TEMPLATE
WS 2016 Project 2

Keep anonymous

1. INTRODUCTION

The objective in the project was to analyze reviews on Lego products, and feedback on these reviews, mined from an online forum[1]. The goal being, being able to determine the sentiment of a forum post, entirely based on mined forum posts, which had manually been evaluated.

2. METHODOLOGY

2.1 Data Mining

Initially, a spreadsheet of 4.900 forum posts were mined and shared via Google Drive with the participants of a Masters Course in Web Science at the University of Copenhagen. Each student then took part in helping to classify the posts, giving them a sentiment value of either -1 (Negative), 0 (Neutral) or 1 (Positive).

In my approach I then copied the spreadsheet, to safeguard from any further edit, and build a Python API using gspread[2] to automatically access the data. Once in my local domain, I strove to keep my computing offline, not depending on webservices to produce my data.

2.2 Sentiment Evaluation

2.2.1 SentiStrength

Initially a great amount of effort was spend on the free to use tool SentiStrength[3], as it offers both sentiment analysis on a sentence based level, and includes a suite of tools for further analysis. SentiStrength also takes some care in processing data which may include misspellings, emoticons and other elements which are a natural part of an online forum. Sadly, though, I found SentiStrength lacking, both in speed and accessibility, and furthermore SentiStrength depends on it's sentiment to be twofold, each sentence having both a positive and negative evaluation, which makes great sense, but sadly was intuitively possible with the given data.

2.2.2 uClassify

The second choice of approach were the tool uClassify[1], which comes in both online and offline versions, me choosing the latter. After having set up a socket server, and modifying code to accompany local access, three classes were made, negative, positive and neutral, and a 3-fold cross validation were preformed, in order to both train and teach uClassify to evaluate the forum posts, the results being elaborated in Section 3.

The method used by uClassify is fairly straight forward, as it will evaluate each sentence parsed, taking note of the provided sentiment and the words occurring in the sentence. Once a sufficient amount of training has been done, it can start making educated guesses on unseen sentences, weighing their value on classifying them as either negative, positive or neutral, based on their likeness on previous posts.

2.3 Deep Learning

Tools like SentiStrength and uClassify word on a naive basis, looking at words in a sentence and evaluating their weight. However, this ignores much of the meaning in a sentence, such as grammar, phrases and so on, and while tools like SentiStrength has some manually coded options to account for this, such as a negating word in front of a positive *"Not Great"*, or slang *"h8- "hate"*, it cannot learn new information unless manually implemented.

Deep Learning aims to solve this problem, by looking at the entire sentence, and the structure within, in order to evaluate a sentence and determine its sentiment. Stanford University has produced a toolkit[2] for evaluating movie reviews. The tool is based upon Deep Learning, and allows for training, much like uClassify, however, in order to train it, trees has to be constructed from each sentence, as the structure of each sentence is now being evaluated. For example the positive sentence:

They look very nice

needs to turn into a tree, where each word has a weight, but the connection between words likewise hold weight

(3 (2 They) (3 (2 look) (3 (2 very) (3 nice))))

The numerical values are as follows:

0	1	2	3	4
Very Negative	Negative	Neutral	Positive	Very Positive

[1] www.eurobricks.com
[2] https://github.com/burnash/gspread
[3] http://sentistrength.wlv.ac.uk/

Web Science 2016 DIKU, Denmark

Once every review is on tree form, training can be preformed, and the new model can be tested. When training, it's an option to use Stanford's provided sentiment model to fill weigh each sentence, and only have the input provided tweak these values, or you can choose to construct an entirely independent model, based purely on the provided date. I experimented with both.

3. FINDINGS

First of. The initial uClassify method was able to correctly evaluate a given review roughly 60% of the time. Table 1 shows the number of correct (hit) and the wrong (miss) evaluations for each of the loops, correct as in matching with the Google Spreadsheet. These evaluations are of every sort,

Table 1: uClassify 3-fold cross validation results

Fold	hit	miss	
1	764	536	0.589
2	780	520	0.600
3	783	517	0.602

positive, negative and neutral, and overall it's not a bad place to start. For a naive model this is what can be expected.

Secondly the Deep-Learning method was tested:
Without using the Stanford Sentiment data already in the toolkit to adjust the trees being constructed, I ended up only having a very few negative reviews when testing, as evident from the first fold's matrix produced when preforming 3-Fold testing, see Table 2. In the first fold, while there's no reviews

Table 2: Deep Learning - Without Movie Review Sentiment

Guess/Gold	0	1	2	3	4	Marg. (Guess)
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	52	176	241	0	469
3	0	818	2005	4893	0	7716
4	0	0	0	0	0	0
Marg. (Gold)	0	870	2181	5134	0	

expected to be 0 (Very Negative) or 4 (Very Positive), the training data did expect 870 Negative reviews, but ended up having 0. On the other hand there's far more Positive (7716) reviews compared to expected 5134. Overall I could say this method failed, and the only reason I get a fairly high hit rate [0.619304, 0.569037, 0.611804] respectively for each fold is due to the above average number of positive reviews in the data.

Next, allowing the trees being generated to be affected by the sentiment already present in the Stanford Toolkit I get a much more varied set of results, as evident from another matrix from the 3-Fold testing, see Table 3.

This time when guessing (testing the training), both Negative, Neutral, Positive and Very Positive reviews are being presented, showing the effect of the movie review sentiment. For all three folds, their values and matrix see Appendix A.

4. CONCLUSION

Table 3: Deep Leaarning - With Movie Review Sentiment

Guess/Gold	0	1	2	3	4	Marg. (Guess)
0	0	0	0	0	0	0
1	114	2878	696	541	17	4246
2	0	225	1372	263	2	1862
3	7	424	254	1290	87	2062
4	0	1	0	4	10	15
Marg. (Gold)	121	3528	2322	2098	116	

As a standalone, the 4900 reviews, with invalid reviews excluded (bad/missing sentiment format), we were not given a large enough dataset to train without including pretrained data, such as the Movie Sentiment from the Stanford Toolkit. However, once included in the building of the data, the reviews give very credible feedback. A hit rate of 67%, without it being due to chance (as in the non-sentiment affected data) is a fair prediction rate, even if the Stanford boasts a 85% success-rate when analyzing movie reviews, our data is simply too small. The data being mined directly from a forum, where people often ends up misspelling, using emoticons and so on, only makes our limited dataset even weaker, as a persons unique way of formulating a review will cause outliers, which are hard to predict and learn from in isolation.

Overall, having a 7% better success-rate when using Deep-Learning is as expected, however I expect expanding the data set will improve success-rate even further.

5. REFERENCES

[1] uClassify classifier tool. https://www.uclassify.com/. Accessed: 2016-03-29.

[2] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

APPENDIX

Fold 1

```
EVALUATION SUMMARY
Tested 299379 labels
  261763 correct
  37616 incorrect
  0.874353 accuracy
Tested 8185 roots
  5550 correct
  2635 incorrect
  0.678070 accuracy
```

	0	1	2	3	4	
0	0	0	0	0	0	0
1	114	2878	696	541	17	4246
2	0	225	1372	263	2	1862
3	7	424	254	1290	87	2062
4	0	1	0	4	10	15
	121	3528	2322	2098	116	

Fold 2

```
EVALUATION SUMMARY
Tested 298018 labels
  262299 correct
  35719 incorrect
  0.880145 accuracy
Tested 8184 roots
  5549 correct
  2635 incorrect
  0.678030 accuracy
```

	0	1	2	3	4	
0	0	0	0	0	0	0
1	129	2761	586	716	34	4226
2	0	359	1482	279	4	2124
3	0	260	116	1294	127	1797
4	1	5	3	16	12	37
	130	3385	2187	2305	177	

Fold 3

```
EVALUATION SUMMARY
Tested 274860 labels
  241540 correct
  33320 incorrect
  0.878775 accuracy
Tested 8184 roots
  5472 correct
  2712 incorrect
  0.668622 accuracy
```

	0	1	2	3	4	
0	0	0	0	0	0	0
1	103	2116	319	382	14	2934
2	1	620	1802	291	4	2718
3	11	544	278	1552	145	2530
4	0	0	0	0	2	2
	115	3280	2399	2225	165	