Embedding and Frequent Itemset based Hierarchical Clustering with Labels and Outliers

Vini Dixit
Freshworks Inc[†]
vini01.dixit@gmail.com

ABSTRACT

Customer support is an essential part of any product company in today's digital era of business and customer interactions. Business growth and customer satisfaction is directly proportional to fast and effective customer support. Modern customer support systems must analyze large amount of textual tickets quickly, organize them into multiple concept levels, and assign them to respective agents for effective resolution.

A ticket may carry issues never seen before, therefore static and predefined problem labels will not always work. These predefined problem labels are often created manually by admins. For a large customer account, roughly 10-35% of the problem labels could get discovered by this manual process and only 5-9% of the available open tickets could get linked to a problem label. So, there is a need of a system where for a new incoming ticket, system should either automatically detect an existing problem topic or systematically generate new problem topic(s) for its assignment.

Our system is based on an incremental approach to mine ticket data streams and assign them to hierarchical soft clusters with auto-generated problem/issue topics. Our system achieves a reasonable increase in problem category discovery with improvement in ticket assignments up to 97%. In order to compare with recent clustering advancements, we've also evaluated our approach on a public dataset [1] having hierarchical categories and achieved comparable results. Source code and results of evaluation are available in detail at: https://github.com/vinidixit/hierarchical-labelled-clustering-evaluation.

CCS CONCEPTS

• Computing methodologies~Artificial intelligence~Natural language processing~Information extraction • Computing methodologies~Machine learning~Learning paradigms~Unsupervised learning~Cluster analysis

[†]Author appreciates and acknowledges her current organization Freshworks to support her personal research interests while maintaining data privacy.

WOODSTOCK'18, June, 2018, El Paso, Texas USA

KEYWORDS

Customer Support Systems, Hierarchical Clustering, Hierarchical Labeling, Topic Detection, Outlier Detection, Frequent Itemsets, Embeddings, Incremental Systems, Microtext

ACM Reference format:

FirstName Surname, FirstName Surname and FirstName Surname. 2018. Insert Your Title Here: Insert Subtitle Here. In *Proceedings of ACM Woodstock conference (WOODSTOCK'18). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

1 INTRODUCTION

Ticket data is an example of Microtext [2] where relevant content gets presented in limited words (4-5 sentences on average). Such content is short but sufficient to get analyzed and detected for underlying intent of the sender. We have preferred a clustering approach over classification because it doesn't expect any prior fixed list of labels, which is hardly possible to maintain for a domain data of dynamic nature. Our idea is to combine benefits of Conceptual clustering [7] to make more interpretable and soft clusters with the power of embeddings to also achieve higher accuracies and maintain semantic relevance for a domain data corpus. To accommodate dynamic nature of incoming data, we have modified and extended clustering mechanism with outlier processing. An outlier is treated as that temporary state for a document which is unassigned at present due to its unpopularity but may change its state with the next data iteration to become a member of mainstream clusters.

Our efforts have been divided into 3 parts –

1.1 Feature Processing – Declutter the noisy ticket data and extract relevant Microtext from it. Apply an NLP pipeline to extract linguistically relevant and unique *phrase* tokenized documents. And, then systematically group semantically identical or similar features by using their embeddings from Fasttext[1] and Bert[2] [8] (transformers fine-tuned on similarity tasks).

1.2 Hierarchical clustering with labels and outliers – Applied Generalized Closed Frequent Itemset based hierarchical clustering to get overlapping but limited and interpretable cluster assignments for documents. The approach and implementation of building Frequent Pattern graph and detect mappings between topic space and document clusters are inspired from [6]. The base approach suggests the mapping in the terminologies where a

[1] https://fasttext.cc
[2] https://github.com/UKPLab/sentence-transformers

document is analogous to a *transaction* and keywords in the documents as *items*. We have chosen TF-IDF scores to select top features from the documents, which has also been a popular choice among various practitioners [6, 7, 9] in the past. Along with that, label assignment scores are also built for a document to choose best topic(s) for a document and use it vice versa to identify an outlier document by a singleton label. And, finally, get grammatically meaningful label names for selected label nodes of the final cluster graph (it won't be a tree because there are multiple and varying number of parent nodes (assignments) for a document at multiple levels).

1.3 Evaluation setup for hierarchical clusters and outliers with overlapping labels – Evaluation metrics for clusters are divided into two parts a. Cluster quality check by calculating document assignment percentage with positively higher label scores and, b. Testing against true labels by generating pairwise constraints similar to how are used in [9] during their experiments. To evaluate outliers, test corpus is selected based on the unpopular categories from historical data. Threshold value for this selection is proportional to the clustering selection thresholds and is already set during step 1.2. Obviously, they both are contrasting and work at opposite extremes of metrics. So, the objective of the evaluation system is to select the hyperparameters that would collectively optimize clustering and outlier prediction quality.

2 RELATED WORK

Clustering and Topic analysis. For topic analysis with documents with mixture of topics, conventionally topic modelling techniques like allocation (LDA) are usually the choice. But, LDA is incapable to be applied on short documents [4]. In the direction of clustering, there have been multiple algorithms and techniques. Although standard clustering techniques such as k-means are proven to be insufficient for soft clustering, getting good accuracies for high dimensional and volume of data. Then to overcome these limitations, conventional hierarchical clustering approaches such as Agglomerative clustering are used, but again they need user to specify certain input parameters like number of clusters that is not applicable for clustering with dynamically changing and uncertain number of clusters. Also, other requirements like getting meaningful labels, soft clustering and keep provision of outliers instead of hard assigning a document. To accommodate above requirements, a Frequent Itemset-based hierarchical clustering fits to a great extent as a choice for interpretable clustering, as explained in [3]. They also systematically succeed to exclude unassignable documents from any cluster and keep them unassigned until they find their match with a certain confidence score. Though there again have been some limitations to use it as is like limited vocabulary, lack of semantic relevance, less recall etc. (a detailed comparison study available at [5, 7]).

Semantic Relevance and Embeddings. Some of the later research work had proposed to use external knowledgebases for semantic relevance [6], that elevated accuracy scores higher. But again, this technique is limited to the availability of updated external knowledgebase periodically, which is even more impractical to have in case of custom domain data like customer tickets. Recent progress in the embeddings and transfer learning in NLP has overcome these limitations by including word embeddings as features by [10] which is also using Bert Token embeddings to extract ranked keyphrases from the text and by [11] using word embeddings for a clustering task.

Labeling and Evaluations. For hierarchical labeling several approaches have been used like in [12] document frequencies and TF-IDF scores are used to build statistics to select labels in hierarchical clusters. To evaluate hierarchical clusters [13] has explained and designed multiple metrics to optimize the correct linkages in a hierarchical cluster. Though, when applied in overlapping clusters, gives irregular results due to finding a document match multiple time. Recently, distance metric learning [14] and triplet approach in constrained based learning have been advantageously used for clustering tasks [9]. In this paper, we explore some of these approaches that worked best in different setups. It could be for getting semantically close feature clusters and building up evaluation setup effectively using pairwise constraint approach for overlapping document clusters.

3 PROPOSED SYSTEM

A typical ticket document comes with a subject and a description where the sender conveys its issues or requests. Figure 1 explains the key steps for discovery of issue categories with ticket linking for effective assignment of agents. It begins with detection of all possible topics and linking of tickets to them using our model. In order to get a reasonable size of assignments and get distinct issue categories, we apply an embedding-based grouping method (explained in section 3.2). Tickets are then assigned to agents based on issue categories.

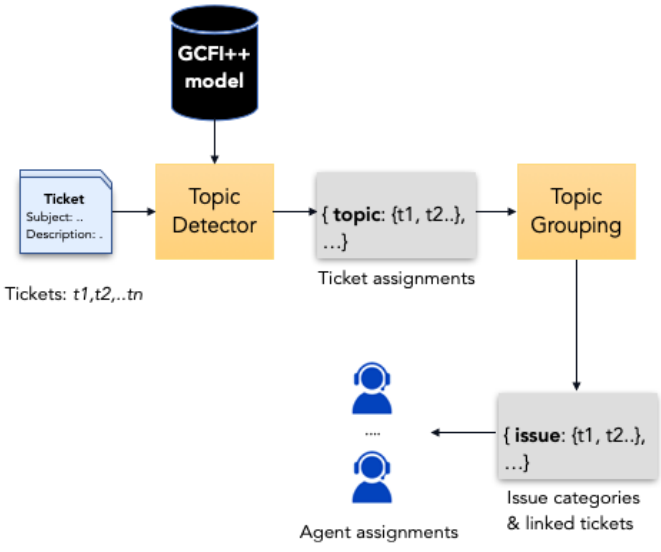


Figure 1: Key steps in the proposed system with our pretrained model

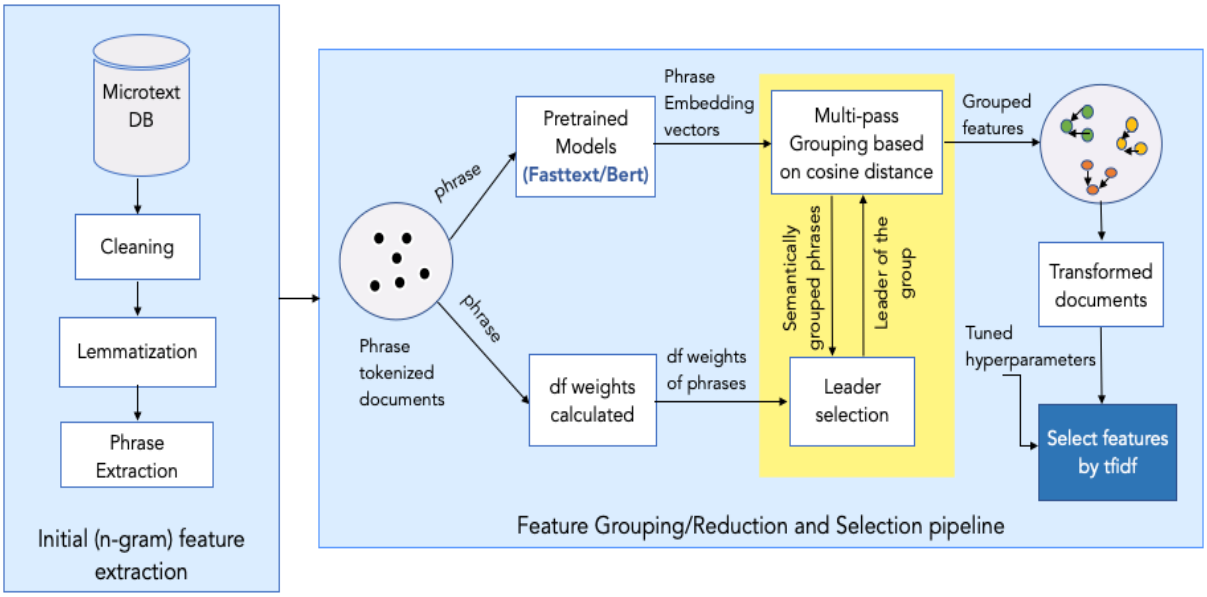


Figure 2: Feature processing pipeline before building Frequent Pattern graph for document clustering

3.1 Initial Feature Processing pipeline

3.1.1. Cleaning - Conversational based data like a ticket or email does not come clean and prepared. It comes with a lot of metadata and irrelevant text like salutations, pleasantries, email footer etc. those instead of adding any signal to textually analyze intent of the ticket rather contributes in building noise and prediction errors.

3.1.2. Lemmatization – To convert tokens to their linguistic source form and select content words by their POS tags (Noun or/and Verb with modifiers like Adjective, Preposition and Adverb). It helps in discarding documents by identifying them as noisy because those had no content words to add any information.

3.1.3. Phrase Extraction – In our context, phrases are the n-grams of sizes from 1 to n. A higher order n-gram is constructed when its probability of occurrence as one term is significantly higher than its occurrence as individual tokens independently. Our approach is based on phrase-based features instead of word tokens. It is often necessary to use phrase-based features because a word changes its meaning completely when it becomes a part in a phrase. For example, *states* and *united* independently mean different from when are used together and in a certain order. We've extracted phrases by approach used in [16]. It is represented by a recurrence function to construct an n-gram confidence score from (n-1)-gram tokenized corpus from n-1th pass.

$$score(ngram) = \frac{(ngram_{count} - minCount) * len_{vocab}}{(n\_1gram\_a_{count} * \ n\_1gram\_b_{count})}$$

where, $ngram_{count}$ represents count of current n-gram candidate joined by n-grams extracted from previous pass: n_1gram_a and n_1gram_b. Counts of joining n-grams from n-1th pass are represented by $n\_1gram\_a_{count}$ and $n\_1gram\_b_{count}$ respectively. $minCount$ stands for minimum collocation count threshold (a tunable parameter) and len_{vocab} stands for the size of vocabulary.

3.2 Feature Grouping and Selection pipeline

3.2.1. Feature Embedding Clustering – Terms or items are the key building block in Itemset based clustering methods where each node is identified by a unique itemset (group of selected items). And, using such unique unigram tokens as is leads to formation of multiple nodes representing the same and hence a duplicate concept in form of either synonyms or paraphrases. Therefore, there is a need to process candidate items before itemset formation. We have taken a bottom-up approach to avoid such occurrences and enhance individual itemset node scores in Frequent Pattern (FP) graph to take part in document clustering. This elevates the number of quality features to get selected and participate in document assignment coverage as well (Figure 5 in results section).

The clustering algorithm for grouping similar features is based on iteratively grouping closer phrases in embedding space. Number of passes is a hyper-parameter with minimum similarity thresholds assigned for each pass in the descending order. In our experiments, we chose up to 4 passes with thresholds between 0.8 to 0.5 inclusive. We have explored and developed phrase groups with Fasttext and Bert based embedding vectors.

$$phrase2vec(p)_{fasttext} = \frac{\sum_{w \in p} w2v_{fasttext}(w)}{phrase\_length}$$

where, $phrase2vec(p)_{fasttext}$ denotes vector representation of a phrase p using Fasttext word embeddings which is represented by

averaging Fasttext word embeddings of word tokens present in the phrase.

In some cases, phrase embeddings without word order and contextual awareness lead to incorrect and unexpected results. For example, "*software license*" and "*driving license*" or "*river bank*" and "*Citi bank*" get unexpectedly closer embeddings by Fasttext phrase vector representation, where added context is insufficient to change sense direction of the resultant vector in both the examples.

To get contextualized phrase embeddings, we explored BERT [17] based models and settled with SBERT [8] model. Sentence-BERT (SBERT) model is a modification of the pretrained BERT network that use Siamese and triplet network trained on language inferencing and semantic textual similarity task to achieve semantically meaningful embedding vector of a sentence (any ordered multi-word structure). We used these pretrained transformers to encode phrases to get their semantic vector representation and overcame errors caused by Fasttext embedding. We used cosine similarity score as a distance metric to measure belongingness of a phrase p_1 in a cluster with phrase p_2.

$$CosSim(p_1, p_2) = \frac{p_1 . p_2}{||p_1|| ||p_2||}$$

We used document frequency (df) scores of phrases in the corpus to select a representative or a leader in a feature group. This leader represents the underlying sense among members of the group. We added two level group membership check, firstly by matching the member candidate with leader of the existing group, which in case is passed then get checked with existing members of the group. This procedure pass through predefined number of iterations and in each pass number of singular and ungrouped features reduces based on similarity thresholds chosen in each pass. During postprocessing, still unassigned singletons get connected to their matching by leader candidate (if available) by making separate groups.

Algorithm 1 Embedding based Feature grouping

Require: phrase vectors from vocab, number_of_passes, leader and member cosine similarity thresholds for each pass
return List of feature groups

```
phrases=sorted(phrases) ▷ ordered in decreasing df scores
for pass ← 1 to number_of_passes do
  for p ∈ phrases(unassigned) do
    leaderCandidates = getNearestNeighbors(
                        p,phrases,leaderThreshes[pass])
    for lc ∈ leaderCandidates do
      membershipCheck = memberSanityCheck(
          p, members(lc), memberThreshes[pass])
      if membershipCheck is True
        leader = leaderSelection(lc, p)
        add p in group of lc
        update leader of the group
  end pass
postprocessing step
```

In algorithm 1, getNearestNeighbors method searches for all the grouping candidates those pass through a minimum threshold given in leaderThreshes for that pass. After that, it checks for membership criteria for pre-existing groups. memberSanityCheck method ensures that current candidate is eligible to become part of existing group by checking its cosine scores with the members of the group with threshold given in memberThreshes for that pass.

Table 1: Example results of feature groups using SBERT embedding on rcv1 data

leader	members
access issue	login password, account locked
information request	information enquiry, asking for details
several	many, multiple, numerous
explain	describe, summarize, understand
disconnected	alienate, aloof, isolate, unmoved, unfazed, unconcerned
technician	mechanic, tech expert, technical analyst

3.2.2. Feature Selection – Ranking and importance of features for a document is calculated by TF-IDF scores from phrase tokenized and transformed documents with sense leaders (processed from previous step). We combine the term frequency and inverse document frequency to produce composite score of a term t in document d is given by:

$$tfidf(t, d) = TF_{t,d} \times IDF_t$$

where $TF_{t,d}$ is term frequency score of term t in document d and IDF_t is inverse document frequency score of t in the corpus.

3.3 Hierarchical clustering

3.3.1. Frequent Pattern Graph Construction – Frequent Itemset-based document clustering approaches have an intuitive and interpretable way of representing document clustering and topic detection as the dual problems [6].

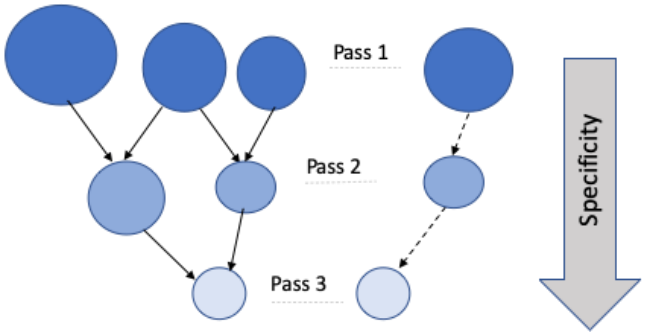


Figure 3: Frequent Itemset graph generated till level 3. As we move from top to down, specificity of cluster assignment to documents increases and number of document assignments per cluster reduces.

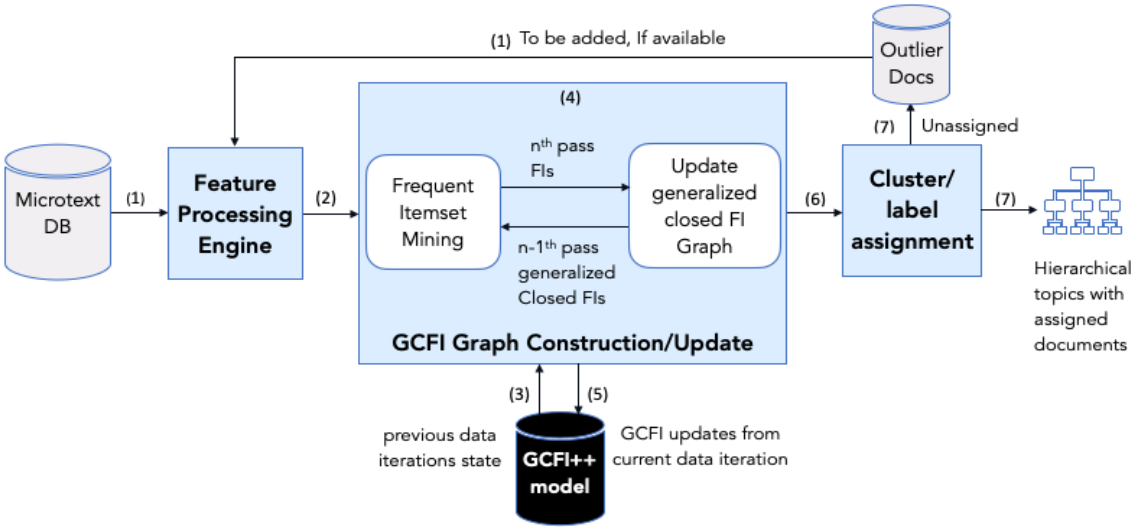


Figure 4: Flow diagram for building GCFI++ cluster model iteratively for document data streams

To construct the frequent pattern graph and mine frequent itemsets from it, we first apply a simple frequent itemset mining algorithm Apriori on all the documents in the first pass. For further passes and higher levels of itemset generation we choose a frequent itemset node based on *generalized closed frequent itemset* property [18] to avoid redundant nodes in the graph. We used approach explained in [6] to select best support thresholds defined for each level and max_dup parameter for maximum allowed overlap in the clusters. In our system, we generated FI[1] graph till 5 levels with thresholds proportional to corpus type - short/long (see experiments section).

Definition 1 Topic/Label/Itemset Node. Itemset node and label or topic node are different states of a node in an FI graph. Itemset is the first state that gets created while FI graph generation. Label node is the term used for those FI nodes that pass through certain criteria of selection and are treated as underlying topic for documents assigned to that node. Cluster is interchangeability used with label node to represent group of documents those got assigned to this label node.

3.3.2. Label Scores and Document assignments - Measurement of assignment quality of document to a certain topic node in above generated graph is based on its *matching* or *coverage* score with that topic. In other words, such assignment score must quantify the measure of confidence for a certain topic node to be represented as a shorthand label for that document. In this case, higher score conveys stronger representation and better cluster assignment quality. Our idea is based on descriptive scoring which is used in various forms for labeling tasks [12].

Definition 2 labelScore. A measurement of document assignment to an itemset node. This score is applied on top of the selected terms (by tfidf) of a Microtext document and the itemset node under consideration. When an itemset node is selected as a topic for the document by this score, it changes its state to a label node. We can formulate this score by using equation 1. We've chosen tf scores to favor popular terms shared across multiple documents for topic building. α, β and γ are tunable multipliers to prefer one weight term over another. In our experiments, we chose their values as 1.

$$labelScore(itemset_{node}, terms_{document}) = \alpha\, weight_{match} - \beta\, weight_{remaining} - \gamma\, weight_{extra} \quad (1)$$

labelScore favors frequent terms over rare terms and is a weighted sum of *matched* terms, *remaining* unmatched terms in the document and *extra* irrelevant terms that come on choosing current itemset node as a label node. Clearly, it penalizes these extra irrelevant terms more than the remaining document terms which can get matched by higher level nodes. *remaining* weight expression (3) ensures that a higher level (more specific) node with more coverage gets higher matching score if gets compared in the options.

Weight scores for each expression are calculated as below –

$$weight_{match} = |match| \times \sum_{term \in match} \log(tf(term)) \quad (2)$$

$$weight_{remaining} = |remaining| \times \sum_{term \in remaining} \log(sqrt(tf(term))) \quad (3)$$

$$weight_{extra} = |extra| \times \sum_{term \in extra} \log(tf(term)) \quad (4)$$

[1] In this paper, FI is used as an abbreviation for Frequent Itemset for better readability

Hypotheses 1. labelScore scoring function quantifies the confidence of document assignment to a cluster aka label node or itemset node in an FP graph and on using higher scoring cluster assignments by labelScore would improve quality of cluster prediction.

Proof. We'll experimentally prove in the results section that our hypotheses are aligned with our results.

3.4 Hierarchy Generation and Labeling of clusters

We have taken a bottom-up approach to generate a Frequent Itemset graph by only including semantically distinct and relevant features using embeddings and then using only top features per document selected from TF-IDF to take part in the Frequent Itemset graph generation. So, we need not to follow additional steps as postprocessing like in [6] to reduce document duplication later. Rather, we added a new level to score by equation 1 and select effective label nodes and building label names from these nodes.

Label names are generated using n-gram orders of the terms used in the label nodes. The most popular n-gram order is chosen for label name generation.

Algorithm 2 Label Name Generation

Require: query labelNode, n-gram tokenized corpus documents with assigned label nodes hierarchy
return label name for the input cluster key/label node

1: D_L=D(ngrams, labelTree) : labelNode ∈ labelTree
2: freqMap = {}
2: **for** doc ∈ D_L **do**
3: ngrams = getDocumentTokens(doc) ▷ preprocessed
4: L(item, order) = getTermOrder(labelNode.items, ngrams)
5: labelName = transformByOrder(L(item, order))
6: freqMap[labelName] += 1
7: end for
8: select labelName by highest frequency from freqMap

3.5 Outlier Detection

Unassigned documents from our clustering procedure are due to absence of features in the document because of two reasons – 1. features failed to get selected by tfidf, or 2. features not selected by frequent itemset algorithm. Cause of first case is due to weak TF-IDF scores and second case is due to not passing through *support* thresholds which are proportional to the maximum frequent itemset count in 1-itemsets. And, reason behind both the cases is rarity of the document feature terms in the corpus. These terms are singletons and isolated in the feature space those did not get any connect with other features neither *linguistically* nor *semantically*. Hence, we can conclude these rare terms as follows:

$$t\ (term = "rare") = \begin{cases} t \in Noise \\ t \in new\ term \end{cases}$$

In clustering algorithms that do not account for the presence of outliers, they make a hard assignment for above documents to any group out of the available clusters. In such situations, if these documents are weak because they are containing noise, then it exploits the entire cluster quality. In other clustering algorithms where outliers are detected during clustering process like DBSCAN, treat all outliers as noise and re-clustering is the only way to reassign them during the next data update.

Therefore, in our system, we keep a separate state for outlier/unassigned documents to be revaluated to become part of main clusters in the next data iterations. And, it doesn't require to reconstruct the clustering procedure for the previous data, instead it is a matter of new FI updates only. And, if the reason of the un-assignment was due to novelty of document topic, these document topics quickly get the support in the next few data batches and they become part of the main clusters and lose its state from the outliers.

4 EXPERIMENTAL EVALUATION

We have run more than 300 experiments per dataset to select and evaluate best choices and step (components in our multi-step system) parameters. We have setup MLFlow[1] experiments on 17 parameters consisting of - a. *step parameters* (hyperparameters and implementation choices) and, b. *evaluation parameters* to evaluate individual steps and system quality by 35 measures. Source code and evaluation setup used to perform these experiments are available at https://github.com/vinidixit/hierarchical-labelled-clustering-evaluation to help with reproducibility. In our experiments we aim to address below points:

- How size of the document affects quality of frequent itemset based clusters. And, how Microtext documents get more advantageous results than larger size documents using this clustering technique.
- How use of embedding fills the need of semantic relatedness in frequent itemset space and how it takes part in achieving better clustering quality from previous implementations.
- How pairwise constraint approach helped in building evaluation data for overlapping and multi-label assignments effectively.
- How outlier detection is an iterative process rather a static state for a document.

4.1 Datasets

Reuters-10K: In order to compare our clustering results with other implementations on clustering, we built our experiment setup same as used in [9]. We randomly selected a sample of 10000 examples with 4 root categories: corporate/industrial, government/social, markets and economics having sub-categories hierarchy till 5 levels of depth. We maintained two versions of this dataset in our experiments –

- **long** – This is the original version of the documents with headline and text body.
- **short** – Because our system is based on a Microtext corpus and favors short documents, a short version the

[1] https://mlflow.org

documents was created for the comparisons including the headlines.

Tickets Problem Dataset (TPD): This dataset was created using ~7k tickets with subjects and descriptions. These tickets were previously linked by the system admin with 38 distinct problem labels.

4.2 Evaluation Measures

We generated 10000 pairs of instances with *must-link* to same cluster(s) and *cannot-link* to any common cluster, termed as positive pairs and negative pairs. If $label(d_i)$ is the set of true labels d_i is assigned to in the dataset, then link between pair of documents d_i and d_j w.r.t. true labels is defined as follows:

$$link(d_i, d_j) = \begin{cases} pos, & if\ label(d_i) \cap label(d_j) \neq \phi \\ neg, & if\ label(d_i) \cap label(d_j) = \phi \end{cases}$$

We evaluated clustering quality on mainly two criteria -

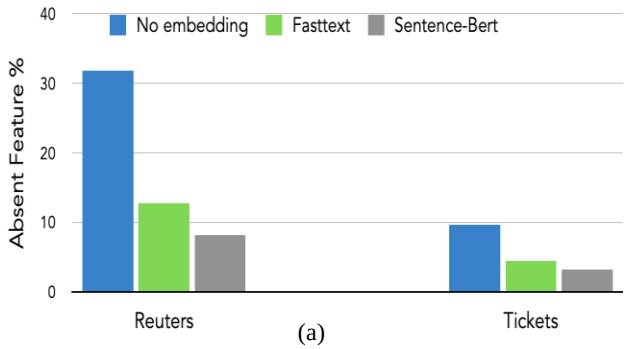
a. Document Assignment quality: document assignment to any cluster depends upon presence of features in it, to be able to participate in cluster assignment process, which in turn also affects possibility of cluster(s) assignment to that document.

b. Cluster prediction quality: Let (d_i, d_j) be a pair of documents in the collection of generated evaluation pairs. If $link_{cluster}(d_i, d_j)$ is the predicted link and $link_{label}(d_i, d_j)$ is the true link between the document pair, the truth function for two documents d_i and d_j can be defined as follows:

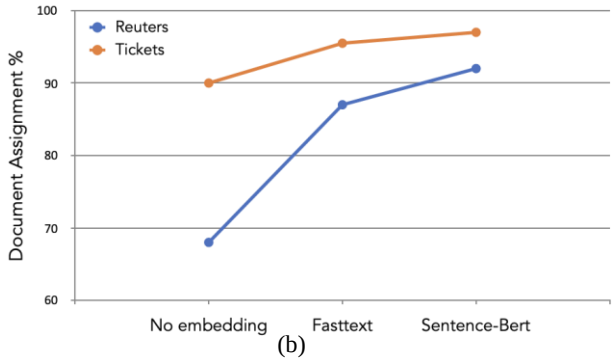
$$truth(d_i, d_j) = \begin{cases} 1, if\ link_{label}(d_i, d_j) = link_{cluster}(d_i, d_j) \\ 0, \quad otherwise \end{cases}$$

5 RESULTS

In Figure 5, we show comparison of various feature processing techniques and their effect on feature selection and document assignment. Here, we can clearly see that embedding options are outperforming in comparison to basic version (No-embedding) for both the corpuses. For TPD corpus, our approach reported up to 97% of ticket assignment to at least one problem category.



(a)



(b)

Figure 5: Comparing different feature processing options with Reuters-short and Tickets (TPD) corpus on (a) availability of features in the documents and, (b) document assignments in the clusters.

In Table 2, we have taken scores from baseline algorithm GCFI[1] [6] and a recent clustering approach DCC [9] and made a comparison with our approaches (GCFI++) for Reuters dataset [1]. In our basic version, we differ from GCFI [6] by phrase-based preprocessing step with use of labelScore (Eq.1) for cluster matching and assignment scores.

Table 2: Metrics comparison across various methods (under study) for Reuters dataset

	GCFI*	DCC	GCFI*++ (Ours)		
			basic	+Fasttext	+SBERT
F-score	0.701	-	0.836	0.868	0.893
Accuracy	-	[0.78, 0.95]	0.824	0.856	0.881
Specificity	-	-	0.901	0.952	0.983

* Implementations deliver hierarchical clustering of documents with multilevel topic detection.

In Table 3, we have reported results for a Ticket dataset (TPD) collected by our model with experimentally selected hyperparameters and implementation choices across various steps in our system. We have compared our results against two implementation choices using labelScore. In the first option, there is no filter or selection made by using labelScore metric and in the second row, we have selected only high scoring (positive) cluster assignments for documents.

Table 3: Result comparison wrt choice of label filter on TPD dataset

Label-filter	Accuracy	Precision	Specificity
None	0.781	0.746	0.848
Positive only	0.797	0.834	0.957

Reported results indicate their alignment with the hypothesis (1) and therefore we can conclude that using labelScore during clustering assignment improves clustering quality.

From point of view of customer support, our system could discover 102 broad and distinct problem categories. This resulted in better coverage for earlier unassigned tickets reported by

[1] Abbreviated Generalized Closed Frequent Itemset based Clustering with GCFI for better readability

previous manually generated 38 labels for account used in the experiments section.

Final grouping of topics to get agent assignable problem categories is based on Algorithm 1. This grouping method is flexible (by changing number of passes) to get a reasonable number of problem categories according to capacity of agents in the system.

6 CONCLUSION

We have, in this work proposed a solution that deals with data streams and mine them for dynamic topic discovery with soft assignment of documents in hierarchies. We have shown potential of itemset mining approach to get interpretable clusters with topics for a Microtext corpus. Strength of phrase embeddings is utilized twice – i) to get meaningful features for itemset mining and, ii) while compressing topics to get agent assignable problem categories. By including outlier state in the system, we are neither accepting hard assignments for weak matches nor discarding them for good, rather keeping them for possible support in next data passes. An interesting future direction would be to explore nature of outliers during further data iterations.

ACKNOWLEDGMENTS

The author is grateful towards her post-grad institute IIIT Hyderabad, India and its professors for giving incredible guidance and mentorship in building a broad viewpoint towards research and data mining to apply in industry problems with relevance.

REFERENCES

[1] Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. In: JMLR, 2004.

[2] Ellen, Jeffrey. (2011). All about Microtext - A Working Definition and a Survey of Current Microtext Research within Artificial Intelligence and Natural Language Processing.. 1. 329-336.

[3] M, Benjamin & Ke, Fung & Ester, Wang. (2003). Hierarchical Document Clustering Using Frequent Itemsets. 10.1137/1.9781611972733.6.

[4] Trinh, Trung & Quan, Tho & Mai, Trung. (2019). Nested Variational Autoencoder for Topic Modeling on Microtexts with Word Vectors.

[5] http://www.ijesrt.com/issues%20pdf%20file/Archive-2016/November-2016/51.pdf

[6] R, Kiran & Shankar, Ravi & Pudi, Vikram. (2010). Frequent Itemset Based Hierarchical Document Clustering Using Wikipedia as External Knowledge. 6277. 11-20. 10.1007/978-3-642-15390-7_2.

[7] Pérez-Suárez, A., Martínez-Trinidad, J.F. & Carrasco-Ochoa, J.A. A review of conceptual clustering algorithms. *Artif Intell Rev* **52,** 1267–1296 (2019). https://doi.org/10.1007/s10462-018-9627-1

[8] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks arXiv:1908.10084 **[cs.CL]**

[9] Zhang, Hongjing & Basu, Sugato & Davidson, Ian. (2020). A Framework for Deep Constrained Clustering - Algorithms and Advances. 10.1007/978-3-030-46150-8_4.

[10] Si, Sun & Xiong, Chenyan & Liu, Zhenghao & Liu, Zhiyuan & Bao, Jie. (2020). Joint Keyphrase Chunking and Salience Ranking with BERT.

[11] Agarwal, Lucky et al. "Authorship Clustering using TF-IDF weighted Word-Embeddings." *FIRE '19* (2019).

[12] Treeratpituk, Pucktada and James P. Callan. "Automatically labeling hierarchical clusters." *DG.O* (2006).

[13] Zhao, Y., Karypis, G. & Fayyad, U. Hierarchical Clustering Algorithms for Document Datasets. *Data Min Knowl Disc* **10,** 141–168 (2005).

[14] Movshovitz-Attias, Yair et al. "No Fuss Distance Metric Learning Using Proxies." *2017 IEEE International Conference on Computer Vision (ICCV)* (2017): 360-368.

[15] Mikolov, Tomas & Grave, Edouard & Bojanowski, Piotr & Puhrsch, Christian & Joulin, Armand. (2017). Advances in Pre-Training Distributed Word Representations.

[16] Mikolov, Tomas & Sutskever, Ilya & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems. 26.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[18] Pudi, V., Haritsa, J.R.: Generalized Closed Itemsets for Association Rule Mining, In Proc. of IEEE Conf. on Data Engineering. (2003)

[19] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A com- parison of extrinsic clustering evaluation metrics based on formal constraints. Information retrieval 12, 4 (2009), 461–486.