Collaboratively Tracking Interests for User Clustering in Streams of Short Texts

Parul Rathour, Pallavi Bansal and Gaurav Raj
Department of Computer Science and Engineering
Jaypee University, Anoopshahr, U.P.

Abstract— User clustering has been studied from different angles: behavior based, to identify similar browsing or search patterns, and content based, to identify shared interests. User clustering adaptively tracks changes of each user’s time-varying topic distribution based both on the tweets the user posts during a given time period. For evaluation purposes, we work with a dataset consisting of users and tweets from each user. Experimental results demonstrate the effectiveness of our proposed clustering model compared to state-of-the-art baselines.

Keywords—Clustering, Streaming Text, Twitter.

1. INTRODUCTION

In this paper, we study the problem of user clustering in the context of short text streams, where users are taken to be people who post messages on a micro blogging platform(Twitter). Our goal is to infer users’ topic distributions over time and dynamically cluster users based on their topic distributions in such a way that users in the same cluster share similar interests while users in different clusters differ in their interests. In addition, we aim at making the clustering results explainable and understandable. We focus on clustering users at a certain point in time, in the context of streams of short documents.

.

2. CLUSTERING

Clustering/segmentation is one of the most important techniques used in Acquisition Analytics. K means clustering groups similar observations in clusters in order to be able to extract insights from vast amounts of unstructured data. The basic idea of K Means clustering is to form K seeds first, and then group observations in K clusters on the basis of distance with each of K seeds. The observation will be included in the nthseed/cluster if the distance between the observation and the nth seed is minimum when compared to other seeds. When you want to analyze the Facebook/Twitter/Youtube comments of a particular event, it would be impossible to manually look at each and every mention and see where the sentiment regarding a particular brand/event/person lies.

The Process of building K clusters on Social Media text data:

- The first step is to pull the social media mentions for a particular timeframe using social media listening tools (Radian 6, Sysmos, Synthesio etc.). You would need to build query/add keywords to pull the data from social Media Listening tools.
- The next step is data cleansing. This is the most important part as social media comments do not have any specific format. People use locals/slangs etc. on social media to express their emotions, so it's important to be able to see through them and understand the underlying sentiment. Remove punctuations, numbers, stopwords (R has specific stopword library but you can also create your own list of stopwords). Also, remove duplicate rows or URLs from the social media mentions.
- The next step is to create corpus vector of all the words.
- Once you have created the corpus vector of words, the next step is to create a document term matrix.

3. TEXT CLUSTERING

Organizations today are sitting on vast heaps of data and unfortunately, most of it is unstructured in nature. There is an abundance of data in the form of free flow text residing in our data repositories. While there are many analytical techniques in place that help process and analyze structured (i.e. numeric) data, fewer techniques exist that are targeted towards analyzing natural language data. In order to overcome these problems, we will devise an **unsupervised** text clustering approach that enables business to programmatically bin this data. These bins themselves are programmatically generated based on the algorithm’s understanding of the data. This would help tone down the volume of the data and understanding the broader spectrum effortlessly. The algorithm first performs a series of transformations on the free flow text data (elaborated in subsequent sections) and then performs a k-means clustering on the vectorized form of the transformed data. Subsequently, the algorithm creates cluster-wise tags, also known as cluster-centers, that are representative of the data contained in these clusters.

The solution boasts of end-to-end automation and is generic enough to operate on **any dataset**.

The text clustering algorithm works in five stages enumerated below:-

- Transformations on raw stream of free flow text
- Creation of Term Document Matrix
- TF-IDF (Term Frequency – Inverse Document Frequency) Normalization
- K-Means Clustering using Euclidean Distances Auto-Tagging based on Cluster Centers.

The free flow text data is first curated in the following stages:-

Stage 1

Removing punctuations and transforming to lower case

Grammatically tagging sentences and removing pre-identified stop phrases

Removing numbers from the document and stripping any excess white spaces

Stage 2

Removing generic words of the English language viz. determiners, articles, conjunctions and other parts of speech.

Stage 3

Document Stemming which reduces each word to its root using Porter's stemming algorithm.

Once all the documents in the corpus are transformed as explained above, a term document matrix is created and the documents are transformed into this vector space model using the 1-gram vectorizer.

A. TD-IDF(Term Frequency Inverse Document Frequency)Normalization

This step and can be performed in case there is high variability in the document corpus and the number of documents in the corpus is extremely large (of the order of several million). This normalization increases the importance of terms appearing multiple times in the same document while decreasing the importance of terms that appear in many documents (which would mostly be generic terms). Provide the TF-IDF transformation, the document vectors are put through a **K-Means clustering algorithm** which computes the **Euclidean Distances** amongst these documents and clusters nearby documents together.

Term Frequency (TF) = (Number of times appears in a document /(Number of terms in the documents).

Inverse Document Frequency (IDF) = log(N/n), where, N is the number of documents a term t has appeared in. The IDF of a rare word is high, whereas the IDF of a frequent word is likely to be low. Thus having the effect of highlighting words that are distinct.

TF-IDF value of a term as = TF * IDF.

B. Euclidean Distance

It is just a distance measure between a pair of samples *p* and *q* in an *n*-dimensional feature space:

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

The Euclidean is often the "default" distance used in e.g., K-nearest neighbors (classification) or K-means (clustering) to find the "k closest points" of a particular sample point. Another prominent example is hierarchical clustering, agglomerative clustering (complete and single linkage) where you want to find the distance between clusters.

C. Gibbs Sampling

Gibbs sampling is a Markov Chain Monte Carlo (MCMC) algorithm where each random variable is iteratively re-sampled from its conditional distribution given the remaining variables. It's a simple and often highly effective approach for performing posterior inference in probabilistic models.

Gibbs sampling starts with some x and then repeats, choose a variable j uniformly at random. Update xj by sampling it from its conditional, xj ∼ p(xj |x−j). **Analogy**: sampling version of coordinate optimization: Transformed d-dimensional sampling into 1-dimensional sampling. Gibbs sampling is probably the most common multi-dimensional sample. For UGMs these conditionals needed for Gibbs sampling have a simple form,

$$p(x_j = c|x_{-j}) = \frac{p(x_j = c, x_{-j})}{\sum_{x_j=c'} p(x_j = c', x_{-j})} = \frac{\tilde{p}(x_j = c, x_{-j})}{\sum_{x_j=c'} \tilde{p}(x_j = c', x_{-j})},$$

because the Z is the same in the numerator and denominator terms. And UGMs it further simplifies due to the local Markov property,

$$p(x_j|x_{-j}) = p(x_j|x_{\mathsf{MB}(j)}).$$

Thus these iterations are very cheap.

D. Data Processing

- Tokenization—convert sentences to words
- Removing unnecessary punctuation, tags
- Removing stop words—frequent words such as "the", "is", etc. that do not have specific semantic
- Stemming—words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix.
- Lemmatization—Another approach to remove inflection by determining the part of speech and utilizing detailed database of the language.

Word2Vec is a group of models which helps derive relations between a word and its contextual words. Two important models inside Word2Vec: Skip-grams and CBOW. One of the simplest techniques to numerically represent text is **Bag of Words. Bag of Words (BOW):** We make the list of unique words in the text corpus called vocabulary. Then we can represent each sentence or document as a vector with each word represented as 1 for present and 0 for absent from the vocabulary. One of the major disadvantages of using

BOW is that it discards word order thereby ignoring the context and in turn meaning of words in the document.

General-purpose emotion lexicons (GPELs) that associate words with emotion categories remain a valuable resource for emotion detection. However, the static and formal nature of their vocabularies make them an inadequate resource for detecting emotions in domains that are inherently dynamic in nature. This calls for lexicons that are not only adaptive to the lexical variations in a domain but which also provide finer-grained quantitative estimates to accurately capture word-emotion associations. The lexicon is the bridge between a language and the knowledge expressed in that language. Every language has a different vocabulary, but every language provides the grammatical mechanisms for combining its stock of words to express an open-ended range of concepts. Different languages, however, differ in the grammar, the words, and the concepts they express. Grammars and words belong to the province of linguistics, but the concepts they express belong to the extra-linguistic knowledge about the world. For each language, the lexicon must provide the links that enable a language processor to carry messages from one province to the other.

4. Dataset

We work with a dataset collected from Twitter.[1] The dataset contains 1,375 active users and their tweets spanning a particular time period. Most of the users are being followed by 2 to 50 followers. In total, there is millions of tweets with timestamps. The average length of a tweet is 12 words. The dataset contains 30322 tokenized entries.

5. Evaluation Metrics and Settings

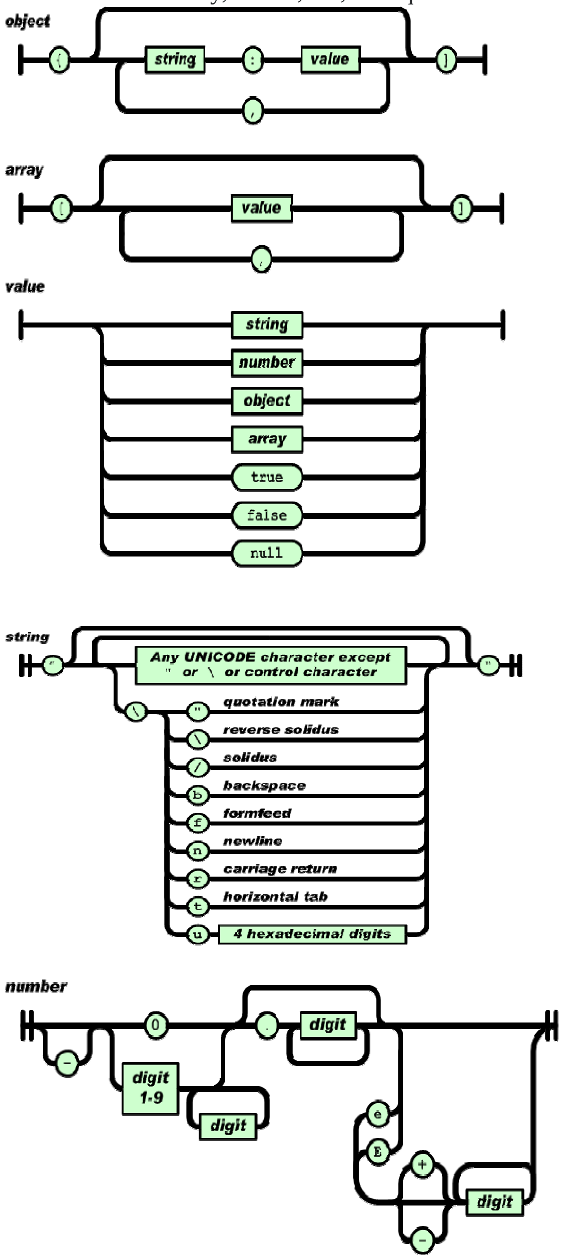
We use Precision, Purity, NMI (Normalized Mutual Information), and ARI (Adjusted Rank Index) to evaluate the performance of user clustering, all of which are widely used in the literature [44]. Higher Precision, Purity, NMI scores indicate better user clustering performance. **Normalized Mutual Information** is a good measure for determining the quality of clustering. It is an external measure because we need the class labels of the instances to determine the NMI. Since it's normalized we can measure and compare the NMI between different clustering having different number of clusters. The **adjusted** Rand **index** has the maximum value 1, and its expected value is 0 in the case of random **clusters**. A larger **adjusted** Rand **index** means a higher agreement between two partitions. The **adjusted** Rand **index** is recommended for measuring agreement even when the partitions compared have different numbers of **clusters**.

6.

A. JSON

JSON (JavaScript Object Notation) is a lightweight format that is used for data interchanging. It is based on a subset of JavaScript language (the way objects are built in JavaScript).JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, structure, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.



B. COLLECTION

First up we have the collections module. If you've been working with Python for any length of time, it is very likely that you have made use of the this module; however, the batteries contained within are so important that we'll go over them anyway, just in case.

Being able to use attributes to access data inside the tuple is much safer rather than relying on indexing alone; if future changes in the code added new fields to the named tuple, the tup.count would continue to work.

The collections module has a few other tricks up its sleeve, and your time is well spent brushing up on the documentation. In addition to the classes shown here, there is also a Counter class for easily counting occurrences, a list-like container for efficiently appending and removing items from either end (deque), and several helper classes to make subclassing lists, dicts, and strings easier.

C. K-nearest neighbors

In pattern recognition, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

k-NN is a type of instance based learning , or lazy learning , where the function is only approximated locally and all computation is deferred until classification. Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor. The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data.

Parameter Selection The best choice of *k* depends upon the data; generally, larger values of *k* reduces effect of the noise on the classification,[5] but make boundaries between classes less distinct. A good *k* can be selected by various heuristic techniques (hyper-parameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when *k* = 1) is called the nearest neighbor algorithm.

The accuracy of the *k*-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithm to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.

In binary (two class) classification problems, it is helpful to choose *k* to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal *k* in this setting is via bootstrap method.

D. Scikit Learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. Pre-processing refers to the transformations applied to your data before feeding it to the algorithm. In python, scikit-learn library has a pre-built functionality under sklearn.preprocessing. It provides efficient versions of a large number of common algorithms. Scikit-Learn is characterized by a clean, uniform, and streamlined API. The Scikit-Learn API is designed with the following guiding principles in mind

- *Consistency*: All objects share a common interface drawn from a limited set of methods, with consistent documentation.
- *Inspection*: All specified parameter values are exposed as public attributes.
- *Limited object hierarchy*: Only algorithms are represented by Python classes; datasets are represented in standard formats (NumPy arrays, Pandas DataFrames, SciPy sparse matrices) and parameter names use standard Python strings.
- *Composition*: Many machine learning tasks can be expressed as sequences of more fundamental algorithms, and Scikit-Learn makes use of this wherever possible.
- *Sensible defaults*: When models require user-specified parameters, the library defines an appropriate default value.

E. NLTK(Natural Language Tool Kit)

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. It guide through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure.

7. SILHOUETTE SCORE

Silhouette analysis can be used to study the separation distance between the resulting clusters.This measure has a range of [-1, 1].Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

8. BIC SCORE

In statistics, the **Bayesian information criterion** (**BIC**) or **Schwarz information criterion** (also **SIC**, **SBC**, **SBIC**) is a criterion for model selection among a finite set of models; the model with the lowest BIC is preferred. The BIC is formally defined as

BIC = ln(n)k -2ln(L^)

Characteristics of the Bayesian information criterion

i. It is independent of the prior or the prior is "vague" (a constant).
ii. Measures the efficiency of the parameterized model in terms of predicting the data.
iii. It penalizes the complexity of the model where complexity refers to the number of parameters in model.
iv. It is approximately equal to the minimum description length criterion but with negative sign.
v. It can choose the number of clusters according to the intrinsic complexity present in a particular dataset.
vi. It is also related to other penalized likelihood criteria such as RIC and the Akaike information criterion (AIC).

BIC has been widely used for model identification in time series and linear regression. It can be applied quite widely to any set of maximum likelihood-based models. In many applications ,for example, selecting a black body or power law spectrum for an astronomical source, BIC simply reduces to maximum likelihood selection because the number of parameters is equal for the models of interest.

TABLE I. **ABBREVIATION**

+ NOTATIONS	**NOTATIONS USED IN BIC SCORE**
	GLOSS
L^	Maximum value of likelihood function of the model
x	The observed data
n	The number of data points in x,the number of observations,or equivalently the sample size.
k_k	. The number of parameters estimated by the model.

9. CONCLUSION

In this paper, we studied the problem of dynamically clustering users in the context of streams of short texts. Our models can effectively handle both the textual sparsity of short documents, and the dynamic nature of users' and their followers' interests over time. To effectively infer users' dynamic interests, we proposed two collapsed Gibbs sampling algorithms for the two collaborative interest tracking topic models. We evaluated the performance of the proposed models in terms of clustering, topical representation and generalization effectiveness, and made comparisons with state-of-the-art models. Our experimental results demonstrated that the models can effectively cluster users in streams of short texts and tracking users' interests based on their topic distributions at longer previous time periods helps to enhance the clustering performance.

REFERENCES

[1] Shangsong Liang, Emine Yilmaz , Evangelos Kanoulas, "Collaboratively Tracking Interests for User
Clustering in Streams of Short Texts."

[2] Shangsong Liang, Emine Yilmaz, Zhaochun Ren, Jun Ma, Maarten de Rijke, "Explainable User Clustering in Short Text"

[3] Python Data Science Handbook by Jake VanderPlas, "Introducing Scikit-Learn".

[4] Vivek Kalyanarangan, "Text Clustering".

[5] Etebong P. Clement, Department of Mathematics and Statistics University of Uyo, Akwa Ibom State, Nigeria, "Using Normalized Bayesian Information Criterion (Bic) to Improve Box - Jenkins Model Building".

[6] Mark Schmidt, University of British Columbia, "CPSC 540: Machine Learning Gibbs Sampling, Variational Inference".

[7] Tita Risueno, Inbound and digital Marketing, "What is the difference between Stemming and lemmatization?"

[8] Javaid Nabi, "Machine learning –Text processing".

[9] Bird, Steven, Edward Loper and Ewan Klein (2009), " Natural Language Processing with Python. O'Reilly Media Inc."