Artificial Intelligence in Clinical Medicine
Serpensoleum

Mohamed Abdalla, Munir Al - Dajani, Alex Adam, Alex Urda

[1]University of Toronto – Department of Computer Science
27 King's College Circle – Toronto – ON – Canada

{mohamed.abdalla, munir.al.dajani, alex.adam, alex.urda}@mail.utoronto.ca

Abstract. *The web is an increasingly popular platform for delivering health care information. In this work, we introduce Serpensoleum, a Google Chrome add-on, to assist in identifying medically trustworthy search results from those which are untrustworthy, using only textual features to distinguish between different classes of web pages. First, we investigated the differences between various parsing and querying techniques, and second, we experiment with two main approaches to the classification of web pages into three categories of: trustworthy (contains factual medical claims), untrustworthy (contains false medical claims), or irrelevant (containing no claims related to the query). Both techniques (sequential binary classifiers, and clustering) show promise with the task of identifying web pages containing misinformation, with the sequential binary classification approach achieving 80% accuracy, and preliminary results showing the clustering approach achieving 88% accuracy.*

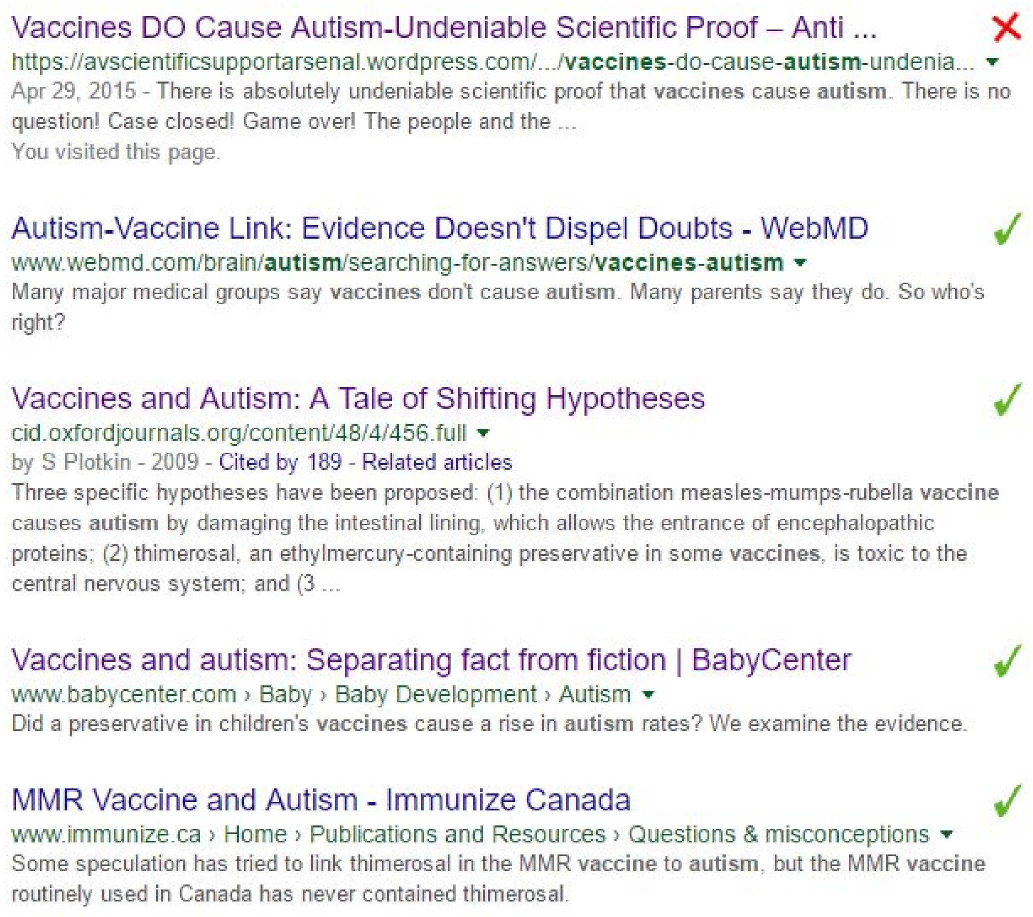
1. Introduction

The web is an ever-increasing platform for healthcare information, with an estimated 87% of internet users saying they have looked online for health information within the past year [Fox and Duggan 2013]. Although this may seem like an excellent way for someone to make quick and informed decisions about their health, some sites may actually provide information that is not trustworthy. As an example, Chung *et. al.*'s study showed that 28% of sites provided inaccurate health information while performing Google searches on infant sleeping safety. [Chung et al. 2012] The informed internet reader most likely knows to verify information by checking more than one source, but not everyone has the time to investigate different viewpoints of a particular topic. Ultimately, users would greatly benefit from a tool to provide them alerts when the web page they are browsing contains false health claims.

In this work, we describe Serpensoleum, a tool that gives users a better understanding of web pages[1] by identifying any highly disputed medical claims. Serpensoleum is a Google Chrome extension that a user can install through their browser. After installation, Serpensoleum will augment Google search results with a colour indicator placed beside the result: green checkmark for pages that contain truthful medical information, red "X" for pages that make false medical claims, and grey circle for pages that are irrelevant. (Figure 1).

Serpensoleum is designed as a client-server model (pipeline shown in Figure 2). The client resides as the plugin for the user's browser. It monitors the user's browsing activity

[1]In the report website is used to mean only single web pages.


Vaccines DO Cause Autism-Undeniable Scientific Proof – Anti ... ✗
https://avscientificsupportarsenal.wordpress.com/.../**vaccines**-do-cause-**autism**-undenia...
Apr 29, 2015 - There is absolutely undeniable scientific proof that **vaccines** cause **autism**. There is no question! Case closed! Game over! The people and the ...
You visited this page.

Autism-Vaccine Link: Evidence Doesn't Dispel Doubts - WebMD ✓
www.webmd.com/brain/**autism**/searching-for-answers/**vaccines-autism**
Many major medical groups say **vaccines** don't cause **autism**. Many parents say they do. So who's right?

Vaccines and Autism: A Tale of Shifting Hypotheses ✓
cid.oxfordjournals.org/content/48/4/456.full
by S Plotkin - 2009 - Cited by 189 - Related articles
Three specific hypotheses have been proposed: (1) the combination measles-mumps-rubella **vaccine** causes **autism** by damaging the intestinal lining, which allows the entrance of encephalopathic proteins; (2) thimerosal, an ethylmercury-containing preservative in some **vaccines**, is toxic to the central nervous system; and (3 ...

Vaccines and autism: Separating fact from fiction | BabyCenter ✓
www.babycenter.com › Baby › Baby Development › Autism
Did a preservative in children's **vaccines** cause a rise in **autism** rates? We examine the evidence.

MMR Vaccine and Autism - Immunize Canada ✓
www.immunize.ca › Home › Publications and Resources › Questions & misconceptions
Some speculation has tried to link thimerosal in the MMR **vaccine** to **autism**, but the MMR **vaccine** routinely used in Canada has never contained thimerosal.

Figure 1. Serpensoleum automatically labels search results

for Google searches. Upon a new search, the search query along with every search result URL is sent to the server in order to retrieve a reliability rating. The server analyses each URL and classifies it as "Truthful", "False", or "Irrelevant". A false claim is a medical claim that is not backed by scientific evidence. For example "vaccines cause autism", "black salve can cure skin cancer", or "cell phones cause cancer" are all false claims.

Here, we describe the design of Serpensoleum, the consequent hurdles that must be surmounted in order to make tools like our add-on work well (such as data collection, parsing, and prediction), the results obtained by applying different machine learning techniques, as well as discuss the possible improvements for future implementations.

2. Background & Related Work

In this section we address previous publications which in some way deal with both our general goal, and high-level approach. We also consider more fine-grained/narrow-focused publications (usually of newly developed methods) which while not applied to our goal (in their original publication), show potential to serve as methods used during the development of our model.

2.1. Previous Solutions

2.1.1. Content-Based Systems

Content-based systems are systems which attempt to arrive at a classification through the careful examination of the content on a chosen website (usually its text). Such approaches usually use machine learning methods (such as Support vector machine (SVM), Bayesian models, and Neural Networks) and labeled data to analyze the text and classify the website [[Ennals et al. 2010],[Abbasi and Chen 2009],[Shen et al. 2006]].

One such content-based approach used by Ennals et al. [Ennals et al. 2010], developed a Firefox add-on which would highlight disputed claims on the web page. Unlike our planned approach, they first developed a corpus of disputed claims using both user inputted claims (an add-on feature), and mining such claims from websites like Snopes and Politifact. Their add-on would compare all of the sentences on a selected web page with disputed claims from their database using "Textual entailment algorithms" to find matches. The textual entailment algorithm used is discussed in detail under 'Claim Comparison' below. While their application was met with enthusiasm, its inability to function properly/up to expectations (no quantifiable metrics) caused disappointment.

Another content-based approach by Boyer and Dolamic [Boyer and Dolamic 2014] developed a model which applied machine learning methods to the task of detecting the trustworthiness of a website based on the HONCode [Baujard et al. 2010] criteria of conduct. The HON code of conduct is a set of ethical principles defined by the HON (Health on the Net) foundation used to assess the quality of health information online. There are 8 quality principles (outlined in the paper) that need to be checked and deemed acceptable for a website to be certified by the HONCode. They manually curated (both positive and negative) sentences which satisfied each of the 8 quality principles and built binary classifiers for each of the principles. Negative examples for a given criterion were documents that supported criteria other than the one being considered. The machine learning algorithms that they used were Naive Bayes classifiers, and C-SVC SVMs with a radial basis function (Gaussian) as the kernel. They trained multiple different classifiers using these models by varying the feature types, and feature reduction levels. Preprocessing of the data included stemming, and removal of stop words. They experimented with actual representation of the data looking at: (1) Bag of Words, (2) Bi-gram Model, and (3) Co-occurrence matrix model. They chose features based on document frequency. Only features whose document frequency exceeded a predefined threshold (t) (after pre-processing) were kept. They experimented with $t = 30\%$, $t = 50\%$, and $t = 80\%$.

The results were expressed using precision, recall, and F-measure. The two classifiers used had very high differences in certain criteria and data-representation + feature selection threshold. For most criteria and for the generally best performing data-representation and feature selection threshold, the difference between the SVM and NB didn't exceed 15%.

The precision is at or above 73% with an F-measure at or above 71% for all criteria except for “References” and “Justifiability”, for which the precision values were 65% and 69%, with F-measure values of 64% and 63% respectively. The paper cites an article which states that the agreement of manual classification between two persons rarely gives more than 72%, which means that the best system reported in the paper outperforms manual classification.

Our own system will be a Content-Based system examining only the text of websites. Unlike the majority of Content-Based Systems, we will aim to provide a more fine-grained classification of which exact sentences are incorrect/unproven as opposed to just labelling the whole site/page as untrustworthy.

2.1.2. Graph-Based Systems

Graph-Based Systems are systems which use the underlying graph/network structure from a web page and its links (both to and from a website) to determine legitimacy. This idea is at its core an inspired version of the PageRank algorithm [Page et al. 1999] developed by Google to determine authority score. In fact, many publications [[Gyöngyi et al. 2004],[Gyongyi et al. 2006]] use publications of some sort PageRank derivative to classify websites. These systems usually only serve as baseline models, as more advanced web administrators can leverage “link-farms” to inflate their authority score, and fool naive PageRank-inspired systems.

2.1.3. Hybrid Systems

Hybrid Systems usually combine the previous two approaches. They have a model which examines the content of the website in question, and another model which looks at its underlying graph structure. [Abbasi et al. 2012] is an example of such a hybrid system and use an algorithm called “Recursive trust labeling (RTL)” which fuses content and graph classifiers to arrive at a classification. RTL works by having two classifiers (graph-based and content-based) and “recursively” re-running training on the dataset and changing weights until the algorithm converges. This approach gets around 90% accuracy individually and 94% when combined, beating other models by a margin of 5% on their curated test set of 1000 legitimate and fake medical websites.

Another hybrid approach taken by Sondhi *et al.* [Sondhi et al. 2012], combined both the reliability criteria from the HON foundation, with additional features including graph-based ones. They provide some rationale behind the selection of feature groups, such as those looking to sell are more likely less reliable than those who are just offering information. The models experimented with were all SVMs trained on different feature set combinations, with varying amounts of training data, and different weighting on each of the feature sets. The optimized configuration (using all features) resulted in an overall weighted accuracy of 80% on the test set. Moreover, Google’s reliability Mean Average Precision (MAP) over 22 queries was found to be 0.753, and after re-ranking, the reliability MAP improved to 0.817.

After looking at approaches which directly relate to our goal, we also decided to look into publications which discussed methods which we could make use of. Our approach contains two basic steps to arrive at classification for websites. Firstly, we must extract the claims from a given website (discussed in 'Claim Extraction and simplification'), and then we must compare these claims with other claims sourced from the Internet (discussed in 'Claim Comparison').

2.2. Claim Extraction, Simplification, & Comparison

2.2.1. Claim Extraction

In our task, one of the steps we must take is identification of claims. Claims are defined as phrases/sentences which assert/state something. They may or may not provide proof for their statements. Before we are able to assess whether or not a site's claims are true, we will need to be able to classify the given sentence. There are methods which could possibly be used for the identification/extraction of claims.

Recurrent neural networks (RNNs) are commonly used in language tasks of both sentence classification, and sentence generation [[Pang and Lee 2004], [Tai et al. 2015], [Karpathy and Fei-Fei 2015]]. Shen and Zhang [[Shen and Zhang 2016]] conducted an empirical evaluation of some of the most commonly used RNN architectures on multiple sentence classification tasks such as sentiment analysis, subjectivity classification, and question classification. They conducted an empirical study of different yet commonly used RNN structures and their effects on sentence classification.

The structures they compared included "Tail Model", "Pool Model", and a hybrid model. No model had superior performance across all the tests, however we chose to focus on the subjectivity test (determining whether or not a sentence is subjective or objective) as we believe it is closer to our task than sentiment analysis or the other tested tasks. The biggest challenge to implementing the methods described by Shen and Zhang is the data collection. All of these methods are supervised and require a lot of data to prevent over fitting.

2.2.2. Claim Simplification

Claim simplification is the task of simplifying of sentences to more readable forms that approximate a sentence's semantic meaning. Such a task can be useful in the preprocessing stage of a machine learning pipeline as a way of eliminating noise caused by syntactic differences among sentences. This is an approach inspired by the historical usage of suffix removal to improve information retrieval processes (IR) [Porter 1980]. There are many such approaches to sentence simplification [[Narayan and Gardent 2015], [Siddharthan 2010], [Vanderwende et al. 2007]] which could be used for claim simplification, but we chose to focus on a single one.

One of the most important features of the approach taken by [[Narayan and Gardent 2015]] is that it is completely unsupervised, so the model being built doesn't rely on being given an unsimplified sentence and its simplified counterpart as input. Furthermore, the method used allows for the splitting of simple sentences into multiple sentences. This is a way of making the text more modular, and is crucial to identifying different semantic statements made within a sentence.

The first step is to provide a sentence as input. Once this is done, the sentence undergoes a lexical simplification phase where words are replaced with simpler words while still retaining the context and meaning of the sentence. Then, the sentence is split into two or more sentences if possible. In the final phase, some words are removed from the sentence(s) completely in such a way that their semantic meaning is (relatively) unchanged.

The authors used a number of metrics to evaluate the performance of their model in comparison to existing models. They tested their model on a test dataset of 100 complex sentences and their 131 parallel simple sentences from Wikipedia and Simple Wikipedia. and reported the following metrics: number of splits overall, number of sentences not simplified, and Levenshtein distance between input and output. They achieved a much better simplification rate compared to existing methods. When evaluating the modules separately, it was reported that the word deletion module is the most effective in terms of simplifying sentences.

2.2.3. Claim Comparison

Another major task that we face once we have extracted the claims from pages on similar/equivalent topics is to compare the claims to find agreement/disagreement. Here we return to Ennals et al.'s "Textual entailment algorithms". Their own approach used a very simplistic method called local lexical matching (LLM)[Jijkoun and De Rijke 2006]. Their implementation of the algorithm (which works on the sentence level) removes stopwords, applies a stemmer, and then attempts to match words. They take the presence of negation words to imply disagreement. They admit that LLM is "far from being state of the art", but it serves well as a benchmark. As said above, their results were sub-par, and although no quantifiable metrics where stated, they stated that the tool had both low precision and low recall during their qualitative user studies.

Ennals et al. also discuss possible improvements on their algorithm by suggesting other more modern approaches including the construction of syntax trees, treating sentences as logical formulas and attempting logical proofs, and building learning models which can infer if two phrases agree or not.

3. Methods & Results

Creating the add-on requires addressing a number of challenges, including:
Data collection: To provide accurate warnings to users, a large corpus of scientifically proven medical information must be created. If this set is too small, Serpensoleum may rarely identify any false claims. If some items in the set are not credible, it may identify

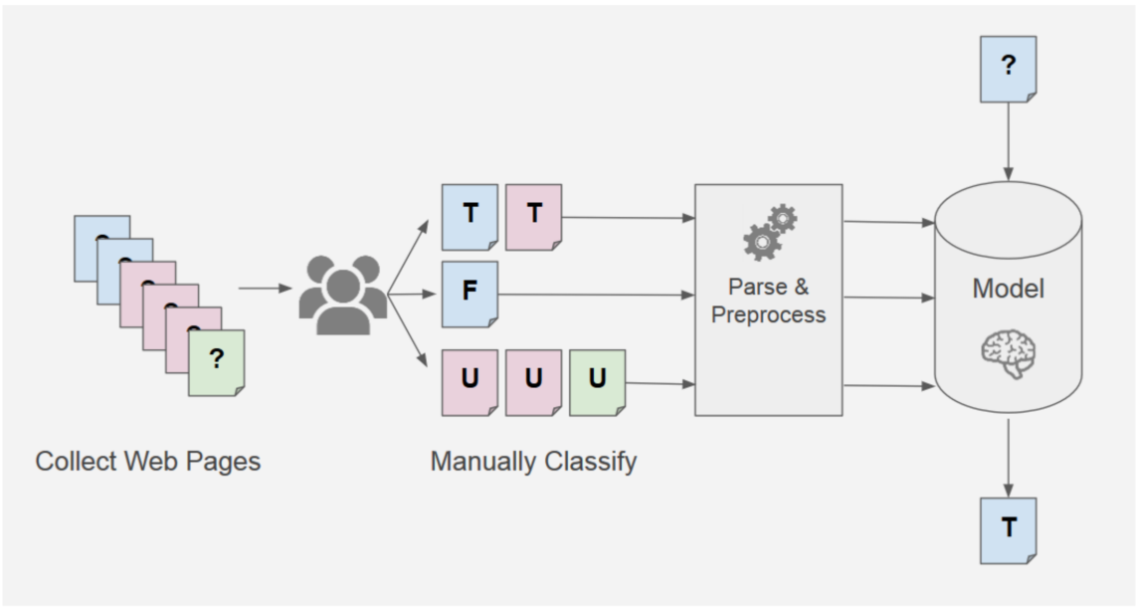


Figure 2. The pipeline for the add-on

truthful claims as being false, leading to misinformation.

Parsing: Given a particular website, how do we determine which section is the main body of text? Can we perform some pre-processing techniques such as removing stop words or stemming in order to provide more consistent data?

Detecting false claims: Once the web pages have been parsed and pre-processed, we aim to determine whether or not a given web page is trustworthy (contains factual medical claims), untrustworthy (contains false medical claims), or irrelevant (containing no claims related to the query).

These individual challenges are not unique to Serpensoleum, and can be applied to any tool that identifies false claims, and is not limited to just medical claims.

In the rest of this section, we discuss the different methods tested for solving the above problems, as well as their benefits or shortfalls.

3.1. Data Collection

Since the data plays such a critical role towards the success of subsequent steps in our process, we must ensure that the examples we collect are both numerous and relevant. In order to train models for detecting false claims, a list of 496 labeled websites was created by our group. Different techniques were tested in order to generate a high concentration of websites that contained either false or factual claims. The list of websites was generated by performing Google searches of controversial medical topics and downloading the top 25-50 results. Some examples of topics searched are: “vaccines and autism”, “cell-phones and cancer”, and “black salve cancer treatment”.

After collection, members of our group classified each result by visiting the website and marking the page as either “True”, “False”, or “Not Relevant”.

A page would be marked as Not Relevant if its content did not relate to the original search query. Examples include the web page being a video, or the body of the web page not containing query related discussions.

Pages that are considered relevant to the query and that are backed up by facts or a proven belief are classified as True.

Finally, a False page is one that is still relevant to the query, but makes claims that are highly questionable and not backed by scientific facts.

3.1.1. Baseline

As a comparative baseline, neutral queries on controversial topics were used to get a total of 83 Google search results. A neutral query is one that does not mention any additional information that could incite more results from trustworthy or untrustworthy websites. An example of a neutral query is “vaccines and autism”. For illustrative purposes, a non-neutral query could be: “proof vaccines cause autism”.

After collecting a total of 83 results on 3 different myths for our baseline, 48% of results were True, 28% False, and 24% Not Relevant.

3.1.2. Trustworthy Examples

Two main methods of collecting a high distribution of factual websites were tested, each with their advantages and disadvantages.

The first method we tested was the use of Schema.org filters. Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on web pages on the Internet. More specifically, we chose to filter all Google search results to display only websites that implemented the MedicalEntity schema [2]. Schema.org estimates that around 100-1000 websites currently adhere to the MedicalEntity schema.

After collecting a total of 86 results on 1 myth, 33% of results were True, 23% False, and 55% Not Relevant. The results were surprising since we did not expect a lower percentage of true results than our baseline. In fact, the majority of websites gathered

[2]https://health-lifesci.schema.org/MedicalEntity

using this method were found to be irrelevant to the original query. We believe the cause of this is the limited number of websites implementing the MedicalEntity schema. Relevant results are scarce, and this is reflected when looking at the final numbers. It is also worth mentioning that any website, factual or not, can implement this schema since there is no oversight. This means there is no guarantee that these websites are truthful without manually checking.

The second method we evaluated overcame the weakness that Schema.org had with small number of total sites and lack of oversight. We used a list of over 7,000 websites accredited by HONCode (as described in the Background and Related Work section) to generate our results.

After collecting a total of 165 results on 4 myths, 55% of results were True, 0.1% False, and 44.9% Not Relevant. We can see a very large decrease of false websites, and an increase of truthful ones. This is expected since all sites follow the 8 HONCode principles for online medical and health information.

3.1.3. Untrustworthy Examples

Similar to developing methods to find trustworthy examples, two methods were tested to collect a high distribution of false examples.

The first method we tested was very similar to our baseline approach, but instead of using neutral queries, we added keywords to promote false website results. As an example, a search on the topic “vaccines and autism” can be turned into “proof vaccines do cause autism”. We call this method of applying a negative bias to the search query ‘Negsearch’.

After collecting a total of 44 results on 1 myth, 41% of results were True, 43% False, and 16% Not Relevant. Compared to the baseline, there was an overall increase in the percentage of false websites. Surprisingly, the percentage of true websites remained almost the same. We speculate that this is due to the fact that Google has incentive to reduce the amount of false medical information in its search results [3]. This could mean that even if the search query is biased towards getting false results, Google will compensate to still show truthful ones as well.

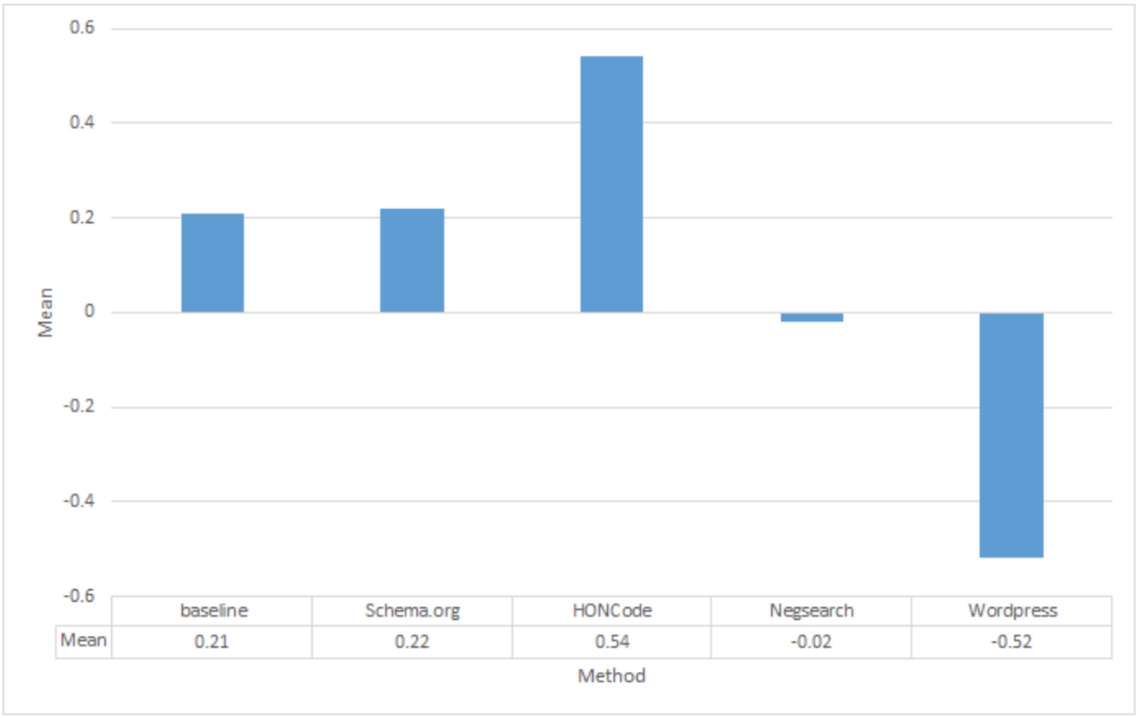
The second method we investigated was filtering search results for only Wordpress blogs. After performing health related searches in the past, our group noticed that a large amount of false websites were blogging sites, typically hosted by Wordpress. We decided to test if this observation was true by actually analysing the results.

After collecting a total of 118 results on 4 myths, 13% of results were True, 64% False, and 23% Not Relevant. Compared to Negsearch, there is a drastic difference in results;

[3]https://blog.google/products/search/im-feeling-yucky-searching-for-symptoms/

many more websites are false. These results also help demonstrate the sources of a lot of fake health information. Blogs posts are not regulated and thus anyone who has an opinion on a medical matter could publish an article. An uninformed person could later visit these blogs and gain the impression that the content is true, leading to the further spreading of misinformation.

3.1.4. Statistical Significance



	baseline	Schema.org	HONCode	Negsearch	Wordpress
Mean	0.21	0.22	0.54	-0.02	-0.52

Figure 3. Means of Different Data Collection Methods

In order to quantitatively state that one method of data collection was had a higher proportion of True, or False websites than the baseline, we devised a way to quantitatively describe these categories. Every category was assigned an integer value of 1, 0, -1 for True, Not Related, and False categories respectively. Thus, we can now calculate the mean of every data collection method since every website has a value of 1, 0, or -1. The closer the mean is to 1, the higher the proportion of True websites in that category. The opposite is true for False websites.

To determine if the results for each data collection method are significant, a *t*-test was performed between the baseline and each of the other different methods using the Scipy Python library. The sample set for each method was the list of classifications for each website converted to its respected integer value. For example, if our HONCode category had the following 4 examples True, True, False, Not Related, the sample set would be $\{1, 1, -1, 0\}$. This distribution would then be tested for significance against our baseline.

Table 1. Null Hypothesis Table

	Schema.org	HONCode	Negsearch	Wordpress
p-value	0.89	0.0013	0.18	2.64E-09
H_0 rejected?	No	Yes	No	Yes

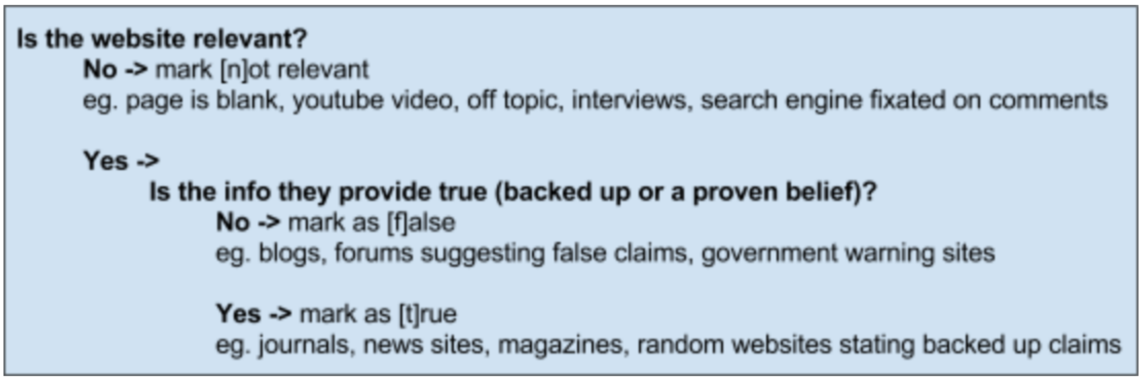
The following hypothesis was tested for each method:

H_0 = *The distribution of true, false, and not related websites is similar between the collection method and the baseline*

The results in Table 1 are from Scipy, with a significance level of 0.05.

Using Figure 2 and Table 1, we can conclude that HONCode has a significantly higher proportion of truthful websites, while Wordpress has a significantly higher proportion of false websites when compared to our baseline.

3.1.5. Inter-Rater Agreement Measure



Is the website relevant?
No -> mark [n]ot relevant
eg. page is blank, youtube video, off topic, interviews, search engine fixated on comments

Yes ->
Is the info they provide true (backed up or a proven belief)?
No -> mark as [f]alse
eg. blogs, forums suggesting false claims, government warning sites

Yes -> mark as [t]rue
eg. journals, news sites, magazines, random websites stating backed up claims

Figure 4. Categorization Flowchart

Because of the large amount of websites to manually categorize, we split this task among everyone in the group. Every member followed the categorization flowchart in Figure 4 in order to mark each website as True, False, or Not Related. However, since this is a largely subjective task, it does introduce a potential source of misclassifications between different people.

In order to assess the reliability of agreement between different members of our group, we performed Fleiss' kappa statistic to quantify our accuracy. Fleiss' kappa was used rather than Cohen's kappa since there are more than 2 categories each person can choose. The measure was performed on 21 websites that were rated by at least 2 group members, independently. We calculated $\kappa = 0.81$, which can be interpreted as "almost perfect agreement" [Landis and Koch 1977].

3.1.6. Final Observations

Task	Approach	Expected Majority	% Truthful	% Untruthful	% Unrelated
Baseline	Google Search	-	48%	28%	24%
Gathering True Sites	**HONCode**	Truthful	**55%**	0.1%	44.9%
	Schema.org		33%	12%	**55%**
Gathering False Sites	**Wordpress**	Untruthful	13%	**64%**	23%
	Negsearch		41%	**43%**	16%

In conclusion, we were successful in finding data collection methods that return a significantly high concentration of either true or false websites. Together with the irrelevancy module (described below in the Irrelevancy Module section), HONCode results can be filtered for websites which are not relevant to produce a large dataset of true sites. On the other side, Wordpress is currently our best source for retrieving examples of false websites.

3.2. Parsing

Given that categorized examples from the Data Collection section are saved as HTML files, the relevant content of these pages must be extracted. It is important to reduce input noise by ensuring that the majority of the main content is kept, and things like ads, images, and scripts are ignored.

The HTML will be parsed by the Python library BeautifulSoup. Python was chosen since the machine learning aspects of the project will also be done in Python, so having consistency between preprocessing and learning is desirable. Secondly, BeautifulSoup was chosen due to its simplicity, abundance of documentation, and many recommendations on websites such as stackoverflow.com. BeautifulSoup is an HTML parser that is also capable of representing the DOM as a Python object so that queries can easily be made. An alternative to this would be Python's built in minidom class, but that has several limitations since its focus is XML. BeautifulSoup enables the use of CSS selectors when searching the DOM, and this can save a lot of development time in comparison to minidom.

3.2.1. Approach 1: HTML Tags

The first approach was to assume that all of the relevant text is contained in <p>(paragraph) tags. Through the BeautifulSoup library, a simple query was performed to obtain all of the text inside <p>tags. Then, using regular expressions to check for punctuation, so long as the paragraph text was longer than 5 words, the parsed text would be included in the output result.

3.2.2. Approach 2: Regular Expressions

The second approach was less limiting in terms of assuming a certain structure for the HTML document. Instead of assuming that content is contained in<p>tags, this regular expression based approach uses the notion that content must be in elements which have no children. This enables us to find content like lists which can not be found by the first approach. Every DOM element in the body is iterated over and, if it has no children, a regular expression is used to separate its text content into sentences. These sentences are then recombined into a paragraph and included in the output.

3.2.3. Results

	Precision	**Recall**
Approach 1	72.4%	65.2%
Approach 2	60.4%	83.3%

Precision and recall for both parsing approaches were performed. A total of 10 websites were randomly selected and parsed. After, the output files were checked by our team sentence-by-sentence to see if the parsed text reflected the main body of text on the HTML page. Determining the precision and recall was not a subjective task, since we considered a sentence to be correct if, and only if, it was in the main body of text.

Approach 1 is more precise than Approach 2, but it has a much lower recall. This is because the parsing technique is much more strict in terms of filtering. Since our group preferred to cast a wider net, we determined that the better method is Approach 2. Many websites, such as blogs, are highly unstructured. In order to gather as much information out of the HTML pages as possible, we needed to use an approach with a high recall. The sacrifice of having a lower precision rate can be counteracted by preprocessing techniques.

3.3. Detection of Controversial Claims

Given the results of the webcrawler and parser which obtain and process HTML pages, we aim to determine whether or not a given web page is trustworthy (contains factual medical claims), untrustworthy (contains false medical claims), or irrelevant (containing no claims related to the query). At this stage of our research we limited the scope of our algorithm to predicting a single label for the entire website, however in the future we could extend some of our techniques to detect and classify individual claims at the sentence level for a more fine-grained browsing experience.

While there has been considerable work on detecting fraudulent web pages (as discussed in Related Works), our approach is unique in that we aim to make such detection only through the usage of linguistic features. We developed two experimental models for this task.

3.3.1. Staggered Approach

The first experimental approach is a classifier which is composed of two separate machine learning models applied in sequence to arrive at a classification. The first of the two models is an "Irrelevancy Classifier". The aim of this classifier is to determine whether or not a web page is relevant to a given query. This is important for two main reasons:

1. By having an accurate irrelevancy classifier we would be able to automatically gain True examples for many health topic by using HONCode and then removing all the irrlevant web-pages, thereby arriving at an almost 100% True dataset, and
2. Often times Google's search results (especially after the first page of results) are barely related to the query. We believe that our add-on should tell our users if such a web page is to be looked at given the query (as unrelated could be treated as False for a given query, since any advice given would not apply and/or could result in negative medical consequences).

After a web page has been classified as relevant, it is passed onto our second model, the "True/False model" (abbreviated as T/F model), to predict whether or not a given website contains factual and trustworthy. At this stage of our research we only arrive at a classification for the whole web-page but in the future hope to be able to arrive at a fine-grained sentence classification.

3.3.1.1. Irrelevancy Module

For this module there were many different possible approaches that we could take to determining whether or not a web page is relevant. We decided to use term frequency inverse document frequency (TF-IDF) to create vectorized representations of the web pages.

$$\text{TF-IDF}$$
$$\text{For a term } i \text{ in document } j$$
$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{1}$$
$$tf_{i,j} = \text{Number of occurence of term } i \text{ in document } j$$
$$df_i = \text{Number of documents containing } i$$
$$N = \text{ total number of documents}$$

TF-IDF (1) is a standard measurement often used to determine which documents are relevant to a given query [Ramos 2003]. TF-IDF works by determining the relative frequency of given words in a document and comparing it to the inverse proportion of that word over the entire document. The aim of this approach is to determine how important a word is in determining relevance by observing how common such a word is to a particular document when compared to all the documents together.

The models tested include:

- Gaussian Naive Bayes,
- Linear SVM, and
- Simple Feed-Forward Neural Network.

Throughout the experiments performed we tested the models using 5-fold Cross Validation.

We experimented with different vector representations of the documents to determine which feature (or combination of features) would be most effective at determining relevancy.

The first sub-experiment were performed only the topic of Vaccines and Autism, because at the time of development we had not yet manually classified the other topics of the data. For each of the experiments below we performed 5 fold cross validation and reported the averaged micro-average F1-Score (Equation 2).

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2}$$

Experiment 1: Closest Sentence TF-IDF
We would measure how “relevant” each sentence in the document was to the query, and use the value of closest sentence a 1x1 vector representation of the document.

Table 2. Experiment 1 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.592	0.592	0.592

Experiment 2: Closest 5 Sentences
We would measure how .”relevant” each sentence in the document was to the query, and use the values of the closest 5 sentences as a vector representation of the document.

Table 3. Experiment 2 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.592	0.592	0.762

Experiment 3: Using only the title
We would measure how “relevant” each sentence in the document was to the query, and use only the value of the title as a 1x1 vector representation of the document.

Table 4. Experiment 3 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.742	0.638	0.762

Experiment 4: Using all the features above
We would measure how "relevant" each sentence in the document was to the query, and use the values of all the features experimented above in combination as a vector representation of the document.

Table 5. Experiment 4 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.74	0.766	0.78

Experiments on all classes

Experiment 5: Using all the features
Once all of the other topics' data had been classified we were able to experiment on them. We chose to experiment with our most successful features from the "Vaccines and Autism" topic to the combined set of all the features. The values are averaged over 5 runs.

Table 6. Experiment 5 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.704	0.708	0.708

Individual class analysis

As we can see there was a 5-10% decrease in the F1-score when we incorporated all of the classes. To determine if this change was caused by a single low outlier class, or if there was variance amongst all of the topics, we chose to experiment on each of the topics individually.

Topic: Vaccines & Autism
Experiment 6: Using all the features.

Table 7. Experiment 6 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.74	0.766	0.78

Topic: Alkaline Water
Experiment 7: Using all the features.

Table 8. Experiment 7 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.664	0.71	0.642

Topic: Black Salve
Experiment 8: Using all the features.

Table 9. Experiment 8 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.64	0.692	0.716

Topic: Cellphones & Cancer
Experiment 9: Using all the features.

Table 10. Experiment 9 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.532	0.774	0.814

As can be seen from the individual class analysis there appears to be large variance between all of the classes. Such a result would have to be further studied to see if it is caused by the data collected (possibly of lower quality, or fewer examples, different distributions), or if there are just different topics which are inherently (for some reason) are harder to classify as relevant or irrelevant.

3.3.1.2. True/False Module

The experiments we conducted for the T/F module could be split into two main groups:

- manually extracted features, and
- traditional bag of words (BoW).

3.3.1.2.1. Manually Extracted Features

Here we experimented on features that we manually extracted ourselves. The first group of manually extracted feature were sentiment based features. We hypothesised that pages which were less trustworthy were more likely to rely on emotional pitches to convince the readers of their viewpoint, while trustworthy sites would be more neutral as previous research [Castillo et al. 2011] used sentiment based features to analyze credibility on Twitter.

To this end we used the Affective norms for English Words (ANEW) to measure the sentiment of words. [Bradley and Lang 1999] ANEW is a dictionary dataset developed to provide a set of emotional ratings for a large group of words of the English language. Every word in ANEW is measured on the three scales of valence (which ranges from pleasant to unpleasant), arousal (which ranges from calm to excited), and dominance (which ranges from in control to out of control). These values were arrived through a nonverbal pictographic measure: the Self-Assessment Manikin, and ranged from numeric values of 0 to 8 for each category.

For this task any words which were not found in the ANEW dictionary were treated as having a value of None for feature extraction. For each dimension of the ANEW dictionary we extracted 3 features per web page:

- Sum of all the words which had a value in the dictionary,
- Average ANEW value (assigning words not in the dictionary a value of 0).,
- Average ANEW value (not including words which were not in the dictionary).

The other two manually extracted features were synonyms of the word buy, and the average count of weasel word/phrase occurrences per document. [Sondhi et al. 2012] demonstrated that simply looking for synonyms of the word buy (purchase, order, etc.) would allow a classifier to distinguish between trustworthy and untrustworthy sites. The motivation behind this point is that often times websites which are not medically accurate are trying to profit of their misinformation through the selling of products (health guides, supplements, etc).

Weasel words are words or phrases which present themselves as meaningful and specific statements of fact, while truly being only vague and ambiguous. We hypothesised that using weasel words would help in distinguishing as fake websites would be more likely to use weasel words when compared to trustworthy medical sites. Using weasel words is equivalent to tergiversating which is defined as "making evasive statements". The list of weasel words which were used in our algorithm were manually curated from Wikipedia [Wikipedia 2004].

The first set of experiments were conducted upon the topic of "Vaccines and Autism" as this was the only dataset which was completely labelled at the time.

Experiment 1: Valence Features

Table 11. Experiment 1 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.53	0.438	0.53

Experiment 2: Arousal Features

Table 12. Experiment 2 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.656	0.33	0.562

Experiment 3: Dominance Features

Table 13. Experiment 3 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.592	0.438	0.518

Experiment 4: All of the ANEW Features

Table 14. Experiment 4 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.664	0.368	0.616

Experiment 5: Synonyms of the word buy

Table 15. Experiment 5 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.672	0.494	0.572

Experiment 6: Weasel Words

Table 16. Experiment 6 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.3	0.494	0.596

Experiment 7: All manually extracted features

Table 17. Experiment 7 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.594	0.414	0.494

3.3.1.2.2. Bag of Words Approach

After seeing how poorly manually extracted features performed, we decided to also experiment using the traditional bag of word approach. In this traditional NLP approach documents (web pages) are represented as a bag (multiset/list) of its individual words, without caring for word order or grammar, but noting multiplicity. So using this approach the vector representation of the web pages would be vectors storing the counts of every word token in that web page.

Experiment 8: BoW

Table 18. Experiment 8 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.798	0.792	0.812

Experiment 9: Preprocessed & Tokenized BoW

Table 19. Experiment 9 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.818	0.852	0.844

On all classes

Seeing as how the BoW experiments performed the best out of all the models tested (greatly so when compared to manually extracted features) this is model we chose to test on the entire collection of manually labelled dataset.

Experiment 10: BoW

Table 20. Experiment 10 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.658	0.718	0.744

Experiment 11: Tokenized BoW & Preprocessed

Table 21. Experiment 11 Results

	Gaussian Naive Bayes	Linear SVM	Neural Network
F1-Score	0.66	0.718	0.72

3.3.1.3. Feature Significance

Table 22. Feature Signifcance

	False Pages $\mu(\sigma)$	True Pages $\mu(\sigma)$	p-value
Average Arousal	3.99 (0.12)	4.03 (0.24)	0.18
Average Arousal by All Words	1.26 (0.13)	1.3 (0.17)	0.1
Average Dominance	5.31 (0.2)	5.27 (0.3)	0.32
Average Dominance by All Words	1.67 (0.18)	1.71 (0.22)	0.24
Average Valence	5.4 (2.1)	5.41 (0.33)	0.95
Average Valence by Total of all words	1.7 (0.18)	1.76 (0.22)	0.12
Number of Synonyms to Buy	0.05 (0.04)	0.04 (0.05)	0.29
Total Arousal	23.69 (5.18)	25.86 (7.89)	0.06
Total Dominance	31.38 (6.58)	33.94 (10.22)	0.08
Total Valence	31.85 (6.72)	34.71 (10.51)	0.06
Weasel Words	0.01 (0.02)	0.01 (0.04)	0.83

After calculating different features for each document, their values were compared between the true and false categories to see if there is a statistically significant difference. The mean and standard deviation was also calculated for each category.

No significant features were identifying after our analysis. We had hoped that these manually extracted features would have significant difference between these two classes (as previous literature [Sondhi et al. 2012] had suggested). It was unexpected than none of the features that were considered were significantly different between the two classes. However this does explain how poorly these features performed when compared to the traditional BoW approach which did not (explicitly rely on any of these features).

3.3.2. Clustering Approach

Our secondary experimental approach to classification was termed "Clustering Approach". This approach was developed in tandem with our previous approaches and is novel in its approach to classify documents (and their sentences) as true, false, or irrelevant. Unlike previous approaches the clustering approach performs 3 way classification instead of applying binary classifiers in sequence.

3.3.2.1. Clustering Approach Architecture

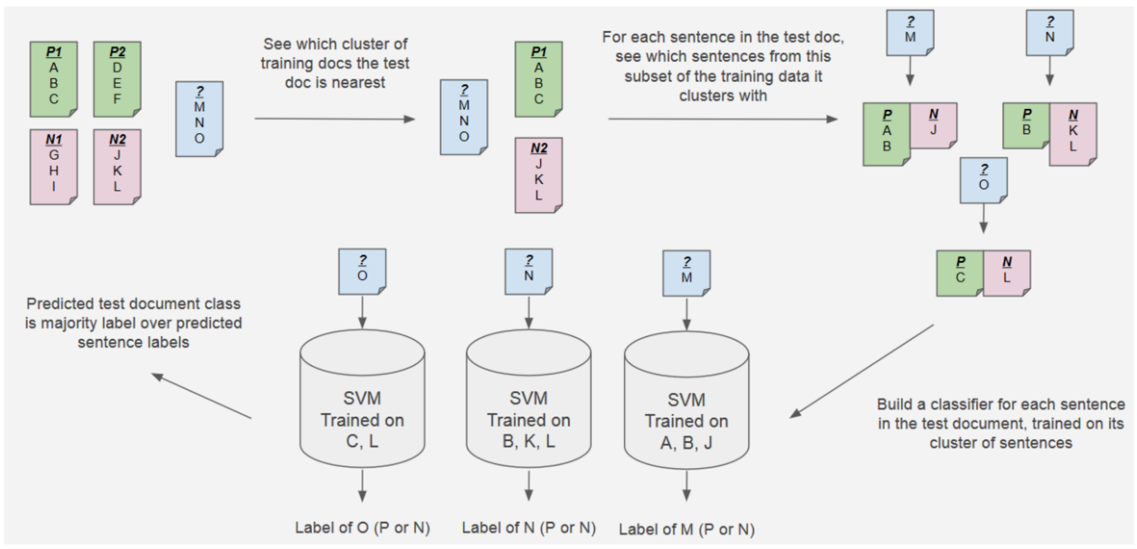

P1 A B C
P2 D E F
N1 G H I
N2 J K L
? M N O
See which cluster of training docs the test doc is nearest
? M N O
P1 A B C
N2 J K L
For each sentence in the test doc, see which sentences from this subset of the training data it clusters with
? M
P A B
N J
? N
P B
N K L
? O
P C
N L
Build a classifier for each sentence in the test document, trained on its cluster of sentences
? O
SVM Trained on C, L
Label of O (P or N)
? N
SVM Trained on B, K, L
Label of N (P or N)
? M
SVM Trained on A, B, J
Label of M (P or N)
Predicted test document class is majority label over predicted sentence labels

Figure 5. Clustering Approach Model Architecture

The clustering approach (Figure 5) relies on the presumption that we have many documents regarding many diseases are already labelled and stored in our database. While this is a limitation at the moment, we can expand the algorithm to collect many documents for every query which is inputted, and also automate the classification of strictly positive and negative examples (as discussed below in future work).

The approach works as follows:

Given a test document T and labelled training data $\{(X_i, Y_i)\}_{i=1}^{n}$, where $\forall i, X_i$ is the TF-IDF vector representation of document i, and $\forall i, Y_i$ is the label of X_i. The following procedure follows:

1. In order to ensure that we can use any Machine Learning techniques on the set of X_i, all vectors that have dimensionality less than the dimensionality of the maximum dimensional vector, have been padded with 0 values in order for all data to be in the same space.
2. Use affinity propagation to estimate the number of clusters (suppose this is k) of $\{X_i\}_{i=1}^{n}$, then use the KMeans algorithm with that number of clusters to group the

training documents. From this we get $\{(C_j, \{(X_{i,j}, Y_{i,j})\})\}_{j=1}^{k}$, where C_j is the cluster center of the j^{th} cluster, and $\{(X_{i,j}, Y_{i,j})\} \subseteq \{(X_i, Y_i)\}_{i=1}^{n}$ is the subset of the training data that belongs to that cluster (labels included).

3. Determine the cluster center, which is nearest to T, where distance is measured by the Euclidean metric.

$$C' = \text{argmin}_{C_j}\left\{\sqrt{\sum_{i=1}^{d}(T_i - C_{j_i})^2}\right\}_{j=1}^{k}$$

   Where d is the dimensionality of T. Let $R = \{(X_i, Y_i)\} \subseteq \{(X_i, Y_i)\}_{i=1}^{n}$ be the subset of the training data belonging to cluster C', this the subset of the training data which we assert contains the 'relevant' documents, with topics most similar to T.
4. Let $S_T = \{s_i\}_{i=1}^{N_T}$ be the set of TF-ISF (this transformation is analogous to TF-IDF except that instead of looking at inverse document frequency, we consider inverse sentence frequency) vector representations of the sentences in T (there are N_T sentences). Again, these have been padded in similar fashion as in step 1 of the algorithm (such that they may all have the same dimensionality).
5. Let $S_R = \{(s_j, Y_j)\}$ be the set of TF-ISF vector representations of the sentences in all of the documents in R. Note that sentences are given their labels as determined by the document they come from (for all sentences coming from a document with label P, the sentences are labelled as P. This is extended for all classes).
6. Ensure that all sentences in S_R and S_T have the same dimensionality, use the same padding process we have used thus far to guarantee this.
7. Use affinity propagation to estimate the number of clusters (suppose this is k') of $\{s_j\} \subseteq S_R$, then use the KMeans algorithm with that number of clusters to group these sentences. From this we get $\{(c_i, \{(s_{i,j}, Y_{i,j})\})\}_{i=1}^{k'}$, where c_i is the cluster center of the i^{th} cluster, and $\{(s_{i,j}, Y_{i,j})\} \subseteq S_R$ is the subset of S_R that belongs to that cluster (labels included). Note that this step is analogous to step (2) of the algorithm, except on a sentence level with a limited scope of the training data (only relevant document sentences being considered).
8. Repeat the following process $\forall s_i \in S_T$:
   (a) Determine the cluster c_k to which s_i belongs (determined using the equation in step (3))
   (b) Let $\{(s_{i,j}, Y_{i,j})\}$ be the set of document-label pairs that belong to c_k. Train a support vector machine with a polynomial kernel of degree 3 on this set of document-label pairs.
   (c) Predict the label of s_i using the trained SVM.
9. Predict the label of T to be the mode over the list of predictions made for each $s_i \in S_T$.

3.3.2.2. Clustering Approach Results

The biggest bottleneck for the analysis of this approach is its extremely slow running time. As our approach builds a classifier for every sentence in our test document certain websites can take up to 30 minutes to classify (depending on the number of sentences they have). Because of this time limitation we were unable to perform an in-depth analysis of the algorithm, and as such, any results discussed here are only tentative pending further experimentation.

We have been able to tentatively verify that the document clustering works as expected, with the closest documents to any test document being those from within the same medical topic. Eg.,

Test Doc Title:

> *Fight Cancer w/ Alkaline Water*

Titles of Docs clustered with it:

> *Is The Alkaline Diet an Effective Cancer Treatment?*
> *Ionised, Alkaline Water*
> *Benefits of Alkalinity in Water*

In almost all cases, the documents clustered with a random test document are of the same topic, however its important to note that there are instances wherein there are also documents of other topics falling into the same cluster (this is due to the way in which affinity propogation determines the number of clusters).

Similarly, we have noticed that sentences tend to cluster both by topic & by form of sentence. E.g.,

Test Sentence :

> *Results larget, expensive animal study on link cellphones cancer.*

Training sentences clustered with it:

> *The NTP assess potential health hazards exposure cell phone radiation.*
> *An important new study linked cell phone radiation cancers brain heart.*
> *It, however, give researchers evidence lead research impact cell phone radiation people.*
> *Maybe study really major turning point understanding risk cell phone usage.*

In most cases, sentences of similar topic & form are clustered together, however, the same failing of affinity that we observed in the document clustering appears here. Note that these sentences are somewhat illegible to due to the removal of stop words in the pre-processing stage.

In our assessment of this algorithm, we only gave it a subset of the training data of size 60, and tested it by iteratively drawing random test cases (one at a time), training on the rest of the data, then predicting the test label (we ensured that no test case was drawn more than once). Due to how slow the approach of this algorithm was (testing on 5 random examples would take about 5 hours depending on the lengths of the randomly selected test examples), the results we have are preliminary, but they are promising. The algorithm seems to perform very well when it comes to identifying examples of positive class (achieving approximately 88% accuracy - averaged over 3 test runs each consisting of 5 randomly drawn test examples), however, does a worse job of distinguishing between examples of false & irrelevant class (approximately 40% accuracy in distinguishing between irrelevant & false - we suspect that this is due to the imbalance with the documents and the topics which we used in our training data - Alkaline water & Cancer, Vaccines & Autism, Cellphones & Cancer).

4. Discussion & Future Work

This paper demonstrates how linguistic features of websites could be used to determine the validity of a medical site. This work combines both novel techniques, and traditional natural language processing methods to expand work previously presented in other papers regarding the detection of fraudulent websites. Unlike previous approaches, we aimed to do so using only textually extractable features, and attempted to determine validity of the claims in a given medically related site.

4.1. Regarding the Add-on and Data

Currently, users can install our Chrome add-on and all Google search results will become augmented with our validity check. As mentioned previously, this extension is only a client that contacts a server for the actual result. This introduces the potential for a slow response time, since all requests must be sent over the Internet. To add to this, every web page requested must then be downloaded server-side, parsed, and then categorized. The range of wait times for classifications is currently 1-7 seconds, which is not ideal in a real world environment. Caching and parallelization of requests can be implemented in the future in order to address these concerns.

4.2. Regarding the Sequential Approach

There is also no guarantee that the add-on works as expected with health myths not explicitly trained on, as this was outside the scope of our project. We hope to improve our methods to perform automated training and testing. Using the knowledge learned from our data collection methods, we can combine HONCode results and filter irrelevant results using the irrelevancy module to obtain, on average, a 75%/25% split between true and not relevant websites. This can be automated so that we can build a large dataset of true examples, hopefully expanding our scope for different medical topics. The set of false websites can also be expanded, although not as accurately as truthful ones, by looking more blogging sites other than just Wordpress, and then filtering for irrelevant websites using the irrelevancy module.

Currently our T/F module of the first approach only works for the English language. This is because the first approach we attempted (manually extracted features) relied on the facts: i) ANEW or some alternative existed which is the case for some languages [Redondo et al. 2007], ii) we looked for synonyms of the word buy in the language of the website instead of English, and most importantly iii) that the classification boundaries and differences between classes was the same in each language. It is plausible that different classes (true vs false) would have different trends in different languages. Our second experiment to the T/F module would not work at all, seeing as we used a bag of words to train the model, and would need to retrain on a different bag for every single language. All of these would need to be studied and adjusted for if the add-on was to be used for other languages.

4.3. Regarding the Clustering Approach

There is much work to be done regarding our clustering approach. The most important work relates to making the algorithm perform much faster so that we may truly test it on all of the data that we have (something which we have been unable to do, due to the run-time constraints). Currently, the reason behind the approach being slow is that for each test document, the process outlined in section 3.3.2.1 is done, this involves a lot of redundant expensive computations. We are presently working on the following approach that would make building the model take a long period of time, but once the model is built, predictions will be very fast:

Given an already partitioned subset of all our data which will be our training data (suppose an 80%/20% split) $\{(X_i, Y_i)\}_{i=1}^{n}$, where this set obeys the same constraints mentioned in 3.3.2.1 (TF-IDF form, matching dimensions)

1. Use affinity propagation to estimate the number of clusters the set of $\{X_i\}_{i=1}^{n}$ will fall into (suppose k clusters), proceed to use the KMeans algorithm to generate $\{(C_j, \{(X_{i,j}, Y_{i,j})\})\}_{j=1}^{k}$, where C_j is the cluster center of the j^{th} cluster, and $\{(X_{i,j}, Y_{i,j})\} \subseteq \{(X_i, Y_i)\}_{i=1}^{n}$ is the set of training example-and-label pairs that fall into that cluster.
2. Repeat the following process for each cluster C_j
   (a) Represent the sentences in $\{X_{i,j}\}$ (the set of documents that fall into C_j in TF-ISF form, and ensure their dimensionality's match. Call this set of sentence-label pairs $S_R = \{(s_i, Y_i)\}$, where a sentence s_i has label Y_i if the document from which it comes has label Y_i.
   (b) Use affinity propagation to estimate the number of clusters that these $\{s_i\}$ fall into (suppose this is k'), and use the KMeans algorithm to cluster them, giving us $\{(c_j, \{(s_{i,j}, Y_i)\})\}_{j=1}^{k'}$ where c_j is the cluster center of the j^{th} cluster, and $\{(s_{i,j}, Y_i)\}$ is the set of sentence-label pairs that fall into that cluster.
   (c) Repeat the following process for each cluster c_j
      i. Train a support vector machine with a polynomial kernel of degree 3 on the set of sentence-label pairs that fall into this cluster ($\{(s_{i,j}, Y_i)\}$)

Note: Step 2, and 2.(c) can be parallelized.

Once we have done the above, given a test example, to predict its label:

1. Let T be the TF-IDF vector representation of the test document. If the dimensionality of the TF-IDF vectors used in the training data is larger than the dimensionality of T, then pad T with 0's, if the dimensionality of T is larger, then use PCA with the number of components being the dimensionality of the TF-IDF vectors used in the training data.
2. Since our documents are pre-clustered, find the cluster which is nearest (by Euclidean distance) to T (note this is an $\mathcal{O}(dk)$ time operation - where k is the number of document clusters & d is the dimensionality of the document vectors). Let this cluster and its documents be R.

3. Represent the sentences in T as TF-ISF vectors, ensure that the dimensions match, this will yield $t = \{s_i\}$ where s_i is the i^{th} sentence in T. We have ensured that within t all sentences have the same dimensionality, but they must also have the same dimensionality as the sentences that have been pre-clustered (the sentences from the documents in R), so compare the dimensionality of any $s_i \in t$ with the dimensionality of the pre-clustered sentences, if the pre-clustered sentences have larger dimensionality, then pad all the s_i vectors with 0's such that they match, if the s_i vectors have larger dimensionality, then use PCA with the number of components being the dimensionality of the pre-clustered sentences.
4. For each $s_i \in t$ repeat the following process:
    (a) Determine the cluster c_j of pre-clustered sentences that is nearest to s_i (by Euclidean distance - this is an $\mathcal{O}(d'k')$ time operation, where k' is the number of sentence clusters, & d' is the dimensionality of the sentence vectors.)
    (b) The SVM is for this cluster is already trained, use it to predict the label of s_i.
5. Predict the label of T to be the mode over the list of predictions made for each $s_i \in t$.

If we compare the speed of the approach described above, to the cost of simply predicting directly with a pre-trained SVM, then the time complexity is only greater by a factor $\in \mathcal{O}(dk + d'k')$, where d and d' are the dimensionalities of the document & sentence vectors, respectively. k and k' are generally relatively small values (always smaller than 100), d can be large (the number of words in the max word document), but not unreasonably so - usually less than 5000. d' will always be relatively small (the number of words in the max word sentence) - always less than 50. If we aggregate these average case upper-bounds, the effective added runtime is quite minuscule in comparison to a direct prediction by a SVM. Note that the case described is one where PCA is not applied, this is because as our training data grows, the likelihood of observing a test document that has dimensionality larger than any one of our training documents becomes very unlikely. In the worst case, if PCA is applied twice (note that this is extremely likely), then the added time complexity would be $\in \mathcal{O}(dk + d'k' + (d^2n + d^3) + (d'^2s + d'^3))$, where n is the number of training documents, and s is the number of sentences (the rest of the terms are the same as before). This is since the major computations involved in PCA are the covariance matrix computation (which for n examples of dimensionality d is $\in \mathcal{O}(d^2n)$), and the eigen-value decomposition ($\in \mathcal{O}(d^3)$) $\implies$ PCA $\in \mathcal{O}(d^2n + d^3)$. Having to perform PCA even once can add substantial overhead, but due to how unlikely this case is to occur, it is not that large of a concern.

Speeding up the clustering approach would open the door for many analyses at the sentence level. This is something we hope to pursue in the future as we believe it could greatly improve the performance (in terms of accuracy), and usability of our add-on. Particularly, we observe that a specific test document may have all sentences be predicted as 'True' except for one, and this one sentence is an obviously 'False' statement, and is the reason why the true label is 'False', yet our clustering model will predict that it is 'True'; we hope to experiment with approaches rooted in information theory in order to find a subset of the sentence predictions which when removed, maximize the increased entropy of the label, allowing us to conclude that this set is the one with significant sentences that may have a more direct semantic effect in determining the true label (then perhaps taking the mode over this set, as opposed to taking the mode over all predictions). We hope that in the future, improvements upon our sentence classifier will allow us to predict and highlight claims which are false, allowing for a more fine-grained browsing experience.

4.4. Final notes

Despite the ever increasing amount of medical advice on the internet, there is still no sure way of ensuring that the results of search queries contain factual information. We introduce an add-on which attempts to assist internet users seeking medical advice to ensure that any advice found online is trustworthy. Our work suggests that it is possible to differentiate between trustworthy and untrustworthy web-pages using only linguistic features. Future work can also extend into other fields such as news sources (which is something being considered by tech giants such as Google and Facebook).

References

[Abbasi and Chen 2009] Abbasi, A. and Chen, H. (2009). A comparison of fraud cues and classification methods for fake escrow website detection. *Information Technology and Management*, 10(2-3):83–101.

[Abbasi et al. 2012] Abbasi, A., Zahedi, F., and Kaza, S. (2012). Detecting fake medical web sites using recursive trust labeling. *ACM Transactions on Information Systems (TOIS)*, 30(4):22.

[Baujard et al. 2010] Baujard, V., Boyer, C., and Geissbühler, A. (2010). Evolution of health web certification, through the honcode experience. *Swiss Medical Informatics*, 26(69):53–55.

[Boyer and Dolamic 2014] Boyer, C. and Dolamic, L. (2014). Feasibility of automated detection of honcode conformity for health related websites. *IJACSA*, 5(3):69–74.

[Bradley and Lang 1999] Bradley, M. M. and Lang, P. J. (1999). Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Technical report C-1, the center for research in psychophysiology, University of Florida.

[Castillo et al. 2011] Castillo, C., Mendoza, M., and Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

[Chung et al. 2012] Chung, M., Oden, R. P., Joyner, B. L., Sims, A., and Moon, R. Y. (2012). Safe infant sleep recommendations on the internet: Let's google it. *The Journal of pediatrics*, 161(6):1080–1084.

[Ennals et al. 2010] Ennals, R., Trushkowsky, B., and Agosta, J. M. (2010). Highlighting disputed claims on the web. In *Proceedings of the 19th international conference on World wide web*, pages 341–350. ACM.

[Fox and Duggan 2013] Fox, S. and Duggan, M. (2013). Health online 2013. *Pew Research Center*. Accessed: 2016-12-15.

[Gyongyi et al. 2006] Gyongyi, Z., Berkhin, P., Garcia-Molina, H., and Pedersen, J. (2006). Link spam detection based on mass estimation. In *Proceedings of the 32nd international conference on Very large data bases*, pages 439–450. VLDB Endowment.

[Gyöngyi et al. 2004] Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. (2004). Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment.

[Jijkoun and De Rijke 2006] Jijkoun, V. and De Rijke, M. (2006). Recognizing textual entailment: Is word similarity enough? In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 449–460. Springer.

[Karpathy and Fei-Fei 2015] Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

[Landis and Koch 1977] Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

[Narayan and Gardent 2015] Narayan, S. and Gardent, C. (2015). Unsupervised sentence simplification using deep semantics. *arXiv preprint arXiv:1507.08452*.

[Page et al. 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: bringing order to the web.

[Pang and Lee 2004] Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

[Porter 1980] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

[Ramos 2003] Ramos, J. (2003). Using tf-idf to determine word relevance in document queries. Technical report, Technical report, Department of Computer Science, Rutgers University.

[Redondo et al. 2007] Redondo, J., Fraga, I., Padrón, I., and Comesaña, M. (2007). The spanish adaptation of anew (affective norms for english words). *Behavior research methods*, 39(3):600–605.

[Shen et al. 2006] Shen, G., Gao, B., Liu, T.-Y., Feng, G., Song, S., and Li, H. (2006). Detecting link spam using temporal information. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 1049–1053. IEEE.

[Shen and Zhang 2016] Shen, L. and Zhang, J. (2016). Empirical evaluation of rnn architectures on sentence classification task. *arXiv preprint arXiv:1609.09171*.

[Siddharthan 2010] Siddharthan, A. (2010). Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 125–133. Association for Computational Linguistics.

[Sondhi et al. 2012] Sondhi, P., Vydiswaran, V. V., and Zhai, C. (2012). Reliability prediction of webpages in the medical domain. In *European Conference on Information Retrieval*, pages 219–231. Springer.

[Tai et al. 2015] Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

[Vanderwende et al. 2007] Vanderwende, L., Suzuki, H., Brockett, C., and Nenkova, A. (2007). Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

[Wikipedia 2004] Wikipedia (2004). Weasel words — Wikipedia, the free encyclopedia. [Online; accessed 15-December-2016].