Classifying web-pages based on their interestingness

Ben Plotke **Renato Stoco**

Abstract

This project describes how to classify web pages based on the users interests. The goal is to use supervised learning algorithms(Naive Bayes or Support Vector Machines) to perform a multi label classification task were the algorithm can classify a web page into 4 different options of interest.

Introduction

The idea that conceded the first model of this research was the question: What if we could select only the most relevant webpages from the internet based on the users interest? Then, after reflecting on the ways that we could use any NLP tasks to solve this problem and if there were datasets to support this idea, we came with the solution for this project. The system will use two categorization task pass, one having the 20NewsGroup training dataset and the other using a training set that we are going to build using the first pass.

The reasoning behind this approach is to generate a good and more "real" training dataset for the interestingness categorizer. For the NLP tasks, we will be using Naïve Bayes and Support Vector Machines since they are proven to be the bests in text classification.

Related Work

Web Page Classification: Features and Algorithms XIAOGUANG QI and BRIAN D. DAVISON

Describes how webpages text classification are far more difficult than normal text classification. In addition, it also describes how and what features to use in order to better classify the web pages. The research used the bag of words and set of words model as the main features and they used different classification algorithms to compare the results.

Text Categorization with Support Vector Machines: Learning with Many Relevant Features Thorsten Joachims

Describes the use of Support Vector Machines in the text classification task. The research details how SVM can achieve the lowest true error and how it handles with a high-dimensionality space making SVM one of the best text classification algorithms. In addition, most of the text categorization problems are linearly separable: which makes the argument of SVM being one of the best algorithms even more plausible.

Data

The system uses two different datasets. One dataset comes from the 20NewsGroup Journal and it will be used to do the first categorization task. The 20NewsGroup dataset has 20 different category labels, and for each label there is a bag of word model (unigram) with only non-stop words. In addition, we have the labeled test data with over 7000 samples in order to calculate the precision, recall and f-score.

Figure 1: All Category Labels

20 Newsgroups
Class
alt.atheism
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.crypt
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc

One random sample from the training dataset:

comp.os.ms windows.misc:

Windows crashing article apr oracle oracle ebosco oracle Eric bosco writes ebosco oracle Eric bosco subject windows crashing […]

The second Dataset will be built based on the webpages that we classified from the first categorization pass. We will extract the unigrams and bi-grams from each webpage and use this data as features for the Interestingness Classifier.

For further discussion, measuring interest is really difficult because is not something that can be said it to be exactly right or not. So, having that in mind, for the second categorizer part we will use our own judgment to tell if the system is working as intended.

System Description

This project will make use of the NLTK toolkit and the sci-kit toolkit. We will use NLTK to filter the web pages that are not in English and to remove stop words from the content.

The system also uses the Beautiful-soup library to better parse the web pages and extract the text data.

Sci-kit was chosen because it has almost all the famous supervised learning algorithms optimized, and once the data is formatted in the right way, we can just run the algorithms and compare the results. Sci-kit will be used to train using Naive Bayes and Support Vectors machines. In addition, sci-kit will be used to calculate the precision, recall and f-score of the system.

Also, we built our own Naive Bayes categorizer to make comparisons with the sci-kit algorithms.

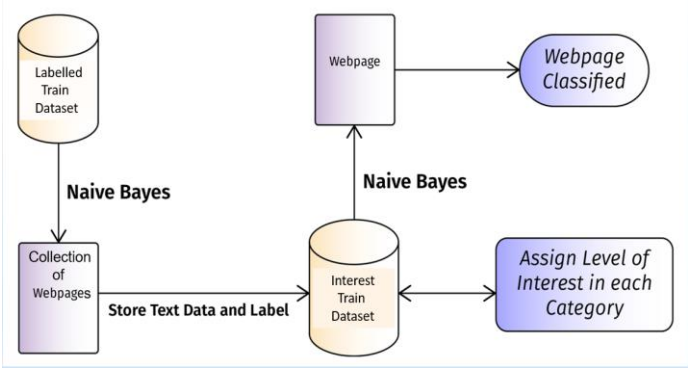
In order to make the data fit into sci-kit format we had to convert all our data into a specific data structure. The data structure is described below:

```
Class dados (object):
    data = []
    target = []
target_names=['alt.atheism','comp.graph
ics','comp.os.ms-windows.misc, […]']
```

The structure is composed by the field “data” which contains all the features. The field target describes to what label or category the feature belongs to. And, since target uses an integer index we will use target_names to map each index to its matching name.

The system follows the architecture of the following Diagram:

Figure 2: System Architecture



We will use Naive Bayes/Support Vector Machines (SVM) on the 20NewsGroup training data. Once the model is ready, we will scrap web pages over the Internet categorizing them as one of the possible category labels and then store the web page text into a file.

After having enough of web pages data the user will input, for each category, their desired level of interest. Then, once again the system will use Naive Bayes/SVM to build a training model based on 4 different types of interest: 1: Not Interested, 2: Maybe Interested, 3: Interested and 4: Very Interested.

Figure 3 Labels Dictionary: each number corresponds to a level of interest

```
def binning(string):
    interest={
        "alt.atheism" : 1,
        "comp.graphics" : 3,
        "comp.os.ms-windows.misc" : 3,
        "comp.sys.ibm.pc.hardware" : 2,
        "comp.sys.mac.hardware" : 4,
        "comp.windows.x" : 4,
        "misc.forsale" : 2,
        "rec.autos" : 3,
        "rec.motorcycles" : 2,
        "rec.sport.baseball" : 2,
        "rec.sport.hockey" : 2,
        "sci.crypt" : 2,
        "sci.electronics"   : 3,
        "sci.med"   : 3,
        "sci.space" : 4,
        "soc.religion.christian" : 1,
        "talk.politics.guns" : 1,
        "talk.politics.mideast" : 1,
        "talk.politics.misc" : 1,
        "talk.religion.misc" : 1
    }
    return str(interest[string])
```

Once the second categorizer model is built we can input random web pages into the system to classify them as one of the interest labels.

The reason behind all this approach is to make easier to the user build its training data that contains what he believes to be interesting instead of manually picking the web pages that might be interesting or not.

In addition, almost all the labeled datasets that have category labels are from formal journals which are not the way most web pages are written.

The Scrapper

The experiment was done many times mostly because one of the main problems with our system was how to select the best web pages to extract the necessary text information.

In this section of this article we are going to describe the scrapper part of our system, but since it does not belongs to almost any NLP related task we are no longer going to mention this part on this paper.

The main role of the scrapper function in our system is to collect all the relevant web pages that can be considered a good source of text information to build the dataset that will be used for training the interestingness model.

The scrapper is composed by two functions: the "scrapper" main function and the "extended link" function. The scrapper function takes an initial URL (the seed) and it extracts all the possible URL within it. Once it have all the URLs we will use each one of them in the second function, which is a recursive function, that will extract all the URLs contained in that web page and the text data categorizing it at the same time. After the system have collected the necessary data it will store the text data with the label, and will call the extended link function on all the URLs.

This process is very costly and it takes a lot of time to build the training data. To better filter the information and the size of the "crawl tree" we used some heuristics to better prune the worst paths. First, the initial seed will be the webpage http://www.theverge.com/, this option was chosen because it contains information about all the general category topics which we believe that could return better results. The second step in the filtering was to not allow a huge disproportion in the training set by having almost all of the dataset composed by, for example, the comp.microsoft category label. And for the last step, we only allowed a maximum of 100 iterations on webpages that were from the same domain.

Experiment

Once we got our training dataset built, with no disproportions and with over than 1000 samples with each sample containing a large paragraph we decided to run the last part of our system that is the interestingness categorizer. For convenience, we used the same scrapper function but with only one more line added onto it. This line represents the call of the interestingness classifier.

Results and Discussion

Our system got really good results on the 20NewsGroupt raining set. We computed the accuracy of our Naïve Bayes approach, the sci-kit approach and the sci-kit SVM approach.

For the testing purposes below we are using the bag of words Model and the test data from the 20NewsGroup that contains almost 7150 Labeled samples.

Our Naïve Bayes code build from scratch got:

For one hundred random samples: 0.80 Raw Accuracy.

For five hundred random samples: 0.816 Raw Accuracy.

For one thousand random samples: 0.815 Raw Accuracy.

It took over 15 minutes to complete the accuracy test with a thousand samples, but since the deviation from one test to another were pretty much the same we decided that we no longer needed any more testing. So our Naïve Bayes approach got an 80% Raw Accuracy.

Now let's take a look on how much the sci-kit SVM can achieve.

Figure 4: SVM Accuracies

Svm Results with our Train/test Data
Raw Accuracy SVM: 0.930184368525

	precision	recall	f1-score	support
alt.atheism	0.93	0.88	0.90	799
comp.graphics	0.94	0.90	0.92	973
comp.os.ms-windows.misc	0.84	0.93	0.88	966
comp.sys.ibm.pc.hardware	0.89	0.84	0.87	982
comp.sys.mac.hardware	0.94	0.93	0.94	963
comp.windows.x	0.96	0.91	0.94	985
misc.forsale	0.89	0.92	0.91	975
rec.autos	0.93	0.96	0.95	989
rec.motorcycles	0.97	0.98	0.97	996
rec.sport.baseball	0.99	0.98	0.98	994
rec.sport.hockey	0.96	0.99	0.98	999
sci.crypt	0.94	0.99	0.97	991
sci.electronics	0.97	0.89	0.93	984
sci.med	0.98	0.99	0.98	990
sci.space	0.94	0.98	0.96	987
soc.religion.christian	0.81	0.97	0.89	996
talk.politics.guns	0.88	0.98	0.93	909
talk.politics.mideast	0.95	0.99	0.97	940
talk.politics.misc	0.99	0.84	0.91	775
talk.religion.misc	0.98	0.59	0.74	628
avg / total	0.93	0.93	0.93	18821

As we can see from the table above, sci-kit SVM achieved a raw accuracy of ~93% way better than our Naïve Bayes built from scratch. In addition, we also measured the precision, recall and F-score to have a better understanding of the accuracies and even these scores stays in the 93% line.
For a better comparison, now let's see how much the Naïve Bayes approach from the sci-kit toolkit can accomplish.

Figure 5: Naive Bayes accuracies

Raw Accuracy Naive Bayes: 0.935391318208

	precision	recall	f1-score	support
alt.atheism	0.90	0.88	0.89	799
comp.graphics	0.95	0.93	0.94	973
comp.os.ms-windows.misc	0.92	0.94	0.93	966
comp.sys.ibm.pc.hardware	0.86	0.93	0.89	982
comp.sys.mac.hardware	0.97	0.95	0.96	963
comp.windows.x	0.97	0.95	0.96	985
misc.forsale	0.96	0.88	0.92	975
rec.autos	0.97	0.98	0.98	989
rec.motorcycles	0.99	0.99	0.99	996
rec.sport.baseball	0.99	0.99	0.99	994
rec.sport.hockey	0.98	0.99	0.99	999
sci.crypt	0.94	0.99	0.96	991
sci.electronics	0.96	0.94	0.95	984
sci.med	0.99	0.98	0.98	990
sci.space	0.97	0.98	0.97	987
soc.religion.christian	0.75	0.99	0.85	996
talk.politics.guns	0.85	0.99	0.92	909
talk.politics.mideast	0.95	0.99	0.97	940
talk.politics.misc	0.98	0.83	0.90	775
talk.religion.misc	0.99	0.36	0.53	628
avg / total	0.94	0.94	0.93	18821

To our surprise, the Naïve Bayes surpassed the SVM approach. However, not by a huge margin and this may be biased by some random event. Still, it achieved 1% more accuracy on precision/recall measures and slight better on the raw accuracy.
From all methods the sci-kit Naive Bayes was the one with the higher accuracy and because of that we will be using the 10-fold cross validation on it to see what the real accuracy is.

Figure 6: Naive Bayes 10-fold Cross Validation

```
[['category' 'precision' 'recall' 'f-score']
 ['alt.atheism' '0.89' '0.89' '0.89']
 ['comp.graphics' '0.92' '0.9' '0.91']
 ['comp.os.ms-windows.misc' '0.88' '0.95' '0.91']
 ['comp.sys.ibm.pc.hardware' '0.75' '0.92' '0.82']
 ['comp.sys.mac.hardware' '0.91' '0.97' '0.94']
 ['comp.windows.x' '0.96' '0.91' '0.93']
 ['misc.forsale' '0.94' '0.85' '0.89']
 ['rec.autos' '0.93' '0.97' '0.95']
 ['rec.motorcycles' '0.97' '0.96' '0.97']
 ['rec.sport.baseball' '0.99' '0.96' '0.97']
 ['rec.sport.hockey' '0.96' '0.98' '0.97']
 ['sci.crypt' '0.87' '0.99' '0.92']
 ['sci.electronics' '0.95' '0.81' '0.87']
 ['sci.med' '0.98' '0.89' '0.94']
 ['sci.space' '0.96' '0.96' '0.96']
 ['soc.religion.christian' '0.73' '0.99' '0.84']
 ['talk.politics.guns' '0.85' '0.98' '0.92']
 ['talk.politics.mideast' '0.96' '0.97' '0.96']
 ['talk.politics.misc' '0.98' '0.82' '0.88']
 ['talk.religion.misc' '1.0' '0.38' '0.55']
 ['Average' '0.92' '0.9' '0.9']]

Process finished with exit code 0
```

As we can see from the table above, using the 10-fold cross validation on sci-kit Naïve Bayes approach made the accuracy from all the measures decrease approximately 3% which means a lot from a precise point of view. However, this is a realistic accuracy and it is still above 90% so we will be using this method to improve our system.

For that reason, we can finally run the last part of the system, the interestingness classifier.

It's necessary to note that the next results are based on the authors' idea of interest, in other words, the results were judged by how likely a web page that we think is interesting will be categorized. In addition, we also judged if the system was returning only the same labels on all selected web pages.

In the first run with no modified data and using the levels of interest mentioned on illustration 3, we collected the results below:

Figure 7: Raw System Results

http://www.polygon.com/2016/3/7/11173396/star-wars-galaxy-topps-trading-cards-jack-kirby-steve-ditko Maybe Interested
http://www.theverge.com/2016/3/7/11175074/apple-scott-forstall-broadway-eclipsed-lupita-nyongo Maybe Interested
http://www.theverge.com/2016/3/7/11175002/oldest-chameleon-amber-99-million-years-baby-lizard Very Interested
http://www.theverge.com/2016/3/7/11174802/solar-eclipse-march-2016-alaska-airlines-flight-870 Very Interested
http://www.theverge.com/2016/3/7/11174624/justice-league-casting-j-k-simmons-commissioner-gordon Maybe Interested
http://www.theverge.com/2016/3/7/11174350/google-photos-live-photos-ios-support Very Interested
http://www.theverge.com/2016/3/7/11174254/4chan-founder-chris-poole-joins-google Maybe Interested
http://www.theverge.com/2016/3/7/11173828/google-project-fi-open-to-everyone Maybe Interested

As we can see, right now the web pages are only falling into two categories: Maybe Interested or Very Interested. This is probably because Naive Bayes is biased towards some of the categories. Also, it is important to have in mind that we are using, once again http://www.theverge.com , and since this news journal is more likely to have news about tech and medicine this fact can contribute to the system being more biased towards one level of interest.

We thought that maybe removing one or two category labels we could improve the performance of the system. Therefore, after observing which changes in the interest levels affected the system, we concluded that the categories “comp.sys.mac” and “sci.crypt” were the ones that are skewing our results.

Figure 8: Results with comp.sys.mac and Sci.crypt labels removed

http://www.theverge.com/2016/3/7/11176566/apple-fbi-encryption-appeal-new-york Interested
http://www.theverge.com/2016/3/7/11176346/hillary-clinton-apple-fbi-data-encryption Interested
http://www.theverge.com/2016/3/7/11175764/microsoft-sql-server-linux-version-launching Interested
http://www.theverge.com/2016/3/7/11176006/virginia-daily-fantasy-sports-law-draftkings-fanduel Not Interested
http://www.theverge.com/2016/3/7/11174232/mercury-carbon-crust-discovered-nasa-messenger-dark-gray Not Interested
http://www.theverge.com/2016/3/7/11175250/the-new-york-times-pop-up-ad-blockers-test Interested
http://www.polygon.com/2016/3/7/11173396/star-wars-galaxy-topps-trading-cards-jack-kirby-steve-ditko Not Interested
http://www.theverge.com/2016/3/7/11175074/apple-scott-forstall-broadway-eclipsed-lupita-nyongo Interested
http://www.theverge.com/2016/3/7/11175002/oldest-chameleon-amber-99-million-years-baby-lizard Not Interested
http://www.theverge.com/2016/3/7/11174802/solar-eclipse-march-2016-alaska-airlines-flight-870 Very Interested
http://www.theverge.com/2016/3/7/11174624/justice-league-casting-j-k-simmons-commissioner-gordon Interested

Finally we got interesting results, as we can see from this sample now the results are better distributed and they match with the levels of interest that we decided on the beginning of this research.

Future Work

The results that we got are far from being the perfect goals. However, we achieved what we wanted, in other words, after removing the biased labels we classified the web pages around what we thought to be interesting.

In the future, we probably will find clever ways to build the training dataset in such a way that Naive Bayes doesn't become biased towards one category or another. Also, it's necessary to have in mind that even though SVM has a slight lower accuracy it is less biased towards any class labels [2]. Thus, in the future we will be using SVM as the approach to go on our system. Also, trying a different category dataset might be one solution to our bias problem.

Conclusion

This research described one approach to classify the levels of interest of a given webpage. The logic behind this approach was to make easier for the user to change his interest, if he wanted, in the future, and also to create a better training dataset since the ones that are available are not from ordinary web pages.

Moreover, after some pruning on the system we collected good results such as web pages classified around our interest table.

Therefore, we conclude that the system achieved its goal because even though web categorization task is much harder than normal text classification task, we got plausible results from our system and the path is not over yet. In the future we will be refining this project with new methods and datasets to further increase the accuracy of our system.

References

[1] QI, XIAOGUANG, and BRIAN D. DAVISON. *Web Page Classification: Features and Algorithms*. Lehigh University. Web. <https://www.cs.ucf.edu/~dcm/Teaching/COT4810-Fall%202012/Literature/WebPageClassification.pdf>.

[2] Joachims, Thorsten. "Text Categorization with Support Vector Machines: Learning with Many Relevant Features."*Http://web.cs.iastate.edu/~jtian/cs573/Papers/Joachims-ECML-98.pdf*. Universitat Dortmund, web

[3] Ana Cardoso Cachopo's Webpage http://ana.cachopo.org/datasets-for-single-label-text-categorization, web