Clustering and labeling of online product reviews

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

[1] (Ph.D. 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 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 baseline labeling. In the evaluation, all of our labeling methods outperform the baseline method.**

**Keywords-**

1. INTRODUCTION

Nowadays, online shopping is a favorite 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 a machine learning algorithm to provide a user with 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 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 similar reviews. 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 CLUSTER LABELING

A core concept and terminologies of information retrieval and natural language processing have been applied in this thesis. We have shown the literature review of work on clustering and related approaches to 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 the 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 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.

Figure 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 summarized by the reviewer Unix Review Time and review Time are time in Unix format and time in raw format. We have only used the content of review Text for the clustering and labeling purposes. We have taken three products for manual annotation. The name of products is 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 the above counting, the annotator has divided aspects into several groups. We call these groups as aspect 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. Figure 2 shows manually annotated aspects clusters of the 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 in 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 the whole text of the review, we split each review into sentences. Sentences collecting from all reviews are the candidate for our clustering algorithm. For example, the 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 the dictionary. But BOW representation does not consider word semantics. In BOW representation, “city”, “town” and “apple” all words have the 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--------------------------------- (1)

WordV ector(t) (1) Where T is the set of qualified words in a sentence after applying to pre-process. WordVector returns a 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 the 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 in 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 the 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: The pre-processing unit accepts a sentence and returns a set of qualified words. The set of words has 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 different cases. Tokenizing split a sentence into words. We have used the default 3 implementations 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 clusters [23]. But in our context, people can write more on one aspect than others. So it is normal to have size variation among clusters. We do two-phase clustering. The first phase is the 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 semantical meaning of two words or sentences, cosine distance is a good choice than other distance metrics like Euclidean distance. Cosine distance between two vectors is bounded within limits of [-1,1] [1]. Also used cosine similarity [27] to measure the similarity between terms or document pair. Cosine similarity is the complement of cosine distance. Complete is used as the 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 the 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 that have at least n (n = 5) number of sentences in it. A small cluster does not contain much information that is central to our interest. We merge all these small clusters and label as none or Miscellaneous because all of these small clusters contain several miscellaneous topics. In the second phase, we run the same clustering algorithm again with the 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 clusters.

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 the clustering algorithm to group similar sentences 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 phrases 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.

* 1. Using TF-IDF with stemming

We have used TF-IDF term weighting scheme to extract essential terms from a cluster. The term frequency is calculated based on the cluster. How many times a term appears in a cluster is the tf value of a term in thecluster. A term is a unigram token. The term frequency of a term t for a cluster is the following:

--------------------------------------------------------------- (2)

idf value determines the overall importance of a term. We have calculated idf value of each term by using following formula:

--------------------------------------------------------------- (3)

Where nr is the total number of reviews and df(r; t) is the document frequency which is the number of reviews that contain the term. To calculate idft value of a term t, reviews have used instead of sentences. Idf value of each term remains the same for further TF-IDF calculation. To get rid of unnecessary terms, stop words were removed. Instead of using a complete term, we have also applied stemming to each term so that two almost similar terms are considered as same. For example, “sounding” and “sounded” is two forms of “sound”. With stemming “sounding”, “sounded”, and “sound” all three appears as “sound”.

The weight of a term is its TF-IDF value. Top-weighted 10 terms are collected from the cluster.

* 1. Combine similar terms for TF-IDF calculation

We have applied stemming to words to treat syntactically related words similarly. If two words are syntactically totally different but their semantics meaning is the same, with stemming we can not treat them similarly. For example, “Russia” and “Soviet” are two semantically closer words although syntactically they are not closed with each other. Treating two words separately those has same meaning while determining TF-IDF, can potentially hamper the weighting criteria. To avoid this, we grouped syntactically and semantically identical words. We use Word2Vec [20] embedding to represent a word with a 300-dimensional vector. And then apply Hierarchical Agglomerative Clustering on them with complete linkage and cosine as distance metric with a specific threshold value (0.5 in our case). Cosine distance is looking into the angle of a vector, and complete linkage will ensure every term in a cluster has a similar meaning.

For example, consider the following unigram tokens taken from Headphone reviews:

“noise”, “sound”, “loud”, “cancel”, “canceling”, “cancellation”, “head-phones”, “earbuds”, “earbud”, “ipod”, “cord”, “cable”, “cords”, “cables”, “jacket”, “cheap”, “price”, “color”, “plastic”, ’tangle”

After applying HAC, they are group together and the result is: [cancel, cancelling, cancellation], [cheap, price], [noise, loud, sound], [color], [cord, cords, cable, cables], [plastic], [headphones, earbuds, earbud], [ipod], [jacket], [tangle].

**Calculating document frequencies**

As we previously clustered uni-gram tokens, we can apply similar words meaning to the document frequency calculation process. To do that first we have transformed all reviews in our collection into count vector (Bag of words model) on vector space of a fixed number of previously determined features where a feature will represent the number of times a term appears. So we’ll have a two-dimensional vector where rows represent the number of reviews and column represents the number of features. If two terms have a similar meaning, we have taken their column matrix and element-wise summed together. Then replace their column matrix with the summed column matrix. The number of times a term appears in a cluster is its TF value. To consider similar words in TF calculation, each cluster is represented by a vector of tf number of features where tf is the same features used in IDF calculation. In vector space, each row represents a cluster and column represents a feature. If two terms have a similar meaning, we have taken their column matrix and element-wise summed together. Then find the term in similar terms which has the maximum value. The maximum value has been replaced by summed value and others value in the row have been replaced by zero.

* 1. Using TextRank

TextRank [35]is a graph-based unsupervised ranking model which can be used for keyword extraction. TextRank [35] uses a similar technique as PageRank where the rank of a vertex is calculated recursively using global knowledge instead of knowledge of local vertex. The basic idea of the graph-based model is an implementation of voting or recommendation. When one vertex is connected to another vertex, it is casting a vote to another vertex. The importance of the vertex is determined by the number of votes casting for the vertex. Formally, let G = (V;E) is a directed graph with a set of vertices V and set of edges E. For a given vertex Vi, let In(V ) is the set of incoming vertices (set of vertices that point to it) and Out(Vi) is the set of outgoing vertices (set of vertices that vertex Vi points to). The score of vertex Vi is defined in equation 4 [24].

--------------------------------------------------------------- (4)

Where d is a damping factor. The value of d can be assigned between 0 and 1. The role of the damping factor is the probability of jumping from a given vertex to another random vertex. The value of d is usually assigned to 0.85 [25].

To extract keywords from a cluster using TextRank [35] unigram token is represented as vertices. To build an edge between vertices, co-occurrence relation is used. Two vertices are connected if their uni-gram tokens co-occur within a window of maximum Nw words. The value of Nw can be set between 2 and 10. Graph vertices are also restricted with syntactic filters which only allow unigram tokens of a certain part of speech. [35] had experimented with various syntactic filters, they had reported the best result for using nouns and adjectives [35]also considered different values of Nw, they have reported finding the best result when using 2 as the value of Nw. We have used the same value for Nw and syntactic filters reported [35].

The TextRank [35] keyword extraction algorithm follows the following steps to find the keyword from the text. First, the text is tokenized as unigram token. Unigram token is selected to reduce the number of vertices. Multi words keywords are constructed in the post-processing step. Then each token is annotated with part of speech tags. Only nouns and adjective tokens are considered for building vertices. Then an edge is added between vertices when two vertices are co-occurred within a window of Nw tokens where the value of Nw is set to 2.

The graph, that has been building is an undirected unweighted graph. A score for each vertex is initialized with the value of 1 and then ranking formula described in equation 4 is run on the graph for several times until it converges at a threshold of 0.0001. Usually, it takes 20-30 iterations to converge. Once we get the final score for each vertex, vertices are sorted in reversed order of their score. Top 10 vertices (unigram tokens) are selected for the post-processing step. In post-processing [35] used adjacent words to build multi-words keywords. Instead of adjacent words, we have used noun phrases to find labels of a cluster. Details of constructing cluster labels from a set of unigram tokens.

* 1. Labeling of a cluster

Unigram terms are not plausible labels. Phrases are more plausible than unigram terms reported [27] [35] had reconstructed multi-word keywords from unigram terms. By their method, adjacent terms are collapsed into a multi-word keyword. For example, consider the following text “The picture quality is good”. If both “picture” and “quality” is selected as potential keywords, they are collapsed into one single keyword “picture quality”, since they are adjacent. We are considering noun phrases instead of adjacent terms because a noun phrase is more natural as a cluster label. A noun phrase is a phrase that acts as a noun. In our approach, top-weighted 10 terms are primarily selected as a set of labels for the cluster. Then the system finds matching noun phrases which are the combination of top-weighted terms. If a noun phrase is found with this criterion, then it is added to the set of labels and terms that are present in the noun phrase are removed from the labels set. To find the matching noun phrases, first, all noun phrases are extracted from the cluster. Each remaining noun phrase is checked if they are built with top-weighted terms. While checking a noun phrase against top-weighted terms, stop words are not considered. For instance, consider the following text “The picture quality is good”. Extracted noun phrase for this sentence is “the picture quality”. If both “picture” “quality” is appeared on top weighted terms set, then we consider “the picture quality” as a label and remove “picture” and “quality” from the set of labels.

4spaCy is an open-source python module for NLP. On their website https://spacy.io/, they claimed spaCy as industrial-strength NLP. The module is an active development with more than 200 developers.

* 1. Mean squared errors (MSE)

We determine a cluster quality by using mean squared errors which is average of squared distances from the cluster centroid to its all data points. A cluster centroid u is defined as the mean of the data points in a cluster w :

--------------------------------------------------------------- (5)

In our case, the data point is sentence vectors in a cluster. Then we calculate the distance from every sentence vector to its centroid u . All distances are summed and averaged to obtain mean squared error.

--------------------------------------------------------------- (6)

A low MSE value means a compact cluster and a high value means disperse cluster.

* 1. Sub-clustering

We make sub-clusters of a cluster and Figure 6 shows the sample of HAC, when a cluster has a high value of mean squared errors (MSE). Mean squared errors of a cluster is decreased with the increased number of clusters. Selecting a large number of sub-clusters produces a large number of labels. A large number of labels for a cluster is not user-friendly. So we select a small number of sub-clusters for a given cluster. We limit the number of sub-clusters to 5 (a small number of sub-clusters). 5 sub-clusters are not being used for each cluster, instead, we find the value for the number of sub-clusters between 1 to 5. A minimum number of sub-clusters with near minimum average MSE are chosen. To select the optimum number of sub-clusters, a similar method like Elbow method is used [17]. A near minimum average MSE means an average MSE value that is close to the minimum average MSE. In Figure 7 x-axis shows number of sub-clusters and y-axis shows average MSE. The figure contains graph for five clusters. Shows in different colors. For each cluster, we gradually apply sub-clustering from 1 to 5 number of sub-clusters and find the average MSE for sub-clusters. For example, for the first cluster, we divide the cluster into two sub-clusters. Then measured the mean squared error of both sub-clusters and take the average. Then find the average MSE for three, four and five sub-clusters respectively. Figure 7 shows five main clusters with average mean squared errors for 1 to 5 sub-clusters. For each cluster, we choose a minimum number of sub-clusters with near minimum average MSE. Table 1 shows the number of sub-clusters in the blue cluster and their corresponding SubClusterThreshold.

To find an optimum number of sub-clusters, minimum average MSE of 1 to 5 sub-clusters is divided by corresponding average MSE. A result of 1 means minimum average MSE but most of the time minimum average MSE is found with the highest number of sub-clusters. Our aim is to find a value close to 1 that we called near minimum average MSE.

--------------------------------------------------------------- (7)

Where MinimumAverageMSE is a minimum average MSE of different amounts of sub-clusters and AverageMSE is average MSE of the corresponding sub-cluster.

For example, in Figure 8, the x-axis indicates the number of sub-clusters and y-axis indicates values that are calculated from minimum average MSE of different sub-clusters divided by average MSE of the corresponding sub-cluster(SubClusterThreshold). If we consider the blue colored cluster in Figure 8, the minimum average MSE is found with 5 sub-clusters. However, with 2 sub-clusters the SubClusterThreshold is 0.90 which is a big improvement from 0.61 in the case of 1 sub-cluster. To find the optimum number of sub-clusters we choose minimum numbers of sub-clusters with SubClusterThreshold of at least 0.8 (0.8 is very close). Table 2 shows an optimum number of sub-clusters for the different clusters.

Figure 8 shows normalized diversity with a different number of sub-clusters.

* 1. Combining important terms for a cluster and labeling

Amount of sentences of each sub-cluster in a cluster is not equal. One sub-cluster could contain most numbers of sentences where others can contain less. To combine terms weights from different sub-clusters, the weight of a term from a sub-cluster is multiplied by the ratio of the number of sentences in the sub-cluster and the cluster. If the same terms appear on two or more sub-clusters, their weight simply added together.

--------------------------------------------------------------- (8)

Top-weighted nsc terms from each sub-clusters are collected where the value of nsc is calculated using equation 8 In the equation 8, the ratio of the number of sentences in sub-cluster and the cluster is multiplied with 10 then ceiling value of the result is the number of terms that we have collected from the corresponding sub-cluster. Combined terms from all sub-clusters are used to derive phrases.

1. EVALUATION AND RESULTS
   1. Clustering evaluation

Sum-of-Squared-Error (SSE) criterion has used to evaluate the performance of the clustering algorithm. The method is an intrinsic evaluation method where the quality of clusters is measured based on how compact the clusters are. We have chosen the approach because it does not require any gold standard data. Summing over the squared distances between the clustering objects and cluster centroid is a standard cost function. The following equation defines the SSE:

--------------------------------------------------------------- (9)

Where, K is the number of clusters, !k is k0th cluster, ~(wk) is the centroid of k0th cluster. A cluster centroid is the mean of the documents in a cluster [17]. Although our clustering evaluation method allows us to only compare results with each other we are nevertheless able to determine which approach performs better.

* 1. Baseline: TF-IDF based labeling

To compare our labeling quality, we have used TF-IDF based labeling as the baseline. We tried to keep the baseline as simple as possible but yet an effective and well-established approach. [1] also used TF-IDF based labeling as the baseline. We have followed their footsteps. The cluster has used unigram token as the term and later extracts phrases from essential terms. Phrase extraction based on terms could introduce another layer of complexity. To make the baseline labeling simple enough, noun phrase has used to represent the term. This approach is similar to reported [1]. In the baseline labeling, tf (term frequency) is the number of times a term appears in a cluster. And idf (inverse document frequency) is computed based on the number of reviews that contain the term. The term frequency of a term t for a cluster ! is the following:

-------------------------------------------------------------- (10)

idf value of a term t is:

-------------------------------------------------------------- (11)

where nr is the total number of reviews and df(r; t) is the document frequency. which is the number of reviews that contain the term t. Terms are used as labels for a cluster. Top-weighted 10 terms is used to represent a cluster.

* 1. Automatic evaluation of cluster labeling

In the automatic evaluation, machine-generated cluster labels are compared with gold standard labels from the cluster. We calculate the similarity of each machine-generated labels with clustering and labeling of online product reviews gold standard labels. Word vectors are used to compute the similarity. A machine-generated label is represented by the summation of Word2Vec [20] vector of each word in the label. The same way also represents gold standard labels. *Cosine similarity* is used to measure the similarity between two labels. Two identical labels produce a similarity score of 1.0. We take each machine-generated label and check the similarity with all of the golden labels for the cluster. The maximum score is taken for each label. Then the average similarity score of each label is the final score for cluster labeling score.

-------------------------------------------------------------- (12)

Where ~v(l) is the vector for the label l. And ~x 2 l represents set of word vector from the label.

Let Lg is the set of gold standard labels for a cluster ! and Lm is the set of labels generated by our proposed method for the cluster !. Each label of Lm will be compared with each label of Lg by using cosine similarity.

-------------------------------------------------------------- (13)

-------------------------------------------------------------- (14)

The average score of all clusters is considered for the evaluation.

5Source code of our implementation can be found on <https://github.com/billahmasumcu/clustering-reviews-labeling> 6Manually annotated by Md Masum Billah and Khaled Hossain, University of Duisburg-Essen

* 1. Results

The results of running an evaluation method for evaluating clustering and labeling algorithm in various settings. TABLE 3 and 4 show the result of the clustering evaluation for two different sets of reviews. Each table contains the result of various settings. Various settings are: making sentences vector with considering stop words removed from the sentence, only considering nouns for a sentence, consider nouns and adjectives, considering nouns and verbs, considering nouns, adjectives, and verbs. The lower value of SSE means good results. Each method of word selection also contains two-column, one is without removing noise sentences and the other is with noise sentences removed. TABLE 4 Evaluation of clusters for Kindle 500 reviews Table 5 shows SSE value of manually annotated clusters. The table also contains the number of sentences those are not a member of the Miscellaneous cluster. While calculating SSE, sentences of Miscellaneous clusters were not considered. Table 6 shows the quality of clustering with or without removing noise sentences for different datasets. TABLE 5 SSE of manually annotated cluster TABLE 6 Effect of noise removing while clustering Table 7contains evaluation result of baseline labeling for clusters of Headphone 100 reviews and clusters of Kindle 500 reviews. Table 8 and 9contains labeling evaluation results for three different methods with and without applying sub- clustering. TABLE 7 Evaluation of baseline labeling TABLE 8 Labeling evaluation of Headphone 100 reviews TABLE 9 Labeling evaluation of Kindle 500 reviews.

1. REFERENCES

[1] Aker, Ahmet et al. Automatic label generation for news comment clusters In FProceedings of the 9th International Natural Language Generation Conference, pages 61–69. 2016.

[2] Christopher d. manning, prabhakar raghavan, and hinrich "Introduction to information retrieval" In Springer, 2010

[3] Baker, L Douglas and McCallum, Andrew Kachites "Distributional clustering of words for text classification" In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pages 96–103. 1998

[4] Barker, Emma and Paramita, Monica Lestari and Aker, Ahmet and Kurtic, Emina and Hepple, Mark and Gaizauskas, Robert "The SENSEI annotated corpus: Human summaries of reader comment conversations in on-line news" In Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue,pages 42–52, 1998

[5] Bezdek, James C and Ehrlich, Robert and Full, William "The fuzzy c-means clustering algorithm" In Computers & Geosciences,pages 191–203, 1984

[6] Bezdek, James C and Ehrlich, Robert and Full, William "Natural language processing with Python: analyzing text with the natural language toolkit" In O’Reilly Media, Inc., 2009

[7] Carmel, David and Roitman, Haggai and Zwerdling, Naama "Enhancing cluster labeling using wikipedia" In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, pages 139–146, 2009

[8] Chin, Ong Siou and Kulathuramaiyer, Narayanan and Yeo, Alvin W "Automatic discovery of concepts from text" In 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI’06), pages 1046–1049, 2006

[9] Cover, Thomas M and Thomas, Joy A "Elements of information theory" In John Wiley & Sons, 2006

[10] Cutting, Douglass R and Karger, David R and Pedersen, Jan O and Tukey, John W "A cluster-based approach to browsing large document collections" In ACM SIGIR Forum, ACM New York, NY, USA, pages 148–159, 2017

[11] Freedman, David A "Statistical models: theory and practice" In cambridge university press, 2009

[12] Gabrilovich, Evgeniy and Markovitch, Shaul and others "Computing semantic relatedness using wikipedia-based explicit semantic analysis" In IJcAI, pages 1606–1611, 2007

[13] Geraci, Filippo and Pellegrini, Marco and Maggini, Marco and Sebastiani, Fabrizio "Cluster generation and cluster labeling for web snippets: A fast and accurate hierarchical solution" In Springer, pages 25–36, 2006

[14] Jurafsky, Dan and Martin, James H "Speech and language processing. Vol. 3" In Pearson London London, 2014

[15] Lance, Godfrey N and Williams, William Thomas "A general theory of classificatory sorting strategies: 1. Hierarchical systems" In PThe computer journal, pages 373–380,1967

[16] Liu, Xin and Gong, Yihong and Xu, Wei and Zhu, Shenghuo "Document clustering with cluster refinement and model selection capabilities" In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pages 191–198,2002

[17] Mogotsi, IC "Christopher d. manning, prabhakar raghavan, and hinrich schütze: Introduction to information retrieval" In Springer, 2010

[18] Chen, Feng-Yi and Tsai, Pi-Fang and Chen, Keh-Jiann and Huang, Chu-Ren "Sinica Treebank" In Computational Linguistics and Chinese Language Processing, pages 87–104, 1999

[19] Chen, Feng-Yi and Tsai, Pi-Fang and Chen, Keh-Jiann and Huang, Chu-Ren "Textrank: Bringing order into the text" In Proceedings of the 2004 conference on empirical methods in natural language processing, pages 404–411, 2004

[20] Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff "Distributed representations of words and phrases and their compositionality" In Advances in neural information processing systems, pages 3111–3119, 2013

[21] Mikolov, Tomas and Chen, Kai and Corrado, Greg and Dean, Jeffrey "Efficient estimation of word representations in vector space" In arXiv preprint arXiv:1301.3781, 2013

[22] Pedregosa, Fabian and Varoquaux, Gaël and Gramfort, Alexandre and Michel, Vincent and Thirion, Bertrand and Grisel, Olivier and Blondel, Mathieu and Prettenhofer, Peter and Weiss, Ron and Dubourg, Vincent and others "Scikit-learn: Machine learning in Python" In a journal of machine learning research, pages 2825–2830, 2013

[23] Pedregosa, Fabian and Varoquaux, Gaël and Gramfort, Alexandre and Michel, Vincent and Thirion, Bertrand and Grisel, Olivier and Blondel, Mathieu and Prettenhofer, Peter and Weiss, Ron and Dubourg, Vincent and others "Scikit-learn: Machine learning in Python" In a journal of machine learning research, pages 2825–2830, 2013

[24] Porter, Martin F and others "An algorithm for suffix stripping." In Citeseer,pages 130–137, 1980

[25] Ratnaparkhi, Adwait "A maximum entropy model for part-of-speech tagging" In Conference on Empirical Methods in Natural Language Processing, 1996

[26] Ratnaparkhi, Adwait "Term-weighting approaches in automatic text retrieval" In Information processing & management,pages 513–523, 1988

[27] Salton, Gerard and Buckley, Chris "Term weighting approaches in automatic text retrieval" In Cornell University,1987

[28] Syed, Zareen and Finin, Tim and Joshi, Anupam "Wikipedia as an ontology for describing documents" In UMBC Student Collection,2008

[29] Treeratpituk, Pucktada and Callan, Jamie "Automatically labeling hierarchical clusters" In Proceedings of the 2006 international conference on Digital government research, pages 167–176, 2006

[30] Willett, Peter "Document clustering using an inverted file approach" In Journal of Information Science, pages 223–231, 1980

[31] Xu, Wei and Liu, Xin and Gong, Yihong "Document clustering based on non-negative matrix factorizatio" In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pages 267–273, 2003

[32] He, Ruining and McAuley, Julian "Fusing similarity models with markov chains for sparse sequential recommendation" In 2016 IEEE 16th International Conference on Data Mining (ICDM), pages 191–200 2016

[33] Harris "Bag-of-embeddings for text classification." In IJCAI, pages 2824–2830 1954

[34] Loper, Edward "NLTK: Building a pedagogical toolkit in Python" In PyCon DC, 2004

[35] Mihalcea, Rada and Tarau, Paul "Textrank: Bringing order into text" In Proceedings of the 2004 conference on empirical methods in natural language processing, pages 404–411 2004