Clustering and labeling of online product reviews

[1] Mohammad Nuruzzaman Bhuiyan, [2] Md Masum Billah, [3] Md. Aktarujjaman

[1] (PhD Fellow), [2] University of Duisburg-Essen Duisburg, Germany, [3] BSc in Software Engineering, Daffodil International University, Bangladesh

[1] mdnuruzzaman2001@yahoo.com, [2] billah.masumcu@gmail.com, [3] akhtarspondon@gmail.com

***Abstract-*We have presented an unsupervised approach to cluster reviews of products collected from Amazon and then generate labels of each cluster. Instead of using a complete review we split a review into sentences and consider all sentences from the reviews as inputs for clustering. We use Hierarchical Agglomerative Clustering to cluster sentences. Our cluster labeling approaches are also unsupervised. For labeling, we have used three different methods to find out a limited number of important words for each cluster. Extracted important words are used to construct phrases. Constructed phrases are used as labels for each cluster. To evaluate our labeling result, we have compared the result of our labeling method with a baseline labeling. In the evaluation, all of our labeling methods outperform the baseline method. Our cluster labeling approaches are also unsupervised. For labeling, we have used three different methods to find out a limited number of important words for each cluster. Extracted important words are used to construct phrases. Constructed phrases are used as labels for each cluster. To evaluate our labeling result, we have compared the result of our labeling method with a baseline labeling. In the evaluation, all of our labeling methods outperform the baseline method.**

**Keywords-**

1. INTRODUCTION

Nowadays, online shopping is a favourite way of purchasing for many people. But when a buyer wants to judge various aspects of a product, only the product description is not sufficient. Reviews written by other buyers give extra confidence to a new buyer to make his/her decision about purchasing. For instance, when a new buyer wants to buy an iPhone by reading its reviews which has been written by existing users, he/she might know what other people said about the display, battery life or camera quality and so on. Usually, a product contains a considerable number of reviews, but all of them are not possible for a reader to go through. Most of the time a reader just read a couple of reviews to know more about the product. We can improve this situation using machine learning algorithm to provide a user a compressed summary of all reviews or most essential aspects that reviews contain. To give a user in-depth insight of reviews one approach could be grouping the similar reviews. This grouping approach is called clustering where similar reviews appear in the same cluster and reviews from two different clusters. By inspecting the contents of a cluster, a user can be able to know what aspect the cluster is talking about. By reading a comparably small amount of reviews of each cluster, a user can have a general idea about the feasibility of the product instead of reading all reviews of the product. The reason is, a good clustering algorithm should group the similar reviews together. To label a cluster, first we select some essential uni-gram terms from the cluster, then these essential terms are used to construct phrases that are used as cluster labels. We have also considered sub-clustering of a cluster when intra-similarity of a cluster content is reduced. Making a small number of sub-clusters of a cluster helps to maintain good intra similarity in each sub-cluster. From each sub-cluster of a cluster, a set of important uni-gram terms are extracted. These important uni-gram terms of sub-clusters are used to determine a combined set of important terms for the cluster. The combined important terms are used to build human plausible phrases for cluster labeling.

1. RELATED WORKS IN CKUSTER LABELING

A core concepts and terminologies of information retrieval and natural language processing which have been applied in this thesis. We have shown the literature review of work on clustering and related approach of finding labels of clusters. So far, a limited amount of research has been done on cluster labeling compared to the clustering algorithm. There exist several ways for cluster labeling. A very common approach for cluster labeling is extracting important terms from the cluster content that can describe the cluster in contrast to other clusters [10]. Important terms can be identified by finding the most frequent terms in the cluster or from the top weighted terms in the cluster centroid [10]. Any other statistical techniques for feature selection can also be applied [12]. Frequent phrases are also considered for cluster labeling in several works [29].

Figure 3 shows a dendrogram of HAC where documents (document index) are shown in x-axis and distance is shown in the y-axis. A horizontal line represents the merge of two clusters. The y-coordinate of a horizontal line is the distance between two clusters that were merged. We call this distance the combination distance of the merged cluster. To consider both descriptive and discriminative power of a label of the cluster [13] selected candidate words for each cluster by using a modified version of the information gain measure [9] which allows word selection that mostly represent the cluster’s contents and is least representative of the contents of other clusters. Instead of using a list of top-ranked words to describe a cluster, their system uses substring that best matches to the selected top-scoring words from the titles. Beside important terms for a cluster, the title of the document which is mostly close to the centroid of the cluster is considered as cluster label [10].

1. DATASET

Our dataset is formed of online product reviews on Amazon. We have considered dataset reported by [32] that contains 142.8 million reviews collected from Amazon1 panning from May 1996 to July 2014. In the dataset, each line contains a single review in JSON (JavaScript Object Notation) format.

Fiqure 1 shows a sample of a review in JSON format where reviewer ID is a unique id of the reviewer, a sin is a unique id of the product, reviewer Name is the name of the reviewer, helpful is the helpfulness rating of the review, review Text is the actual review written by the reviewer, overall is the rating of the product, summary is a compact summary summerized by the reviewer unix Review Time and review Time are time in unix format and time in raw format. We have only used content of review Text for the clustering and labeling purpose. We have taken three products for manual annotation. Name of products are Kindle Fire HD 7, Panasonic ErgoFit In-Ear Earbud Headphones and SanDisk Memory Card. These three products have been chosen because each of them has more than 10 thousand reviews. More product reviews enable us to run the clustering and labeling algorithm on more data.

1. Manual annotation: Our manual annotation, we have taken 500 reviews for Kindle Fire HD 7, 100 reviews for Panasonic ErgoFit In-Ear Earbud Headphones and 100 reviews San-Disk Memory Card. To avoid very small or extensive sized reviews, we have chosen reviews which have character length between 150 to 1500. An annotator has split each review into sentences and assigned topics to each sentence. A topic is a group of words or phrase that indicates what the sentence is talking about.

We have provided the annotator a python script that collects all annotated aspects with the counting of their occurrences. For instance, “sound quality”: 76, “price”: 50, “comfortability”: 26. It means “sound quality” appears in 76 sentences, “price” appears in 50 sentences and so on. With the help of above counting, the annotator has divided aspects into several groups. We call these groups as aspects cluster. We have limited the number of maximum groups to 15 because we think more than 15 clusters will not be user-friendly to read. In Figure 2 shows manually annotated aspects clusters of Headphone dataset.

The annotator aspects divided into several groups. These groups as aspects cluster. Aspects clusters have been used to cluster sentences. A python script has been used to iterate each sentence of every review. Every sentence those do not contain any aspect has been assigned to a single cluster named Miscellaneous. Other sentences have been clustered according to their first aspect. When the first aspect does not appear on aspects clusters then we consider the second aspect and so on. In this way, we have achieved our manual clusters of sentences.

1. CLUSTERING

Our whole process starts with clustering. On an online shopping site like Amazon, people tend to write several aspects of products in a single review. If we make hard clusters with the complete text of reviews, it won’t give us the right result because a complete review is a potential candidate for multiple clusters. Instead of clustering whole text of review, we split each review into sentences. Sentences collecting from all reviews are the candidate for our clustering algorithm. For example, following review is collected from a dataset [32] of Amazon product reviews.

Amazon (https://www.amazon.com/) is an online shopping site founded in 1994.

1. Sentence Representation: A common approach to represent a document in a vector space is using Bag-of-words (BOW) model [33] where each document is represented in a fixed sized feature vector. Features are extracted from the collection of all documents, and they are called as the dictionary. But BOW representation does not consider word semantics. In BOW representation, “city”, “town” and “apple” all words have same representation although “city” and “town” are more semantically closer than “city” and “apple”. We consider using Word2Vec [20] words embedding to represent a sentence in a vector space. Word2Vec[20] has been used because of its potentiality of considering word semantics .To cluster collected sentences, we have represented each sentence with a fixed length vector. A sentence vector is derived from its words vectors. In word vector, each word is represented by Word2Vec [20] vector. A sentence vector is vector summation of its words vectors.

SentenceV ector =X t2T

WordV ector(t) (1) Where T is the set of qualified words in a sentence after applying pre-processing. WordVector returns 300-dimensional Word2Vec [20] vector of the word.

1. Word2Vec Word Vector: each word in a sentence is represented by Word2Vec vector which is a distributed representation of words introduced by [20] . We use Word2Vec because represented vector can capture both syntactic and semantic relations between words. Word2Vec[20] uses the Skip-gram neural network model [21] which is an efficient method for learning high-quality vector representations of words that capture a large number of precise syntactic and semantic word relationships from large amounts of unstructured text data. The aim of Skip-gram model is to find word representations that are useful to predict surrounding words in a document or sentence. Alternatively, the neural network is going to be trained such a way that if a word from a sentence is given, the network can tell us the probability for other words in the vocabulary of being a nearby word. A nearby word is determined by window size. A window size of nw means, nw words behind and nw words after the targeted word in a sentence. With the Word2Vec representations of words, semantic similarities between two words can be captured. For example, vectors of “Soviet” and “Russia” has more similarity than vectors of “Soviet” and “Cat”. The Word2Vec representations of words vectors explicitly encode many linguistic regularities and patterns. Many of these patterns can be represented as linear translations. For example, the result of a vector calculation vec("Germany") - vec("Berlin") + vec("Bangladesh") is closer to vec("Dhaka") than to any other word vector. We have used the publicly available Word2Vec vectors for a vocabulary of 3 million words and phrases.
2. Pre-processing: Pre-processing unit accept a sentence and returns a set of qualified words. The set of words have been used to obtain a sentence vector. An input sentence is first lowercased and tokenized into unigram word. Lowercasing ensures two same words are represented in the same way even they are written in the different cases. Tokenizing split a sentence into words. We have used the default 3 implementation of word tokenizer in NLTK [34] python library. Then stop words are removed from the return set of words. Stop words are considered non-meaningful words concerning the text. The set of qualified words contains meaningful words only. Removing stop words also reduces noise.
3. The Clustering: After having the vector for each sentence we cluster sentences by using Hierarchical Agglomerative Clustering. Hierarchical clustering has been used instead of flat clustering because flat clustering like K-Means [30] tends to produce similar size cluster [23]. But in our context, people can write more on one aspect than other. So it is normal to have size variation among clusters. We do two-phase clustering. The first phase is cleaning phase. We cluster all sentences using HAC where the distance between sentence vectors is measured by cosine distance [27] and we use complete [15] as linkage. Highly frequent words have more magnitude than less frequent words. With cosine distance, the angle between two vectors is measured, it does not depend on magnitude. So when we want to calculate the semantically meaning of two words or sentences, cosine distance is the good choice than other distance metric like Euclidean distance. Cosine distance between two vectors is bounded within limits of [-1,1] [1]. Also used cosine similarity [27] to measure similarity between terms or document pair. Cosine similarity is the complement of cosine distance. Complete is used as linkage method although the method is sensitive to an outlier. Complete linkage distance d between two cluster means the maximum distance between any two data points could be d in the cluster. Every document is maximal d distance away in the group. Small cut off value of d ensures the cluster is congested. <https://code.google.com/archive/p/word2vec/>

Figure 4 shows sample dendrogram of HAC with the cut off value of 0.8 for the first phase. The HAC applied on sentences of 100 reviews of “Panasonic ErgoFit In-Ear Earbud Headphones.” HAC applied to sentences vectors with d cut off value. We have chosen a large value of d which is 0.8. A large cut off value produces less inter-similar clusters. If a cluster contains very few sentences they supposed to be highly dissimilar to other clusters. We consider those clusters as noise. We only consider clusters which have at least n (n = 5) number of sentences in it. A small cluster does not contain much information those are central to our interest. We merge all these small clusters together and label as none or Miscellaneous because all of these small clusters contain several miscellaneous topics. In the second phase, we run same clustering algorithm again with remaining sentences. We can get a fixed number of clusters from agglomerative clustering. In our test case, we have generated 10 clusters (a small rounded number). If the computation is not a problem, instead of a predefined fixed number of clusters we can search for an optimum number of cluster.

Figure 4 HAC dendrogram after removing noise sentences the number of sub-clusters from one to five. We can apply the same method to find an optimum number of clusters between five to fifteen (a small number of clusters).

1. CLUSTER LABELING

For cluster labeling, we have considered several approaches. Each of the approaches. Cluster labeling is independent of clustering. Although we have applied clustering algorithm to group similar sentences but our cluster labeling algorithm can be applied independently to any set of clusters. As a content unit, unigram token is an excellent choice but as a label, unigram token is not plausible, reported by Salton and Buckleyp [27]. The approach that we have followed is derived phrase from sentences based on provided unigram token. We have considered three variations to extract important terms (unigram tokens) from a cluster, they are:

* 1. Using TF-IDF with stemming.
  2. Using TF-IDF while combining similar unigram term by using Word2vec [20] embedding and
  3. Using TextRank [35] to extract keywords (unigram token).

Extracted unigram tokens have been used to derive phrases. Derived phrases act as a set of labels of a cluster.

sajdojldflsdhlfkhsdlkfhljaskdh