete Project proposal, prepare workspace on MCS. Setup fresh Python environment on WSL, familiarize with Ubuntu. Initial research into Unified Verb Index, logical system, and any subtasks in the NLP-extraction module.

**Deadlines**

- Proposal Deadline – 23rd October, 12 PM.

**Epoch 1** - *1st November – 13th November*

Get refamiliarized with Python and investigate appropriate NLP frameworks for each task. Prepare datasets for use.

*Milestone:* Datasets prepared for evaluation. Full pipeline sketched out with specific functions set out.

## Implementation

**Epoch 2 - 14th November – 27th November - NLP-Extraction module**

Implement AllenNLP's SRL model, handle required subtasks including Coreference resolution, temporal reasoning, named entity recognition (in preparation for NLU). Implement standard SpaCy pipeline (including POS-tagging & Dependency Parsing if/where required).

*N.B. NLP unit of assessment coursework issued 11th November, Due 4th December.*

*Milestone: Natural Language relations extracted from sample claims*

**Epoch 3 - 28th November – 11th December - NLU**

Draft preparation chapter of dissertation. Establish Knowledge Base system compatible with NLTK's logic system. Implement entity linking.

*Milestone: Entities identified in Natural Language relations and are linked to KB ID.*

**Epoch 4 - 11th December – 25th December - NLU**

Handle conversion of natural language triples (i.e. output of SRL) into KB compatible triples with aid of Unified Verb Index. Implement primitive web-crawler to permit preliminary validation testing.

**Epoch 5 - 26th December – 8th January**

Complete outstanding work on NLP-Extraction and NLU modules. Vacation & revision of Michaelmas lectures.

*Milestone: Logical formulae obtained from input claims.*

**Epoch 6 - 9th January – 22nd January - Reasoner & Web Crawler**

Implement complete web crawler with reliable sources set and Knowledge Base querying supported. Introduce reasoning system and perform end-to-end tuning.

*Milestone: Web crawler and reasoning system instituted, end-to-end tuning completed.*

**Epoch 7 - 23rd January – 5th February - Overflow, Evaluation & Progress Report**

Begin evaluating against success criteria. Draft Implementation chapter of dissertation. Compose Progress Report and rehearse presentation.

*Milestone: Evaluation against criteria begun. Progress report completed.*

**Deadlines**

- Progress Report Deadline – 5th February, 12 noon.

**Epoch 8 - 6th February – 19th February - Evaluation & Extensions**

Continue evaluation & implementation of any extensions.

**Epoch 9 - 20th February – 5th March - Extensions**

Complete extension implementation, complete drafting of implementation chapter.

*Milestone: Implementation completed.*

**Deadlines**

- Progress Report Presentations – 11th-16th February, 2 PM.

**Epoch 10 - 6th March – 19th March - Extensions**

Complete evaluation of extensions, begin drafting evaluation chapter.

*N.B. Expecting Cybercrime unit of assessment work to peak around this point.*

*Milestone: Evaluation completed.*

## Dissertation

**Epoch 11 - 20th March – 2nd April**

Vacation, Exam revision, first draft of uninitiated chapters of dissertation, excluding Conclusion.

**Epoch 12 - 3rd April – 16th April**

First full draft of dissertation completed and submitted to supervisors for feedback.

*Milestone: Dissertation submitted to supervisors.*

**Epoch 13 - 17th April – 30th April**

Finalize appendices & Bibliography, action suggested feedback. Convert drafts to finalised text.

**Epoch 14 - 1st May – 14th May**

Final text review, submit code and dissertation document.

*Milestone: Dissertation submitted.*

**Deadlines**

- Dissertation Deadline – 14th May, 12 noon.
- Source Code Deadline – 14th May, 5 PM.

# Resources Declaration

## Hardware & Backups

In the case of loss of connection to University services or other unforeseen circumstances, I intend to use my personal machine (Dell XPS 9560 running Windows 10 with WSL2, Intel i5-7300HQ CPU, 8GB RAM, 1+0.125TB SSDs + 1TB HDD). I will use version control with a private GitHub repository & weekly whole-project incremental backups to a private Google Drive account.

I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.

I will request access to the HPC facility to permit efficient training of a custom SRL model if required.

## Datasets

- Politihop dataset – Publicly available, under (workable) licence terms [6].
- Politifact-speech dataset - Publicly available.[4]
- FEVER dataset – Publicly available, under (workable) licence terms.[5]
- OntoNotes5 dataset – Available under licence from the LDC. Securely kept on MCS systems, having been obtained.[6]
- BOLT dataset - Available under licence from the LDC. Would be securely kept on MCS systems if obtained.[7] I am in the process of sourcing this through the university, however **this is not considered critical** as should BOLT not be obtained then the 'Broadcast conversation' and 'Telephone conversation' subsets of OntoNotes can be used instead.
- AllenNLP models – Publicly available, under (workable) licence terms.[8]

  A clean copy of the required data/models will be kept on an external hard drive in the case of their sudden unavailability (excluding where my right to use them has been withdrawn as a result of a licensing change). The datasets will additionally be kept in adherence to the terms set out in the Ethics Application.

[4] [7], data at https://hrashkin.github.io/factcheck.html
[5] https://s3-eu-west-1.amazonaws.com/fever.public/license.html
[6] https://catalog.ldc.upenn.edu/LDC2013T19
[7] https://catalog.ldc.upenn.edu/LDC2020T21
[8] https://github.com/allenai/allennlp-models/blob/master/LICENSE

# References


[1] Naser Ahmadi et al. *Explainable Fact Checking with Probabilistic Answer Set Programming*. 2019. arXiv: 1906.09198 [cs.DB].

[2] Mevan Babakar and Will Moy. "The state of automated factchecking". In: *Full Fact* 28 (2016).

[3] Zapan Barua et al. "Effects of misinformation on COVID-19 individual responses and recommendations for resilience of disastrous consequences of misinformation". In: *Progress in Disaster Science* 8 (2020), p. 100119. ISSN: 2590-0617. DOI: https://doi.org/10.1016/j.pdisas.2020.100119. URL: http://www.sciencedirect.com/science/article/pii/S2590061720300569.

[4] Alexandre Bovet and Hernán A. Makse. "Influence of fake news in Twitter during the 2016 US presidential election". In: *Nature Communications* 10.1 (Jan. 2019). ISSN: 2041-1723. DOI: 10.1038/s41467-018-07761-2. URL: http://dx.doi.org/10.1038/s41467-018-07761-2.

[5] Cecilia Kang and Adam Goldman. *In Washington Pizzeria Attack, Fake News Brought Real Guns*. Dec. 2016. URL: https://www.nytimes.com/2016/12/05/business/media/comet-ping-pong-pizza-shooting-fake-news-consequences.html.

[6] Wojciech Ostrowski et al. *Multi-Hop Fact Checking of Political Claims*. 2020. arXiv: 2009.06401 [cs.CL].

[7] Hannah Rashkin et al. "Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2931–2937. DOI: 10.18653/v1/D17-1317. URL: https://www.aclweb.org/anthology/D17-1317.

[8] Karishma Sharma et al. *Combating Fake News: A Survey on Identification and Mitigation Techniques*. 2019. arXiv: 1901.06437 [cs.LG].

[9] James Thorne et al. "The Fact Extraction and VERification (FEVER) Shared Task". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 1–9. DOI: 10.18653/v1/W18-5501. URL: https://www.aclweb.org/anthology/W18-5501.

[10] Soroush Vosoughi, Deb Roy and Sinan Aral. "The spread of true and false news online". In: *Science* 359.6380 (2018), pp. 1146–1151. ISSN: 0036-8075. DOI: 10.1126/science.aap9559. eprint: https://science.sciencemag.org/content/359/6380/1146.full.pdf. URL: https://science.sciencemag.org/content/359/6380/1146.