Classifying American English Regional Usage Using Reddit

Kristopher Rutherford
Department of Information & Library Science
1320 E. 10th Street, LI 011
Bloomington, IN 47405
klruther@iupui.edu

Abstract

Using Reddit posts from the top cities around the United States, this experiment hopes to classify Reddit posts into regional categories by their word usage. Using Naïve Bayes and Support Vector Machine classifiers, this experiment shows that traditional approaches may not be sufficient for identifying regional variances in word usages.

1 Introduction

Regional variances of the spoken word are easily identifiable. Accents, body language, and immediately heard slang provide immediate feedback about a change in regional setting. The written word though loses all of these immediate effects making it more difficult to notice regional variances in expression.

The social media platform Reddit provides sub forums or sub Reddits for almost any topic imaginable. In theory, this should be perfect for identifying regional variances in the written word. This paper hopes to explore and identify if this is actually possible using traditional Naïve Bayes (NB) and Support Vector Machine (SVM) classification approaches.

2 Related Work

Although there does not appear to be a lot of research involving the use of Reddit for language variation, there is related work in the area. Work by Gabriel Doyle proposes a method of identifying dialect variants through the use of Twitter user location data and their associated tweets [6]. He uses a Bayesian method to form a probability distribution model to identify dialect variations. There is also work using Twitter to identify racial variations of language by identifying racial characteristics in a Tweeter's profile. The goal of that paper was to identify how racial word usage propagates across social media [7].

3 Dataset

One of the nice things about using Reddit sub forums is that the data essentially self-annotates. The reason for this is that Reddit sub forums are traditionally hyper-specific for a specific topic. Therefore, if one wants information about Tallahassee, Florida, and finds that sub Reddit, that person will most likely find other people talking about only Tallahassee, Florida. That is the assumption that this paper takes. This paper also assumes that an individual who is actively seeking out a specific forum and participating in it either will be from that region or strongly affiliated with it and familiar with the region's customs.

For this experiment, the dataset was mined from scratch using the PRAW python library to access Reddit's API [1]. The forums mined were the Reddit sites for each of the states in the US, except Alaska and Hawaii, and the top 5 cites in each of those states by population size determined by the US 2010 Census [2]. If a city did not have a sub Reddit, it was disregarded. For each of the sub Reddits, up to the top 1000 posts for the year were pulled, with each post's respective comments. Order was not maintained for the post and its comments. The final dataset consisted of about 2.5 million Reddit comments.

The data also went through the following preprocessing measures:

1. All text was lowered.
2. Multi-letter words such as 'looool' were consolidated to their base versions of 'lol'.
3. All URLs were replaced with 'URL' and added to the list of stop words.
4. Double quotes were removed to not interfere with delimiting text data.

5. Text was then tokenized using NLTK's twitter tokenizer to account for emoticons and other social media tidbits [3].
6. All tokens were then split on white space and then concatenated together again without white space. For example, ':)' would be consolidated to ':)'. This was done to make the word count process easier and more accurate.
7. Punctuation, NLTK English stop words, 'URL', and '[Deleted]' tokens were removed. ([Deleted] is the token left behind when someone removes a Reddit post.)
8. Remaining tokens, if any, were then concatenated together, separated by single white space, and paired with their respective state label.

To determine regional categories, this experiment used the US Census Regions and Divisions map. The state's label paired with each Reddit post was replaced with one of the following numerical representations based off of their determined region on the map [4].

New England	0
Middle Atlantic	1
South Atlantic	2
East South Central	3
West South Central	4
East North Central	5
West North Central	6
Mountain	7
Pacific	8

After categorization, the regions had the following total Reddit posts associated with them:

New England	163,955
Middle Atlantic	309,537
South Atlantic	330,197
East South Central	161,338
West South Central	346,487
East North Central	341,082
West North Central	249,418
Mountain	242,336
Pacific	410,206

4 Experiment

For this experiment, Scikit-learn's Multinomial NB and Linear SVM classifiers were used [5]. These classifiers were chosen because of their ease of implementation and their good baseline results for determining the viability of a text classification exercise. Each classification exercise was undertaken against a unigram and bigram token model. The tested holdout percentages were 10% and 20%. To normalize word frequencies and their document significance, Term Frequency- Inverse Document Frequency (tf-idf) scoring was applied to the entire corpus. The classification exercise was also tested against the entire corpus and a random 150,000 sampling of each category to test against categories that had a uniform number of features.

5 Results

The top results for each classification exercise ran against each of the two datasets. The top results will be determined by which unigram/ bigram holdouts have the highest average f-score. As a possible optimization technique, the experiment was ran again while using only 25,000 words in each corpus. These results will be shared as well.

5.1 Support Vector Machine

5.1.1 > 1 Word Frequency

5.1.1.1 Precision/ Recall/ F-Score

SVM 10% Hold Out Bigram-Full				
class	precision	recall	f1-score	support
0	0.44	0.33	0.38	16263
1	0.4	0.42	0.41	30692
2	0.42	0.4	0.41	32730
3	0.38	0.3	0.33	15981
4	0.38	0.41	0.4	34317
5	0.43	0.39	0.41	33815
6	0.48	0.45	0.46	24724
7	0.4	0.35	0.37	24038
8	0.39	0.52	0.44	40672
avg/total	0.41	0.41	0.41	253232

5.1.1.2 Confusion Matrix

SVM 10% Hold Out Bigram Confusion Matrix									
	0	1	2	3	4	5	6	7	8
0	5432	1661	1332	514	1603	1362	848	870	2641
1	1036	12935	2305	854	3090	2472	1579	1491	4930
2	983	2994	12956	1245	3655	2634	1694	1788	4781
3	463	1342	1835	4750	1949	1493	918	919	2312
4	978	2877	2958	1211	14165	2695	1922	1854	5657
5	1037	3275	2795	1112	3674	13104	1936	1769	5113
6	603	1812	1723	761	2321	1971	11132	1271	3130
7	702	1852	2013	850	2653	1926	1350	8359	4333
8	1151	3438	2839	1083	4083	2818	1953	2320	20987

N-4 Cross Validation

Partial Corpus				
Unigram	0.3379	0.3414	0.3426	0.3421
Bigram	0.3731	0.3734	0.3752	0.3760
Full Corpus				
Unigram	0.3548	0.3541	0.3569	0.3550
Bigram	0.4020	0.4024	0.4013	0.4013

5.1.2 Top 25,000 Word Frequency

5.1.2.1 Precision/ Recall/ F-Score

SVM 10% Hold Out Unigram-Full				
class	precision	recall	f1-score	support
0	0.38	0.29	0.33	16263
1	0.39	0.35	0.37	30692
2	0.39	0.34	0.36	32730
3	0.35	0.26	0.30	15981
4	0.35	0.37	0.36	34317
5	0.35	0.37	0.36	33815
6	0.37	0.31	0.34	24724
7	0.33	0.33	0.33	24038
8	0.35	0.47	0.40	40672
avg/ total	0.36	0.36	0.36	253232

5.1.2.2 Confusion Matrix

	0	1	2	3	4	5	6	7	8
0	4772	1482	1247	506	1608	1675	891	1162	2920
1	1193	10744	2325	899	3170	3262	1652	1939	5508
2	1063	2617	10974	1268	3733	3573	1860	2434	5208
3	521	1139	1633	4128	1908	1868	1066	1234	2484
4	1062	2473	2879	1158	12600	3632	2015	2524	5974
5	1189	2765	2598	1078	3594	12476	2113	2340	5662
6	820	1884	1888	823	2737	2916	7659	1862	4135
7	797	1616	1822	770	2592	2503	1427	7977	4534
8	1302	3008	2808	1170	4287	3727	2104	3015	19251

5.1.2.3 N-4 Cross Validation

Partial Corpus				
Unigram	0.3038	0.3045	0.3051	0.3048
Bigram	0.2804	0.2816	0.2828	0.2821
Full Corpus				
Unigram	0.3134	0.3125	0.3145	0.3130
Bigram	0.2903	0.2933	0.2897	0.2904

5.2 Naïve Bayes

5.2.1 > 1 Word Frequency

5.2.1.1 Precision/ Recall/ F-Score

Naïve Bayes 10% Hold Out Bigram-Partial				
class	precision	recall	f1-score	support
0	0.41	0.41	0.41	15000
1	0.36	0.38	0.37	15000
2	0.39	0.31	0.34	15000
3	0.36	0.4	0.38	15000
4	0.39	0.28	0.33	15000
5	0.33	0.35	0.34	15000
6	0.41	0.42	0.41	15000
7	0.34	0.41	0.37	15000
8	0.36	0.37	0.37	15000
avg/ total	0.37	0.37	0.37	135000

5.2.1.2 Confusion Matrix

Naïve Bayes 10% Hold Out Bigram Confusion Matrix									
	0	1	2	3	4	5	6	7	8
0	6132	1464	785	1110	690	1233	966	1396	1224
1	1283	5703	885	1173	857	1386	1122	1230	1361
2	1080	1367	4598	1760	868	1320	1247	1565	1195
3	920	1086	1061	6038	836	1391	1180	1525	963
4	1114	1374	1045	1541	4250	1363	1251	1628	1434
5	1224	1429	874	1472	826	5208	1252	1514	1201
6	1010	1099	805	1275	789	1235	6330	1467	990
7	1030	1047	840	1370	795	1227	1148	6114	1429
8	1179	1430	814	1109	912	1194	1086	1713	5563

5.2.1.3 N-4 Cross Validation

Partial Corpus				
Unigram	0.3550	0.3523	0.3533	0.3527
Bigram	0.3628	0.3622	0.3629	0.3626
Full Corpus				
Unigram	0.3471	0.3468	0.3473	0.3462
Bigram	0.2907	0.2907	0.2906	0.2906

5.2.2 Top 25,000 Word Frequency

5.2.2.1 Precision/ Recall/ F-Score

Naïve Bayes 10% Hold Out Unigram- Partial				
class	precision	recall	f1-score	support
0	0.42	0.37	0.39	150000
1	0.35	0.38	0.36	150000
2	0.41	0.3	0.34	150000
3	0.33	0.4	0.36	150000
4	0.32	0.32	0.32	150000
5	0.34	0.33	0.33	150000
6	0.4	0.34	0.37	150000
7	0.34	0.39	0.36	150000
8	0.34	0.38	0.36	150000
avg/ total	0.36	0.36	0.36	135000

5.2.2.2 Confusion Matrix

Naïve Bayes 10% Hold Out Unigram Confusion Matrix									
	0	1	2	3	4	5	6	7	8
0	5607	1505	750	1242	1098	1129	873	1390	1406
1	1183	5638	741	1333	1214	1310	879	1150	1552
2	985	1302	4454	1907	1391	1150	984	1493	1334
3	811	1126	990	6019	1265	1257	981	1412	1139
4	931	1394	875	1714	4803	1201	1026	1461	1595
5	1137	1427	831	1626	1248	4904	1108	1349	1370
6	917	1336	754	1613	1289	1227	5124	1468	1272
7	897	1062	824	1549	1271	1092	978	5831	1496
8	1000	1427	685	1257	1306	1063	916	1602	5744

5.2.2.3 N-4 Cross Validation

Partial Corpus				
Unigram	0.3239	0.3249	0.3236	0.3232
Bigram	0.2976	0.2970	0.2975	0.2981
Full Corpus				
Unigram	0.3362	0.3362	0.3363	0.3374
Bigram	0.3129	0.3127	0.3126	0.3120

6 Conclusion and Future Work

Across the board, the results are quite poor. There are though some interesting observations that can influence future work. Looking at the Naïve Bayes classifier, it can be seen that it performs the best with a uniform, smaller dataset. This shows that Naïve Bayes might be a good candidate for targeted, low feature count situations.

The SVM also shows some interesting results. Performing the best out of both of the classifiers, it attained an average f-score of .41 using the full corpus in a bigram format. Using a smaller amount of features in a unigram format, the overall score dropped to .36. This was marginally better than the .35 average for the bigram reduced dataset (not recorded in paper). This shows the SVM classifier may be appropriate in situations where we have a very large corpus and willing to use an >1 n-gram format.

To try and dig deeper into why the classifier performed so poorly, the similarity between each class of documents can be compared. Looking at the diagrams below, one can see that the pruned word feature matrix and the base feature matrix are all extremely similar.

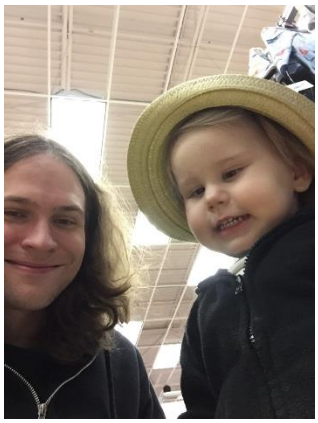
Cosine Similarity Matrix: > 1 Words									
	0	1	2	3	4	5	6	7	8
0	1.0000	0.9834	0.9843	0.9793	0.9815	0.9847	0.9834	0.9795	0.9792
1	0.9834	1.0000	0.9838	0.9789	0.9843	0.9880	0.9865	0.9752	0.9839
2	0.9843	0.9838	1.0000	0.9896	0.9878	0.9885	0.9904	0.9848	0.9794
3	0.9793	0.9789	0.9896	1.0000	0.9838	0.9851	0.9880	0.9857	0.9768
4	0.9815	0.9843	0.9878	0.9838	1.0000	0.9873	0.9870	0.9820	0.9820
5	0.9847	0.9880	0.9885	0.9851	0.9873	1.0000	0.9901	0.9814	0.9818
6	0.9834	0.9865	0.9904	0.9880	0.9870	0.9901	1.0000	0.9833	0.9805
7	0.9795	0.9752	0.9848	0.9857	0.9820	0.9814	0.9833	1.0000	0.9821
8	0.9792	0.9839	0.9794	0.9768	0.9820	0.9818	0.9805	0.9821	1.0000

Cosine Similarity Matrix: Top 25,000 Words									
	0	1	2	3	4	5	6	7	8
0	1.0000	0.9836	0.9844	0.9795	0.9817	0.9848	0.9836	0.9797	0.9794
1	0.9836	1.0000	0.9839	0.9791	0.9844	0.9881	0.9866	0.9753	0.9840
2	0.9844	0.9839	1.0000	0.9898	0.9879	0.9886	0.9905	0.9850	0.9795
3	0.9795	0.9791	0.9898	1.0000	0.9840	0.9852	0.9882	0.9859	0.9770
4	0.9817	0.9844	0.9879	0.9840	1.0000	0.9874	0.9871	0.9821	0.9821
5	0.9848	0.9881	0.9886	0.9852	0.9874	1.0000	0.9902	0.9815	0.9818
6	0.9836	0.9866	0.9905	0.9882	0.9871	0.9902	1.0000	0.9835	0.9807
7	0.9797	0.9753	0.9850	0.9859	0.9821	0.9815	0.9835	1.0000	0.9822
8	0.9794	0.9840	0.9795	0.9770	0.9821	0.9818	0.9807	0.9822	1.0000

These similarities also explain the erratic nature of the previously mentioned confusion matrixes. Document categories that are not easily differentiated from one another makes for a difficult classification problem.

In future work, possibly using the mean word count frequency as a starting point and using features within two standard deviations will produce better results. This will allow for a good mixture of unique and common words, perhaps strengthening the classifiers. Also enriching the dataset with other social media outlets could diversify the corpus with more unique word phrases, thus making each region stand out more.

7 About Author



I've been working with data management systems for almost six years. I'm very passionate about Semantic Web ontologies and graph databases. I've been developing a strong interest in NLP technologies and how they can be used to create intelligent help systems. In my off time I like to canoe, go to the park, and go on adventures with my daughter.

https://github.com/pumpkinslayer12

Reference

1. PRAW Development Team (2016). Python Reddit API Wrapper (PRAW), GNU General Public License. https://github.com/praw-dev/praw
2. U.S Census Bureau. 2010 Census Data. 2010. <http://www.census.gov/2010census/data/>
3. Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.
4. U.S. Energy Information Administration. "Commercial Buildings Energy Consumption Survey (CBECS)." n.d. U. S. Census Regions and Divisions. Image. 14 April 2016.
5. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
6. Bamman, David, Chris Dyer, and Noah A. Smith. "Distributed representations of geographically situated language." (2014): 828.
7. Eisenstein, Jacob, et al. "Diffusion of lexical change in social media." PloS one 9.11 (2014): e113114.