Sensing Topics in Tweets

Amar Budhiraja
Vikram Ahuja
Manas Tewari
Team - 39

Problem Statement

- Identifying topics in Tweets is a major problem as it poses the following challenges:
    - The length of a Tweet is very small
    - Noisy data
    - Uneven Typing (Use of words like “U”, “em” etc)
- We plan on comparing three Topic Detection in Twitter schemes from available literature.
    - Boosted LDA (From Probabilistic Methods)
    - Boosted N-Gram (From Statistical NLP Methods)
    - Soft Frequent Set Mining (From Data Analytics)
- The goal of the project is to understand the tradeoff between the three approaches when it comes to topic detetction in Tweets.

Dataset

Tweets were extracted for two topics:

- FA Cup 2015 (~2000)
- US Elections(~2000)
- SUPER TUESDAYS (~2000)
- TOTAL - 6231

Classifiers Used

- LDA(Unigrams Only)
- KNN(Calculating for Unigrams, Bigram and Trigram)
- K-Means (Unigrams Only)
- Naive Bayes(Calculating for Unigrams, Bigram and Trigram)
- Hashtag Mirroring and Proper Nouns Boosting done for every Classifier.

Latent Dirichlet Allocation

1. Each Document is considered as a probability distribution over topics which in turn are distributions over words.
2. Every Document is considered as a bag of terms which are only observed variables in the model.
3. LDA requires the expected number of topics k as input.

Naive Bayes Classifier

- Assumes independence among features
- Implemented as a Tweet being a BAG of Words Representation
- We did not consider any smoothing function over the Bayes Classifier - Neglected the words which were not seen earlier.
- For each Tweet the following Bayes equation was modelled based on its words

$$c_{MAP} = argmax P(c) \times \Pi P(x_i|c)$$

Clustering using K-Means

- Clustering algorithms aim at finding the natural grouping of a given set of elements with a notion of distance between each pair of elements.
- We used COSINE similarity between each pair of Tweets to perform clustering

K-Nearest Neighbor

- For each iteration of KNN algorithm, distance of input point is computed with every other data point in the set and the K closest neighbors are noticed
- Usually, a majority vote is taken into consideration to assign a class to the incoming point
- Odd value of K is preferred

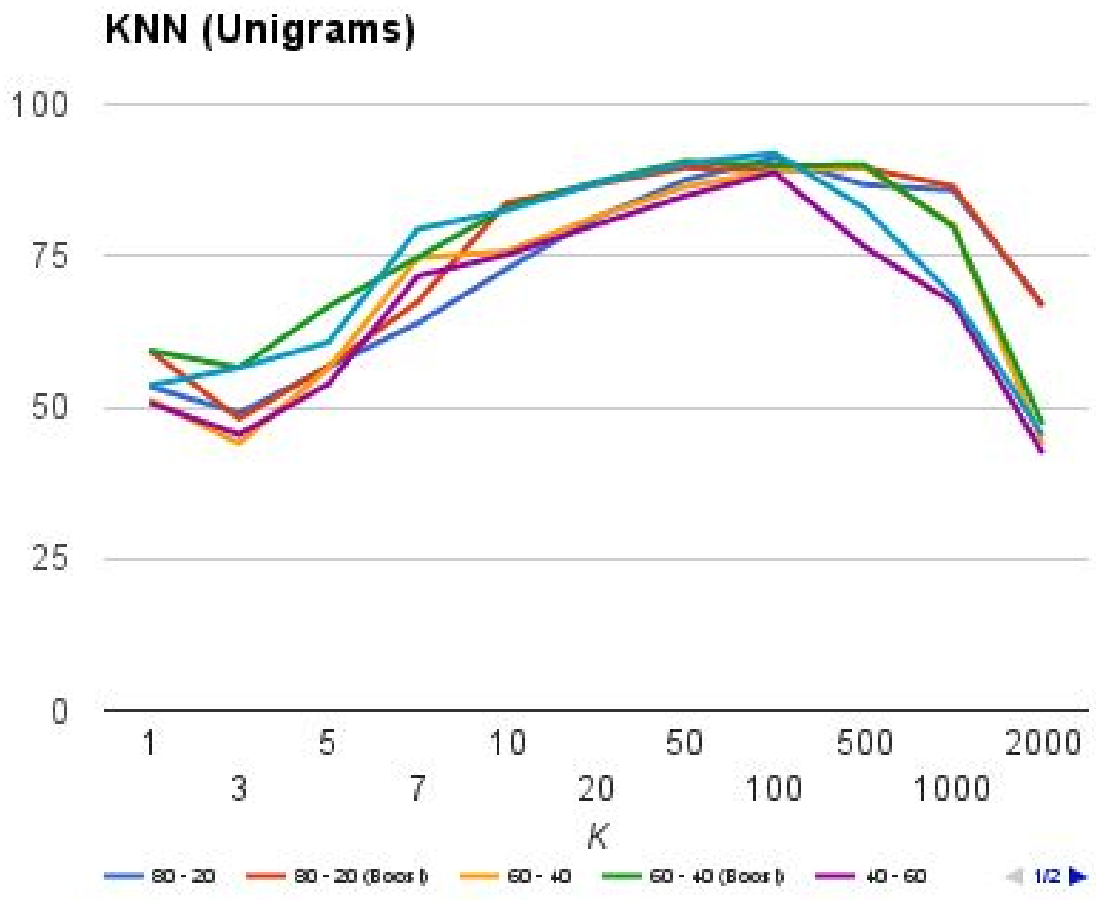
Our Approaches

- Stop word removal
- Boosting using duplication of hashtags and proper nouns
- Representing strings as numbers using TF*IDF
- An extremely sparse matrix created for TF * IDF. 14000*6000 in case of Unigrams, 113000 * 6000 in case of Bigrams and 189000*6000 in case of Trigrams.
- Checking for accuracy over different ratios training to test data (80-20, 60-40, 40-60)
- Cosine similarity used for K-means with number of clusters taken as 3
- Total number of dimensions in the features is equal total number of unigrams , bigrams and trigrams in their respective cases.

Results

KNN(Unigram)

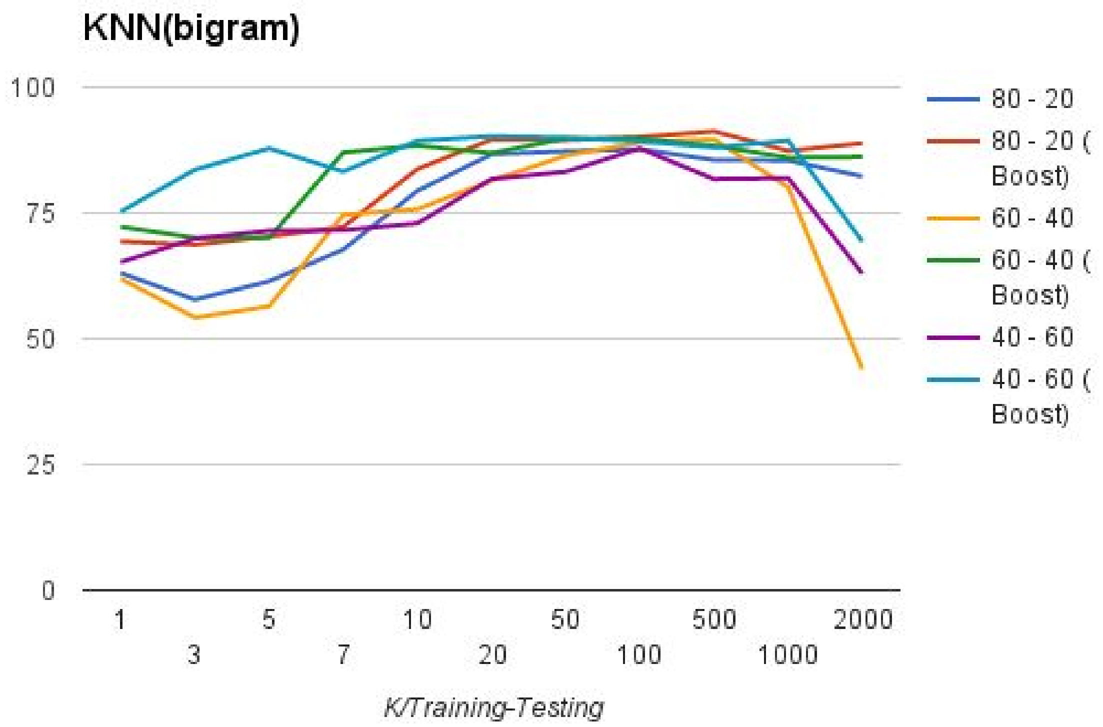
K/Training-Testing	1	3	5	7	10	20	50	100	500	1000	2000
80 - 20	53.45	49.06	56.67	63.89	72.87	81.29	87.5	91.1	86.72	85.8	66.911
80 - 20 (Boost)	59.39	48.16	56.78	67.43	83.74	86.76	89.51	89.05	89.46	86.43	66.66
60 - 40	51.28	44.1	56.43	74.64	75.77	81.45	86.4	89.01	89.62	80.11	43.93
60 - 40 (Boost)	59.39	56.56	66.66	74.74	82.97	87.13	90.63	89.87	90.02	79.8	47.24
40 - 60	50.74	45.54	53.89	71.65	75.19	80.12	84.87	88.73	76.53	67.24	42.43
40 - 60 (Boost)	53.63	56.52	60.76	79.417	82.54	86.93	90.23	91.88	82.92	68.34	45.41



KNN (Unigrams)
100
75
50
25
0
1
3
5
7
10
20
50
100
500
1000
2000
K
80 - 20
80 - 20 (Boost)
60 - 40
60 - 40 (Boost)
40 - 60
1/2

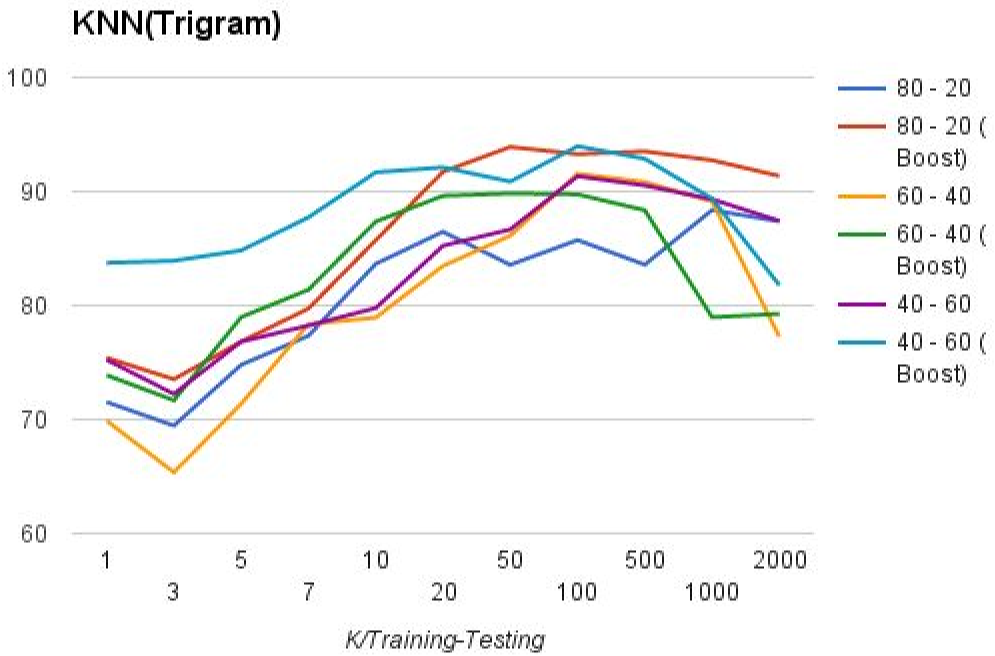
KNN(Bigrams)

K/Training-Testing	1	3	5	7	10	20	50	100	500	1000	2000
80 - 20	63.07	57.76	61.43	67.72	79.41	86.68	87.25	87.66	85.53	85.45	82.27
80 - 20 (Boost)	69.33	68.67	70.33	72.22	83.74	89.62	89.51	90.1	91.23	87.33	88.88
60 - 40	61.93	54.2	56.43	74.64	75.77	81.45	86.4	89.01	89.62	80.11	43.93
60 - 40 (Boost)	72.23	70.08	70.02	87.01	88.403	86.89	89.83	89.75	88.36	85.99	86.15
40 - 60	65.23	69.89	71.46	71.564	72.94	81.75	83.21	87.82	81.73	81.93	62.94
40 - 60 (Boost)	75.27	83.61	87.83	83.25	89.35	90.25	89.98	89.27	88.02	89.38	69.31


KNN(bigram)
100
75
50
25
0
1
3
5
7
10
20
50
100
500
1000
2000
K/Training-Testing
80 - 20
80 - 20 (Boost)
60 - 40
60 - 40 (Boost)
40 - 60
40 - 60 (Boost)

KNN(Trigram)

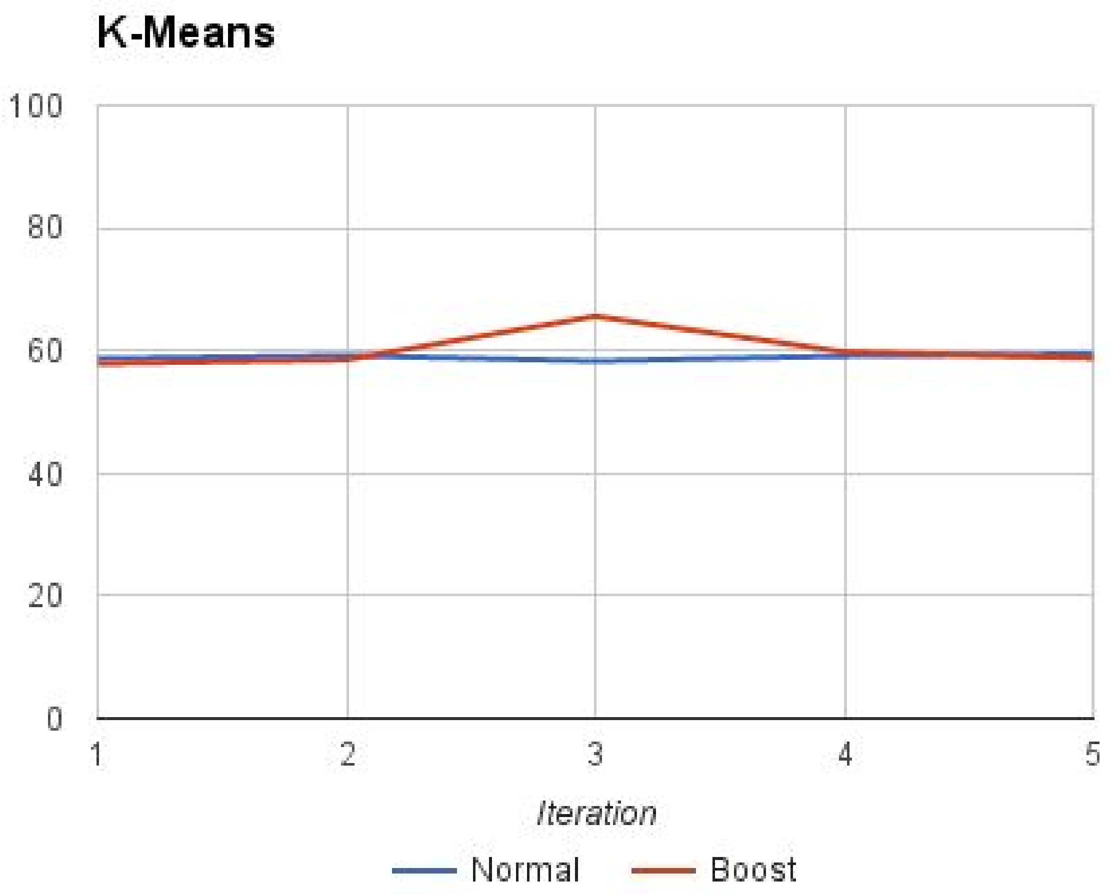
K/Training-Testing	1	3	5	7	10	20	50	100	500	1000	2000
80 - 20	71.53	69.46	74.79	77.34	83.65	86.47	83.57	85.74	83.58	88.37	87.38
80 - 20 (Boost)	75.41	73.53	76.83	79.73	85.72	91.73	93.91	93.28	93.52	92.74	91.36
60 - 40	69.89	65.37	71.37	78.36	78.93	83.47	86.12	91.56	90.83	89.12	77.27
60 - 40 (Boost)	73.89	71.68	78.98	81.38	87.36	89.62	89.83	89.75	88.36	78.97	79.27
40 - 60	75.24	72.24	76.83	78.24	79.78	85.24	86.67	91.34	90.53	89.32	87.42
40 - 60 (Boost)	83.74	83.92	84.83	87.73	91.68	92.12	90.87	93.97	92.87	89.38	81.78


KNN(Trigram)
100
90
80
70
60
1
3
5
7
10
20
50
100
500
1000
2000
K/Training-Testing
80 - 20
80 - 20 (Boost)
60 - 40
60 - 40 (Boost)
40 - 60
40 - 60 (Boost)

KMeans

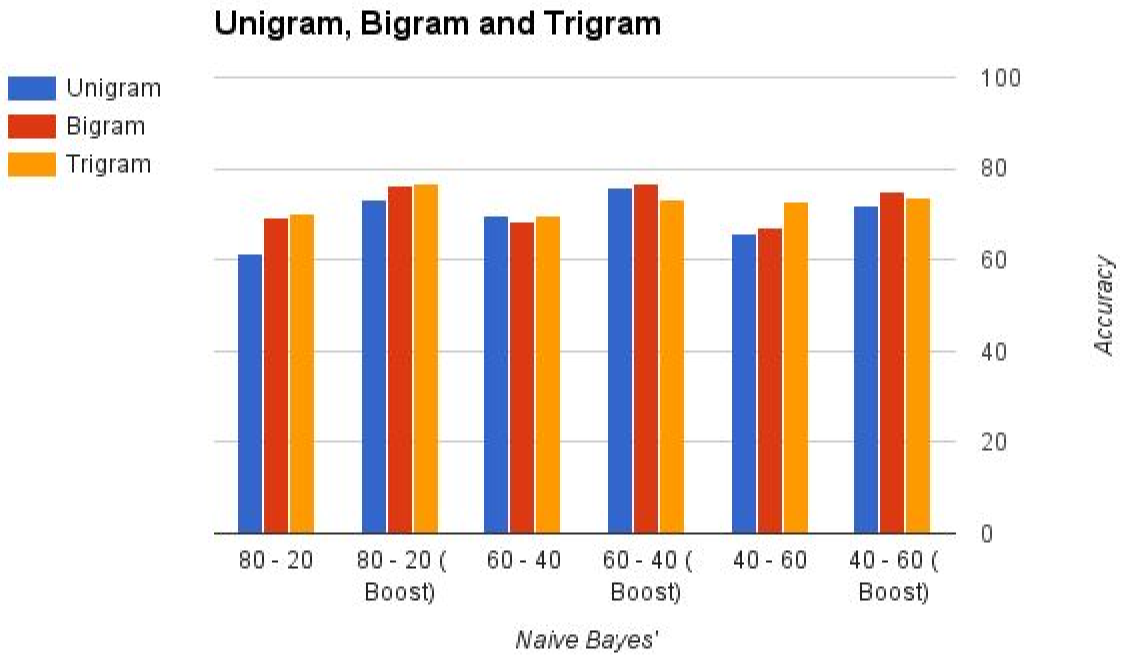
For 3 Clusters

Iteration	1	2	3	4	5
Normal	58.72	59.23	58.32	59.15	59.43
Boost	57.88	58.53	65.6	59.8	58.8


K-Means
100
80
60
40
20
0
1
2
3
4
5
Iteration
Normal
Boost

Naive Bayes

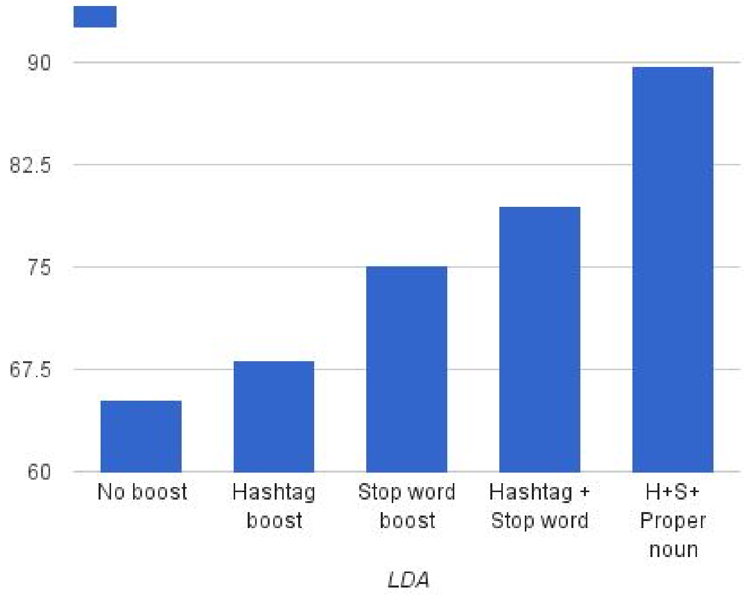
Naive Bayes'	Unigram	Bigram	Trigram
80 - 20	61.4	69.44	70.05
80 - 20 (Boost)	73.12	76.53	76.71
60 - 40	69.86	68.51	69.55
60 - 40 (Boost)	75.85	76.7	73.21
40 - 60	65.87	66.94	72.96
40 - 60 (Boost)	71.84	74.97	73.77



Unigram, Bigram and Trigram
Unigram
Bigram
Trigram
100
80
60
40
20
0
Accuracy
80 - 20
80 - 20 (Boost)
60 - 40
60 - 40 (Boost)
40 - 60
40 - 60 (Boost)
Naive Bayes'

LDA

LDA	
No boost	65.32
Hashtag boost	68.16
Stop word boost	75.14
Hashtag + Stop word	79.41
H+S+Proper noun	89.74



Conclusion

- Boosting proved too be a valuable modification to data on almost all occasions.
- In KNN we observe that for all the unigram, bigram and trigram the accuracy increases till a specific value of K and then decreases for large values of K.
- Trigram gave better results than bigrams and bigrams gives a better accuracy than unigram for all the cases.
- K-means gave least accuracy as it is an unsupervised method
- LDA performance increased drastically by boosting
- Trigrams provide the best result as most of the tweets are semantically similar and syntactically similar in case of Retweets.
- Naive bayes’ accuracy increases greatly by both boosting and increasing n-gram size

Thank you!
Questions?