Project Report

SeaStar
Ashwini Karappa (axk142031)
Sagar Deshpande (ssd140830)

1. Problem Description

Most of the current email servers have the capability to block the spam mails by making use of some filters. However, some spam emails still sneak through the spam filters. This is mainly because the spammers manipulate the spam filters by appending certain words to the mail which rarely occur in spam mails, or substituting certain characters of the spam words, or by adding the synonyms of the words which occur in the spams, to fool the spam filters.

2. Proposed solution

Our main idea is to design a baseline system that will train the machine and classify the emails as ham or spam based on naïve bayes training. The emails identified as spam would be kept aside as correct spams, whereas, the ham emails would be sent to the NLP engine that we will design to classify them further. We propose to check some lexical, syntactic, and semantic features to help the bayes engine classify them correctly.

3. Full implementation details

For performing spam detection, we downloaded six datasets on which we trained and tested our baseline system as well as the NLP engine we created. Reference for these datasets is http://www.aueb.gr/users/ion/data/enron-spam/ . Each of these datasets have spam and ham emails. We use one of these datasets for training and any of the others for testing.

3.1. Baseline system

We designed a baseline system using Naïve Bayesian classifier. Naïve Bayes classification is based on bag of words concept, which does not take into consideration the sequence of words or their semantics. It totally depends on frequency of all words computed during training phase for each class in which document is to be classified. When we applied this naïve technique without digging into any more features we observed good accuracies on our datasets, like in and around of 90%. Although the baseline system using machine learning techniques can classify the emails into ham and spam emails with a good accuracy, it is not fool-proof. The reason behind this is that these techniques do not take the content into considerations. We tried to add few NLP features on top of our baseline system to check how its accuracy gets affected and analyse the reasons behind it.

3.2. Improvement strategy

We attempted to improve the naïve technique by applying lexical, syntactic and semantic features by looking at the content of the emails.

3.2.1. Lexical Features

At the very beginning, we tokenize the email into words and initialize their ham and spam word count based on their placement, whether in ham or spam. In the next step, we train our system by using some lexical features. For that purpose, we maintain the following 3 lists:

1. “stop words”
2. “swear words”

3. “common spam phrases”

3.2.1.1. Stop words removal

Stop words are words which do not add into much of a meaning to the topic and yet appear most frequently in the documents like articles or prepositions. We maintain a stop words list. During training phase, we skip the word if it is a stop word, and do not consider it in the Bayes probability calculation.

3.2.1.2. Swear words handling

Mostly, the spam emails may contain swear words, which make them categorized as spam emails. But this may not always be the case. So, while training, we make a note of the swear words occurring in the ham emails, and increase their ham count. This will increase the weight of that swear word being in ham.

3.2.1.3. Common spam phrases handling

We maintain a list of common spam phrases. We, scan the emails line by line, and check whether it contains the commonly occurring spam phrases. The occurrence of the spam phrase in the mail, increases the probability of the email being a spam. So, we accordingly increase the spam probability in the Bayes calculation.

3.2.2. Syntactic features

Further, we tried to use the syntactic constructs of the emails. We performed the following syntactic analysis:

3.2.2.1. N-Gram POS tagger

We tagged all the emails’ content with their part of speech. Then, we considered a window of 5 tags and scanned the emails by moving the window one tag at a time. These combination of tags and their frequency of occurrence was recorded. And, once all the emails were scanned for this tag window, the n-gram probability (with n = 5), was used to test the emails.

3.2.2.2. Head word calculation

Our idea of head word computation was, that we computed the headwords for every sentence of every email. Using its number of occurrences as head word in spam or ham email and the level on which it was present in the tree, we planned to test the email. We used Stanford core nlp parser for getting the syntactic parse tree and then getting the head word for the sentence. But computing parse tree for even a single sentence is time consuming. Computing parse tree for our email dataset for huge amount of data is thus time consuming. So, though we tested this feature.

3.2.2.3. Bigram probability based computation

Baseline system uses unigrams, but our idea was to analyse bigram based naïve Bayesian. We designed a generic classifier which could take “n” gram count and will apply Bayes classification on n-grams instead of unigrams. We show the accuracy using bigrams.

3.2.3 Semantic Features

We observed that spammers use synonyms or hypernyms of spam words in emails.

3.2.3.1 Synonyms

If the data dictionary from the training phase does not contain the any word from test email, we look if we can find the synonyms of the test word in data dictionary, using WordNet. Then using the found synonyms in data dictionary, we average out their ham and spam probabilities and assign them to the test word.

3.2.3.2 Hypernyms

We use similar concept as that of synonyms. Likewise, we compute the hypernyms of the words of test email. And, calculate the average ham and spam count of those occurring in the training dictionary. Then, we use the

3.3 Examples

3.3.1 Email with Stop words

Look at the email below containing stop words.

"I am **abc**. I work at **xyz** company. I have a great business idea in which, if you **invest the money**, you will get great **returns on investment**.

So hurry up! And Click here"

In the above email you can see there are very few important concepts and rest all of them are stop words. Including these stop words as well for computing ham or spam probability may give misleading results.

3.3.2 Swear words

Generally ham emails do not contain swear words. So, if email contains swear words for example: bad, fucking, slutty etc. then it has the high probability of getting categorized to spam

3.3.3 Spam phrases

Most of the spam emails ask you to perform some action by using promoting phrases like "FREE FREE FREE", "Offer available only till night", "Hurry up", "Best sale prices".

We identify such phrases in the email and give it a higher probability of being spam

3.3.4 POS Tagger

We observe that some of the spam emails just contain random text, some words are not even a valid English words, do not form valid English grammar rules.

So, we tagged the emails with POS tagger and tried to gather all valid and invalid grammar rules in the window of five and maintained its counter for spam and ham and measured the accuracy.

Example:

My/PRP dog/NN also/RB likes/VBZ eating/VBG sausage/NN

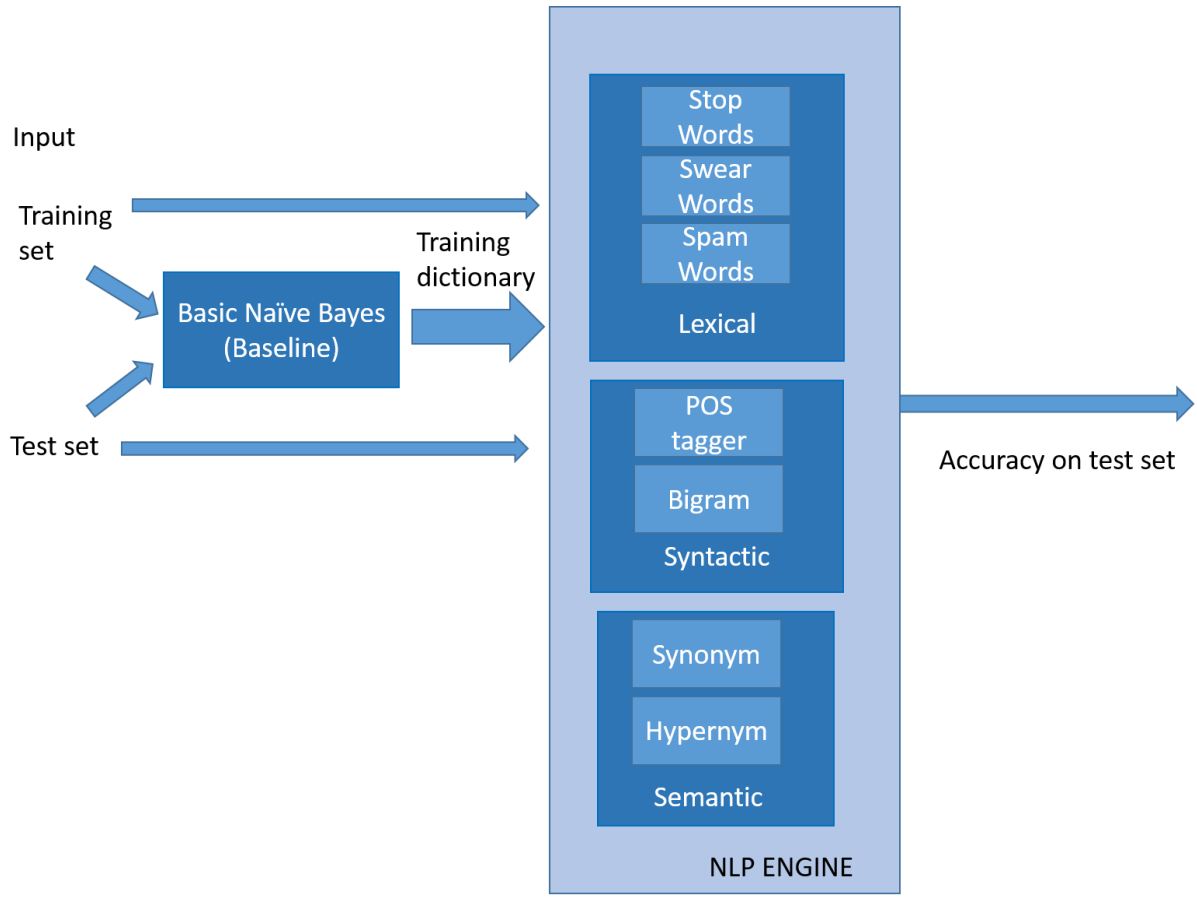
If this line occurs in ham email. We have PRP_NN_RB_VBZ_VBZ, NN_RB_VBZ_VBZ_NN two rules out of it and maintain the count of their occurrences.

3.4 Programming tools (including third party software tools used)

We used java for developing our spam filter. Third party tools we are using are:

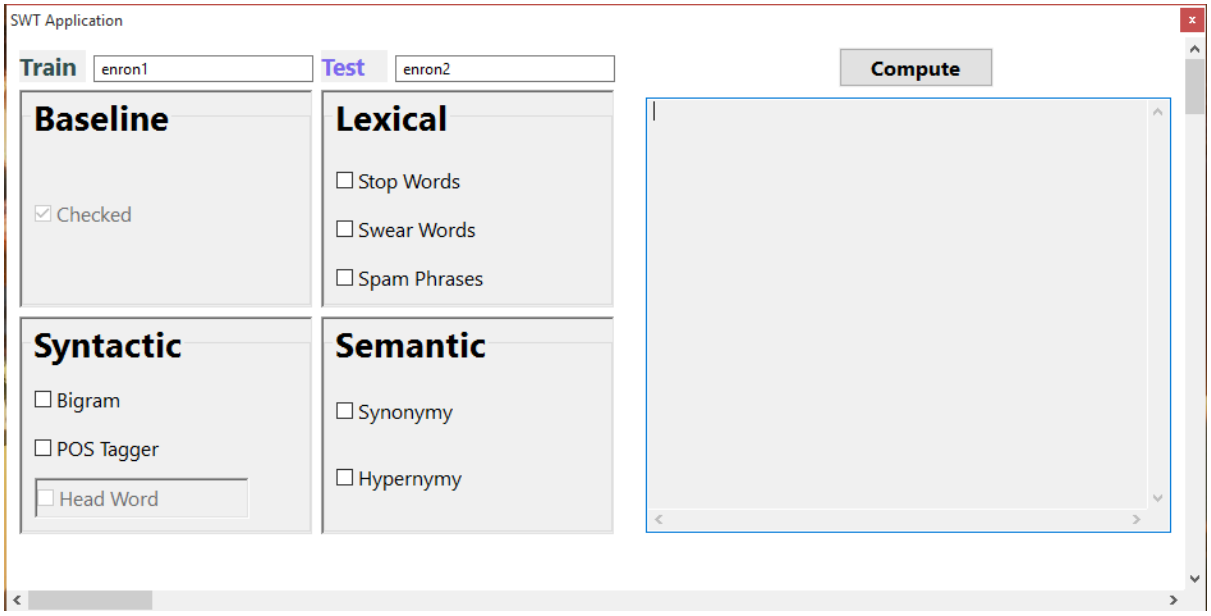
1. Stanford-core-nlp-3.5.2
2. WordNet Dictionary
3. MIT JWI API

3.5 Architectural Diagram

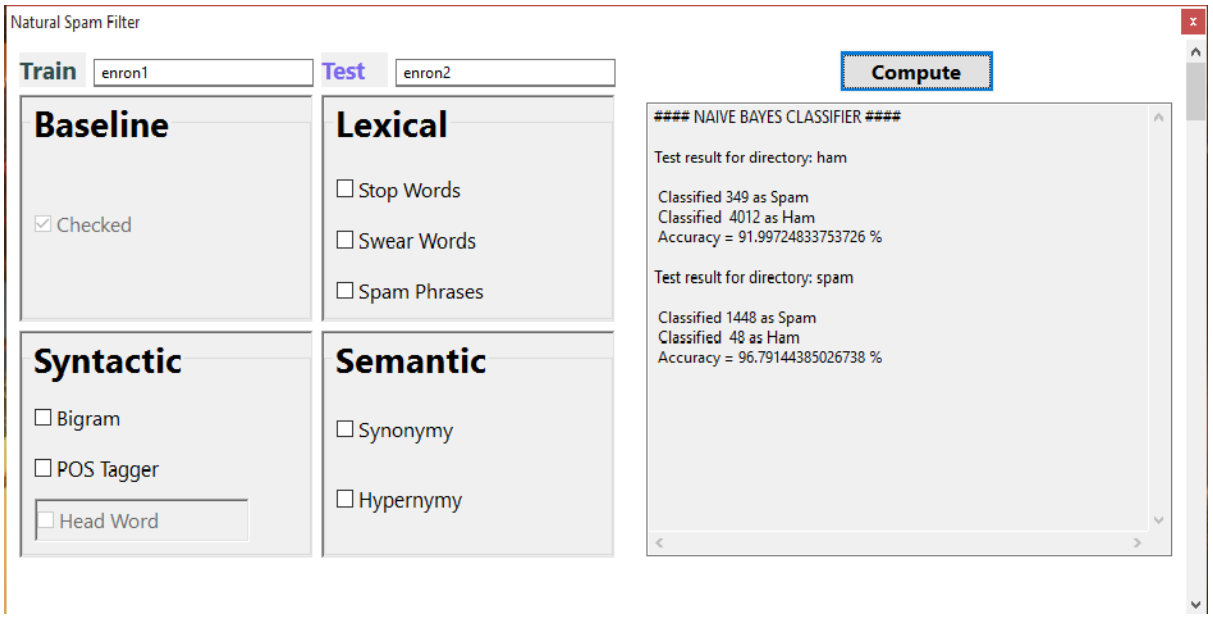


3.6 Results

Following is the GUI for our Spam filter. Window below compute will show the results when features on left side are selected and compute button is hit.

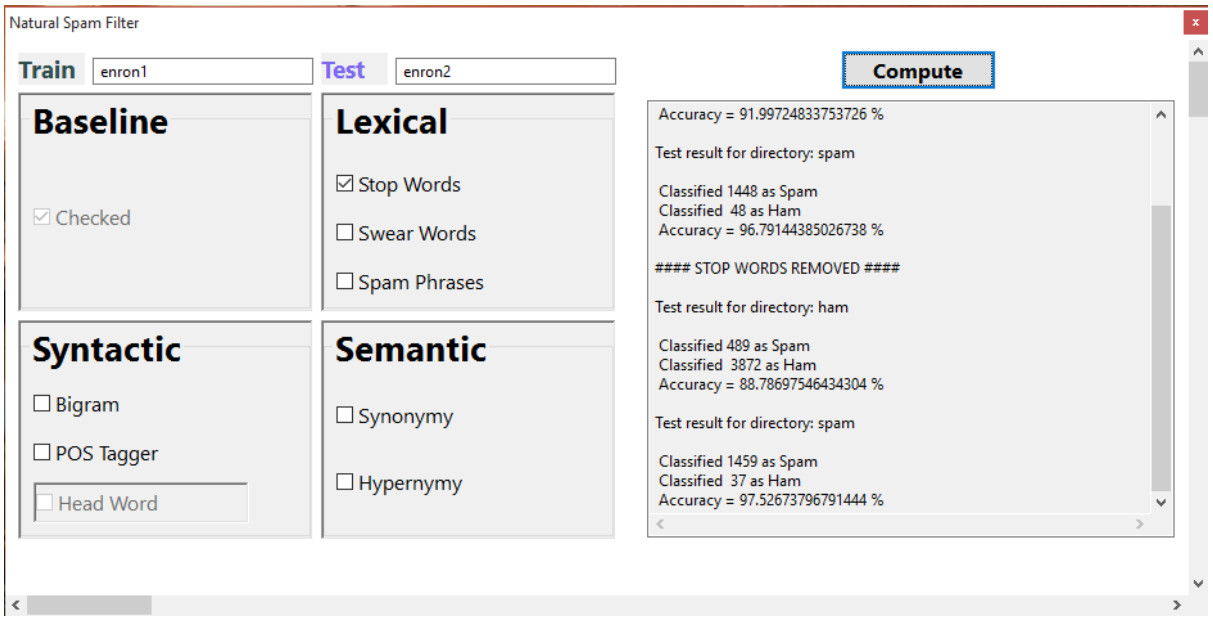


1. Baseline accuracy:



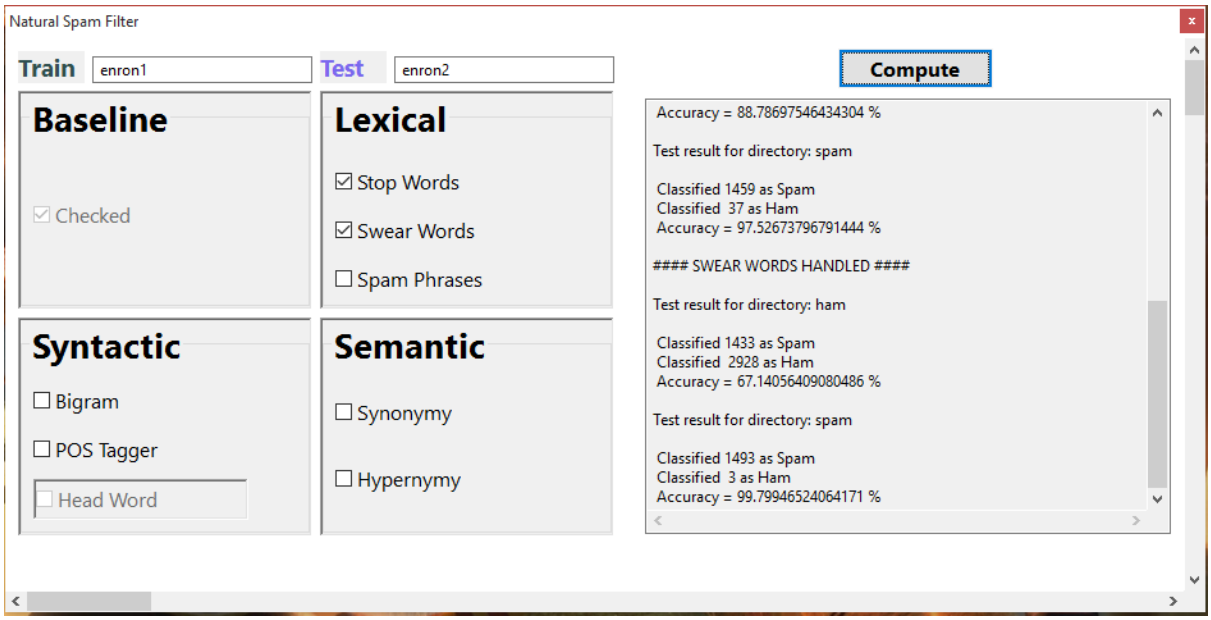
2. Accuracy after removing stop words

We can observe that accuracy over spam emails is increased whereas the accuracy over ham emails is decreased. The reason could be: legitimate emails may generally have a lot of content and thus result in a lot of stop words whereas spammers mostly are focusing on redirecting the users to some fake website. So, in general we can say spam emails have less stop words. If we remove stop words which earlier used to contribute more towards ham probability, is now resulting into decreased ham accuracy and increased spam accuracy.



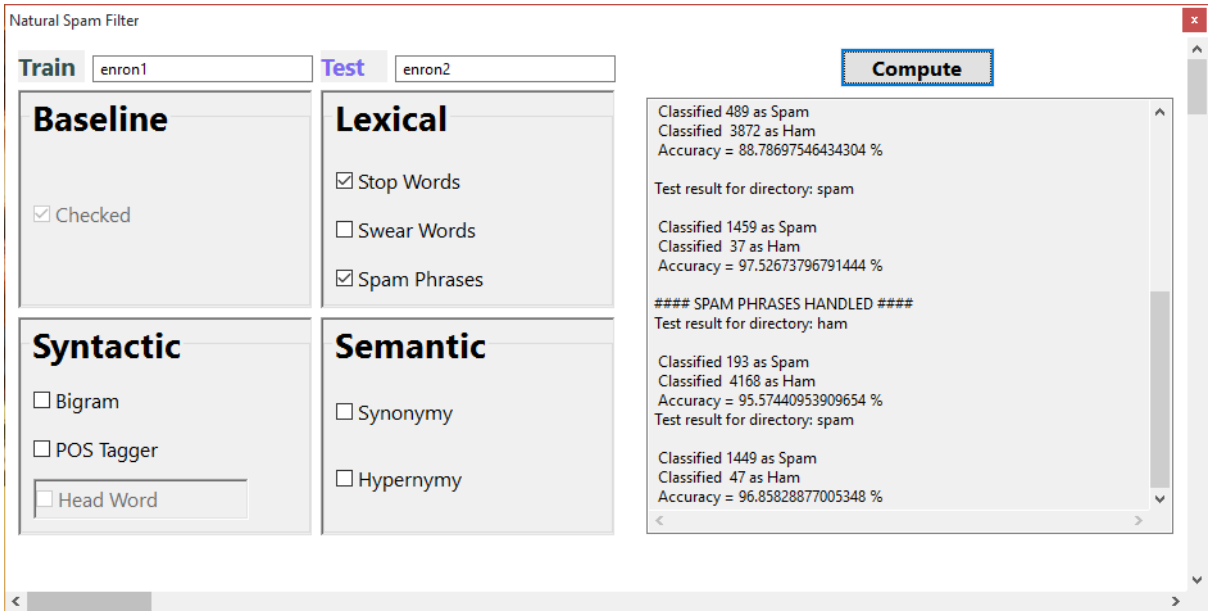
3. After removing swear words

As pointed out earlier, swear words are mostly found in spam mails, thus when test email having swear words is given higher probability to be categorized as spam, we have improved the accuracy over spam. But we see, the accuracy over ham emails drastically decreased, the reason could be that ham emails may also contain few swear words, which during testing haven't been considered at all.



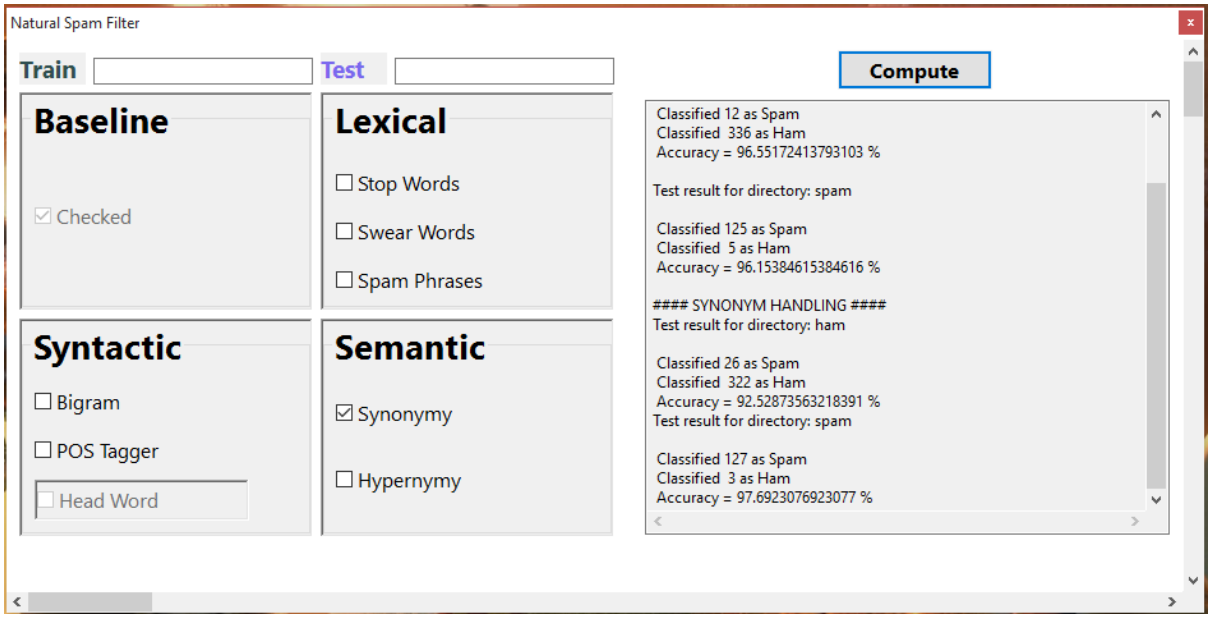
4. Handling spam phrases

We can see acuuracy over the ham emails has increased by a good number, the reason is spam phrases have words which can be catch the attention of the reader. When such words are appearing in spam emails, we double the probability of email to be spam by increasing the weight of the words to be in spam.But such words may appear in ham emails as well, so we increase the weight of the words to be in ham by a some value.



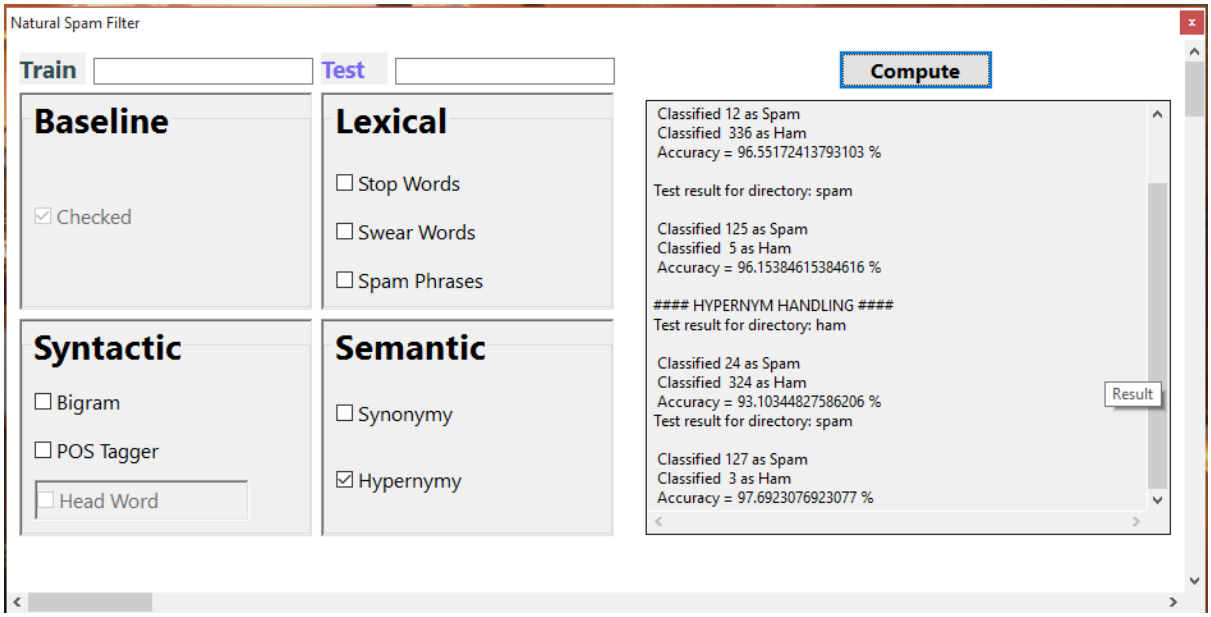
5. After handling synonyms

Accuracy over spam increased whereas ham decreased. Possible reason: Ham words' synonyms may not be in the trained dictionary to take the average or else synonyms are not making much sense for content of the ham emails, but still we are considering the average.

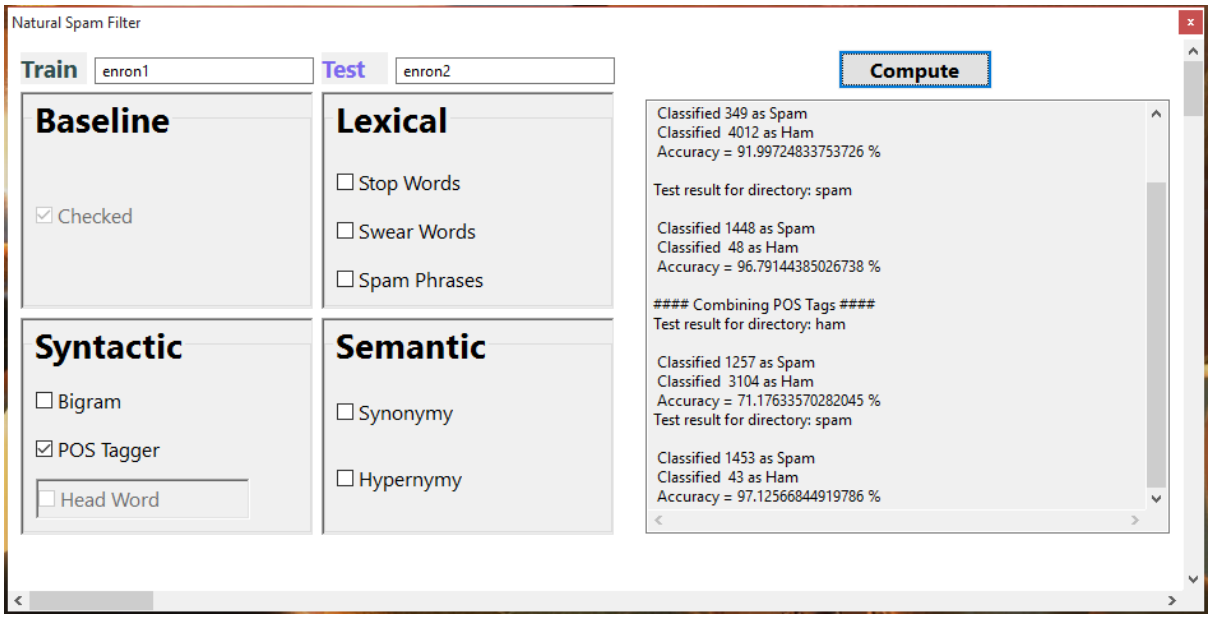


6. <u>After handling Hypernyms</u>

Accuracy over spam increased whereas ham decreased. Possible reason could be similar to that of synonyms mentioned above in point number 5.



7. After POS tagging



3.7 Summary of problems encountered

1. Problem in finding syntactic parser and the way for parsing the semantic tree. We had to do some trials with Stanford NLP parser to understand the way which is not easily explained anywhere over the internet.

2. We had some problems in coming up with semantic and syntactic features that could be useful for improving spam detection. We tried to use the features known and experiment.

3.8 Pending Issues

One pending issue that we have is regarding head word computation. We used Stanford parser for parsing the sentence and giving the head word. It took us a lot of time for parsing a small email containing 10 lines. So, when we applied parsing on our smallest data set of training, we waited like for 10 mins and it was still parsing the emails. So, this is a pending issue and can be solved by some efficient means of head word computation.

3.9 Potential improvements

We came across some document over the internet which tells about the ontology concept for spam detection. Classification based on this "Ontology" would be something interesting and worth exploring for potential improvement. The implementation of this concept requires knowledge of Weka Explorer and Jena. Once these things are correctly understood and well documentation for implementing this ontology concept is available, this feature could prove to be an important improvement.

4. References

[1] Spam Detection using Natural Language Processing 1Rohit Giyanani, 2Mukti Desai.

[2] Auto-Coding and Natural Language Processing by Richard Wolowitz - 3M Health Information System - White Paper 2011

[3] http://nlp.stanford.edu/

[4] http://projects.csail.mit.edu/jwi/

[5] https://www.safaribooksonline.com/library/view/doing-data-science/9781449363871/ch04.html