HIGH SCHOOL EDITION
Journal of Student Research
Volume 13 Issue 4 (2024)

Machine Learning for Text Categorization to Enhance Emergency Response Efficiency

Aditya Patra[1] and Professor Arijit Das[#]

[1]California High School, San Ramon, CA, USA
[#]Advisor

ABSTRACT

This research paper attempts to improve the emergency response efficiency of an organization by employing text categorization methods to group related requirements from different departments of the organization. The paper evaluates multiple machine learning-based text categorization methods to find which method can best group requirements from various departments that work towards the same goal allowing departments to synchronize operations and work more efficiently. The two methods that will be reviewed in this paper are text-embedding-based clustering and LDA models. Each data sorting method is trained, if necessary, and tested on a dataset of sample requirements from multiple departments of an organization.

Introduction

When a multidepartment organization responds to an emergency, it is difficult for the departments to work synchronously as each department would follow its own set of protocols leading to tasks being completed in an inefficient manner. The goal of this research paper is to find how the requirements from the protocol of multiple departments can be categorized through machine learning to allow for tasks with a similar end goal to be grouped together. This would allow departments to work in a more collaborative manner, increasing emergency response efficiency.
This research paper compares the process and results for data categorization using text embeddings and an LDA model. Each algorithm is tested on a dataset of requirements dictating the procedure for dealing with emergency situations for different departments of an organization. The requirements are stored in an Oracle database and the results of categorization will be stored and displayed in a 3-D graph. The categorization methods will be graded on how well they can group requirements from multiple departments that work towards the same goal.

Background

Machine learning is a branch of computer science dedicated to creating algorithms that allow computers to automate tasks without explicit instruction. It allows the user to replace millions of lines of code that need to account for every possibility with a simple dataset of sample runs. By analyzing the dataset and detecting patterns, a computer can be taught to find the most efficient way to complete a task. Since its introduction the field of machine learning has grown rapidly and is now an essential component in many aspects of our everyday lives. Prime examples of this are chatbots such as ChatGPT and Perplexity as well as recommendation engines and object detection algorithms.
As the field of machine learning is becoming more and more popular, machine learning technology is being used for a wider range of purposes. One such purpose is text analysis. Although text analysis through machine learning has existed since the 1960s in the form of natural language processors, it only became popular in the 1990s and 2000s. Since then, computer scientists have created various new algorithms for text analysis such as BERT(Bidirectional Encoder Representations from Transformers) embeddings and LDA(Latent Dirichlet Allocation) models. With many

new machine learning-based text analysis technologies being developed, it is imperative to evaluate the different algorithms to find which algorithm best suits our needs.

Dataset

This paper utilizes a dataset of sample requirements for how various departments should respond to emergency situations. For each requirement, the requirement text, number, and organization departments are stored in an Oracle database table.

Categorization Methods Tested

Text Embedding

Text embeddings are a method of finding out how related two sentences or phrases are. First, each sentence is tokenized using a method known as BERT(Bidirectional Encoder Representations from Transformers) which converts phrases to something that is easier for a computer to use. To simplify the text for the machine, BERT converts each word into a set of tokens that each signify a common phrase or root word. In order to use the tokenized sentence, the sentence lengths are normalized by adding as many filler tokens as necessary to the end of each sentence. After normalizing the sentence lengths, the embedded converts each token into a vector using a sentence transformer. This is done by looking at the context of each token. Based on what other token values are surrounding it, it can be assigned different vectors. The vectors of the sentence are then combined into an array. The embedder can then find closely related phrases by comparing the locations of the vector arrays. For example, if there is a 3-word phrase where each word is assigned a token and vector, each of the vectors would be compared with vectors of other phrases to find the phrase with the most similar set of vectors. As the sentence transformer used to vectorize the tokens is pretrained, there are no adjustments that can be made to the algorithm to provide an output more suited to the scenario discussed in this paper.

LDA Model

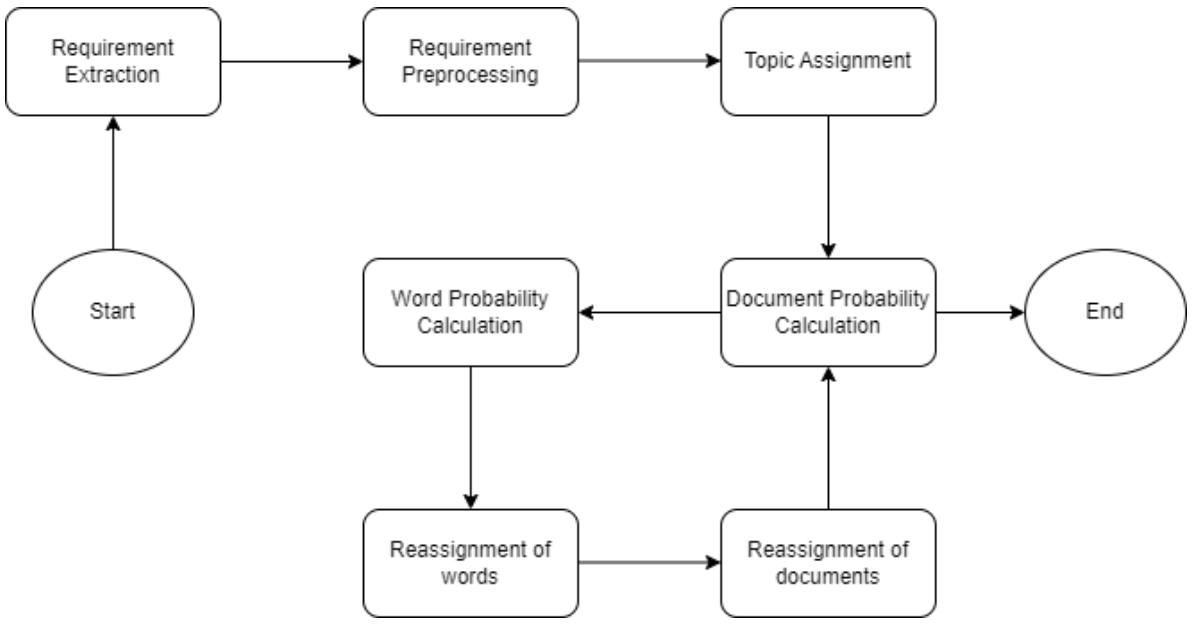

Figure 1. Graph showing flow of LDA model training

LDA Models are a way to find a correlation between documents and their respective topics. In order to do this, the model needs to be fed multiple documents as well as a predetermined number of topics. The model then goes through each document and randomly assigns each unique word to one of the topics. If the same word appears in multiple documents, it can be assigned to multiple topics. During this process, words that contain no value such as the, or, and because are skipped as they cannot be used to classify the document. Once the random assignments are complete, the LDA model cycles through the documents and assigns each document a set of topics that each have a corresponding probability of the document belonging to that topic based on the common words. At this stage, although the classifications of the documents are random, documents with similar subjects should be assigned some of the same topics due to keywords. Once all the documents have an assigned topic, the model then goes through each document and calculates two values.

First, it evaluates the current topic distribution within the document using the following equation:

$$P(\theta_d|\alpha) = \frac{1}{B(\alpha)}\prod_{k=1}^{K}\theta_{d,k}^{a_k-1}$$ where θ_d is a vector representing the topic distribution(probability of each topic belonging to the document) for document d. α is the Dirichlet distribution vector which determines how related a document has to be to a topic to be assigned to the topic. Smaller α values result in fewer topics being assigned to the document. $\theta_{d,k}$ represents the probability of topic k belonging to document d and is set to the power $(a_k - 1)$ where a_k represents the dirichlet distribution parameter for topic k. The equation uses product notation to find the product of all numbers from topic 1 to topic K where K represents the total number of topics. This is then multiplied by $\frac{1}{B(\alpha)}$ which is a normalization term that makes sure the sum of all probabilities of topics belonging to the document d is 1. This value is stored in $P(\theta_d|\alpha)$ which determines how probable the current topic distribution is.

Next, it evaluates the current word distribution within each topic using the following equation:

$$P(\phi_k|\beta) = \frac{1}{B(\beta)}\prod_{w=1}^{V}\phi_{k,w}^{\beta_w-1}$$ where ϕ_k is a vector representing the word distribution(probability of each word belonging to the topic) for topic k. β is the Dirichlet distribution vector which determines how related a word has to be to a topic to be assigned to the topic. Smaller β values result in a lower chance of each word being assigned to the topic. $\phi_{k,w}$ represents the probability of word w belonging to topic k and is set to the power $(\beta_w - 1)$ where β_w represents the dirichlet distribution parameter for word w. The equation uses product notation to find the product of all number from word 1 to word V where V represents the total number unique words within the documents. This is then multiplied by $\frac{1}{B(\beta)}$ which is a normalization term that makes sure the sum of all probabilities of words belonging to the topic k is 1. This value is stored in $P(\phi_k|\beta)$ which determines how probable the current word distribution is.

After calculating $P(\theta_d|\alpha)$ for every document and $P(\phi_k|\beta)$ for every topic, the LDA model is able to use the values to recalculate the probability of each word belonging to each topic. The LDA model then reassigns the documents based on the new word distributions. Although the classifications of words from each document to each related topic start off random, as this process is repeated, the probabilities of words belonging to each topic change based on how often it is used in documents related to the topic and the documents are correctly categorized.

Procedure

- Training: Train models on dataset
- Testing: Test models on dataset

- Analysis: Compare model results
- Evaluation: Describe overall performance of each model

Training

As text embedding has built-in tokenization and vectorization features, it does not require training on the dataset.

Training the LDA model requires the user to first determine the number of topics the data will be categorized into. Through trial and error, we decided to create 11 topics. Before the LDA model can be trained, the data needs to be converted to a format easily understood by the model. First, we created a list of stopwords(common words that carry no meaning relevant to model categorization) by importing a premade list and appending words we found to be common throughout the dataset. We also imported a natural language processor that can tokenize the data to separate the words from punctuation and remove any accentation. After tokenizing the requirements, each token is lemmatized, meaning the root of each word is stored. This process implements the stopwords list created early to only store meaningful tokens. The python script then uses preprocessing techniques to convert the token list to tuples that can be easily understood by the LDA model. The script also creates a dictionary to match tuples with corresponding words. We then passed the tuple list and dictionary into the LDA model and set the pass count to 200 to allow the model to accurately determine keywords for each topic.

Testing

Text Embedding

To create text embeddings, the python script uses the all-MiniLM-L6-v2 transformer. This model is set to have a maximum sequence length so that input text size is limited allowing for the model to process text more quickly, but larger pieces of text need to be split into sections to capture the entire meaning. After creating the text embeddings, the python script is able to categorize the data by using the euclidean distance formula to compare the embedding values. To make the embedding categories easier to visualize, we reduced the vector dimensions to three prior to implementing the distance formula to make the embedding values graphable. To categorize the data, we used 0.3 as an arbitrary distance between points for them to be in the same category. To categorize the data, we used a python script to cycle through the list of embedding values. The python script checks each embedding against the rest of the list to find data points that are in the same category. After finding all the points in the same category, the python script removes the points from the main embedding list and stores them in a 2D list containing the categories and corresponding data values. The script continues to cycle through the data list until all embedding values have been categorized.

LDA Model

To assign topics to the requirements using the LDA model, we passed the requirements through a trained LDA model which returns a list of possible topics for the requirement as well as the corresponding probabilities. We assigned the topic with the highest probability to each requirement.

Results

To allow for simple analysis of the results, we created an HTML webpage to display how the requirements were categorized by different methods. The webpage contains 4 sections each performing a different function.

Clustering

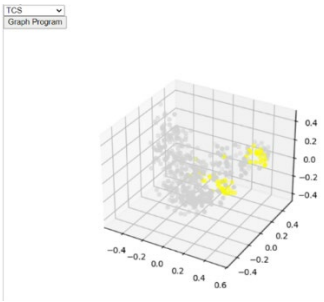

Figure 2. Clustering tab highlighting requirements from specific departments to portray text embedding categorization

This tab features a dropdown menu listing the different departments for which the requirements provide emergency protocol instructions. Choosing a department and submitting the form displays a 3D graph of the different requirements in the database in which the requirements from the selected department are highlighted in yellow. This tab shows how the text embedding categorization algorithm treats requirements from the same department as part of the same category.

Clustering2

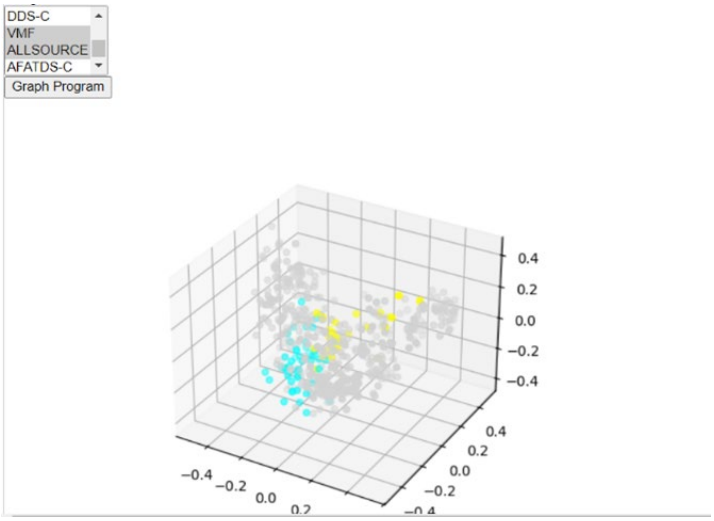

Figure 3. Clustering2 tab highlighting requirements from multiple departments in different colors

This tab has the same functionality as the Clustering tab with a dropdown menu that allows for multiple selections. The user can choose as many organization departments as they want and when the form is submitted, the application will display a 3D graph in which each selected department is highlighted in a different color. The departments and corresponding colors are listed in a legend below the graph. This tab shows how the text embedding categorization algorithm groups requirements from different departments into different categories.

LDA

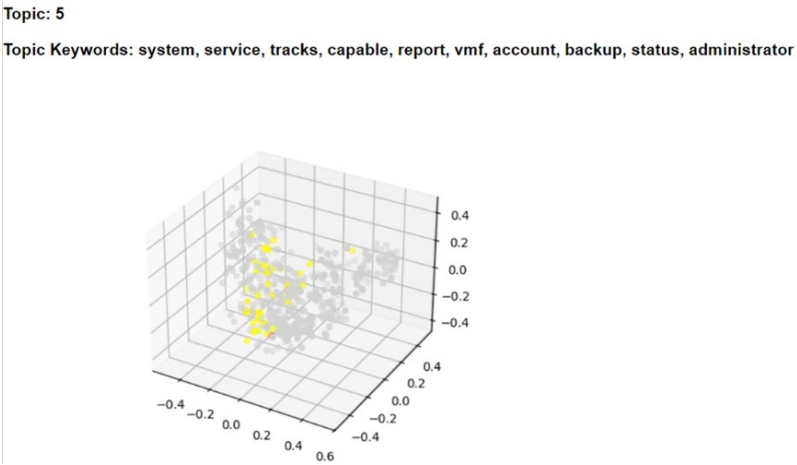

Figure 4. LDA tab highlighting requirements from specific topic and providing topic keywords

In the LDA tab, there are two dropdown items in the form: one for the Program of Records, and one for the requirement. When the program of records is chosen, the dropdown of requirements is changed to only contain requirements from the selected category. Once the user chooses the requirement and submits the form, the application displays the requirement's LDA topic and the topic's corresponding keywords. It also renders a 3D graph that displays the requirement point highlighted in red and other requirements with the same topic highlighted in yellow. This tab shows how requirements grouped together by the LDA model come from various categories.

LDA2

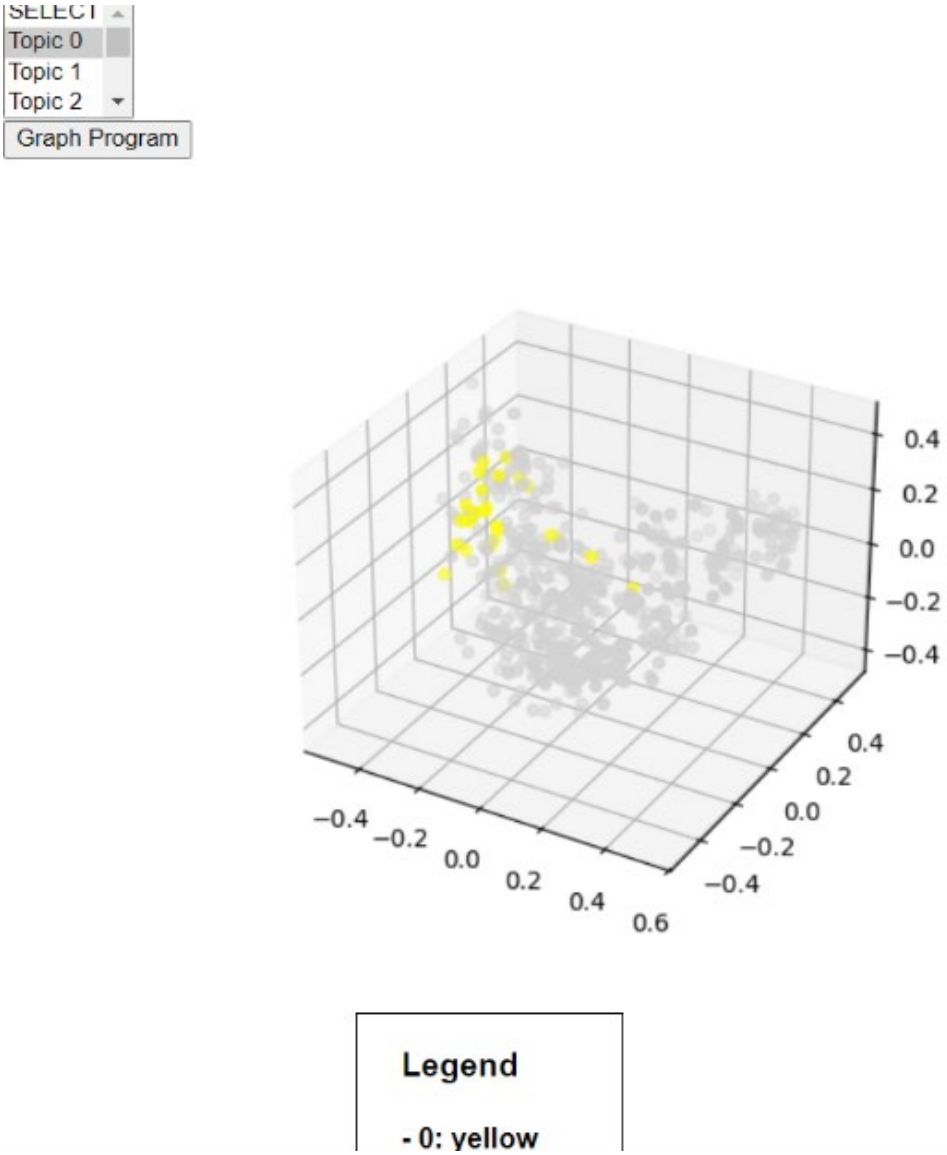

Figure 5. LDA2 tab highlighting requirements from multiple LDA topics in different colors

The LDA2 tab contains a form that features a multi-select dropdown menu which contains a list of topics. Once the user selects the topics they want and submits the form, the application creates a 3D graph of the requirements with the selected Programs of Records highlighted in different colors. The Programs of Records and corresponding colors are listed in the legend.

Text Embedding Clusters

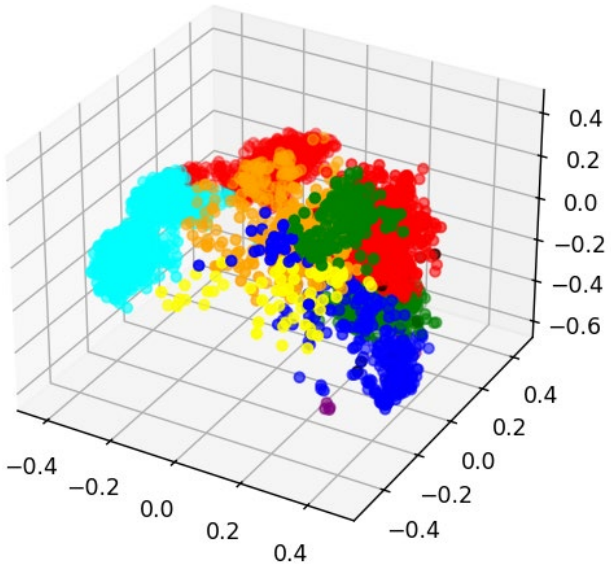

Figure 6. Tab displays graph of different categories created by text embedding algorithm highlighted in different colors

The Text Embedding Clusters tab displays a 3-D graph of the data points as categorized by the text embedding algorithm. As stated previously, the algorithm groups together data points that are a set distance from each other or closer. The groups are highlighted in different colors on the graph. This tab shows how the text embedding algorithm clusters the requirements. As the requirements are categorized based on a predefined distance, the categories are all relatively the same size.

Conclusion

The different sections show the variation between text embedding categorization which is depicted by the location of the points and LDA categorization which is depicted by the color of the points. Upon analysis of the categorization results of both methods, it becomes evident that the LDA model is better suited for our goal. As the text embedding method analyzes data semantically, it places requirements that perform similar functions in the same category rather than requirements that are targeted toward the same end goal. It is also more inaccurate as categorization through text embeddings requires the distance between data points to be predetermined. This makes it difficult to group data properly as points from different categories could be spaced differently resulting in the grouping or separation of multiple categories.

The LDA model, on the other hand, can categorize the data as desired because it looks at keywords which are common between requirements. As requirements working towards the same goal usually have more words in common, the LDA model can accurately assign data to various topics making the LDA model better suited for our goal of enhancing emergency response efficiency in an organization through text categorization.

Acknowledgments

I would like to thank my advisor Professor Arijit Das for encouraging me to pursue this research project. He helped spark my interest in this field of machine learning and provided guidance, support, and encouragement throughout the writing of this paper.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

Kulshrestha, Ria. “Latent Dirichlet Allocation(Lda).” *Medium*, Towards Data Science, 28 Sept. 2020, towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2.
Latent Dirichlet Allocation, www.jmlr.org/papers/volume3/blei03a/blei03a.pdf. Accessed 29 Aug. 2024.
“Latent Dirichlet Allocation.” *Latent Dirichlet Allocation - an Overview | ScienceDirect Topics*, www.sciencedirect.com/topics/computer-science/latent-dirichlet-allocation#:~:text=Latent%20Dirichlet%20Allocation%20(LDA)%20is,these%20layers%20and%20observed%20documents. Accessed 28 Aug. 2024.
Turing. “Word Embeddings in NLP: A Complete Guide.” *Word Embeddings in NLP: A Complete Guide*, Turing Enterprises Inc, 10 Feb. 2022, www.turing.com/kb/guide-on-word-embeddings-in-nlp.
Uma, Chisom. “What Is Text Embedding for Ai? Transforming NLP with Ai.” *DataCamp*, DataCamp, 3 June 2024, www.datacamp.com/blog/what-is-text-embedding-ai.
Wang, Liang, and Correspondence to {wangliang. *Improving Text Embeddings with Large Language Models*, arxiv.org/html/2401.00368v2. Accessed 28 Aug. 2024.