designs

MDPI

Review

Using NLP for Fact Checking: A Survey

Eric Lazarski, Mahmood Al-Khassaweneh * and Cynthia Howard

Department of Engineering, Computing and Mathematical Sciences, Lewis University, Romeoville, IL 60446, USA; ericmlazarski@lewisu.edu (E.L.); howardcy@lewisu.edu (C.H.)
* Correspondence: khassaweneh@ieee.org or malkhassaweneh@lewisu.edu

Abstract: In recent years, disinformation and "fake news" have been spreading throughout the internet at rates never seen before. This has created the need for fact-checking organizations, groups that seek out claims and comment on their veracity, to spawn worldwide to stem the tide of misinformation. However, even with the many human-powered fact-checking organizations that are currently in operation, disinformation continues to run rampant throughout the Web, and the existing organizations are unable to keep up. This paper discusses in detail recent advances in computer science to use natural language processing to automate fact checking. It follows the entire process of automated fact checking using natural language processing, from detecting claims to fact checking to outputting results. In summary, automated fact checking works well in some cases, though generalized fact checking still needs improvement prior to widespread use.

Keywords: natural language processing; machine learning; fact checking

Citation: Lazarski, E.; Al-Khassaweneh, M.; Howard, C. Using NLP for Fact Checking: A Survey. *Designs* **2021**, *5*, 42. https://doi.org/10.3390/designs5030042

Academic Editors: Camelia Delcea, Liviu-Adrian Cotfas and Gabriella Ferruzzi

Received: 3 June 2021
Accepted: 6 July 2021
Published: 14 July 2021

1. Introduction

Fact checking has been becoming more popular since the early 2000's, however it has grown in popularity greatly in recent years. As of 2016, there were 113 active fact checking groups across the globe, 90 of which were established post 2010 [1]. With the rise of social media and "fake news" spreading throughout the web [2], fast and accurate fact checking is now more imperative than ever. This is difficult, however, as the claims that need to be checked are written in human languages. Hence, by using Natural Language Processing (NLP), claims can be fact checked autonomously.

Methods already exist to do this, and they are implemented in production and can receive good reviews from their users [3]. The current state of the art approach appears to be Claim Buster, as other papers compare themselves to this application in regard to performance. Claim Buster is available to use for free [4] and an open API is also available. However, a simple query for the web-based fact-checker of "From 2016–2020 Donald Trump was President of the United States of America" returned no results that indicate that the query is true, most of the results being related to his age. This is an undesirable result for a simple query.

This paper examines how NLP is used to fact check claims in written text, from the start of defining claims and determining those worthy of verification, to how parsed claims are verified against facts that are known to be true.

2. Review Studies on Using NLP for Fact Checking

This section is subdivided into several sections. They are related to background information, claim identification and extraction, defining sources of evidence, and various processes of claim verification itself.

2.1. Background

As mentioned above, fact checking has entered the foreground of public thought in recent years. To be clear, fact checking "directly engages in adjudicating factual disputes by

publicly deciding whose claim is correct or incorrect" [5]. This has become more important in recent years, as false information has become easier to spread and false statements and "fake news" can be actively created to advertise [2] or cause dissent. Currently, there are approximately 160 fact checking organizations globally [3], and several online based fact checking applications, however false information continues to spread on the internet. Of the fact checking organizations, many launched in recent years. Between the years 2014 and 2015 alone over 50 were launched [1]. However, these groups are not always able to respond in time to new false stories that are spreading on social media, so there is a need to automate the process.

NLP Methods

Possibly the simplest and most common method used in NLP tasks is to use n-grams. N-grams are used in statistical methods and are pairings of n words. Bigrams (pairs of 2 words) and trigrams (sets of 3 words) are used most frequently. In a similar vein, entire sentences and parsed sentences can be grouped like this, as well.

Neural Networks are also popular with NLP tasks. These are black box methods that require extensive training data to effectively work on real data. Layers of simple perceptrons that take in a 0 or a 1 and output the same are created with connections between each perceptron in adjacent layers. Weights between the perceptrons in each layer are learned, then over time the entire system learns to classify input. NLP has seen the advent of bi-directional long short-term memory neural networks in recent years. These networks take the input normally and reversed and have the ability to use information from previous inputs when generating their output.

Broadly speaking, NLP methods work on matrices and vectors. Text is tokenized where each word is a token and, with this, sentences, phrases, and other groupings of words are converted into vectors. Combining the vectors allows for the creation of matrices. Combining vectors creates two dimensional matrices that represent token occurrences over multiple word groupings and facilitates the comparison of one word grouping to another.

2.2. Claim Identification and Extraction

Some methods of automated fact checking assume that a claim is given as input, however in the context of fully automated fact checking this will not be the case. The three sections below discuss using machine learning to search for check-worthy claims in text and extracting meaningful data from them, then converting them into machine-readable formats, as well as a mathematical method that achieves both steps when working with statistical properties.

2.2.1. Searching for Claims

This first step is not always implemented in autonomous fact checkers. For example, researchers in [6] assume that their model is provided claims to check, feeding them directly into a long short-term memory (LSTM) neural network. Claims are the first input to their system. However, this cannot always be the case, as some autonomous fact checking applications are designed to work on articles or on comment sections of social media, so the first step must be to determine what is and is not a claim, and then whether or not that claim is worthy of being fact checked.

Inconveniently, even though research has gone into claim detection, there is no formal definition of what a claim is yet. Instead, the definition of a claim is defined by researchers on a per-project and organization basis [3]. This causes it to be difficult to replicate the work of other researchers.

Full Fact is one fact checking organization that collaborated with researchers to work on a claim detecting algorithm. As of the 2015 UK election, their definition of a claim was "an assertion about the world that can be checked" [3]. This definition is too broad when developing an autonomous fact checker, so the researchers defined a typology that would

capture the different types of claims. There are seven major categories of claims, each with several subcategories in their model.

Annotated claims based on the classification system defined in Table 1 were provided by volunteers that were keen on fact checking [3]. These categories reduce into binary Claim/Not A Claim to ease and speed up initial classification, as seen in Table 2.

Table 1. Claim typology with examples adopted from [3].

Main Category	Subcategory	Example
Not a claim		"Give it all to them, I really don't mind."
Other	Other	"Molly gives so much of who she is away throughout the film."
	Support/policy	"He has advocated for a junk food tax."
	Quote	"The Brexit secretary said he would guarantee free movement of bankers."
	Trivial claim	"It was a close call."
	Voting record	"She just lost a parliamentary vote."
	Public opinion	"A poll showed that most people who voted Brexit were concerned with immigration."
	Definition	"The unemployed are only those actively looking for work."
Quantity	Current Value	"1 in 4 people wait longer than 6 weeks to see a doctor."
	Changing quantity	
	Comparison	
	Ranking	
Prediction	Hypothetical statements	"The IFS says that school funding will have fallen by 5% by 2019."
	Claims about the future	
Personal experience	Uncheckable	"I can't save for a deposit"
Correlation/causation	Correlation	"Tetanus vaccine causes infertility"
	Causation	
	Absence of a link	
Laws/rules	Public Institutional procedures	"The UK allows a single adult to care for fewer children than other European countries."
	Rules/rule changes	

Table 2. Reducing 7 categories into 2 classes.

Claim	Not a Claim
Quantity	Personal experience
Correlation/causation	Other
Laws/rules	Not a claim
Prediction	

Researchers used sentence embeddings captured from InferSent, which differ from normal sentence embeddings as they factor word order [3]. Part of speech (POS) and named entities were retained in each sentence embedding, represented as the count of each in the corpus. Several classifiers were tested, all with default settings in Python's scikit-learn library. Logistic regression performed the best, however POS and named entity features did not make a difference in performance. The F1 score using only logistic regression was 0.83 [3].

Similar work was done by researchers in [7], who developed Claim Buster. The dataset was created using sentences from U.S. general election presidential debates, with three categories. These categories were: Non-Factual Sentences (NFS), Unimportant Factual

Sentences (UFS), and Check-worthy Factual Sentences (CFS). Examples of each sentence type are available in Table 3.

Table 3. Claim categories and examples adopted from [7].

Sentence Category	Example
NFS	But I think it's time to talk about the future. You remember the last time you said that?
UFS	Next Tuesday is Election day. Two days ago we ate lunch at a restaurant.
CFS	He voted against the first Gulf War. Over a million and a quarter Americans are HIV-positive.

The training dataset contained sentences from debates between U.S. presidential candidates prior to a general election, starting from the 1960 debate. The sentences were labelled by humans to fit into one of the three categories. Several features were also extracted to the sentences and added to the data, such as sentiment scores (positivity vs. negativity in a sentence), sentence word count, word frequency, POS tags for words, and named entity types. Using a random forest classifier to determine the most important features, the top 30 were selected for use in training a model [7].

Several supervised learning methods were tested; however, a support vector machine (SVM) had the best performance overall. On the CFS class alone, the SVM correctly classified the sentence 72% of the time.

2.2.2. Claim Extraction

The above section explains how claims are found, but not how they can be parsed to be comparable to whatever the researchers consider a valid source of truth is. Popular reductions are to convert claims into triplets such as {London, capital of, UK} or sets of short sentences as these tend to represent the claims themselves better [8]. In the case of the latter, the claims themselves can be used as input moving forward, assuming that they are short enough. However, converting the detected claims to triplets requires some extra work.

The basic form of a triplet is: {subject, predicate, object}. Seemingly, the most efficient triplet extracting algorithms require that sentences be parsed into tree structures, where POS tags are applied to words and syntactic structure is added. Sentences are represented by tree parsers as a root with three children: a noun phrase (NP), a verbal phrase (VP) and the full stop (.). The root of the tree is denoted as S [9].

To extract this information, researchers in [9] utilized existing parsers, namely the Stanford Parser and OpenNLP. The algorithm searches for the first noun in the NP and assigns that as the main noun, then it associates attributes to it based on available adjectives and related subtrees and uses this as the subject. Next, it searches for the VP predicate by finding the deepest verb in the VP subtree, again associating adjectives and related subtrees as attributes. This becomes the predicate in the triplet. Finally, the object is defined by searching by looking for NP, participle phrases (PP), and adjective phrases (ADJP) that are siblings to the VP. Attributes are extracted and applied here, as well.

With the above algorithm, the sentence "A rare black squirrel has become a regular visitor to a suburban garden" results in the triplet {squirrel, become, visitor} with attributes found in Figure 1.

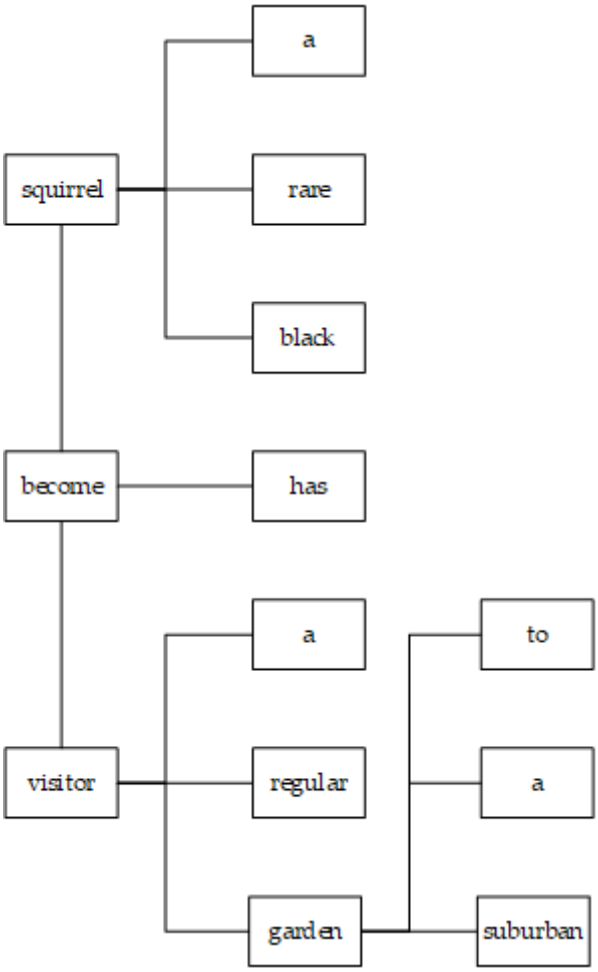

Figure 1. Triplet conversion of the sentence "A rare black squirrel has become a regular visitor to a suburban garden".

2.2.3. Mathematical Method for Statistical Properties

Researchers in [10] developed a method to identify and extract simple claims about statistical properties, such as the population of a nation. The claim identification portion relies on discovering textual patterns between an entity and a numerical value used to express the property of interest [10]. When attempting to extract information about the population of a nation, several different wordings are valid and can be seen in the left column of Table 4. If these were examples from textual sources, the values from those sources appear in the right column of Table 4.

Table 4. Various phrasings and example data for the population of several nations.

Pattern	Data
The population of *X* is *y*	France: 66,000,000 Russia: 140,000,000 Iceland: 325,000
X's population is estimated at *y*	France: 66,030,000 Russia: 140,000,000
X has *y* inhabitants	Russia: 140,000,000 Iceland: 300,000

To train their mathematical method, a set of entity-value pairs are required that act as a knowledge base (KB) containing the actual values of the properties, in this case

population. At the time of their experimenting, these values were: France: 66,028,467, Russia: 143,700,000, Iceland: 325,600. A set of textual patterns with their respective values are also required, this is the data in Table 4 as well as others, such as the pattern "*X*'s inflation rate is *y*" with data "France: 0.9, Iceland: 4.0" [10]. Because the values can be estimated in the text, the method aims to find patterns that predict the values well rather than finding patterns that state their exact values. The algorithm ranks all text patterns according to how well they predict the values for each property, then greedily selects those patterns until the aggregate predictions by the patterns stop improving.

To rank the patterns for various properties, the predicted entity-value (EV') pairs are compared against the actual entity-value (EV) pairs using the mean absolute percentage error (MAPE) as seen in the following equation:

$$MAPE(EV,\ EV') = \frac{1}{|E|}\sum_{e\in E}\frac{|v_e - v'_e|}{|v_e|} \tag{1}$$

where E and E' refer to the entity sets in EV and EV' respectively. The v and v' refer to the values for each entity, e. This equation only utilizes the values for entities in both E and E', so when calculating with the phrase "*X* has *y* inhabitants" from Table 4 only the values for Russia and Iceland are used.

In order to alleviate an issue where the MAPE equation would improperly score a wrong pattern highly, such as "*X* has had *y* tourists this year" with France: 66,000,000, the MAPE equation is adjusted to take into account the number of values used in the calculation so that the end user or designer of a system can adjust how important this factor is. This adjusted MAPE equation is seen the following equation:

$$adjustedMAPE(EV,\ EV') = \frac{c}{c+N}MAPE(EV, EV') \tag{2}$$

where N is the number of values used in calculating the original MAPE and c is the adjustable parameter. Setting c to a lower value puts higher importance on the number of values, making the eventual algorithm seen in Figure 2 more reliable [10].

The algorithm that uses the MAPE equation can be seen in Figure 2. To start, it makes a best guess at the value given the KB and the textual pattern. This guess is the mean of the training data, their median, or 0, whichever produces a lower MAPE score. Next, all patterns are ranked based on their MAPE score for the property in question, and this ranking is iterated over. During each iteration, the MAPE score is calculated for each pattern combined with the entity in question and a list is populated of patterns that predict the value well. As soon as the MAPE value is increased with a possible pattern, the algorithm exits, and the selected list of patterns is returned [10].

Pattern generation for this method can be accomplished as a batch process. Using Freebase, now known as WikiData, as a KB, queries can be generated for statistical properties that exist for many entities, and then these queries can be sent to search engines. The top results can be downloaded and parsed with an NLP toolkit and patterns can be extracted from them. This method was used by researchers in [10] when developing this model. With their testing, this algorithm achieved an overall precision score of 0.6, with strong variation across different properties [10].

2.3. Sources of Evidence

Various research projects into automated fact checking have used several different sources of evidence to compare claims to. This section enumerates these methodologies (listed below) as well as explaining how they have been curated by various researchers. The types of evidence discussed here are fact databases, the internet, other external sources, as well as the originator of the claim and the language that they use. Each source receives its own section below.

```
Input  : Entity-values for property
         EV = {(e1, v1), (e2, v2), …}
         patterns P = {p1, p2, …}
         entity-values for pattern p: EVp
Output : Selected patterns Psel
1  : vdef = InformedGuess(EV)
2  : priorityQueue q = Ø
3  : foreach pattern p in P do
4  :         push(q, (p, MAPE(EV, EVp)
5  : Psel = Ø
6  : pred = predict(E, Psel)
7  : mp = MAPE(EV, pred)
8  : while q not empty do
9  :         pattern p = pop(q)
10 :         P'sel = Psel U {p}
11 :        pred' = predict(E, P'sel)
12 :         mp' = MAPE(EV, pred')
13 :         if mp' > mp then
14 :                 break
15 :         else
16 :                 mp = mp'
17 :                 Psel = P'sel
18 : function predict(entities E, patterns Psel)
19 :         ev = Ø
20 :         foreach e in E do
21 :                 sum = 0
22 :                 count = 0
23 :                 foreach p in Psel do
24 :                         if (e, v) ∈ EVp then
25 :                                 sum += p{e}
26 :                                 count++
27 :                 if count > 0 then
28 :                         ev = ev U (e, sum/count)
29 :                 else
30 :                         ev = ev U (e, vdef)
31 :         return ev
```

Figure 2. Claim identification algorithm adopted from [10].

2.3.1. Fact Databases

Fact databases store pre-checked claims or world knowledge and can sometimes be augmented by claims from trustworthy sources. The stored facts can be stored in numerous methods, from triplet representations to unaugmented, previously fact checked claims.

Triplet representations in databases have been used to perform fact checking on simple, numerical, claims by researchers in [10,11]. Researchers in [12] generated their machine learning dataset by downloading the subset of the Freebase database that relates to the "statistical_region" entity type. Next, for each region they formed a web query and downloaded the top 50 results, then extracted statistical triplets from the pages that relate to the region currently being processed. Researchers in [11] extended the researchers of [12] by including temporal tags to factor the changing of statistics with time.

Previously fact checked claims are another way that knowledge can be stored. This is the case for Claim Buster, where part of their fact checking process involves searching for closely related or identical claims that have already been fact checked by professionals [7]. These were curated from various fact checking websites, such as PolitiFact and AZCentral.

Researchers in [8] developed a dataset similar to this named FEVER, based on the introduction sections of Wikipedia pages. Their dataset was generated by human annotators who, given a sentence from Wikipedia, were asked to generate a set of claims containing a single piece of information, focusing on the entity that the page is about. They also generated mutations of those claims, the types of mutations being: paraphrasing, negation, entity substitution, and making the claim more general or specific [8]. This dataset is labeled for machine learning applications. Each claim is labeled as supported, refuted, or not-enough-info. The first two labels are in relation to any Wikipedia page an annotator could find.

Text, such as encyclopedia articles, verified news, and scientific papers can also be utilized in the fact checking process and be stored in a fact database. For example, researchers in [13] developed a dataset containing articles and journals. They then proceed to only utilize the headlines in their task of rumor debunking.

Entire corpora are available, as well. In the first Fake News Challenge, researchers in [12] processed entire articles in their task of stance detection. Stance detection, in this case, being the process of determining how much an article agrees with its headline.

2.3.2. The Internet

Like searching entire corpora for information, the internet itself can be used as a source of evidence when fact checking claims. Claim Buster utilizes web results to augment its fact checking when its existing claim database does not contain a similar enough claim. To implement this, a question generation tool is used, then valid questions are sent to the question answering engine Wolfram Alpha. Simultaneously, the questions are sent to Google and the snippet result is used. Google is also sent the claim itself, and the resulting pages are searched for similar claims. This information is sent directly to the end user [7].

2.3.3. Other External Sources

It is possible to achieve fact checking relying only on external sources, specifically just the snippets returned from Google and Bing searches. With the methods implemented by researchers in [6], parsing web pages yielded no better results than parsing only the snippet returned by searching a shortened version of the claim.

Wikidata is an online KB with a public API that allows users to query for information. Since it launched, it became one of the most active Wikimedia projects. As with all Wikimedia projects, data is updated and managed by the community. This is crowd sourcing data acquisition. Data is stored in property-value pairs, each property relating to an entire Wikidata page containing labels, aliases, and descriptions [14].

2.3.4. Originator and Language

While not evidence towards truthfulness, the originator of a claim and the language that is used in it can be used to augment the process by which a claim is determined to be true or false. Including the metadata of speaker's truthfulness history along with the claim itself has been shown to increase the accuracy of convolutional neural networks in spotting false statements [15]. The language of the claim itself is also useful in predicting veracity.

Trusted sources are more likely to use assertive words and less likely to use hedging words than untrusted sources. They are also more precise when describing events [16].

2.4. Claim Verification

The majority of methods used for automated fact checking are based on supervised machine learning [8]. This chapter is broken down into sections that each describes a different method of determining whether or not a claim is likely to be true or false. The methods explored are the language of the claim, comparing the claim to fact databases, comparing the claim to pre-checked claims, and comparing the claim to external sources.

2.4.1. Language Analysis

As described above, the language of a claim can indicate its veracity. Analysis indicates that false claims often arise from differences in phrasing rather than outright fabrications, so language itself is useful to analyze. For example, subjective words can be used to dramatize or sensationalize a story [16]. First and second person pronouns are used more often in less reliable news sources, as news writers generally try to appear indifferent. Words used to exaggerate are also used more by "fake news", whereas concrete figures are found more often in truthful news [16]. This information has been used to develop several models that take a claim and term frequency-inverse document frequency (TF-IDF) data as input and predicts the PolitiFact rating, a scale consisting of completely false, false, mostly false, half true, mostly true, and true. A similar set of models were designed that only outputs a true or false verdict on a claim. These models were trained on PolitiFact and similar websites. The models in question are an LSTM, Naïve Bayes, and Maximum Entropy. Results of the testing by researchers in [16] is seen below in Table 5. These models are not reliable when dealing with PolitiFact's 6 classes of truthfulness, however when simply determining if a statement is likely true or false the models perform better.

Table 5. F1 scores of various models by [16] when classifying claims based on their language.

	True/False		**6 Classes of Truthfulness**	
	Text	**Text + TF-IDF**	**Text**	**Text + IDF**
Naïve Bayes	0.44	0.58	0.16	0.21
Maximum Entropy	0.55	0.58	0.20	0.21
LSTM	0.58	0.57	0.21	0.22

Researchers in [17] take this further by including references to claims and how claims are cited. For example, if one source cites another and the second source is truthful, the citing source's likelihood of being valid is increased. The inverse also applies, as false stories will propagate through related sources whereas true ones will likely do the same. Their method utilizes a triplet format for claims and outputs believability scores. Specifically, these are subject-predicate-object (SPO) triplets, where the predicate in the triplet is a verb or verb phrase. Synonyms to these triplets are used going forward, using alternate wordings of the same statement. These are referred to as fact candidates. These candidates are assigned a believability score, which is a combination of its objectivity score and its co-mention score. The objectivity score is defined as:

$$O(f_i) = \log|D_i| \cdot \frac{\sum_{d_k \epsilon D_i} O(d_k)}{|D_i|} \quad (3)$$

where f_i is the fact candidate, D_i is a set of documents that the candidate is mentioned in, and each document where $d \epsilon D_i$ has objectivity $O(d)$. The document objectivity score $O(d)$ is calculated using a logistic regression classifier, and it can only be between 0 and 1, where 0 shows that the source is subjective [17]. Taking the logarithm of the absolute value of D_i ensures that many mentions of a candidate in low objectivity sources do not end up

with higher objectivity scores than those with fewer mentions in high objectivity sources, as well as making sure that candidates with many mentions in high objectivity do not have higher objectivity scores than those with a single mention in a higher objectivity source. The co-mention score aims to ensure that fact candidates mentioned in similar sources have similar believability scores. For example, if f_i is mentioned in several highly objective sources and f_j is stated in one highly objective source that also contains f_i, then f_j should have a higher believability score. This co-mention score is defined as:

$$u(f_i) = p(f_i) + \sum_{f_j \epsilon\ F} w_{ij} u(f_j) \tag{4}$$

where $p(f_i)$ is the normalized mention frequency of f_i, and w_{ij} is the propagation weight. This weight is the average of the objectivity of the sources that mention both candidates.

$$w_{ij} = average\big(O(d_k) : d_k \epsilon\ (D_i\ \cap\ D_j)\big) \tag{5}$$

To avoid unrelated co-mentioned candidates increasing each other's scores, w_{ij} can only be greater than 0 if f_i and f_j are on the same topic, defined by the verb in the triplet [17]. To leverage inter-dependencies among related co-mentioned fact candidates, the solution is modeled as a graph ranking method, where each candidate is a node with edges between each pair of related candidate nodes f_i and f_j with w_{ij} as the edge weight. With this, Equation (5) can be reformulated as $u = M * u$, where u is the co-mention score vector and M is a Markov matrix [17]. To implement this, the equation is iteratively applied until the change in the score is less than a pre-defined threshold. This results in final co-mention scores of fact candidates.

Finally, the overall believability score of a fact candidate is defined as:

$$B(f_i) = \gamma O(f_i) + (1 - \gamma) u(f_i) \tag{6}$$

where γ is a weighting parameter between 0 and 1, controlling the relative importance of the objectivity and the co-mention scores [17]. Pseudocode for this entire algorithm is in Figure 3. As input, it takes a set of fact candidates based on the original claim, as well as a knowledge base, a set of subject-verb-object triplets, and the internet. It outputs a set of believability rankings for all of the input fact candidates.

2.4.2. Comparing to Fact Databases

Working with their FEVER dataset, researchers in [8] developed a system to perform fact checking based on it. This system consists of three components: document retrieval, sentence-level evidence selection, and textual entailment. Document retrieval was implemented by the existing DrQA system, a k-Nearest-Neighbor algorithm based on unigram and bigram TF-IDF vectors. Sentence selection was implemented using a ranking system relying on TF-IDF vectors, as well.

Textual entailment was implemented using an existing state of the art system that had open-source code and did not require input text be syntactically parsed.

As introduced above, researchers in [8] use a KB to check numerical claims. This knowledge base consists of a set of un-normalized tables, translated into Entry-Predicate-Value triplets. The process of fact checking is broken down into three steps: linking named entities in the claim to entities in the KB, filtering entries with the same entities using relation matching, and then determining if the claim is relevant. Finally, if the claim is relevant, the triplet from the KB is compared to the claim to determine truthfulness.

2.4.3. Comparing to Pre-Checked Claims

A KNN algorithm can be used to assess semantic similarity between statements, as well [18]. This was implemented, not for fact checking, but simply for determining semantic textual similarity (STS) by researchers in [19]. Textual similarity can range from

exact equivalence to completely unrelated, so these researchers grade similarity on a scale from 0 to 5, with 5 being the most similar. This is seen in Table 6.

Input : Set of fact candidates F
Knowledgebase KB
SVO corpus C
Web W
Output : Set R of rankings for each fact f_i in F
1 : $R = \emptyset$
2 : **while** F not empty **do**
3 : f_i = pop(F)
4 : $alts$ = getAlternatives(f_i, KB, C, W)
5 : PriorityQueue $R_i = \emptyset$
6 : **foreach** alternativeFactCandiate f_j in $alts$ **do**
7 : B_j = getBelievabilityScore(f_j)
8 : push(R, (f_j, B_j))
9 : B_i = getBelievabilityScore(f_i)
10 : push(R_i, (f_i, B_i))
11 : $R = R \cup R_i$
12 : **return** R

Figure 3. Pseudocode adopted from [17] for determining believability of a claim based on its language and its existence in other sources.

Table 6. Semantic similarity ratings for sentences adopted from [19].

Similarity Rating	Sentences
5	The bird is bathing in the sink. Birdie is washing itself in the water basin.
4	In May 2010, the troops attempted to invade Kabul. The US army invaded Kabul on 7 May last year, 2010.
3	John said he is considered a witness but not a suspect. “I am not a suspect anymore,” John said.
2	They flew out of the nest in groups. They flew into the nest together
1	The woman is playing the violin. The young lady enjoys listening to the guitar.
0	John went horseback riding at dawn with a whole group of friends. Sunrise at dawn is a magnificent view to take in if you wake up early enough for it.

Two scores were generated during their research with two separate methods. The first, as with Claim Buster, is generated merely by comparing token similarity. The second involves generating XML for each sentence containing named entities, entity classifications, and dates. The similarity score was then calculated over several dimensions of the data. General similarity was generated via the cosine similarity of the TF-IDF of the tokens from all fields [19]. TF-IDF is a numerical statistic intended to reflect how important a word is to a document or a corpus [20]. Author similarity was the cosine similarity of the TF-IDF vectors for the authors. Involved persons, time period, and physical location similarity were calculated with the cosine similarity of TF-IDF vectors of those fields. Event similarity

was the cosine similarity of TF-IDF vectors of all verbs. Subject and description were the cosine similarity of TF-IDF vectors of their respective fields. In the TF-IDF vectors, the IDF values were calculated from a subset of the overall data used for the entire project [19].

There are downsides to comparing input claims to previously checked ones. To start, previously checked claims must already exist and be similar enough to be detected. This involves storing checked claims and having a suitable algorithm for searching through them. It also requires that the checked claims already exist, generally having been checked by a human. This leaves the issue of not being able to keep up with quickly generated false statements on social media and requiring humans to perform the initial fact checking.

2.4.4. Comparing to External Sources

NLP based fact checking can be achieved using only external sources, without existing fact databases, as demonstrated by researchers in [6]. Their system takes a claim and searches for support information on the Web. The steps involved in this are: external support retrieval, text representation, and veracity prediction.

External support consists of generating a query from the input claim and sending it to a search engine. This query is a shortened version of the input claim, as claims tend to be two sentences long on average [6].

To accomplish this, words are ranked using TF-IDF. The IDF values were computed based on a 2015 Wikipedia dump and the English Gigaword. Only the verbs, nouns including named entities, and adjectives are considered. The generated queries are 5–10 tokens long and are sent to the Google and Bing search engines. Examples of these generated queries are visible in Table 7. Finally, the snippets at the top of the results and the URLs from reliable domains are extracted. The reliability of domains was hand generated by these researchers. If the query did not return results, the last token is iteratively dropped until results are returned [6].

Table 7. Generated queries from claims, named entities are in single quotes.

Claim	Generated Query
Texas teenager Ahmed Mohamed was arrested and accused of creating a "hoax bomb" after bringing a home-assembled clock to school	'Texas'—'Ahmed Mohamed'—hoax—bomb—clock—arrested—accused
Is the popular casual dining chain Chipotle closing all locations soon?	'Chipotle'—dining—locations—popular—closed

Next, comparable representations of the claim and external support data must be generated. Three similarities are calculated between the claim and snippet and between the claim and web pages. They are cosine with TF-IDF, cosine over embeddings, and containment. Embeddings are calculated as the average of the embeddings of the words using pre-trained embeddings from GloVe [21]. Also, to compare the Web pages to the shorter claim, the pages are split into sentence triplets and then the similarities are calculated between the triplets and the claims, keeping only the highest scoring triplet. The maximum and average of the three similarities over the snippets and over the web pages are used. The embeddings of the claim, the best-scoring snippet, and the best-scoring sentence triplet from a Web page are used as features. These embeddings are calculated as the average of the embeddings of the words in the text and by using LSTM encodings within a deep neural network [6]. The LSTM structure can be seen in Figure 4.

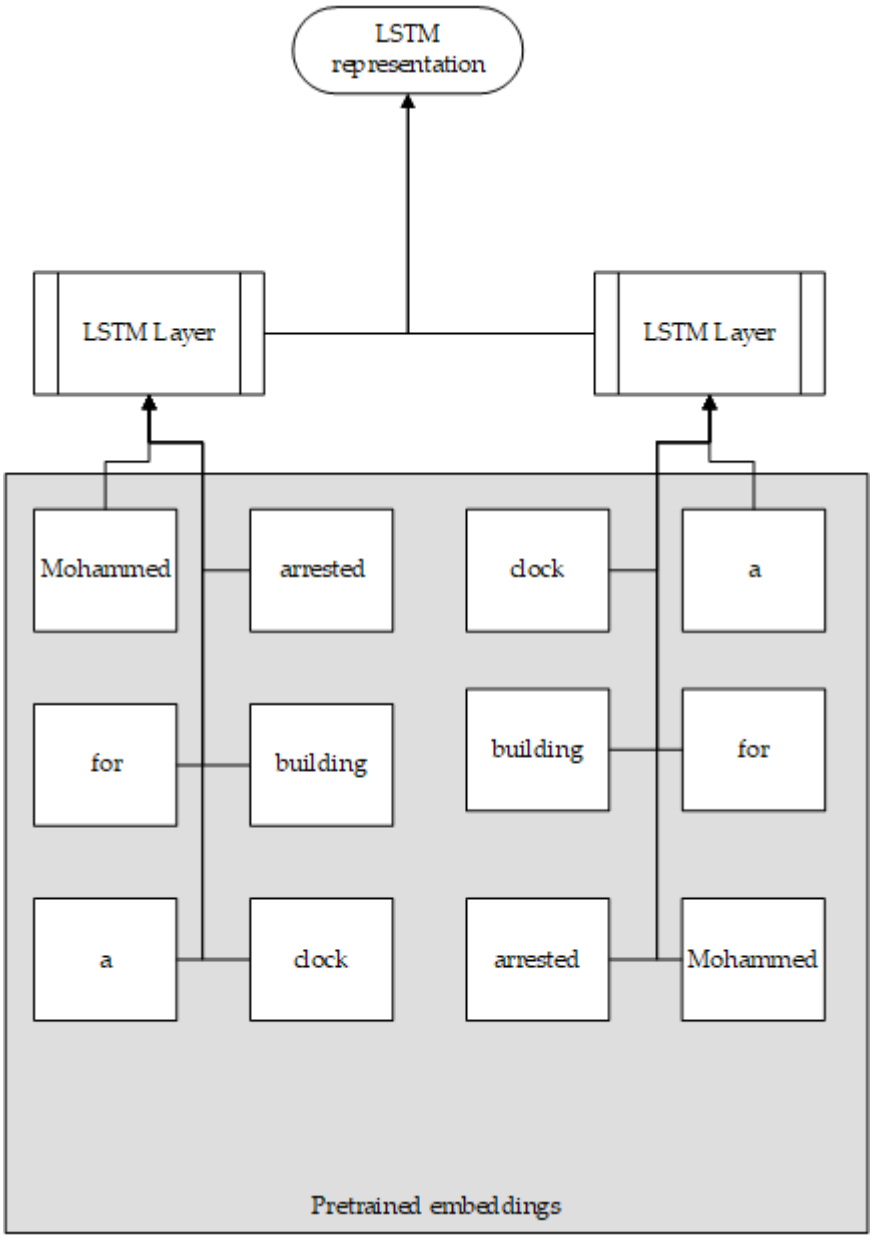

Figure 4. LSTM representation for each external source and claim adopted from [6].

Finally, the veracity of the claim is predicted using several classifiers: a neural network, a support vector machine, and a combination of both models. The neural network consists of five LSTM sub networks, one for each of the text sources from two search engines: the claim, the Google web page, the Google snippet, the Bing web page, and the Bing snippet. The overall structure of this is shown in Figure 5. The claim is fed into the network without changes. As multiple snippets are possible, only the highest scoring one is used. The highest scoring triplet from the web pages is also used. Similarity features are also added to the network as described above. These vectors go through a compact hidden layer, then through a SoftMax output unit that is used to classify the claim as true or false. Each of the sub-LSTM networks is bidirectional and takes as input a sequence of word embeddings [6]. The support vector machine uses an RBF kernel with the same input as the neural network, however the word embeddings are calculated by averaging rather than using a bidirectional LSTM. The combined method augments the SVM by using the hidden layer of the NN as input to the SVM. The bidirectional LSTM embeddings are also

an input to the SVM. This combined method was the most successful of the three with an accuracy of 0.8 [6].

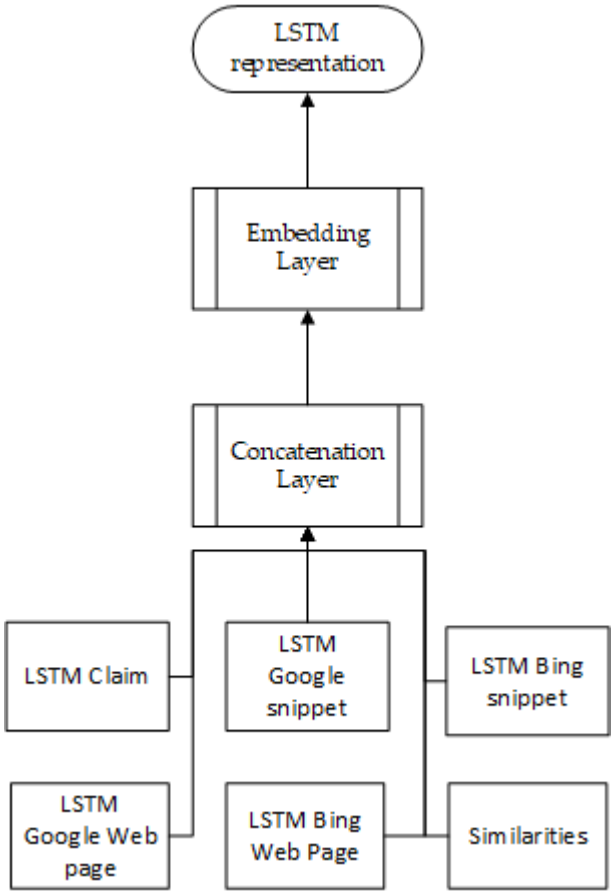

Figure 5. Overall LSTM architecture adopted from [6].

2.5. Aggregate Method: Claim Buster

The current state of the art system for autonomous fact checking is Claim Buster. It uses a KB of previously fact-checked claims, the Web, and a question answering system to deliver a veracity verdict to its end users [7]. This section discusses the portions of the Claim Buster system that have not yet been addressed and is broken down into subsections for each portion. These are question generation and answering, database searching, and external support retrieval and comparison.

2.5.1. Question Generation

It is possible to implement fact checking without language analysis or storing pre-checked facts, but rather by simply searching the web and parsing the results. This is a part of Claim Buster's system, as well as others. Claim Buster uses a question generation tool and then forwards the valid questions to the question answering engine Wolfram Alpha via an API, as well as Google. Wolfram's response and Google's answer are extracted and compared, then sent to the end user [22].

Claim Buster uses an algorithm developed by researchers in [23] to generate questions. These questions are generated via an algorithm consisting of three modular stages, NLP transformation, a Question Transducer, and a Question Ranker, as seen in Figure 6 [23].

This algorithm takes in source sentences, or claims in the case of Claim Buster, and first runs NLP transformations on them to derive new declarative sentences. This is changing sentences such as "Selling snowballed because of waves of automatic stop-loss orders, which are triggered by a computer when prices fall to certain levels" into "Automatic stop-loss orders are triggered by computer when prices fall to certain levels" [23]. The

rules defined in this stage may not apply to some input sentences, so they may not be transformed.

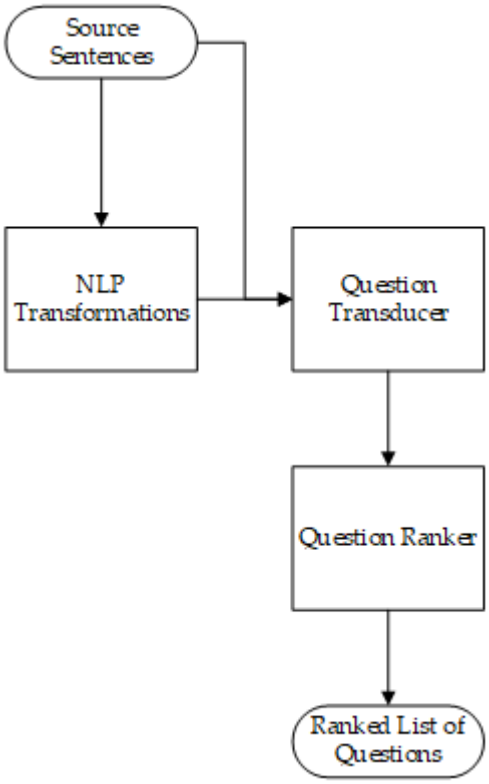

Figure 6. Question generating stages from adopted [23].

Next, declarative sentences are passed through a Question Transducer. This generates a set of possible questions based on its input sentence. It does this by identifying answer phrases, such as "Africa" in the sentence "Kenya is located in Africa," and converts them into question phrases. Answer phrases can be noun or prepositional phrases, enabling who, what, where, when, and how much questions. All possible questions are generated by (1) marking phrases that cannot be answer phrases, (2) removing answer phrases and generating question phrases for it, (3) decomposing the main verb, (4) inserting the subject and auxiliary verb, and (5) inserting a question phrase. This process is shown in Figure 7.

Unmovable phrases are identified by hand-generated grammatical rules. Possible answer phrases are iterated over, and each is transformed into several question phrases. Generating question phrases is accomplished by using the entity types, i.e., person, location, date, etc., and replacing them with an appropriate question word, i.e., who, where, when, etc. Decomposing the main verb does not happen for all sentences, only when an auxiliary verb or modal is not present. It decomposes the main verb into the appropriate form of "do" and the base form of the main verb and then modifies the syntax tree structure of the question appropriately. For example, the sentence "John saw Mary" transforms into "John did see Mary" before finally resulting in "Who did John see" rather than "Saw John Mary" [23]. Subject–auxiliary inversion occurs when the question is a yes–no question or when the answer phrase is a non-subject noun phrase. This allows for more natural questions such as "Following Thomas Jefferson, who was elected the 4th president" rather than "Who, following Thomas Jefferson, was elected the 4th president." When the question is not a yes–no question, the generated question phrases are inserted into a copy of the syntax tree following any leading sentence-level adjunct phrases [23]. Finally, proper formatting and punctuation is added as a post processing step.

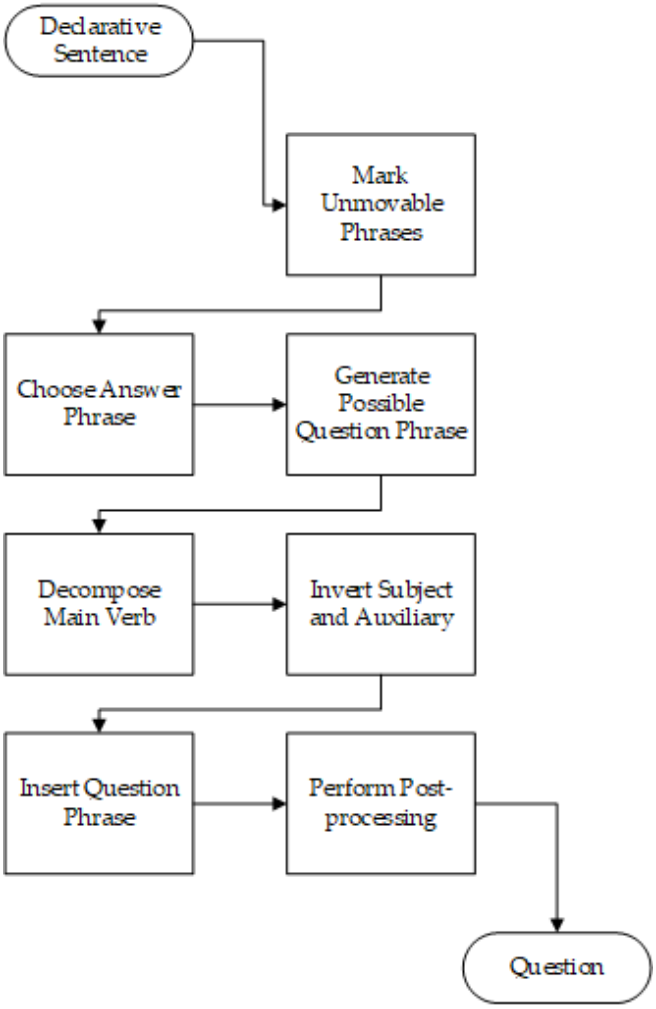

Figure 7. Converting a declarative sentence to a question adopted from [23].

The last step in the question generating algorithm from [23] is to pass generated questions through a Question Ranker prior to outputting a list of valid questions. This step is needed because the process thus far has been to generate as many questions as possible without considering whether they are easy to understand or even grammatically correct. A discriminative reranker based on a logistic regression model is used for ranking questions [23]. It defines a probability of acceptability given a question and a source text. The researchers trained a Boolean classifier, where any type of deficiency in the question resulted in an unacceptable outcome and an aggregate set of classifiers that compute the probability of a question being acceptable for each of the deficiencies that they enumerated, combining the results. The enumerated deficiencies can be seen in Table 8.

Table 8. Question Deficiencies enumerated adopted from [23].

Deficiency	Description
Ungrammatical	The question does not appear to be a valid English sentence.
Does not make sense	The question is indecipherable (e.g., Who was the investment?).
Vague	The question is too vague to know what it is talking about (e.g., What did Lincoln do?).
Obvious answer	The correct answer is obvious or clearly yes.
Missing answer	The answer to the question is not in the original input sentence.
Wrong question word	The question would be acceptable if the correct question word were used (e.g., What is the Eiffel Tower located?).
Formatting	Minor formatting errors with capitalization, punctuation, etc.
Other	Other reasons that a question may be unacceptable.

2.5.2. Question Answering

While Wolfram Alpha and Google's methods for question answering are not available for study by the public, other methods of question answering are published. The AnswerBus system developed in [24] may be similar to the black box solutions that Claim Buster utilizes.

AnswerBus is an open-domain question answering system, meaning it is not restricted to a subset of question areas, that is based on sentence level information retrieval [24]. It uses web searches to provide users with answers to their natural-language questions entered in several possible languages. However, even though it can accept questions in several languages, it only searches the Web in English with an automatically translated question. The first step of the AnswerBus system is to use a simple language recognition module to determine whether the input question is in English and, if it is not, to use the AltaVista translation tool BabelFish to translate it into English [24]. This is followed by four main steps: selecting search engines to use and forming specific queries per engine based on the question, sending the queries to the search engines and retrieving the top results, extracting sentences that potentially contain answers from these documents, and ranking the answers and returning them along with URL links for the user [24]. The structure of the AnswerBus system is shown in Figure 8.

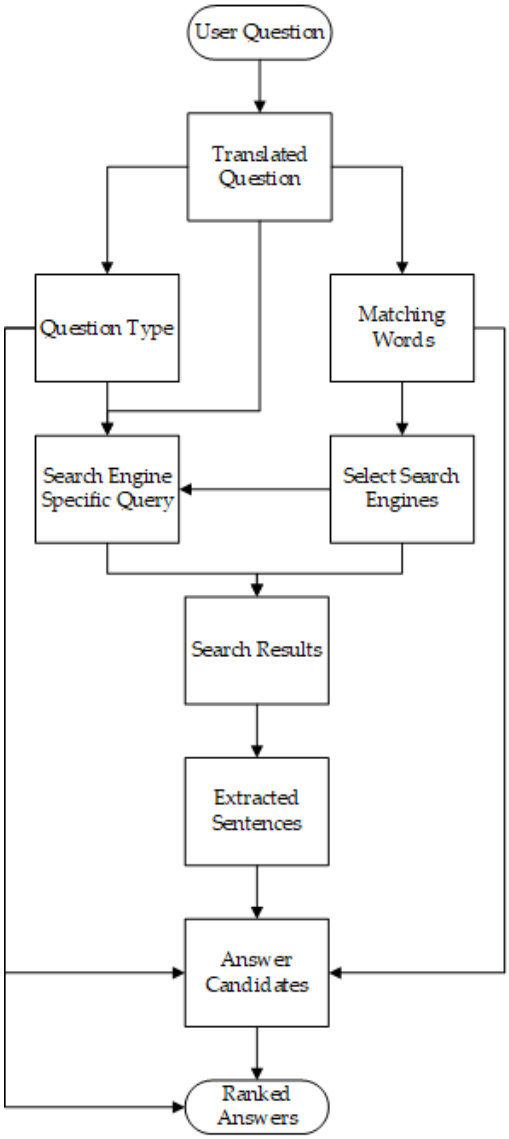

Figure 8. AnswerBus architecture adopted from [24].

Five search engines are available for AnswerBus to select from. They are Google, Yahoo, WiseNut, AltaVista, and Yahoo News. The various engines are used as each search engine has its own domain that it is optimal for searching. For each question, AnswerBus chooses two or three search engines to work with. The engine selector is a simple machine learning model based on result counts for each word in a set of 2000 training questions. The search engines that would return the most results for a query are selected. For example, if Google returns 5 results for word1 but Yahoo only returns 4, the selector is more likely to choose Google as a search engine for a query that contains word1 [24].

Next, search engine specific queries are generated. It is useful to send differing queries to each search engine so that they return optimal results. Optimal, in this case, not being many but rather the most useful in the shortest amount of time possible [24]. For example, if a user is inquiring about the height of Mount Everest, the optimal query for Google may just be the words "Mount Everest" so that it returns the Wikipedia page quickly. As computing speed is most optimal for AnswerBus, single and simple queries are preferred over expanded ones. To generate a query, several methods are combined. First, functional words are deleted. Examples of these are prepositions, determiners, pronouns, conjunctions, interjections, and discourse particles. Phrases that act as functional words such as "kind of" are also deleted. The in-progress query is then further shortened by deleting frequently used words in the query. Finally, some words are converted to another form. Generally, these are verbs, such as "end" to "ended" [24].

At this point, the search-engine specific query is sent to the search engines and the AnswerBus system downloads the top results. Sentences are extracted from the documents using an HTML parser and answer candidates are searched for using the formula below.

$$w \geq \left\lfloor \sqrt{W-1} \right\rfloor + 1 \tag{7}$$

In this equation, w is the number of matching words in the document sentence and W is the number of matching words in the query. For example, if a query contains three words, then an answer candidate must at least have two of them. If a sentence meets the condition in the formula, it is scored based on the number of matching words it contains [24].

Finally, answer candidates must be ranked and returned to the user as output. To start, the sentences that have a score of 0 from the previous step are dropped. Next, the scores for the remaining sentences are refined with influence from several steps. First the question type is determined using a pre-defined Question/Answer database that maps question types such as "Who is . . . ?" and "How far . . . ?" to their appropriate answer units. With this information, answer candidates with a mile unit can be removed from consideration for a "Who is. . . . ?" question. Next, named entities are extracted and sentences with entities related to the question are given a higher score. For example, a "How much money . . . ?" question will cause sentences with a currency entity to receive a score boost. Conference resolution, determining what words such as "he" and "they" are referring to, is applied towards the scores after this. AnswerBus does not employ full conference resolution in the name of speed, opting instead to solve conferences in the adjacent sentences. The later sentences receive part of its score from the previous sentence when a conference is resolved. Another factor in answer candidate score is how its rank in the search results. The first result receives a higher score than the second one, and so on. Finally, redundancy is dealt with among the highest scoring answer candidates, leaving only the top non-redundant results. These results are returned to the end user [24].

2.5.3. Database Searching

Part of the ClaimBuster system's method for fact checking is to compare detected claims against previously fact checked ones. This is the duty of the Claim Matcher component [7]. It searches the repository described in the previous chapter using two approaches to measure the similarity between the input claim and the claims in the repository. The first is based on similarity of tokens between sentences, and the other is based on semantic

similarity. Token similarity is found via an Elasticsearch server, and Semilar is utilized for detecting semantic similarity [7]. When both servers have returned results, these are combined and sent to the end user.

2.5.4. External Support Retrieval & Comparison

The Claim Checker portion of the Claim Buster system collects supporting or debunking evidence from knowledge bases and the Web [7]. It does this by sending the raw input claim to Google as a search query. It then parses the web page at the top result and searches for sentences matching the input claim. These sentences and their surrounding companions are grouped into a context and returned to the end user. This process is similar to the one developed by researchers in [6].

3. Related Task: Speaker & Source Analysis for Fact Checking

While not related to NLP, fact checking can be augmented by taking in metadata with a claim such as its context, its author, and its source. This is demonstrated in [25] with their method of fact checking in community question answering forums, such as StackOverflow, Yahoo! Answers, and Quora. This method was developed to check the veracity of answers on community answer forums.

This is taken as a supervised machine learning problem with a multifaceted data model including, as stated above, the answer context (what is said and how), the author profile (who wrote the answer in question), the rest of the community forum (such as StackOverflow and Yahoo! Answers), and external authoritative sources of information [25]. This is a classification problem where community answers are categorized into six different classes. These are listed in Table 9. When creating the training dataset for their model, only answers classified as good by an existing system were retained and re-annotated. A good answer is one that attempts to address the question, irrespective of its veracity [25].

Table 9. Answer categories, explanations, and examples adopted from [25].

Answer Category	Explanation	Example Question	Example Answer
FACTUAL—TRUE	The answer is true, and this can be manually verified using a trusted external resource.	I wanted to know if there were any specific shots and vaccinations I should get before coming over [to Doha]	Yes, there are; though it varies depending on which country you come from. In the UK, the doctor has a list of all countries and the vaccinations needed for each.
FACTUAL—FALSE	The answer gives a factual response, but it is false.	Can I bring my pit bulls to Qatar?	Yes, you can bring it but be careful this kind of dog is very dangerous.
FACTUAL—PARTIALLY TRUE	Only part of the answer could be verified.	I will be relocating from the UK to Qatar [. . .] is there a league or TT clubs/nights in Doha?	Visit Qatar Bowling Center during Thursday and Friday and you'll find people playing TT there.
FACTUAL—CONDITIONALLY TRUE	The answer is true in some cases and false in others, depending on some conditions that the answer does not mention.	My wife does not have NOC from Qatar Airways; but we are married now so can I bring her legally on my family visa as her husband?	Yes, you can.
FACTUAL—RESPONDER UNSURE	The person giving the answer is not sure about the veracity of their statement.	Will there actually be a raid on Area 51?	I heard that this was going to happen, not sure if anyone will show up though.
NONFACTUAL	The answer is not factual but is rather an opinion, advice, etc. that cannot be verified.	Should I buy this motorcycle?	I made the mistake of buying one like that and I am still paying for that mistake 3 years later, so I do not recommend it.

Five main facets are used when classifying an answer. These are: the user profile, the language of the answer, the context of the answer, external sources of information, and evidence from within the forum itself. These are discussed below [25].

3.1. User Profile

The user profile characterizes the user who posed the answer. This includes their post categories, such as posts in the subject of Computers, Education, etc. The answers in each category of the dataset are used twice, once with the raw number and a second time normalized by the total number of answers that the user has posted. The user's quality of posts is also considered. This is generated by another classifier and stored as several statistical indicators of the ratio of good and bad answers that the user has generated. The user's activity is included and is modeled with the number of answers posted, the number of distinct answers, the number of questions posted, the number of days since registering in the forum, and the number of active days. The time of posts is also factored in as working hours, after work, night, early in the morning, and before noon. The day of the week is also factored in as a weekday/weekend binary feature.

3.2. Answer Context and Language

The answer context models how the answer is written. The linguistic bias, subjectivity, and sentiment are included in this portion of the model. Several linguistic markers in the answer are searched for and a bias is derived. The credibility of the answer is generated using features such as the number of URLs, images, emails, and phone numbers in the answer, and the number of pronouns. Interrogative sentences, noun vs. verb vs. adjective vs. adverb vs. pronoun counts, and the number of words not in word2vec's [26] Google News vocabulary are useful here, as well. Semantic embeddings from Google News are included, as well as semantic embeddings from the database to further add features to the answer context.

3.3. External Evidence

External evidence is gathered via a method similar to the one presented in [6] described above. The Web is used to find evidence via the Bing search engine. The source of this evidence is taken into account, and reputed sources, forum sites, and other websites all have veracity scores stored in separate features.

3.4. Intra-Forum Evidence

This encapsulates the "where it was said" feature when modelling the answer in question. The current thread is analyzed and the cosine similarity between the current answer and other good answers in the thread is calculated. The idea is that if one answer is correct, other correct answers will state the same facts. Forum level evidence from all posts is searched for, and evidence from high quality posts is searched for a second time.

4. Discussion

From this research, automatic fact checking using NLP is possible with today's technology. However, the current methods only perform well when the domain of facts is restricted in one way or another. Regarding generalized fact checking, methods that utilize powerful search engines on the internet perform the best. A comparison of the accuracies is available in Table 10. The related task of analyzing the speaker and source of a claim is included here, as well. This final method outperformed only utilizing search engines, though this is expected as more types of data are available to the fact checking algorithms.

Table 10. Accuracies of the various methods of claim verification [6,8,17,19,25].

Method	Accuracy Score
Language Analysis	0.78
Comparing to Fact Databases	0.51
Comparing to Pre-Checked Claims	N/A—evaluation of STS is an open issue [19]
Comparing to External Sources	0.8
Speaker & Source Analysis	0.83

There are several future challenges that must be tackled prior to implementing NLP based fact checking on a large scale. Possibly the largest problem shared by many of the above methods is that they are designed to work with, and only work with, claims and evidence written or extracted from English text. Fact checking is needed in more places than just the English-speaking world, however implementing automated fact checking has been focused primarily on just that. The only method described above that handles multiple languages is in [24] for question answering. This method can take as input several languages; however, it relies on a translation service to convert input queries into English. While beyond the scope of this paper, there is work in the area of multilingual NLP. The MultiMix framework has been developed by researchers in [27]. At a high level, it aims to solve cross-lingual problems with one known language and one unknown language. It achieves state of the art results with named entity recognition and natural language inference.

Another glaring deficiency in the methods discussed above is their accuracy. Several methods scored low in precision, recall, and accuracy. These were as low as 20% and lower. Very few achieved high scores, and even those rarely reached 90%. If automated fact checking using NLP is to become more mainstream, the ability of the systems to accurately perform their tasks must be increased greatly. The best performing methods discussed here were the ones that used external sources, such as the Web and search engines. These all ranked in the mid 70%'s.

On the subject of using the Web for evidence retrieval, a third challenge is to make fact checking systems operable in real time without humans first determining whether a claim is true or false and without relying on pre-existing fact datasets. While methods that utilize prechecked claims and knowledge bases are valid, they are not scalable to social media where false rumors and "fake news" run rampant.

One more challenge is checking claims and using evidence that is not textual. For example, a claim such as "US population has grown by 33% in the past 10 years" may be only verifiable by looking at a graph, table, or list of populations by year and calculating the growth manually. NLP is required to parse the claim, however other technologies are needed to parse the evidence.

A final difficulty, related to the above, is the sheer complexity of human-based fact checking. At times, complex reasoning is required to determine the veracity of a claim and evidence from multiple sources must be combined. Even a simple statement such as "The UK, by leaving the EU, will take back control of roughly £350 million per week" could require a substantial amount of work in order to properly check it.

Author Contributions: Conceptualization, E.L., M.A.-K. and C.H.; methodology, E.L., M.A.-K. and C.H.; investigation, E.L., M.A.-K. and C.H.; writing—original draft preparation, E.L.; writing—review and editing, E.L., M.A.-K. and C.H.; supervision, M.A.-K. and C.H.; funding acquisition, M.A.-K. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially funded by Lewis University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Graves, L.; Cherubini, F. *The Rise of Fact-Checking Sites in Europe*; Reuters Institute: Oxford, UK, 2016.
2. Vargo, C.J.; Guo, L.; Amazeen, M.A. The agenda-setting power of fake news: A big data analysis of the online media landscape from 2014 to 2016. *N. Media Soc.* **2017**, *20*, 2028–2049. [CrossRef]
3. Konstantinovskiy, L.; Price, O.; Babakar, M.; Zubiaga, A. Towards Automated Factchecking: Developing an Annotation Schema and Benchmark for Consistent Automated Claim Detection. *arXiv* **2020**, arXiv:1809.08193.
4. Claim Buster. Available online: https://idir.uta.edu/claimbuster/ (accessed on 8 July 2021).

5. Shin, J.; Thorson, K. Partisan Selective Sharing: The Biased Diffusion of Fact-Checking Messages on Social Media. *J. Commun.* **2017**, *67*, 233–255. [CrossRef]
6. Karadzhov, G.; Nakov, P.; Marquez, L.; Barron-Cedeno, A.; Koychev, I. Fully Automated Fact Checking Using External Sources. *arXiv* **2017**, arXiv:1710.00341.
7. Hassan, N.; Zhang, G.; Arslan, F.; Caraballo, J.; Jimenez, D.; Gawsane, S.; Hasan, S.; Joseph, M.; Kulkarni, A.; Nayak, A.K.; et al. ClaimBuster: The first end-to-end fact-checking system. In Proceedings of the VLDB Endowment, Munich, Germany, 28 August 2017; Volume 10, pp. 1945–1948. [CrossRef]
8. Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; Mittal, A. FEVER: A Large-Scale Dataset for Fact Extrac-tion and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 809–819. [CrossRef]
9. Rusu, D.; Dali, L.; Fortuna, B.; Grobelnik, M.; Mladeni, D. Triplet Extraction from Sentences. In Proceedings of the 10th International Multiconference, Information Society-IS, Ljubljana, Slovenia, 8–12 October 2007.
10. Vlachos, A.; Riedel, S. Identification and Verification of Simple Claims about Statistical Properties. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; Association for Computational Linguistics: Stroudsburg, PA, USA, 2015; pp. 2596–2601.
11. Thorne, J.; Vlachos, A. An Extensible Framework for Verification of Numerical Claims. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017; pp. 37–40.
12. Riedel, B.; Augenstein, I.; Spithourakis, G.P.; Riedel, S. A Simple but Tough-to-Beat Baseline for the Fake News Challenge Stance Detection Task. *arXiv* **2018**, arXiv:1707.03264.
13. Ferreira, W.; Vlachos, A. Emergent: A novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2016; pp. 1163–1168.
14. Vrandečić, D.; Krötzsch, M.W. Wikidata: A free collaborative knowledgebase. *Commun. ACM* **2014**, *57*, 78–85. [CrossRef]
15. Wang, W.Y. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017.
16. Rashkin, H.; Choi, E.; Jang, J.Y.; Volkova, S.; Choi, Y. 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, 7–11 September 2017; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017; pp. 2931–2937.
17. Nakashole, N.; Mitchell, T.M. Language-Aware Truth Assessment of Fact Candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 1009–1019.
18. Vlachos, A.; Riedel, S. Fact Checking: Task definition and dataset construction. In Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science, Baltimore, MD, USA, 26 June 2014; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 18–22.
19. Agirre, E.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W. SEM 2013 Shared Task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, Atlanta, GA, USA, 13–14 June 2013*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2013; Volume 1, pp. 32–43.
20. Rajaraman, A.; Leskovec, J.; Ullman, J. Mining of Massive Datasets. 2014. Available online: http://www.mmds.org (accessed on 11 November 2020).
21. Pennington, J.; Socher, R.; Manning, D.; Christopher, D. GloVe: Global Vectors for Word Representation. 2014. Available online: https://nlp.stanford.edu/projects/glove/ (accessed on 8 July 2021).
22. Hassan, N.; Arslan, F.; Li, C.; Tremayne, M. Toward Automated Fact-Checking. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; ACM: New York, NY, USA, 2017; pp. 1803–1812.
23. Heilman, M.; Smith, N.A. *Question Generation via Overgenerating Transformations and Ranking*; Defense Technical Information Center (DTIC): Fort Belvoir, VA, USA, 2009.
24. Zheng, Z. AnswerBus question answering system. In Proceedings of the Human Language Technology Conference, San Francisco, CA, USA, 24–27 March 2002; Association for Computational Linguistics: Stroudsburg, PA, USA, 2002; pp. 399–404.
25. Mihaylova, T.; Karadzhov, G.; Atanasova, P.; Baly, R.; Mohtarami, M.; Nakov, P. SemEval-2019 Task 8: Fact Checking in Community Question Answering Forums. In Proceedings of the 13th International Workshop on Semantic Evaluation, Minneapolis, MN, USA, 6–7 June 2019; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 860–869.
26. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
27. Bari, M.S.; Mohuiddin, T.; Joty, S. MultiMix: A Robust Data Augmentation Framework for Cross-Lingual NLP. *arXiv* **2020**, arXiv:2004.13240.