T-verifier: Verifying Truthfulness of Fact Statements

Xian Li[1], Weiyi Meng[1], Clement Yu[2]

[1]*Department of Computer Science, Binghamton University*
Binghamton, NY 13902, USA
{xianli, meng}@cs.binghamton.edu

[2]*Department of Computer Science, University of Illinois at Chicago*
Chicago, IL 60607, USA
yu@cs.uic.edu

Abstract— **The Web has become the most popular place for people to acquire information. Unfortunately, it is widely recognized that the Web contains a significant amount of untruthful information. As a result, good tools are needed to help Web users determine the truthfulness of certain information. In this paper, we propose a two-step method that aims to determine whether a given statement is truthful, and if it is not, find out the truthful statement most related to the given statement. In the first step, we try to find a small number of alternative statements of the same topic as the given statement and make sure that one of these statements is truthful. In the second step, we identify the truthful statement from the given statement and the alternative statements. Both steps heavily rely on analyzing various features extracted from the search results returned by a popular search engine for appropriate queries. Our experimental results show the best variation of the proposed method can achieve a precision of about 90%.**

I. INTRODUCTION

More and more Web users depend on the Web to acquire information. Unfortunately, not all information on the Web is truthful. Concerns about the quality of Web content have been raised for quite some time [6]. Reasons for having untruthful information include unintended mistakes (e.g., typos) and intentionally-spread rumors. The first search result in Fig.1 is an example of a rumor-spreading article that attempts to portrait Barack Obama as a Muslim even though he has been a Christian for a long time. The fact that the Web is an open forum where everyone can publish without prior review makes it easier for untruthful information to enter the Web. Methods are needed to distinguish truthful information from untruthful ones on the Web.

Fig. 1: Results of "Barack Obama is Muslim" (Yahoo.com on 12/20/2009)

Many web pages contain sentences that state a "fact" and these statements will be called *fact statements*. If a fact statement states a true fact, it is called a *truthful statement*; otherwise, it is an *untruthful statement*. In this paper, we are interested in only fact statements rather than *opinionative statements*, which express opinions. More specifically, for a fact statement, we are interested in determining whether or not the statement is truthful, and if it is not, finding out the truthful statement most related to the given statement. The T-verifier system solves this problem based on information collected from the Web. When using the T-verifier system, a user submits a fact statement whose truthfulness is uncertain to the user. Such type of statements will be called *doubtful statements*. In addition, the user also specifies which part(s) of the statement he/she is not sure about; the specified part(s) will be called *doubt unit(s)*. A doubt unit can be a term or a phrase. All the other non-stop terms in the doubtful statement are called *topic unit(s)*. As an example, a user may submit "Barack Obama is a Muslim" as a doubtful statement and specify "Muslim" as a doubt unit. In general, a user may specify multiple doubt units in a doubtful statement. In this paper, we consider the case when only one doubt unit is specified in each doubtful statement and leave the case involving multiple doubt units as future work.

A special case of the problem is that a user has a doubtful statement whose doubt unit has several values and the user knows one of these values is truthful but is not sure which one is. For the above example about Barack Obama, a user may know that either Muslim or Christian is truthful but not sure which one is truthful. In this case, a simple method to determine the truthfulness of a fact statement is as follows. First, by replacing the doubt unit with each of the possible values in the doubtful statement, we can generate multiple statements. Second, we submit each statement to a popular search engine and obtain the corresponding number of hits. Finally, select the statement that has the largest number of hits as the truthful statement. Intuitively, this method is appealing and makes some sense because it is a simple way to leverage the huge amount of data on the Web and there is likely more truthful information than untruthful information on the Web.

TABLE I: NUMBER OF HITS COMPARISON (YAHOO.COM ON 12/20/2009)

Statements	Truthfulness	Number of Hits
Hillary Clinton is the President of United States	Untruthful	30,100,000
Hillary Clinton is the Secretary of State	Truthful	23,600,000

Unfortunately, the above simple method is not sufficiently reliable as frequently untruthful statements may get more hits than truthful ones. As an example, in Table 1, the untruthful statement "Hillary Clinton is the President of United States"

got more hits than the truthful "Hillary Clinton is the Secretary of State". In general, our experiment shows that the above simple method has only a 20% precision (see section V for details). Therefore, more advanced methods are needed to determine the truthfulness of fact statements.

Additional challenges that make determining the truthfulness of fact statement difficult include: (1) Conflicting statements about the same fact may appear on the Web. For example, in the two articles corresponding to the results in Fig.1, one asserts that Barack Obama is a Muslim while the other tries to dispute the assertion. (2) Current search engines rank search results based on their estimated relevance to the user query but they do not take the truthfulness of the results into consideration. As a result, it is unrealistic for ordinary users to utilize an existing search engine directly to reliably determine the truthfulness of their doubtful statements. We need a systematic way to help users verify doubtful statements.

Though untruthful information is very common on the Web, we believe that on the web (1) there must be someone telling the truth of a fact; and (2) truth of a fact is usually more widespread than untruth. Our solution of identifying statement truthfulness consists of two steps: *alternative statement collection* and *statement truthfulness verification*. In the first step, we try to find other statements of the same topic as the doubtful statement but are different on the doubt unit. We call such kind of statements *alternative statements*. In the second step, we will rank all the alternative statements as well as the doubtful statement based on a set of measures related to truthfulness determination and select the one which is most likely to be truthful. For simplicity, in the statement truthfulness verification step, we will consider the user given doubtful statement as one of the alternative statements.

This paper has the following contributions:

(1) We propose a two-step (alternative statement collection and statement truthfulness verification) method to tackle the problem of determining the truthfulness of a fact statement. And it has the capability to find out the intended truthful statement if the given doubtful statement is not true.

(2) Instead of simple string pattern matching techniques as used in [16, 17], we use a feature based method to collect alternative statements from the Web. Besides text features, we perform semantic and correlation analysis on the alternative statements and the doubt unit to filter out unlikely statements and rank the promising statements.

(3) We employ a set of basic rankers to rank all the alternative statements (including the doubtful statement as stated earlier), evaluate and compare different rank merging algorithms (including a new merging algorithm we propose) for selecting the truthful statement – the one with the highest combined ranking score. Based on our experiment, our method can find the truthful statement in 90% of the cases.

For the rest of the paper, we first give an overview of our approach in section II. After that, we discuss our solution on alternative statement collection and statement truthfulness verification in sections III and IV, respectively. The precision of our solution is evaluated based on real data and our experimental results are shown in section V. We compare our work with other published papers on related issues in section VI. We conclude the paper in section VII.

II. Overview of Approach

In this paper, we introduce our T-verifier system which addresses the truthfulness problem on statement level. A system overview is shown in Fig.2. The system allows a user to submit a doubtful statement DS together with a doubt unit DU as input. The system will first generate a keyword query based on topic units in DS and submit it to a search engine. We try different query specification methods for the search engine and use the best in our system implementation. From the retrieved web pages we generate the alternative statements by exploring a set of features. We then submit each of the alternative statements (including the original doubtful statement) as a query to the search engine, rank them by comparing returned page sets on different measures, and show the user the top-ranked statement as the truthful statement found.

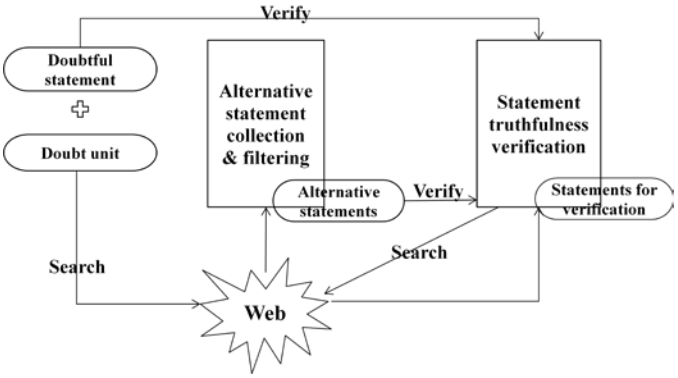

Fig. 2: T-verifier system overview

The purpose of alternative statement collection is to find a list of those possibly truthful statements with respect to the fact. The first challenge comes from the variations of language usage. People may use different ways to state the same fact. Therefore, collecting alternative statements may need natural language understanding techniques. Second, some statements with the same structure as the doubtful one may not be relevant. Still taking the "Barack Obama is Muslim" as an example, a possible alternative statement is "Barack Obama is President", which is truthful but not relevant to the religious topic of the given doubtful statement.

The statement truthfulness verification module is designed to select the most likely truthful statement by ranking all the alternative statements. The ranking is based on the search result records retrieved using each of these statements. There are two issues here. First, what kind of features should we use to perform the ranking and how to combine the ranks produced by individual rankers using different features? These features, derived from the collection of search result records, should be able to distinguish the truthful statement from others. Second, though using a larger number of alternative statements can increase the chance that one of them is the truthful statement, it also increases the effort needed to

perform statement truthfulness verification. Note that for every alternative statement considered, we have to perform a search, collect certain number of search results and analyse them. A good approach should try to minimize the number of alternative statements for the verification step without increasing the chance of missing the truthful statement.

III. ALTERNATIVE STATEMENT COLLECTION

Given a doubtful statement DS, we let the user specify the doubt unit DU. A doubt unit can be a single word, a phrase, a name entity, a number or a date, etc.

Alternative statements to DS are very important to the truthfulness judgment, because they provide the users with different versions of the fact. Especially when DS is not truthful, the truthful one is quite probably one of the alternative statements. Based on our observation, alternative statements should have the following three properties:

- **Same Topic**. An alternative statement should cover the same topic as the doubtful statement. Otherwise, it does not satisfy the user's expectation even when the alternative statement is truthful.
- **Different Values**. An alternative statement should be different from the doubtful statement on the doubt unit. We call the term(s) in place of the doubt unit *alternative unit*. For example, the alternative statement "Barack Obama is a Christian" contains an alternative unit "Christian" which is in place of "Muslim" in the doubtful statement.
- **Term sense/type closeness**. An alternative unit should be a replacement of the doubt unit. Therefore, the alternative unit and doubt unit should have close word sense to each other, in both data type and semantic meaning. For example, "Christian" is closer to "Muslim" than to "President", because both "Christian" and Muslim" are religious people.

The alternative statements may be presented in various ways, while the essential difference between alternative statements and the doubtful one is the alternative units. In order to avoid getting involved in complex natural language processing issues, we turn the problem of collecting alternative statements into the problem of looking for alternative units and use them to generate alternative statements. Hence the key to find relevant alternative statements is to find relevant alternative units from web pages relevant to the same topic.

We partition a doubtful statement into two parts: *doubt unit* and *topic units*. For the doubtful statement "Barack Obama is Muslim", the doubt unit is "Muslim" and topic units are "Barack" and "Obama". Web pages used for collecting alternative units are retrieved by submitting only the topic units as a query to a search engine (we currently use Yahoo! in T-verifier). Note that if the doubt unit was included in the query, the search results would be biased in favour of the doubtful statement. We collect search result records (each consisting of a title and a snippet), SRRs for short, from the search engine result pages. Let $D = \{r_1, r_2 \ldots, r_N\}$ be the set of all SRRs collected, where N is the cardinality of D. We utilize seven different features to rank the terms in the set of SRRs and to find the alternative units from the top ranked terms.

A. Features of alternative units

Initially, all the terms/phrases appearing in the SRR set D are considered as candidates for alternative units. In order to reduce the number of candidates, we could resort to linguistic analysis or some heuristic rules. However, it is not easy to produce a set of widely usable rules and it is time-consuming to do linguistic analysis. In this paper, we pursue a different approach to identify potentially good alternative units. Our observation shows that there are two types of co-occurrence information with alternative units that can be used. First, relevant alternative units frequently co-occur with the topic units. This is because these topic units provide a good context to the alternative units. Second, relevant alternative units often co-occur with the doubt unit. This is because (1) when people have doubt about a fact, they often mention other possible answers to the fact; and (2) when people dispute a controversial point or a common misconception, they often mention their own points or the truthful fact. For example in Fig.3, it shows the top result of "Obama is a Muslim" from the Yahoo search engine, where we can find both "Muslim" and "Christian" being mentioned in the same record. In order to utilize these two co-occurrences in collecting alternative units, we develop the following features to estimate the possibility of a candidate term T (T can be single term or a phrase) as an alternative unit.

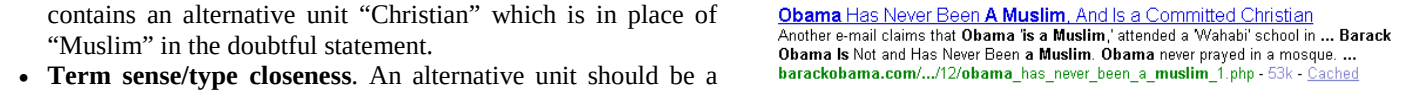
Obama Has Never Been **A Muslim**, And Is a Committed Christian
Another e-mail claims that **Obama** '**is a Muslim**,' attended a 'Wahabi' school in **... Barack Obama Is** Not and Has Never Been **a Muslim**. **Obama** never prayed in a mosque. **...**
barackobama.com/.../12/**obama**_has_never_been_a_**muslim**_1.php - 53k - Cached

Fig. 3: Sample result of "Barack Obama is Muslim" (Yahoo.com on 12/20/2009)

The first four features, including "result coverage", "result query relevance", "SRR ranking", and "term-keyword distance", are introduced to represent the co-occurrence relationship between topic units in query Q (which is the doubtful statement minus the doubt unit) and a candidate term T (i.e., a candidate alternative unit). We use function Cont(r, T) to check whether an SRR r in D contains the term T; Cont(r, T) returns 1 if r contains T and 0 if r does not.

- **Result coverage (RC)**: It is the percentage of SRRs in D that contain T. We assume that all records in D are relevant to the fact. RC shows how frequently T co-occurs with the topic units. The record coverage of T is computed by:

$$RC(T) = \frac{\sum_{i=1}^{N} Cont(r_i, T)}{N}$$

- **Result Query Relevance (RQR)**: It measures what portions of the topic units are covered in SRRs. In the above result coverage computation, all SRRs are considered equally relevant to query Q. In reality, the degree of relevance is different among the SRRs. One way to estimate the relevance of an SRR is to check how many topic units are contained in the SRR, and an SRR that contains more topic units can be considered to have a higher degree of relevance. Intuitively, if more SRRs containing term T have higher degree of relevance, T is more likely to be a relevant alternative unit. We measure RQR by:

$$RQR(Q,T) = \frac{\sum_{i=1}^{N} Cont(r_i,T) * intersect(r_i,Q)/len(Q)}{\sum_{i=1}^{N} Cont(r_i,T)}$$

where $\mathrm{intersect}(r_i, Q)$ is the number of topic units in Q appearing in r_i and len(Q) is the length of Q.

- **SRR ranking (Rrank)**: Another way to estimate the relevance of an SRR is to see where it is ranked by the search engine used. Generally, search engine ranks SRRs based on their likelihood of being relevant to the query. Therefore, we aggregate the ranks of the SRRs that contain term T as the record ranking of T. The Rrank is computed as follows:

$$Rrank(T) = \frac{\sum_{i=1}^{N} Cont(r_i,T) * (1 - pos(r_i)/N)}{\sum_{i=1}^{N} (1 - pos(r_i)/N)}$$

where *pos*(*r*) is the position of r in the search engine's ranking list and *N* is the cardinality of D.

- **Term distance (TD)**: Intuitively, terms that appear far away from the topic units are less likely to be related to the topic units compared to those that appear closer. To capture the proximity information between terms, we consider the size of the smallest window of consecutive words in each SRR (title or snippet) that covers all the topic units contained in the SRR as well as the term T. We use the following formula to compute TD in the snippet of the SRR for term T:

$$TD_{snippet}(T) = \frac{\sum_{i=1}^{N} Cont(r_i^{snippet},T) * (Snippet\_len - min\_winsize)}{N} * intersect(r_i^{snippet},Q)/len(Q)$$

Our TD formula considers the length of the minimum window as well as the number of topic units covered in this window. For the TD measure, while an SRR containing fewer topic units would benefit from possibly smaller window size, it is also penalized for including fewer topic units. Similarly we also compute $TD_{title}(T)$ and add these two parts together as the overall TD value. As a special case, when none of the topic units appears in the same sentence with T, we consider its TD for this record to be 0.

Another set of features capture the relevance between a candidate term T and the given doubt unit DU.

- **Data type matching (DM)**: The alternative units should be of the same data type as the doubt unit because it is reasonable to assume that the user is look for something of the same type. Since the doubt unit provides us an instance of the data type the user is looking for, candidate terms of different data types are less likely to be relevant alternative units. We identify several data types either by their special format or by a dictionary, including date, time, telephone number, email address, person name, place name (e.g. name of state, city and attractions), and number. All the others are considered as common string. In this way, we can filter out those candidates that have a data type different from that of the doubt unit.
- **Sense closeness (SC)**: Data type matching can filter out a great number of candidates, especially when the doubt unit is of a special type, like number, dates or names etc. However, in many cases the doubt unit is a common string. In these cases, filtering by data type does not work effectively. According to our observation, many terms extracted from SRRs are irrelevant with the doubt unit even when they have the same data type, especially when the data type is common string. For example, a user has doubts about the religion of Barack Obama and inputs "Barack Obama is Muslim". When query "Barack Obama is" is submitted to Yahoo, the most frequent term extracted from the returned SRRs that has the same data type is "president". However, "president" is a term irrelevant to religion. Instead, a less frequent term "Christian" is more likely to be a good alternative unit because, similar to "Muslim", it is also about religion. This example shows that terms that have a closely related meaning as the doubt unit are good candidates for alternative units.

In this paper, we utilize WordNet [22] to capture the sense closeness between two different terms. In general, we take special care of the following three relations: direct hypernym/hyponym, instance hyponym, and sibling. Instance hyponym means one term is an instance of the other, like "Europe" is an instance hyponym of "continent". Sibling means two terms share the same direct hypernym or instance hypernym. Except for these three cases, the sense closeness is defined as the Wu-Palmer similarity (wup_similarity) provided by the NLTK [15, 21], which returns a score denoting how similar two word senses are, based on the depths of the two senses in the taxonomy and that of their Least Common Subsumer (most specific ancestor node). For the hypernym and sibling cases, we do not use wup_similarity value because hypernym and sibling terms are often the best choices for alternative unit. Therefore we want to specially differentiate these cases. To summarise, we use the following formula to compute sense closeness between term T and doubt unit DU:

$$SC(T,DU) = \begin{cases} \alpha, & if\ hyper(T,DU) = true\ or\ instanc\_hyper(T,DU) \\ \beta, & if\ sibling(T,DU) = true \\ \gamma, & \gamma = wup_{similarity(T,DU)}, \quad otherwise \end{cases}$$

where α and β are parameters to be tuned empirically for best performance.

- **Term local correlation (TLC)**: It measures the correlation between candidate term T and doubt unit DU. Based on our second observation mentioned before, relevant alternative units are often mentioned together with the doubt unit for various reasons.

Detecting the terms which are highly correlated with the doubt unit can be an effective way in finding likely relevant alternative units. In this paper, we use the correlation coefficient formula [14] to evaluate the correlation strength between candidate term T and doubtful unit DU:

$$TLC(T,DU) = \frac{\sum_{i=1}^{N} (Cont(r_i,T) - RC(T))(Cont(r_i,DU) - RC(DU))}{\sqrt{\sigma(T)^2 \sigma(DU)^2}}$$

Here, RC(T) is the result coverage of T, which we have mentioned before. σ(T) is computed by $\sum_{i=1}^{N}(\mathrm{Cont}(r_i, T) - \mathrm{RC}(T))^2 * \frac{1}{N}$. This formula basically uses the definition of correlation coefficient.

If we consider X (Y) as a random variable which describes the appearance of T (DU) in each SRR, then E(X) is RC(T) and E(Y) is RC(DU). σ(T) and σ(DU) are the standard deviations of X and Y, respectively. According to the correlation coefficient definition, TLC of T and DU is measured by the above formula.

B. Alternative unit selection algorithm

In this paper, we select alternative units by combining the seven features described above in two steps. We first filter the candidate terms by data type matching (DM), i.e., only those terms that match the data type of the doubt unit will be considered further. In the second step, we rank each remaining candidate term based on the other six features. The ranking score of T is computed using the following formula:

$$Alter_{rank}(T) = w_1 * RC + w_2 * RQR + w_3 * Rrank + w_4 * TD + w_5 * SC + w_6 * TLC$$

where w_i, i=1,...,6, are the weights for the features. These weights form a vector called *Alternative Unit Selection Vector* AUSV = (w_1, w_2, w_3, w_4, w_5, w_6). We use training to find the best AUSV that yields the best performance. We evaluate the effectiveness of an AUSV by the position it assigns to the truthful alternative unit in its ranking list. Ideally, the truthful alternative unit should be ranked at the top of the list for each doubtful statement. Each time we go through the list of top 10 candidate terms sorted by $Alter_{rank}$ in descending order. If the truthful alternative unit is ranked the i-th on the list, the fitness of this AUSV will be (1-0.1*i).

A genetic algorithm [4] is used to obtain the best performance AUSV over a set of training statements. Initially, the algorithm randomly generates 30 different AUSVs, each of which contains 6 real numerical components, as the first generation. The evolution from one generation to the next includes three steps: parent selection, crossover and mutation. Every time we choose a pair of AUSVs in the current generation as parents for next generation based on the *Wheel of Fortune* principle. The probability for an AUSV to be selected is its fitness divided by the sum of the fitnesses of all AUSVs in the current generation. The AUSVs with larger fitnesses have a better chance to be selected as parents. The same AUSV is allowed to be selected more than once. For each pair of selected parents, the crossover is done with a probability of 0.75. When performing the crossover, we generate a random binary mask with the same number of components as AUSV, which is used to select the value of an AUSV component from either the first parent or the second parent, respectively. Mutation on the child AUSV is performed with probability of 0.1. If a component mutates, it increases or decreases by 0.1 with equal probability of 0.5. The AUSV with the best fitness in the current generation will be directly moved to the next generation. The genetic algorithm continues until the best fitness meets a predefined threshold τ. The fittest AUSV of the final generation is selected as the "optimal" AUSV.

According to our experiment, the optimal AUSV found by the above procedure performs well on our test dataset. For each of the 50 sample statements tested, the algorithm always ranks the truthful alternative statement among the top five results. Furthermore, in 31 cases, the truthful statement is ranked at the top. As a result, we only need to consider the five statements with the highest $Alter_{rank}$ in the statement truthfulness verification step for each user statement. This not only reduces the effort but also provides a solid basic ranker for ranking the alternative statements for the next step.

IV. STATEMENT TRUTHFULNESS VERIFICATION

In the previous section, alternative statements are generated from the searching results of the query composed of the topic units. The Alternative Unit Selection algorithm aims to ensure that the truthful alternative unit is included among the top five ones on the final ranking list. However, the truthful statement is not always at the top. This means there is a need of further verification on each of the top five statements to find out which one is truthful. In this section, we will discuss our verification process for a given group of alternative statements (including the doubtful statement). Generally, our verification has three steps: first, send every alternative statement as a query to a search engine and collect relevant SRRs; second, employ a number of basic rankers and generate a ranking list of the alternative statements using each basic ranker based on newly collected SRRs; third, use a rank merging algorithm [1, 11] to merge the rank lists into a combined final list.

A. Statement truthfulness verification

Intuitively, verifying the truthfulness of a statement requires relevant pages as evidence. In the web environment, we can use a search engine to retrieve relevant web pages. As we have mentioned in the introduction, truthful information is usually more widespread than untruthful ones and tends to be consistent. We will put emphasis on analysing how the retrieved web pages support their corresponding statement.

Given several alternative statements among which one is true, we can transform the verification problem to a ranking problem: rank the statements based on an estimation of its truthfulness and choose the one that is ranked the highest. In order to perform the truthfulness evaluation, we can use different features as measurements. For example, the simplest way is to compare the number of retrieved pages. Intuitively, the statement retrieving the largest number of results is more likely to be true. Another method is to directly use the ranks of alternative units generated by the alternative unit selection step. In this paper, we will explore different ways to rank the alternative statements and combine these ranks by a rank merging algorithm.

We use a five-element vector to formalize the truthfulness verification problem: [Statements, Supporting SRR sets, Basic rankers, Basic rank lists, Merged rank list]. We show the overview of the verification module in Fig.4.

- **Statements** (Stmts) is the set of selected alternative statements (including the doubtful statement) to be verified. Let Stmts = {s_1, s_2, …, s_n} with $n \leq 5$. All statements are about the same topic but present different versions on the fact. It is assumed that one of the statements in Stmts is truthful but we do not know which one is.

- **Supporting SRR Sets** (Sup_S) is a collection of supporting SRR sets. Specifically, Sup_S = {Sup_1, Sup_2, …, Sup_n}, where each Sup_i contains two components: (1) the number of hits when statement s_i is submitted as a query to the search engine used; and (2) the top N SRRs retrieved by the search engine for s_i. Roughly speaking, all SRRs in Sup_i can be considered as supporting evidence for s_i though with a different degree of relevance. Additional features may help the truthfulness judgment, for example, the publishing time and *domain* of each SRR in Sup_S, Here the domain of an SRR is the internet domain (e.g., edu, com) of the URL of the web page corresponding to the SRR. It is believed that information from certain domains (say edu) is more trustworthy than information from other domains (e.g., com) [5, 18].
- **Basic Rankers** (BR) is a set of basic rankers each of which ranks the statements in Stmts based on their measurement. BR = {$BR^{(1)}$, ..., $BR^{(M)}$}, where M is the number of basic rankers used. Each basic ranker tries to explore different features of the SRR set in Sup_S and their relationships with the corresponding query statement. In section IV.B, we will discuss all the basic rankers used in our verification method.
- **Basic Rank Lists** (BRL) is a collection of rank lists, each of which is a list of ranked statements in Stmts produced by one of the basic rankers. The list produced by basic ranker $BR^{(k)}$ is {r_1^k, ..., r_n^k}, where r_i^k is the rank of s_i given by $BR^{(k)}$.
- **Merged Rank List** (MRL) is the merged rank list of the statements in Stmts by merging all the basic rank lists. Rank merge is a classic problem which has been studied by data fusion researchers. In this paper, we apply voting based techniques to perform merging. For comparison purpose, we also apply machine learning method to the verification problem in experiment and compare its performance with our voting based techniques.

Stmts		s_1,	s_2,	…,	s_n	
Sup_S		Sup_1,	Sup_2,	…,	Sup_n	
BR and BRL	$BR^{(1)}$ {	r_1^1,	r_2^1,	…,	r_n^1	}= BRL^1
	$BR^{(2)}$ {	r_1^2,	r_2^2,	…,	r_n^2	}= BRL^2
	…	…	…	…	…	
	$BR^{(M)}$ {	r_1^M,	r_2^M,	…,	r_n^M	}= BRL^M
MRL		MRL^1	MRL^2	…,	MRL^n	

Fig. 4: Elements for statement truthfulness verification

Overall, our verification method requires a search on the web for each alternative statement. For a given alternative statement s_i, we submit it to the search engine, collect the number of hits returned by the search engine and the top N (N = 200 is used in our experiments) SRRs from the result list as Sup_i. By controlling the number of alternative statements that need to be verified to at most five, the number of searches needed in the verification step is small.

B. Basic Rankers

Given Stmts and Sup_S, ranking the truthfulness of the statements based on the collected SRRs in Sup_S can be done in different ways. Each of these methods captures one or more features that the truthful statement is likely to outrank untruthful ones. We first introduce different basic rankers to rank the alternative statements with their features.

1) Alternative Unit Ranker (AUR): In section III, we discussed the process of extracting and selecting alternative units from the SRRs retrieved by the doubtful statement minus the doubt unit. The alternative unit selection algorithm selects the alternative units by ranking candidates based on a number of features collected from the search result. Since each alternative unit corresponds to an alternative statement, this ranking of the alternative units from the alternative unit selection algorithm can be considered as the ranking of the statements in Stmts, and therefore we call it Alternative Unit Ranker (AUR).

The ranks generated from AUR can be useful here because of four reasons: (a) The web data used for AUR ranking is retrieved by the doubtful statement without the doubt unit, which is different from the data collected in Sup_S. (b) In the process of ranking, we use not only data from the web, but also WordNet to do semantics analysis, reducing the probability of selecting irrelevant terms as alternative units. (c) AUR specially considers the observation that related (either truthful or controversial) alternative units often co-occur with the untruthful ones. (d) AUR has reasonable effectiveness for identifying the truthful alternative units as among the 50 statements used in our experiment, 31 truthful alternative units are ranked at the top by this ranker, yielding a 62% precision.

2) Hits Ranker (HR): A seemingly reasonable method is to rank the alternative statements by the number of hits they retrieve from a search engine. Web users commonly use this method to do quick truthfulness verification. We call this method the Hits Ranker.

One potential problem of this ranker is that it implicitly assumes that all the SRRs in Sup_i support statement s_i. However, it is possible that some of the SRRs are actually against s_i (i.e., saying it is not true). As current search engines retrieve results based on only query words and do not analyse the meanings of texts, the number of hits may not be a reliable indicator for judging the truthfulness of a statement.

3) Text Feature Rankers (TFR): Text feature rankers are a set of rankers measuring the relevance between alternative statements and each SRR. In section III.B, we discussed several text features used to measure the relevance between an SRR and the modified query statement (i.e., the doubtful statement minus the doubt unit). There are mainly four features involved: Result Coverage (RC), Result Query Relevance (RQR), SRR ranking (Rrank), Term Distance (TD). In TFR, we reuse these four features to do the ranking. However, the rankings generated by TFR here are different from those by the alternative statement collection step in section III because these four features are now used against a different set of SRRs, which is retrieved by a different query (i.e., the full alternative statement under consideration).

4) Domain Authority Ranker (DAR): There is a background truth on the web: Some researchers have observed that web pages published by certain domains are more likely to be truthful, such as ".gov", ".edu", etc [5, 18]. This is because websites in these domains claim the responsibility for their published information. Based on this observation, we assign a higher weight to a domain that is considered to be more trustworthy.

In our work, the weights for different domains are learned as follows. First collect two sample SRR sets. One set contains the SRRs for k truthful statements, called True_set, and the other set contains the SRRs for k untruthful statements, called Untrue_set. We treat each domain as described below. Take domain ".edu" as an example. We find the total number of SRRs from the ".edu" domain in True_set, say n_1 and that in Untrue_set, say n_2. Then the weight for SRRs from the ".edu" domain is $W_{.edu} = n_1/(n_1+n_2)$. In other words, if a statement yields one result from ".edu" in its supporting SRR set, it gains a truthfulness score of $n_1/(n_1+n_2)$.

When testing a statement S with unknown truthfulness, its supporting SRR set is Sup with N SRRs included. We aggregate the number of SRRs from each different domain in its Sup. For example, there are m SRRs found from ".edu". These ".edu" pages gain truthfulness score of $W_{.edu}$ * m for S. Overall, the rank position given by DAR for S is the average truthfulness score of an SRR in Sup, $\sum_{d \in domains} {W_d * n_d}/{N}$, where W_d is the weight of domain d trained from the sample set, n_d is the number of SRRs from domain d, and N is the cardinality of the Sup.

C. Rank Merging

In section IV.B, we introduced a set of basic rankers each of which evaluates statement truthfulness with a different measure. Each ranker is applicable to all the alternative statements. However, divergence in statement ranking is inevitable among all these basic rankers. In order to get a comprehensive conclusion, the final rank should combine ranks from all basic rankers. Therefore, an effective rank merging algorithm is a key component in a solution to the statement verification problem. Taking the "Hillary" statements in Table I as an example, though HR ranks the untruthful statement over the truthful one, AUR, TFR and DAR give higher rank to the truthful statement. For this example, combining all the ranks would rank the truthful statement over untruthful ones.

Rank merging is not the only solution to find out the truthful statement. By considering the ranking scores of each statement given by basic rankers as feature values, we can also apply a machine learning algorithm to a set of pre-tagged truthful/untruthful statements to train a classifier, and use it to determine the truthfulness of unknown alternative statements. We will compare rank merging solutions with a decision tree based solution in our experiment (see section V).

1) Baseline Merging Algorithms

Given a set of alternative statements about the same topic, Stmts = $\{s_1, s_2, \ldots s_n\}$, each basic ranker ranks them in its own manner. Say ranker $BR^{(i)}$ ranks Stmts as $\{r_1^i, r_2^i, \ldots r_n^i\}$, where r_j^i is the rank of statement s_j given by ranker $BR^{(i)}$. In general, $r_j^i < r_k^i$ means ranker $BR^{(i)}$ believes that s_j is more likely to be truthful than s_k. Therefore, M rankers would generate M basic rank lists BRL = $\{BRL^1, BRL^2, \ldots, BRL^M\}$. In this paper, we use two commonly used algorithms Borda [1, 2] and Condorcet [3, 11] as the baseline algorithms for merging the rank.

In the basic Borda algorithm (**BaseBorda**), the top ranked statement by $BR^{(i)}$, say s_j, is given score n ($r_j^i = n$) and the second ranked is given n-1 and so on. Therefore, the scores of each ranker's rank list are a permutation of integers in [1, n]. The Merged Rank List (MRL) will be obtained based on the values computed in $\{\sum_{i\in[1,M]} r_1^i, \sum_{i\in[1,M]} r_2^i, \ldots, \sum_{i\in[1,M]} r_n^i\}$. The alternative statement with the largest combined score is considered as the truthful one. This algorithm fully utilizes the position information in every rank list and uses quantified scores to represent position information so as to facilitate the merging process.

Algorithm 1: Baseline Condorcet Algorithm (BaseCond)

```
1. Count = 0; graph G;
2. for each basic ranker r_i, do
3.     if r_i ranks s_i higher than s_k
4.         Count += 1
5.     else Count - = 1;
6. if Count > 0: add s_i→s_k to G
7. else if Count < 0: add s_k→s_i to G
8. compute Hamiltonian path in G
```

The idea of Condorcet algorithm is to choose the statement which is ranked higher than or the same as every other statement as the truthful one. For every pair of statements in Stmts, say (s_i, s_j), their rank relationship is defined as follows. Check the ranks given by each basic ranker. If s_i is ranked higher than s_j by more basic rankers, then s_i is ranked higher than s_j in MRL, denoted as $s_i \rightarrow s_j$, and vice versa. If each of the two statements is ranked higher than the other by the same number of basic rankers, then they are considered to have the same rank in MRL, denoted as $s_i = s_j$. Note that since we use seven basic rankers, the case for $s_i = s_j$ won't occur. Essentially, considering every statement as a vertex and $s_i \rightarrow s_j$ as an edge, the ranking relationships can be represented as a directed graph, called *Condorcet graph*. If there exists a cycle in the Condorcet graph, e.g. $\{s_i \rightarrow s_j \rightarrow \ldots \rightarrow s_k \rightarrow s_i\}$, all the statements included in this cycle are considered as equivalently ranked. Finally, select the statement ranked not lower than any other statement as the top one, which can be carried out by finding a Hamiltonian path in the Condorcet graph. Obviously, it is possible for this algorithm to rank more than one statement at the top. In such a case, our system will select the statement that is placed highest by the most accurate basic ranker.

Both of the above two algorithms can find the top ranked alternative statement fast, especially when the size of Stmts is not large. Two noticeable common features of these two algorithms are that they both treat each basic ranker equally. However, in reality, features adopted by each basic ranker are likely to be relevant with truthfulness judgment with different degrees. Some rankers may produce more accurate ranks than

others. Therefore, we introduce weighted merging algorithms in the next subsection.

2) Weighted Rank Merging

In this subsection, we introduce several ways to incorporate weights of individual rankers into the BaseBorda and BaseCond algorithms to improve ranking precision. Specifically, we consider the following variations to them.

• ***Positional Borda (PosBorda)***. For each basic ranker, the BaseBorda algorithm gives the top ranked statement score n, the second score n-1 and so on. We can interpret it as follows: if a statement is ranked at the top by a basic ranker, it is truthful with probability $\frac{n}{\sum_1^n i}$ for this ranker; if second, then its probability is $\frac{n-1}{\sum_1^n i}$, and so on. We call it "*position probability*", i.e., the probability of a statement being truthful if it is ranked at i-th position.

We may obtain more accurate position probabilities from sample documents. Let D be a collection of sample alternative statement sets, D = {$Stmts_1$, $Stmts_2$, …, $Stmts_X$}. Each $Stmts_i$ contains n alternative statements and only one of them is truthful. We try each basic ranker on D and record the rank position of the truthful statement. Let x_j be the number of truthful statements ranked at the j-th position. Then the probability that a truthful statement is ranked at the j-th position can be estimated to be x_j / X, where X is the total number of Stmts in D. In general, the i-th basic ranker $BR^{(i)}$ gets a set of position probabilities $PP^{(i)}$ = {$P_1^{(i)}$, $P_2^{(i)}$, …$P_n^{(i)}$}, where $P_j^{(i)}$ is the probability that a statement ranked at the j-th position by $BR^{(i)}$ is truthful. We can interpret $P_j^{(i)}$ as the ranking score given by $BR^{(i)}$ to the statement ranked at j-th position; it is an empirical score obtained by training. When given a new Stmts, each alternative statement s in Stmts will receive a score (i.e., the position probability) from each of the basic rankers. The sum of these scores becomes the overall score of s. All statements in Stmts are ranked in descending order of their overall scores.

In summary, suppose we have M basic rankers with BR = {$BR^{(1)}$, $BR^{(2)}$, …, $BR^{(M)}$} and each gets a set of position probabilities PP. If a statement s is ranked at j-th position by $BR^{(i)}$, it will receive a score as $P_j^{(i)}$, which corresponds to r_j^i in Fig.4. The merged rank of the statement s is determined by its overall ranking score $\sum_{i=1}^{M} r_j^i$.

• ***Weighted Borda (WBorda)***. This variation differentiates the importance of different rankers and assigns a weight to each ranker, as was also done in [1]. There are two different ways to get the weights for basic rankers. First, because the weight is used to enhance the influence of more precise basic rankers, we can use the precision of each basic ranker on the sample statement set D as its weight. Same for the sample set D with X groups of alternative statements. If a basic ranker ranks the truthful statement at the top in x groups, its precision is x/X. In such a case, we can get a set of weights, each of which represents the precision of a basic ranker. The other method is to use a genetic algorithm to train a weight vector with M elements, W = {w_1, w_2, …, w_M}, over the training set D to get the optimal weights for rank merging. We tested both methods and the second method had better performance. For the rest of this paper, we use the weights obtained by the second method. After the weight for each basic ranker is obtained, the only difference between WBorda and BaseBorda is that the latter multiplies each score assigned by $BR^{(i)}$ by w_i.

• ***Weighted Position Borda (WPosBorda)***. This variation is a combination of PosBorda and WBorda, which combines the idea of adjustment on the scores assigned to each ranked position and different weight placed on basic rankers. It obtains the Merged Rank List (MRL) based on the values computed in $\{\sum_{i\in[1,M]} w_i r_1^i, \sum_{i\in[1,M]} w_i r_2^i, \ldots, \sum_{i\in[1,M]} w_i r_n^i\}$, where w_i is the weight assigned to the i-th basic ranker, r_j^i is the position probability at position j by basic ranker $BR^{(i)}$.

• ***Weighted Condorcet (WCond)***. We now discuss how to incorporate ranker weights into the Condorcet algorithm. In the process of building the Condorcet graph, we decide the partial order for every pair of statements by combining the ranks given by the basic rankers. When assigning weights to basic rankers, this process would incorporate the weights as parameters. The modified algorithm over Algorithm 1 is described as Algorithm 2 below. The same trained weights for basic rankers that are used in WBorda are used for WCond in our experiment.

Algorithm 2: Weighted Condorcet Algorithm (WCond)

```
1. weightedCount = 0; weighted Condorcet graph G;
2. for each basic ranker r_i, do
3.     if r_i ranks s_i higher than s_k
4.         weightedCount += w_i
5.     else weightedCount - = w_i;
6. if weightedCount > 0: add s_i→s_k to G
7. else if weightedCount < 0: add s_k→s_i; to G
8. compute Hamiltonian path in G
```

3) Probabilistic Rank Merging

• ***Probability Combination (ProbComb)***. This is another way to combine basic rankers. Our basic rankers are based on different types of fairly independent features. Thus we may assume that these rankers are independent. Based on this assumption, we can consider the precision of a basic ranker as the success rate of the basic ranker or the probability that the basic ranker is correct in determining the truthful statement. Let P_i denote this probability of basic ranker $BR^{(i)}$. Then the overall probability that the truthfulness of a statement S is correctly determined (i.e., the probability that at least one of the basic rankers is correct in predicting the truthfulness of S) can be estimated by $1-\prod_{i\in[1,M]}(1-P_i)$.

V. EXPERIMENTS

Our experiments require a set of fact statements that are either truthful or not truthful as well as the specified doubt unit for each of the statements. In order to evaluate the precision of our system, we also need the truthful statement for each case. We use the factoid questions from TREC-8 and TREC-9 Question Answering track [20] as the experiment data repository, which contains a large number of factoid questions as well as correct answers. In our experiments, we randomly choose 50 questions from TREC-8 and TREC-9, transform

them into statements with answers that are either correct or incorrect. We make half of the statements with correct answers while the other half with incorrect ones. The answer part in each statement is specified as the doubt unit. We make our dataset available at [23]. We use our proposed method to determine the truthfulness of each statement and compare the result with the ground truth to compute the precision for our method. We show two sample doubtful statements in our dataset in Table II.

TABLE II: SAMPLE DOUBTFUL STATEMENTS

Doubtful statement	Doubt unit	Truth
Antarctic is the only continent without a desert.	Antarctic	Europe
George C. Scott won the Oscar for best actor in 1970.	George C. Scott	George C. Scott

A. Experiments on Alternative Statement Collection

1) Search methods experiment

Current commercial search engines such as Yahoo often provide more than one way to specify a query and the most common specifications are: "all of the words", "the exact phrase" and "any of the words". As our alternative statement collection method aims to discover possible truthful alternative units from the SRRs returned by the search engine used, it is important to choose the best query specification method to maximize the chance of retrieving better alternative statements. We use the 50 statements in our dataset to find out which specification method would return more SRRs containing the truthful alternative units. Our measure is like this, for every query specification method, we compute the average number of SRRs that contain truthful alternative units among the top k SRRs for all 50 sample statements. In our experiment, we test the cases for k = 10, 50 and 100. The experimental results are shown as Table III.

TABLE III: COMPARISON ON QUERY SPECIFICATION METHODS

Different Query specification methods			
	All of the words	Exact phrase	Any of the words
Top 10 SRRs	2.8	2.32	0.34
Top 50 SRRs	13.2	10.52	0.98
Top 100 SRRs	24.14	17.76	1.78

Based on our experiment, it shows that queries in "all of the words" format achieved the best performance. "Exact phrase" is not as good as "all of the words". The "any of the word" method performs much worse than the other two. For the rest of our experiments, we use "all of the words" method to submit queries to Yahoo! Search.

2) Alternative statement collection experiment

In section II, we introduced six features as well as a data type filter for alternative statement collection. We train the weights for different features based on the training set using a genetic algorithm (see section III). Then we use the optimal weights to do experiments on the testing set. Among the 50 statements, we randomly select 25 of them as training set and the rest as testing set. The collection is performed on the top 200 SRRs for each statement. The results are shown in Table IV. On the training set, in 17 out of 25 cases, the truthful alternative unit is collected and ranked as the top candidate. Among the other 8 cases, the truthful alternative unit is ranked at the second in 7 cases and the fourth for one case. On the testing set, the result is also promising, in 14 of 25 cases, the truthful alternative statement is ranked as top 1 and the rest are ranked from the second to the fourth. The experimental result shows that our algorithm can always rank all truthful alternative statements among the top 4 candidates for our dataset. To further reduce the chance of missing out on the truthful alternative statement for each query, we will consider the top 5 results for each query.

TABLE IV: ALTERNATIVE STATEMENT COLLECTION PERFORMANCE

Training data set		**Testing data set**	
Total cases	25	Total cases	25
Truthful one as top 1st	17	Truthful one as top 1st	14
Truthful one as 2nd	7	Truthful one as 2nd	9
Truthful one as 3rd	0	Truthful one as 3rd	1
Truthful one as 4th	1	Truthful one as 4th	1
Truthful one as 5th	0	Truthful one as 5th	0

We also divided our dataset into 10 subsets to do 10-fold cross-validation. Each subset contains 5 doubtful statements and 200 SRRs for each doubtful statement. In each round of validation, we use 45 statements as training set and 5 statements for testing. The precision of each round is determined as the number of cases in the testing set which rank the truthful alternative unit at the k-th position. Finally, we compute the average precision of all 10 rounds. We run the 10-fold cross validation for 30 times with different ways to divide the dataset into 10 subsets to get the average precision. The experiment results are shown in Table V. Again all the truthful alternative units are ranked among the top 5 results.

TABLE V: CROSS-VALIDATION ON ALTERNATIVE UNIT COLLECTION

10-fold cross-validation	
Truthful one as top 1	0.614
Truthful one as top 2	0.912
Truthful one as top 3	0.944
Truthful one as top 4	0.988
Truthful one as top 5	1.0

Alternative statements are extracted from the top N SRRs. For efficiency consideration, it is better to use a smaller N as it would save the SRR downloading and processing time. However, if N is too small, the truthful alternative unit may not be contained in the top N SRRs and would cause problem in determining the truthfulness of the doubtful statement or finding a truthful alternative statement. Therefore, the value of N is a trade-off parameter that should be considered. We perform experiments with statements in the training set to test what value of N is small enough but can ensure that the truthful alternative unit appears in the top 5 extracted terms using our method. We compare the cases with N selected from {10, 50, 100, 150, 200} and show the result in Table VI. We use the AUSV with the best fitness in our training.

According to our experimental results, in 5 out of 50 cases truthful alternative units are not included in the top 5 alternative units if we only consider the top 100 SRRs. When

the number of SRRs increases to 150, the truthful alternative units are extracted and ranked among the top 5 for all the 50 cases. Moreover, they are ranked among the top 4 with 200 SRRs. Based on our experimental results, 150 is an appropriate number of SRRs to be used for the purpose of ensuring that all truthful alternative units be extracted and ranked among the top 5 results. We set N = 200 in our experiments as it produces a better result than when N = 150 as can be seen from Table VI.

TABLE VI: IMPACT OF USING DIFFERENT NUMBERS (N) OF TOP SRRS

	Top 10	**Top 50**	**Top 100**	**Top 150**	**Top 200**
Ranked as 1st	12	21	29	31	31
Ranked as 2nd	6	12	10	12	16
Ranked as 3rd	5	3	3	3	1
Ranked as 4th	3	1	2	2	2
Ranked as 5th	2	0	1	1	0
Not Among top 5	22	13	5	0	0

B. Statement Verification Experiments

Based on our experiment on alternative unit collection shown in last subsection, we only need to compare five alternative statements with the user-entered doubtful statement to determine which one is most likely to be truthful and consequently decide whether the doubtful one is truthful. For the purpose of verification, we developed seven basic rankers as mentioned in section IV: Alternative Unit Ranker (AUR), Hits Ranker, (HR), four Text Feature Rankers (TFR) including Result Coverage (RC), Result Query Relevance (RQR), Result ranking (Rrank), Term Distance (TD), and Domain Authority Ranker (DAR). First, we use all the 50 sample statements to test the precision of each basic ranker. In other words, use each of the basic rankers to rank the doubtful statement as well as its corresponding alternative ones and check whether it ranks the truthful one as the top result.

TABLE VII: PRECISION OF BASIC RANKERS

Ranker	**AUR**	**TFR(TD)**	**TFR(RC)**	**TFR(RQR)**
Precision	0.62	0.32	0.66	0.6
Ranker	**HR**	**DAR**	**TFR(Rrank)**	
Precision	0.20	0.20	0.62	

In Table VII, we show the precision of each basic ranker. Overall, the precision achieved by each basic ranker is not very good. Result Coverage performs better than others. The domain authority result ranker results in the lowest precision because most retrieved SRRs are from domain ".com". Therefore, using the website domains of the SRRs cannot effectively distinguish truthful statements from untruthful ones. Individual basic rankers do not achieve good performance because each one of them evaluates statements based on only one feature. A better way is to combine them together to make a comprehensive evaluation.

Combining ranks from basic rankers would generate an overall rank based on which the system decides the top ranked statement, either an alternative one or the user given one, as truthful. The precision of the verification module is computed as the percentage of test cases which successfully rank the truthful statement as the top result.

In section IV, we introduced three types of rank merging algorithms, Borda-based, Condorcet-based and ProbComb. BaseBorda and BaseCond are two baseline algorithms used to merge the basic rank list. They do not require training. We directly apply the algorithms to all of the 50 statements. PosBorda adopts the idea of position probability. WBorda and WCond assign weights to different basic rankers. WPosBorda combines both the position probability as well as the rankers' weights. ProbComb combines the success rates of different basic rankers. These five algorithms require training for either the position probability or the rankers' weights/success rates. So we use 10-fold cross validation to evaluate the algorithms' performance. Overall, we run cross validation for 30 times and use the average precision as the final result.

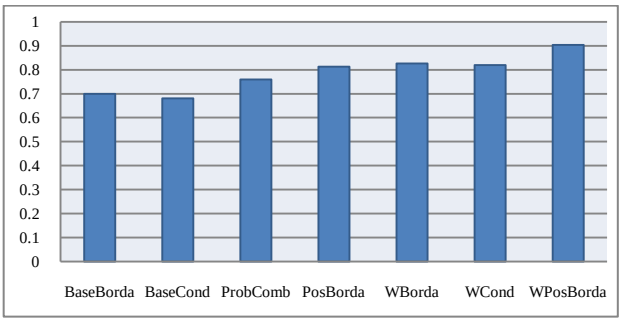

Fig. 5: Precisions of Rank Merging Algorithms

The precisions of all of the seven algorithms are shown in Fig.5. According to the results, the BaseCond gets the worst precision at 0.68 and BaseBorda only achieves 0.7. By incorporating position probability, the ProbComb method gets 0.76 and PosBorda increases this to 0.81. Assigning fine-tuned weights to the basic rankers also improves the precision. The WBorda gets 0.826 and WCond gets 0.82. WPosBorda, which combines both position probability and rankers' weights, achieves the highest average precision at 0.904.

TABLE VIII: LIST OF ERRONEOUS CASES

	Untruthful statements verified as truthful	Truthful
1	*Tom Hanks* was lead actress in the movie 'Sleepless in Seattle'.	Meg Ryan
2	*Apollo* is the first spacecraft on the moon.	Luna2
3	*Sullivan* is the fastest swimmer in the world.	Michael Phelps
4	*Les Paul* invented the electric guitar.	Rickenbacker
5	*English* is the primary language of the Philippines	Filipino

We also evaluated the precision of the ID3 decision tree method [24] for truthfulness verification. For the 10-fold cross-validation, the average precision of ID3 is 0.66, which is much lower than all of our rank merging algorithms.

In Table VIII, we show the 5 cases WPosBorda made incorrect verification. In Case 1, our verification algorithm failed to recognize that "Tom Hanks" is a man which mismatches with "lead actress". In Case 2, Apollo is the first "manned" spacecraft on the moon while Luna2 is the first spacecraft on the moon. "Tom Hanks" and "Apollo" appear much more frequently than "Meg Ryan" and "Luna2" on the

Web, respectively. In Case 3, we expected "Michael Phelps" to be the truthful alternative unit. But our system selects "Sullivan" as truthful because he broke the 50-meter freestyle world record in 2008 and was called the "fastest swimmer in the world" in many news reports. Though Michael Phelps is widely considered as the fastest swimmer today, not many web pages use the phrase "fastest swimmer" to describe him. Note that when TREC-9 was held in 2000, "Sandy Neilson" was given as the correct answer for Case 2, which is way out of date now. The answer for the statement in Case 4 is controversial. TREC gave "Rickenbacker" as the correct answer. However, "Les Paul" is the inventor of the first solid-body electric guitar, while "Rickenbacker" invented semi hollow body electric guitar. So it is difficult to say who the inventor of electric guitar is. Case 5 fails because according to most web pages, both English and Filipino are the official language of the Philippines though TREC 9 gives only "Filipino" as the correct answer. So it could be argued that for Cases 4 and 5, the correct answers are not unique and our system did not actually fail. In the future, we will investigate a systematic solution to deal with the situation where a doubtful statement may have multiple truthful alternative units.

All of the above experiments were based on combining all the basic rankers. Because the precision of each individual ranker differs from one another and we have obtained the precision of each basic ranker, we experimented with merging only the k most accurate rankers (k = 2, 3, …,7) to observe the impact on the overall precision. We use only the WPosBorda merging algorithm in this experiment as it has the best performance among all merging algorithms we tested. The experimental results are shown in Fig.6. According to our experiment, merging the top 5 basic rankers (AUR, TFR(RC), TFR(RQR), TFR(Rrank), TFR(TD)) can achieve 0.9 precision, same as using all seven basic rankers. The fact that adding HR and DAR does not gain any benefits suggests that these two rankers are not effective in differentiating truthful and untruthful statements.

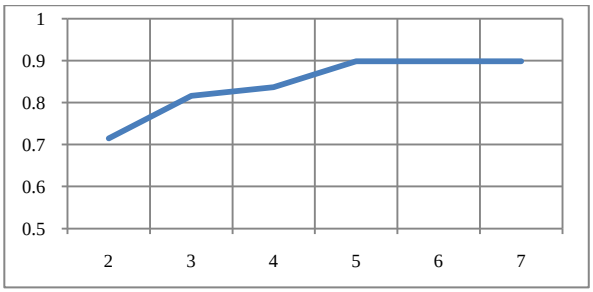

Fig. 6: Precision of partially merging basic rankers

VI. Related Work

The quality of web page content has become a known concern in recent years. Untruthful information on the web often misleads common users and is especially harmful when it spreads misconceptions to the young generation. The web information trust issue could be addressed at three levels: statement level, web page level and website level.

- Some researchers have started discussing the truthfulness of individual statements in recent years. The "Honto" system [16, 17] has a similar objective as ours. We will compare their work with ours in detail in the next paragraph.
- At the page level, current works mainly focus on detecting spam pages based on link analysis and content features for the purpose of filtering out low quality web pages [5, 12]. However, spam and "untruthful" are two relevant but different concepts. Untruthful pages are not necessarily spams. There are many non-spam pages presenting incorrect facts which cause users' confusion. In order to identify untruthful statements, we need pay more effort on analysing the statements in pages.
- To our best knowledge, there is not any published paper discussing the website trust issue except considering the domain of a website.

Recently, researchers from Kyoto University proposed their system "Honto?Search" for the purpose of verifying the truthfulness of facts [16, 17]. The basic idea is the same as ours, which looks for alternative statements via a search engine and finds most likely truthful one from them. Their method is based on hits numbers, temporal information (i.e., when a page is published) [16] and sentimental factors [17]. However, several important issues are not addressed in their work. First, the authors do not provide the actual techniques used for alternative statement collection although they mention that the collection is based on the search results of the doubtful statement. Alternative statement collection is an important step in a complete solution to determine the truthfulness of the doubtful statement and to identify the truthful statement. Second, they do not utilize several piece of relevant information which is essential for truthfulness verification. Specifically, text similarity between the query statement and SRRs, ranking positions of SRRs, semantic closeness and correlation between the doubt unit and each alternative unit are not considered in their solution. They also do not give a systematic algorithm to combine the factors. Lastly, in their experiment, they only performed experiments on statements with temporal doubt unit (e.g. which year event A happens) and the precision of their method is only 62%, which is much lower than the precision of our methods.

Answer verification in question answering is relevant to the statement truthfulness verification problem. [9, 10] proposed a method for the answer verification problem. This method extracts keywords from the question (Qsp) as well as each candidate answer (Asp), computes the co-occurrence of Qsp and Asp on web pages based on the hits numbers from querying Qsp and Asp together and separately. The more often a candidate answer co-occurs with question keywords, the more likely it is the right answer. In our method, we also utilize the co-occurrence information to find alternative units. But it is only one of seven features we use in our alternative statement collection algorithm. Moreover, after obtaining alternative units, we use a verification algorithm to determine the truthfulness of each of them.

To the best of our knowledge, very few papers address the answer verification problem directly, although many systems incorporate answer verification into their candidate answer ranking component. The ranking component scores candidate answers based on certain features (e.g., document frequency), and considers the answer with the highest score as the most likely correct one [7]. We summarize these features as follows. (a) The entity type associated with a candidate answer should match the question type. (b) The number of occurrences of a candidate answer in retrieved relevant document set has an impact on its score as stated in [13]. (c) The rank of retrieved documents. In AnswerBus [19], documents that are ranked higher in the return list are assigned higher scores. (d) Distance between candidate answers and question keywords. In [17], the FDUQA system defines the distance as the distance between answer and question keywords in the grammar parsing tree. [19] assigns a weight to each keyword and scores a candidate answer by its word distance from the keywords multiplied by the corresponding weight of each keyword.

Naturally, our method also uses some of the features used in previous solutions. But our method is significantly different from the existing solutions as summarized below. (1) Our method consists of two carefully designed phases (i.e., alternative statement collection and statement truthfulness verification) which explore different sets of features, including new features not used before such as the semantic closeness between the doubt unit and each alternative unit. None of the existing solutions used all of these features in a single solution and used them like our method. None of them has studied both phases as comprehensively as our work. In fact, most of them studied only one of these phases. (2) Our alternative statement collection algorithm has achieved always including the truthful alternative unit among the top 5 results in our experiment. To the best of our knowledge, similar results have not been reported before. (3) Based on seven basic rankers, we have evaluated and compared a large number of ranking merging algorithms for statement truthfulness verification. Similar studies have not been reported before in related applications. The best merging algorithm (WPosBorda), that we proposed in this paper, achieved a precision of about 90%. And if we counted Cases 4&5 in Table VIII as correct, the precision of WPosBorda would reach 94%.

VII. CONCLUSION

In this paper, we studied the statement level trust problem and proposed a two-step (alternative statement collection and statement truthfulness verification) approach to automatically determine the truthfulness of statements given by users. Both of these two steps exploit a number of features relevant to truthfulness, and our experimental results showed that these features as well as our combining methods are capable of handling most truthfulness identification cases. Besides textual features, we illustrated the effectiveness of utilizing semantic information of the doubt unit and the correlation between the doubt unit and each alternative unit for alternative unit selection. In addition, we proposed a new rank merging algorithm (WPosBorda), evaluated and compared the performance of a large number of merging algorithms for statement truthfulness verification including several Borda-based and Condorcet-based algorithms as well as a probabilistic method. There is still room for improvement, such as negation analysis on statements in SRRs and coping with multiple doubt units in a doubtful statement.

ACKNOWLEDGEMENTS

This work is supported in part by the following NSF grants: IIS-0842608, IIS-0842546 and CNS-0958501. We would like to thank Yadnesh Baviskar for providing the code for comparing different query specification methods.

REFERENCES

[1] J. A. Aslam, M. H. Montague: Models for Metasearch. SIGIR 2001: 275-284.
[2] J. C. de Borda. M´emoire sur les ´elections au scrutin. In Histoire de l'Academie Royale des Sciences. Paris, 1781.
[3] M. de Condorcet. Essai sur l'application de l'analyse `a la probabilit´e des decisions rendues `a la pluralit´e des voix, 1785.
[4] D. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, 1989.
[5] Z. Gyöngyi, H. Garcia-Molina, J. Pedersen: Combating Web Spam with TrustRank. VLDB 2004:576-587.
[6] M. R. Henzinger, R. Motwani, C. Silverstein: Challenges in Web Search Engines. IJCAI 2003: 1573-1579.
[7] C. T. Kwok, O. Etzioni, D. S. Weld: Scaling question answering to the web. ACM Trans. Inf. Syst. 19(3): 242-262 (2001).
[8] S. Liu, F. Liu, C. Yu, W. Meng: An effective approach to document retrieval via utilizing WordNet and recognizing phrases. SIGIR 2004: 266-272.
[9] B. Magnini, M. Negri, R. Prevete, H. Tanev: Is It the Right Answer? Exploiting Web Redundancy for Answer Validation. ACL 2002: 425-432.
[10] B. Magnini, M. Negri, R. Prevete, H. Tanev: Mining Knowledge from Repeated Co-Occurrences: DIOGENE at TREC 2002. TREC 2002.
[11] M. H. Montague, J. A. Aslam: Condorcet fusion for improved retrieval. CIKM 2002: 538-548.
[12] A. Ntoulas, M. Najork, M. Manasse, D. Fetterly: Detecting spam web pages through content analysis. WWW 2006: 83-92.
[13] X. Qiu, B. Li, C. Shen, L. Wu, X. Huang, Y. Zhou: FDUQA on TREC2007 QA Track. Sixteenth Text REtrieval Conference (TREC), 2007.
[14] C. Silverstein, M. R. Henzinger, H. Marais, M. Moricz: Analysis of a Very Large Web Search Engine Query Log. SIGIR Forum 33(1): 6-12 (1999).
[15] Z. Wu and M. Palmer. "Verb semantics and lexical selection". 32nd Annual Meeting of the Associations for Computational Linguistics, pp 133-138. 1994.
[16] Y. Yamamoto, T. O. Tezuka, A. Jatowt, K. Tanaka: Honto? Search: Estimating Trustworthiness of Web Information by Search Results Aggregation and Temporal Analysis. APWeb/WAIM 2007: 253-264.
[17] Y. Yamamoto, T. Tezuka, A. Jatowt, K. Tanaka: Supporting Judgment of Fact Trustworthiness Considering Temporal and Sentimental Aspects. In WISE 2008: 206-220.
[18] X. Zhang, B. Han, and W. Liang: Automatic seed set expansion for trust propagation based anti-spamming algorithms. WIDM 2009: 31-38.
[19] Z. Zheng: AnswerBus Question Answering System. In HLT 2002: 399-404.
[20] http://trec.nist.gov/data/qamain.html
[21] NLTK: http://www.nltk.org
[22] WordNet: http://wordnet.princeton.edu/
[23] www.cs.binghamton.edu/~xianli/doubtful_statement.html.
[24] www.cs.waikato.ac.nz/ml/weka/