CS229 Project Report, Fall 2014 (deepak.zambre@gmail.com), Ajey Shah (ajey.shah@gmail.com)

Topic based comments exploration for online articles

Introduction

With news now being consumed on the Internet, not only are consumers exposed to a wide range of reporting sources but a worldwide peer audience also adds their comments and thoughts about news pieces. These comments are sometimes very insightful and interesting especially on specialized communities like Hacker News. However, while the article may be well formatted and sectioned into different features, the comments are always presented as a long list sorted by last posted time or a voting system. We want to investigate methods and algorithms to expose a new way of navigating comments – one where comments are clustered and arranged based on topics from the article.

To present a rationale as to why this is a real problem: Consider the article where iPhone 6 was released. It had a gamut of information from the new OS features to new hardware features. If I'm a photography lover, I'm probably am first interested in knowing what the community has to say about the new camera features. Our solution will attempt to automatically categorize comments based on such topics extracted from the article. We believe this will bring greater value to the reader and the news site.

Dataset and nomenclature

For this project we will define the following terms:

"Story" – this refers to the actual original article that is being commented on. For example: The report on the new iPhone as published on TechCrunch is a "story".

"Top-level comment" – this is a comment published on hacker news for a particular story such that it has no other parent comment.

"Concept space representation" – For every story and comment, we run it through a concept/entity recognition algorithm as service. The output of this process is what we call the concept space of the data.

The dataset we use is available through https://github.com/HackerNews/API. This is a collection of all user generated discussion and comments, in plain text, for every *story* on the popular tech news site Hacker News. We only consider the *top-level comments* for our work. The data also has a link to the actual article on the web that is being discussed. We downloaded 1.2 million unique stories which together had 5.2 million comments.

For test data, we manually read 375 comments across 5 such stories on HackerNews and clustered them as human end users of the system. This allowed us to capture our users' way of mentally segregating topics in the story as well as comments.

Preprocessing

We crawl the original content of the news article and use basic cleansing API to remove the HTML cruft in the page. Once we have clean story and comment text data, rather than directly operate on bag of words of the original text, we attempt to operate on the concept space as generated by www.alchemyapi.com. Alchemy API approximately works on the principle as defined in [1]. It uses data from Wikipedia, freebase and other such open source datasets to create an entity graph. This graph is used to find important keywords that describe particular entities. Alchemy API uses this knowledge to identity important concepts from free text.

To reiterate our problem statement, we do not want to invent a new way of finding concepts in a text document. Rather what we propose are applications of clustering and prediction techniques on concept space representation.

Unsupervised learning based clustering approach

Random clustering

Randomly assign comments to concept clusters from the story. This was a great baseline score to compare against our more complicated clustering methods.

Cosine Similarity based clustering

Our next method to clustering was a very quick and dirty approach using the cosine similarity matching. Our algorithm is as follows:

```
For each concept in the original story:
        For each comment:
                Do:
                        -Find the similarity between the comment concept space and
                        the concept in the original story
                        -If similarity is greater than 0, add it to that concept
                        cluster.
```

KMeans

Since we are dealing with the problem of clustering comments (unsupervised), kMeans is a candidate algorithm that we could have tried with. We represented each comment as a vector of concept scores and used kMeans to cluster the comments in this concept vector space.

Bag of words (BOW) KMeans

We represented a comment as a vector of counts with each count representing the number of times a particular word appears in the comment. We applied kMeans on this representation of comments. The results for this approach are shown in graphs below. Note : We also tried stemming and removing stop words from comments however, in this document we are reporting results when we didn't stem and remove stop words (because this approach was giving us better v-measure). The motivation for using BOW is to be able to compare BOW approach against concept based approach.

Hierarchical Clustering

Conceptually speaking there is a system of taxonomy associated with our project dataset. We can think of a story with many comments. Each comment has a number of concepts and each concept has a number of attributes that describe it. Based on this theory we wanted to test the performance of hierarchical agglomerative clustering.

Supervised learning

Naïve Bayes

While in the previous sections we experimented with unsupervised learning, we also wanted to experiment with performance of supervised learning. This was particularly interesting in the real world scenario where we would need to be able to cluster comments as they come in after we use the initial set of comments to build out a model.

Results

For this problem we define two goals:

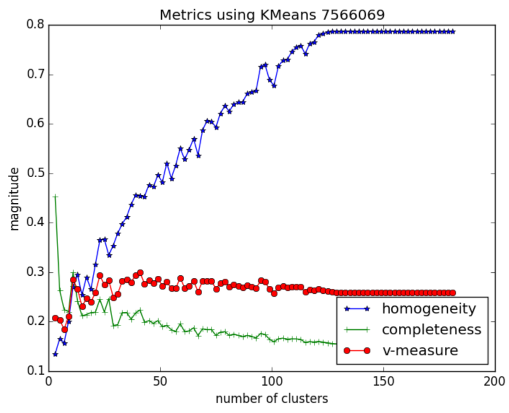
1. Keep conceptually similar comments with each other
2. Minimize number of concepts in comments to match concepts in original story.

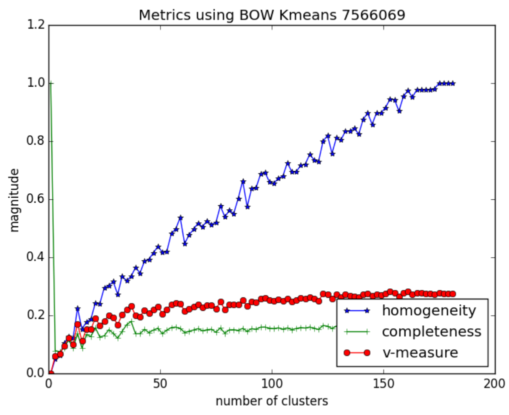
The first metric helps us understand how well we can cluster w.r.t. human expectation whereas the second objective ensures that concepts being discussed in the comments that are not related to the original story are not given too much weightage.

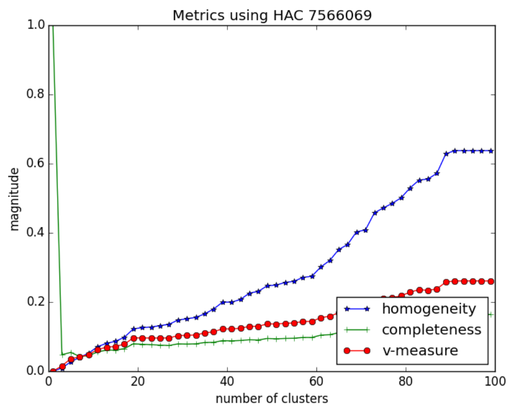
We compare the performance of aforementioned algorithms on 4 stories below with about 50~100 comments in each story. Note that the results reported are for the number of clusters that were expected by the user as defined in the test set. We believe this is a more realistic comparison than one purely based on the best v-measure since it tells us performance of algorithm as expected by an end user rather than for purely academic purposes.

Scores for story 7566069

Training method	Number of clusters	Homogeneity	Completeness	V-Measure
Random	8	0.060	0.053	0.056
Cosine ranking	8	0.012	0.304	0.024
KMeans	**9**	**0.201**	**0.322**	**0.247**
BOW KMeans	9	0.125	0.095	0.108
HAC	9	0.052	0.045	0.048
NaiveBayes	9	0.122	0.324	0.1775



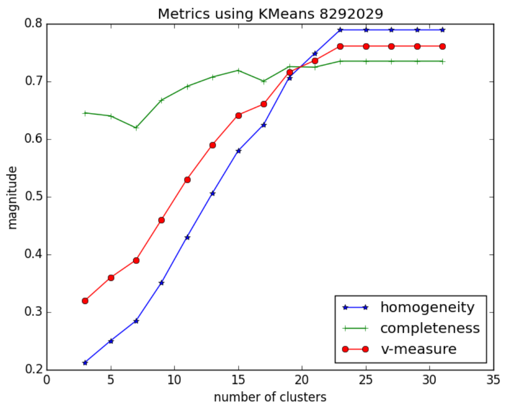


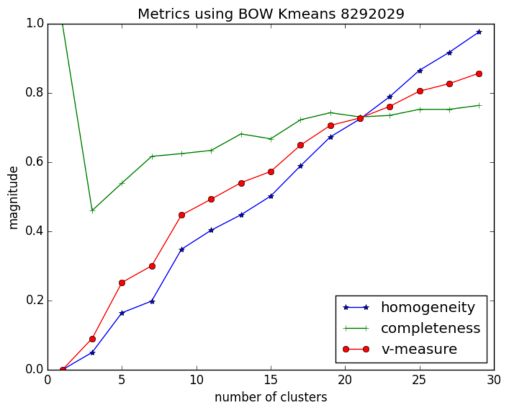


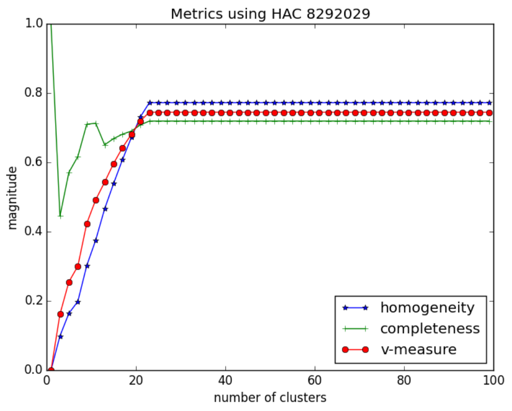
Scores for story 8292029

Training method	Number of clusters	Homogeneity	Completeness	V-Measure
Random	15	0.201	0.655	0.308
Cosine ranking	15	0.100	0.503	0.168
KMeans	**15**	**0.579**	**0.719**	**0.641**
BOW KMeans	15	0.501	0.667	0.573
HAC	15	0.538	0.668	0.596

NaiveBayes	-	-	-	-

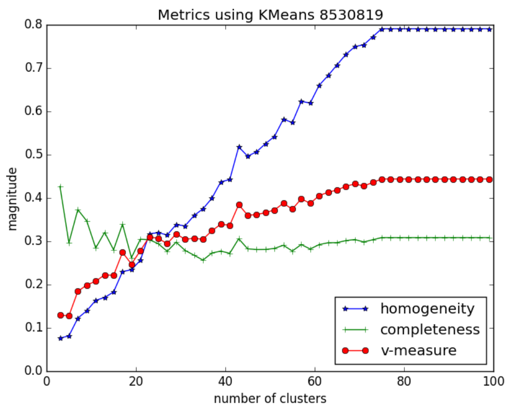


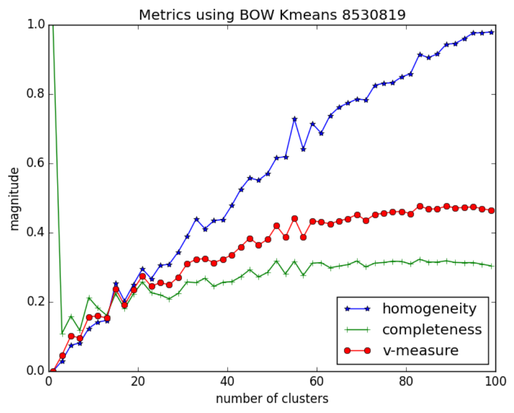


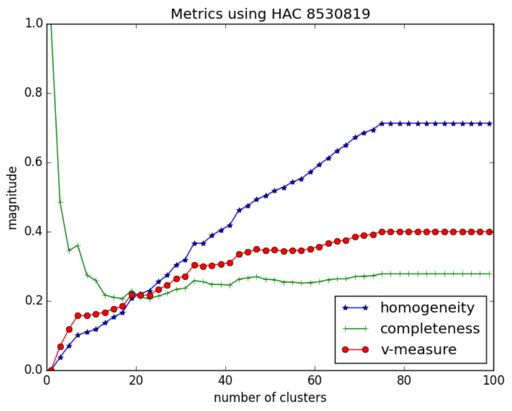


Scores for story 8530819

Training method	Number of clusters	Homogeneity	Completeness	V-Measure
Random	9			
Cosine ranking	9	0.0173	0.262	0.032
KMeans	9	0.234	0.262	0.247
BOW KMeans	9	0.250	0.223	0.236
HAC	9	0.207	0.230	0.218
NaiveBayes	**9**	**0.280**	**0.468**	**0.350**

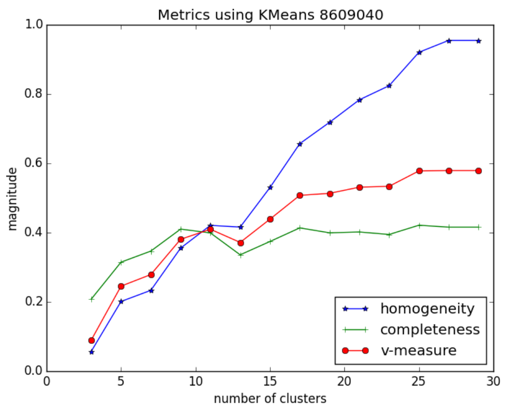


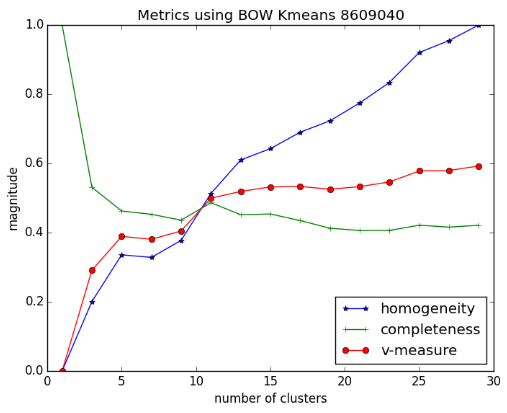


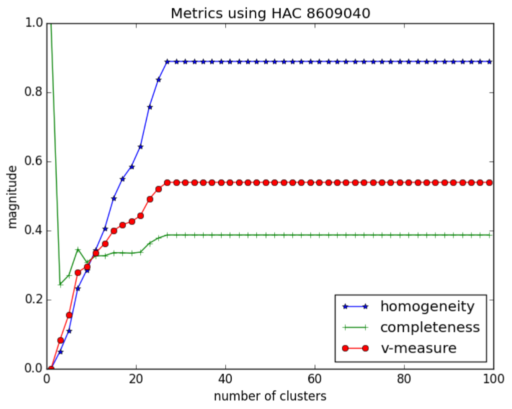


Scores for story 8609040

Training method	Number of clusters	Homogeneity	Completeness	V-Measure
Random	6	0.057	0.102	0.073
Cosine ranking	-	-	-	--
KMeans	7	0.233	0.346	0.278
BOW KMeans	**7**	**0.328**	**0.452**	**0.388**
HAC	7	0.233	0.346	0.278
NaiveBayes	-	-	-	-







Homogenity[1]: Score to determine if each cluster contains only members of a single class
Completeness[1]: Score to determine if all members of a given class are assigned to the same cluster.
V-Measure[1]: The V-measure is the harmonic mean between homogeneity and completeness.

Discussion

Cosine: The way we've implemented cosine ranking, we limit ourselves to actually matching concepts from the story. This is different than some of our later approaches. Without having keywords that describes each concept it's highly unlikely for the alchemy algorithm to detect exact concept matches. Hence we expect this algorithm to devolve into a simple binary lookup to the concept title. This explains the terrible performance of this approach. Getting a baseline result was however good for us to move forward with Kmeans based clustering as well as understanding that restricting clustering to only use story-comment relations and ignore comment-comment relations is a naïve approach.

kMeans and BOW kMeans:
Both, kMeans on concept space and BOW kMeans performed better than any other unsupervised algorithm that we have used. Additionally, kMeans on concept space of comments performed better than BOW kMeans. Intuitively, this observation corroborates clustering process used by humans: a human groups together comments talking similarly about similar entities. In our case, we are able to group comments talking of similar entities ignoring the sentiment. BOW kMeans weighs each word equally from a comment, whereas inherently in a comment, the keywords associated with main entity of the comment are more important than others.

HAC:
Our motivation here was to look for an unsupervised algorithm to beat kmeans score based on a taxonomy approach. Note that while HAC beat cosine and random, it was almost always slightly behind the kmeans based approach. We believe this is due to the fact that the hierarchy of story-

concepts, comments-concepts, and concept-keywords was not well fleshed out in the data set. Importantly we missed keywords that described particular concepts.

Naïve Bayes:
Our Gaussian Naïve Bayes model was built using the concepts used from alchemy API. We split our test set into 60% for training and 40% for testing our model's performance. Note that since we operate on the concept space, we don't use multinomial approach since frequency of a concept is always 1. Naïve Bayes beat other algorithms for story 8503819. Although promising, we believe that a more descriptive model for what defines concepts will prove beneficial in this case as well.

Conclusion

Our results show that it's definitely a net win to use kmeans clustering method on the concept space rather than on the text itself. We can draw this conclusion based on comparisons with the random method and bag of words based kmeans approach. However, getting a single method to always perform better is challenging due to nature of the data set and trained algorithms for concept space (like alchemy api). It was also clear to us that unsupervised methods are not suitable for practical purposes since they can't possible become real time. As for supervised Naïve Bayes, it will be hard to get real-time human judgments on comments to build a model.

Future work:

In the future it would be interesting to train concept space algorithm on specific genres of news and then use these models to provide a richer concept space for clustering. We would also suggest experimenting with more supervised means of clustering like decision trees, neural networks, random forests and online learning methods along with crowdsourced test set generation for initial comments to help build a model. A stretch goal would be to implement novel online learning methods as outlines in [6].

References

[1] Rosenberg and Hirschberg (2007)
[2] Story 7566069 - https://news.ycombinator.com/item?id=7566069
[3] Story 8292029 - https://news.ycombinator.com/item?id=8292029
[4] Story 8530819 - https://news.ycombinator.com/item?id=8530819
[5] Story 8609040 - https://news.ycombinator.com/item?id=8609040
[6] Vuvuzelas & Active Learning for Online Classification, Microsoft Research
[7] Python, scikit-learn