Enhancing Cluster Labeling using Wikipedia

Abhinav Gupta
New York University
abhinavg@nyu.edu

Chirag Maheshwari
New York University
chirag.m@nyu.edu

ABSTRACT

A lot of work has been done on labeling clusters in a way such that they represent what each cluster represent. Wikipedia is a very common tool used to get some semantic relationship among different words. Each wikipedia page has a title and some categories which are semantically close to the queried word.

We exploit this relationship among words to label a set of clusters more efficiently. We use 20-newsgroup dataset which has documents from 5 diverse categories. We apply different types of clustering algorithms on the training set. We extract the important words from each cluster using some distance measure like Jensen-Shannon Divergence (JSD) and cluster-biased-weighting for each term. We call this set of words important labels. We query each of the important term in Wikipedia and extract the title and the categories. We call this set of labels extracted from Wikipedia as candidate labels.

We use 2 types of Judge namely, MI (Mutual Information) Judge and SP (Score Propagation) Judge for scoring both the important labels and candidate labels. Then we rank each of the label for every cluster according to the score obtained from theses judges. Finally we find the similarity with the ground truth labels where we compare each label with every true label using a trained model like Word2Vec or Wordnet.

Keywords

Clustering; labeling; Wikipedia; Information Retrieval; Natural Language Processing

1. INTRODUCTION

There is a vast amount of textual data available in this digital age which brings about the need for efficient methods for processing and organizing in a way that is easily manageable. The popular approach for categorizing textual data into coherent sets of documents is by clustering. The goal of clustering is to partition the set of input documents into multiple coherent sets, where the documents in one set are similar to each other based on a given metric and dissimilar to document in other sets. Note that clustering algorithms largely depend on the distance or similarity metric used.

To work with the clustered documents in an informed way, it is necessary to tag or label the created clusters with human understandable titles. cluster labeling can be seen as extracting important features from the set of documents which best describe the cluster itself. There has been a lot of work done in providing quality cluster descriptors [2, 3, 5, 6]. But, often the cluster descriptors obtained from the textual data itself fail to describe the cluster radically or generally consist of terms which individually may be deemed non-sensible.

In this project we cluster-label the 20-newsgroup dataset using the vast volume of *human curated* articles from Wikipedia and it's search functionality to *improve the quality* of cluster descriptors and also analyze how the different clustering techniques affect these feature extraction or cluster descriptor extraction [1]. In the subsequent section we will provide information on the framework pipeline and its components and the experiments performed and their results.

2. FRAMEWORK PIPELINE

We created an end-to-end pipeline in Python [1] for cluster labeling using Wikipedia and evaluating the generated labels with the ground truth. The general flow of the system is as follows. The system first downloads and retrieves a set of documents from the internet which serves as the input documents. These documents are pre-processed and cleaned to only contain Alphanumeric character as described in section 2.1. This cleaning has been done to improve the performance of clustering algorithm. Next these pre-processed documents are passed to the clustering algorithm with the number of clusters as described in section 2.2. Every cluster is separated and sent to processing for feature extraction or important term extraction which uses Cluster-Biased Term Weighing and Jensen-Shannon Divergence. These techniques are described in detail in the section 2.3. Now that the important terms have been extracted, these terms are searched over the web on Wikipedia giving a big set of Candidate Labels which are then judged based on PMI and SP metric to give a final set of cluster descriptors. These are described in detail in sections 2.4 and 2.5

2.1 Indexing and Document Vector

Input documents are first pre-processed to only contain alphanumeric characters. This has evidently improved clustering of the documents in more coherent sets. After the basic pre-processing, these documents are parsed and tokenized to document vectors which are stored on the disc partitioned by document. The weights of every term is identified by the term frequency-inverse document frequency a.k.a *tf-idf* weighing scheme of the vector space model. This has been done by storing two sets of data, namely, Term Frequency and Inverse Document frequency. This helps us easily query

[1]Code available on github repository: https://github.com/chirag1992m/clustering-labeling

term frequency of a term t in a given document d which is represented as $tf(t,d)$ and inverse document frequency for a given term i in the entire collection represented as $idf(t)$. The set of all documents is represented by D.

In general, this *tf-idf* vector is very sparse for a given document and for any further calculations and algorithms we input the set of documents as a *sparse matrix*. This helps in saving the disc and memory usage while storing these documents and even running algorithms on them.

$$idf(t) = \log \frac{\sum_{d \in D} 1}{\sum_{d \in D} 1_{tf(t,d)>0}}$$

2.2 Clustering

The clustering algorithm is done to partition the set of documents into coherent sets where the document in one set is similar to the documents in the same set and dissimilar to the documents in other sets.

We'll represent the set of clusters as $\mathcal{C} = \{C_1, C_2, \dots C_n\}$ where n is the number of clusters and $\bigcup_{i=1}^{n} C_i = D$. Note that one document can be a part of multiple clusters which happens in Gaussian Mixture Model.

Although, the pipeline is not limited to a specific clustering algorithm; Here we explored multiple clustering algorithms, namely, K-Means with random and K-Means++ initialization, Gaussian Mixture Model, Agglomerative Clustering, and Birch Clustering.

For every clustering algorithm, the input was the Tf-Idf sparse matrix of the documents. The words with very low tf-idf value were ignored and only the most prominent 10000 features were taken into consideration. This was done to fasten the clustering algorithm and also resulted in better clustering of the documents. Other modifications made to the tf-idf vector for clustering was that any term with a very low and high term-frequency were ignored. This removed very common words like 'the', 'a', 'and' and not so common words like '11112121' which were the result of poor pre-processing techniques. The clustering algorithms used will be described in detail in the upcoming sub-sections.

2.2.1 K-Means

K-Means is a clustering algorithm which partitions the given document as vectors into n clusters. It does that by minimizing the sum of *euclidean distance* between document of one cluster over all the clusters. In doing so, it partitions the vector space into voronoi cells and thus a document can only belong to one cluster. The result of the K-means partition can be represented by the cluster centers and one can calculate the cluster participation by finding the cluster center with minimum distance.

2.2.2 Gaussian Mixture Model

Gaussian Mixture Model can be thought of as K-Means on steroids but much slower in converging to a solution. In a Gaussian Mixture Model, every cluster is represented by a gaussian distribution with unknown parameters and is thus a probabilistic model. This allows for a document to be a part of multiple clusters at the same time. The caveat with Gaussian Mixture Model is that it does not scale well with the number of documents.

2.2.3 Agglomerative Clustering

Agglomerative Clustering is a Hierarchical Clustering mechanism which forms clusters by repeatedly forming larger cluster from smaller clusters. Initially all the documents are taken as individual clusters. This algorithm easily scales with the number of documents used.

2.2.4 Birch Clustering

Birch Clustering is again a Hierarchical Clustering which tends to handle *noise* effectively. We hoped that it would easily figure out the miscellaneous category in 20-newsgroup dataset which wasn't the case as explained the in the experiments section.

2.3 Important Label Extraction

This was one of the most important part of the project as it involves feature extraction from the clusters formed which coherently distinguishes one cluster from the other clusters. These important terms/labels will be represented as $T(C)$ where C is a given cluster. Given a cluster $C \in \mathcal{C}$, we try to find $T(C) = t_1, t_2, \dots, t_k$ which is an ordered set. A term t_i can be any n-Gram but for our purposes we have extracted 1-Grams as important terms. We focused on two different techniques namely, cluster-biased weighing and Jensen-Shannon divergence.

2.3.1 Cluster-Biased Term Weighing

A cluster is best represented by the cluster's center; the weights to every term is dependent on the times it appears in the documents of a cluster and inversely proportional to the times it appears in the whole collection. It can be imagined as the Inverse Document Frequency biased towards the whole cluster instead of every document thought of as an individual cluster. The weight is represented as $w(t,C)$.

$$w(t,C) = ctf(t,C) \times cdf(t,C) \times idf(t)$$

where,

$$ctf(t,C) = \frac{1}{|C|} \sum_{d \in C} tf(t,d)$$

$$cdf(t,C) = \log(n(t,C) + 1)$$

$$n(t,C) = \sum_{d \in C} 1_{tf(t,d)>0}$$

In this scheme, the higher the weight the better the term represents that cluster.

2.3.2 Jensen-Shannon Divergence

Jensen-Shannon Divergence is similar to Kullback-Leibler Divergence (KLD) and is a measure of similarity between probability distributions. It is used as it is symmetric and is always non-negative making it ideal to be used as distance metric between two probability distribution. For important term extraction, we search for terms which *maximizes* the distance (Jensen-Shannon Divergence) between the cluster and the whole collection (without the cluster itself). Every term actually participates in the distance but we select the top K terms according to its contribution to the distance. Jesen-Shannon Divergence is represented as $JSD(t,C)$ and is calculated as follows.

$$JSD(t, P(C)||P(\mathcal{C})) = \frac{1}{2}KLD(t, P(C)||\mathcal{M}) + \frac{1}{2}KLD(t, P(\mathcal{C})||\mathcal{M})$$

where,

$$\mathcal{M} = \frac{1}{2}(P(C) + P(\mathcal{C}))$$

$$KLD(t, X||Y) = X(t) \times \log\left(\frac{1 + X(t)}{Y(t)}\right)$$

Here, $P(C)$, $P(\mathcal{C})$ represents the probability distribution of cluster C and collection $\mathcal{C}$ respectively. X and Y are any distribution over the term t.

2.4 Candidate Label Extraction

Now that we have our set of important labels T(C) we need to find some more labels which are semantically close to these important words which can act as a candidate for labeling the cluster.
One way is that we can just use the important terms extracted in the previous section and do the scoring accordingly. But many a times it has been found [4]
that the labels do not totally capture the semantic meaning of the cluster. It is possible to have better terms to act as a representative for the cluster. We try to get as many semantically close words that we can get so as to capture the total content in the cluster.
One of the most common ways to get semantically close words of a given word is using Wikipedia. We just need to query a word or a phrase in Wikipedia which will return a page with a title and a set of categories. Now of course there will be some labels for which a wikipedia page does not exist so we skip those words and just add the word as it is in the candidate label list.
Now for scoring (in the next section) each label we need to store the relative ranking of the label and the categories with respect to the initial important label ranking. So we store each candidate label with a ranking score according to the order of their occurrence in the Wikipedia search list. We will call this set of candidate labels as L(C). We will use the L(C) and the T(C) for scoring in the next section.

2.5 Scoring

At this stage we have both the important labels T(C) and the candidate labels L(C). We need some sort of scoring mechanism which can rank each of the labels (or candidates) according to their individual score and semantic score.
There are two types of judge proposed in earlier work [1] which are described in detail in the next 2 subsections.

2.5.1 MI Judge

The MI (Mutual Information) judge scores each candidate label by the average pointwise mutual information (PMI) of the label with the set of the clusterâĂŹs important terms, with respect to a given external textual corpus. So the PMI score acts as a semantic distance for each label with respect to the cluster content. It is a very common approach which is widely used in labeling cluster of blogs or old newsgroups which have some structural similarity.
The MI judge takes as input a list of important labels L(C), a list of candidate labels L(C) and an external corpus to evaluate each of the label. Given a label l ∈ L(C) and T(C), we find MI(l,T(C)) as follows:

$$MI(l, T(C)) = \sum_{t \in T(C)} PMI(l, t|corpus) \cdot w(t)$$

where w(t) gives the relative importance of the corresponding term in T(C) and $\sum_{t \in T(C)} w(t) = 1$. Although, For calculating MI we need the PMI scores for each (label, term) pair which is defined as follows:

$$PMI(l, t|corpus) = \log\left(\frac{Pr(l, t|corpus)}{Pr(l|corpus) \cdot Pr(t|corpus)}\right)$$

where Pr represents the probability of the corresponding term, label or (label,term) pair which is defined as follows

$$Pr(x|corpus) = \frac{\#(x|corpus)}{\#(corpus)}$$

where $\#(x|corpus)$ represents the count of x in the corpus. Note that here x can be any word-ngram (unigram, bigram, trigram, 4-grams and 5-grams). Depending on the type of ngram the denominator will also change depicting the count of the corresponding ngram in the corpus.
For our experiment we used many corpora including:

- Brown corpus which consists of readings of the English language from Brown University.
- Twitter corpus which includes random tweets from people.
- The complete 20-newsgroup dataset.

Note that all these datasets were preprocessed so that it was possible to find the ngram labels in the corpus since the labels were extracted from the preprocessed dataset.

2.5.2 SP Judge

The SP (Score Propagation) Judge as the name suggests propagates the score from the original document ranking list where we get the important labels and the candidate labels. In this type of judge, we deal with important labels and the candidate labels separately as their scores are mutually independent to each other. We describe the procedure for candidate labels but the same can be applied to the important terms as well.
Given a search query q which was extracted during the candidate label extraction step, we get a list of documents D(q). Now for a given candidate label l, we get the total sum of that label (split into different terms) across all the documents d ∈ D(q) and $l \in d$.

$$w(l) = \sum_{d \in D(q); l \in d} \frac{score(d)}{n(d)}$$

where n(d) represents the total number of candidate labels extracted from the query q and score(d) represents the inverse of the rank of the document in the D(q) list.
After getting the score of each split term, we sum up the score of each label to get the total score corresponding to that label.

$$w(kw) = \sum_{l \in L(C); kw \in l} w(l)$$

Algorithm	Top-5 Important Words using JSD
K-Means	God, Jesus, Israel, Israeli, Jews
Agglomerative	God, Christ, Christians, Bible, Faith
Birch	encryption, clipper, chip, keys, escrow
GMM	Jesus, us, gov, truth, evidence

Figure 1: Comparing clustering algorithms for Middle Eastern Category in 20-newsgroup dataset.

Now we take the average of each candidate label l and find the SP score as follows:

$$SP(l|D(q) = \frac{1}{n(l)} \sum_{k\in} w(kw)$$

Important terms are scored in a similar way using the scores from the distance metric (JSD or cluster-biased weighing) described in section-2.3.

2.6 Evaluation

Now we have predicted labels for each cluster and their corresponding MI or SP judge score, we rank all the labels and assign the best label for each cluster. Since we also have the ground truth labels we can find how close our predicted labels are using some similarity metric.
Word2Vec or Wordnet are very common packages which find the similarity score between two words. So we split the predicted labels and the true labels into single terms and find the score for each (predicted,true) pair. Then we sum up all the scores for a given predicted label and rank each label with the best true label score. Let us denote $d(y', y)$ as the distance between predicted label y' and true label y.

$$s(C) = \min_{y' \in topK(C)} \frac{1}{d(y', y)}$$
$$score = \sum_{c\in\mathcal{C}} s(C)$$

3. EXPERIMENTS

We started with different clustering algorithms and extracted the top-5 terms using JSD. For every category we manually checked if the important terms are in anyway able to describe the category they supposed to. The results are shown in fig. 1. As can be seen from the table, K-Means with K-means++ initialization easily surpasses the performance of every other clustering algorithm. This result made us to go forward with K-Means clustering for the upcoming parts of the pipeline and experiments.

In figure-2, we compare both the distance measure cluster based naive-weighting and the Jensen-Shannon Divergence for extracting the important labels from the cluster. The final metric is based on the Accuracy Score as described in the evaluations. We do not use any wikipedia extracted terms in this analysis.

In figure-3, we compare the performance of both the judges showing the improvement in accuracy of the wikipedia extracted labels. The final metric again is based on the Accuracy Score as described in the evaluations.

As we see from both figure-2 and 3, using wikipedia extracted labels increased the relative accuracy by around 80%.

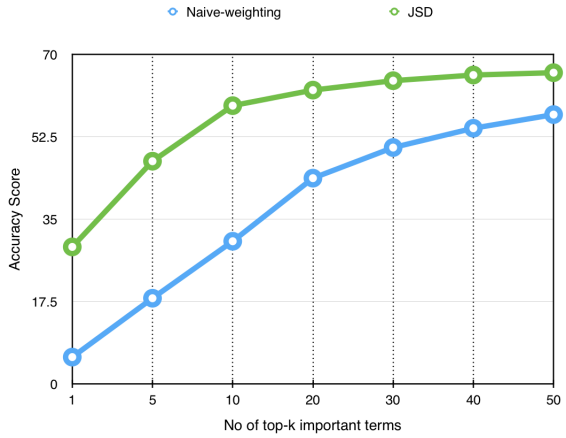

Figure 2: Comparing the two types of distance measure for extracting important terms.

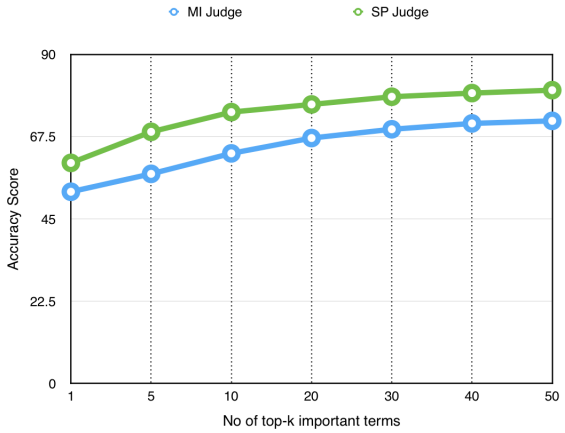

Figure 3: Comparing the scores of the two judges.

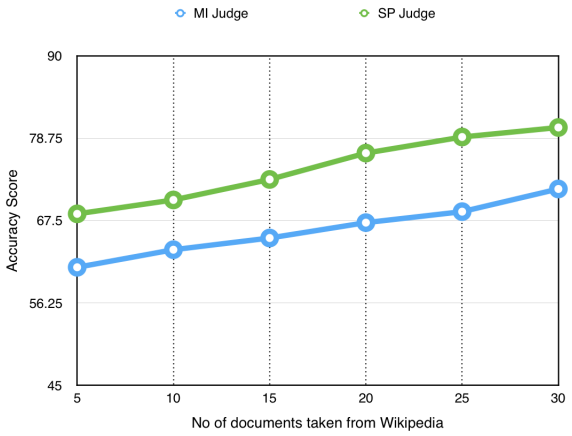

Figure 4: Analysis on the number of documents used from Wikipedia.

In figure-4, we compared the performance by choosing different amounts of wikipedia docs when an important label is queried in wikipedia.

4. CONCLUSIONS

In this project we established that cluster labeling can be greatly enhanced by the vast volume of articles and their associated meta-data available on Wikipedia. They improve the semantic meaning of cluster labels and are even robust to noisy important terms extracted. Experimentation with different clustering algorithms also helps us understand that the underlying structure of clusters is important for a superior performance of cluster labeling.
This method of labeling cluster can be used to cluster and label unsupervised data which can later on be compared to the ground truth labels. The metric defined in this work is also novel and can be extended for comapring different types of datasets.

5. FUTURE WORK

In this work we only considered the title and categories of the Wikipedia articles. Using the hierarchical structure of clusters and labeling them from specific to more general labels as we go up the hierarchical tree could further improve these cluster descriptors. In addition to datasets' hierarchical structure, Wikipedia articles are hyperlinked and connected to each other making a graphical structure which can be further exploited to retrieve semantic relationships between cluster labels.

6. REFERENCES

[1] D. Carmel, H. Roitman, and N. Zwerdling. Enhancing cluster labeling using wikipedia. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146. ACM, 2009.

[2] S.-L. Chuang and L.-F. Chien. A practical web-based approach to generating topic hierarchy for text segments. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 127–136. ACM, 2004.

[3] D. R. Cutting, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 126–134. ACM, 1993.

[4] F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani. Cluster generation and cluster labelling for web snippets: A fast and accurate hierarchical solution. In *International Symposium on String Processing and Information Retrieval*, pages 25–36. Springer, 2006.

[5] E. Glover, D. M. Pennock, S. Lawrence, and R. Krovetz. Inferring hierarchical descriptions. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 507–514. ACM, 2002.

[6] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake. Using web structure for classifying and describing web pages. In *Proceedings of the 11th international conference on World Wide Web*, pages 562–569. ACM, 2002.