CE807-Assignment

Student ID - 2322460

Abstract

This report presents a text analysis project utilizing Latent Dirichlet Allocation (LDA) for topic modeling and Support Vector Machine (SVM) for sentiment classification. The project involves model selection, design and implementation of classifiers, and performance analysis. Key findings and future directions are discussed.

Materials

- Code
- Google Drive Folder containing models and saved outputs
- Video Presentation

1 Task 1: Model Selection

1.1 Selected Models **Unsupervised Model:** Latent Dirichlet Allocation (LDA)

Discriminative Classifier: Support Vector Machine (SVM)

1.2 Justification of Choices **Latent Dirichlet Allocation (LDA):**

Summary: LDA is a technique used to find out topics of a text in a corpus of documents. The documents can be seen as a combination of topics and each topic seen as a distribution over words in the vocabulary. It is specifically helpful when applied to the search for thematic structure when there is no labeled data available at all.

Suitability: Thus, the LDA has been chosen, as it would perform topic modeling, which is an important task of dissecting the underlying themes of the textual data. It is highly flexible so one can easily disentangle topics and keywords that are associated with each topic.

Strengths: As a result of this, LDA is useful in exploring and characterizing data that has not been tagged or classified in any way. It is well scalable to large data sets of data and is relatively very efficient.

Limitations: The limitation of LDA is that it requires specification of the number of topics which may be a challenge especially for cases as it is assumed the topic number is known beforehand. It also presupposes that topics are governed by Dirichlet distribution and this might not be the real distribution in most cases.

Support Vector Machine (SVM):

Summary: SVM is a type of machine learning technique which falls under the category of supervised learning and it is mostly used for classification purposes. It functions by finding the best hyperplane that would separate classes in the feature space and at the same time, make the maximum distance between the nearest data points of different classes, known as the support vectors.

Suitability: SVM was chosen on the basis of the fact that it demonstrated good results in the data with the large number of features, for instance, the TF-IDF vectors of the text material. This makes SVM extra forgiving in the situations, the place lessons are separable linearly.

Strengths: SVM is capable for linear and nonlinear classification depending on the kernel tricks. It also works well when m is much larger than n, which is quite common in the case of text data.

Limitations: The shortcoming of SVMs are that it is computationally intensive particularly when working with large number of data; and not scalable. Furthermore, it is also not designed to produce probabilistic outputs of actions at the native level; however, this might be inconsequential as most packages do not do this either.

1.3 Text Preprocessing and Representation Pre-processing Steps:

1. **Tokenization:** This is a process of breaking down of text to individual words also referred to as tokens.

2. **Stop-word Removal:** Excluding stop words since their content does not have a major impact on the text information (e. g. , “and”, “the”).

3. **Lemmatization:** Translating words to their basic or root form[for instance transforming

"running" to "run"]. These steps result in the reduction of dimension and removal of noise which is very essential for both LDA and SVM models.

Text Representation:

Bag-of-Words (BoW) for LDA: BoW acts as the term frequency of the words within the documents and LDA is able to make assumptions about topics deduced from the occurrence of words.

TF-IDF for SVM: Using Term Frequency Inverse Document Frequency, words are valued across a spectrum of their relevance in a document to a document count; thus assisting Support Vector Machine to fix its attention on goal words that segregate between the classes.

2 Task 2: Design and implementation of classifiers

Dataset	Total	Positive	Negative
Train	3681	3030	651
Valid	454	374	80
Test	409	-	-

Table 1: Dataset Details

Model	F1 Score
LDA	0.83
SVM	0.86
SoTA	0.90

Table 2: Model Performance.

2.1 Development of Classifier Models Latent Dirichlet Allocation (LDA) Model:

Data Preprocessing: Initially the data was imported and then data preprocessing emerged into the picture. These preprocessing steps that have been carried included the tokenization of the input text data, removing stop-words and finally lemmatizing the input text data. These activities were very significant for removing the noise and simplifying the structure of the text data, which made the quality of the input data for the LDA model to be very high.

Feature Extraction: For feature extraction, CountVectorizer was applied for preprocessed text data in which the text data was transformed to maximum feature vectors. This results in a matrix of tokens which in simple terms gives the frequency of words in the documents. We have already eliminated a lot of words in an effort to consider only first thousand words so as to obtain more meaningful words with less number computation.

Model Training: Training of an LDA model was done by utilizing this bag-of-words representation. Regarding the number of topics, a fixed number of 10 was selected with reference to early data analysis and specific knowledge of the research area. The topics model described these topics as what different themes or subjects were in the dataset.

Saving the Model: Subsequently, the LDA model and the fitted CountVectorizer were saved in the use of the joblib library. It also enables the user to reuse the model for future analysis while at the same time making certain that different vocabularies and topics of interest have the expected densities.

Support Vector Machine (SVM) Model:

Data Preprocessing: The same like the LDA model, before applying the SVM model, it was necessary to prepare the text data. The preprocessed text was used to enforce the coherence and to allow to merge models further on.

Feature Extraction and Transformation: The text is represented in form of TF-IDF features before the classification is done using a TfidfVectorizer. Such a representation seeks to attach a value to each word within the document taking into consideration the relative frequency of the word when the documents from the set is under consideration. A TF-IDF representation has been selected instead of the simple count vectors because the former provides better distinctive power of words in the context of classification.

Model Training: Using scikit-learn we deployed the SVM classifier with the help of a pipeline that involved TfidfVectorizer followed by SVC. This method involved the use of the linear kernel, which was used based on it simplicity and efficiency in analyzing text data, which simple linear separators are capable of handling. In the training process the model was fitted, hyperparameter adjusted and optimized for the classification accuracy against the training data. As it has been discussed above the linear kernel was especially appropriate to achieve the best combination of computational complexity and its outcomes.

Model Evaluation: Thus, the model was tested for accuracy in making valid predictions of new

data as a measure of its suitability for predictions as contained in the validation set. To evaluate the over efficiency, several indication of accuracy, precision, recall, F1-score among others were calculated. The result of this evaluation was that the SVM model reached a positive outcome in different aspects; it was especially effective when it came to sorting various pieces of information into diverse categories. Regarding the positive sentiment, the recognition accuracy turned out to be very high.

Model Saving: The generated SVM model and the TF-IDF preprocesser vectorizer are also saved using joblib so that the same model and preprocessing can be served in the future.

2.2 Training and Testing Training Process: Both models were stamped with the transformed and preprocessed texts as inputs. LDA model was used for extracting the topics of the entire given set of data and SVM model was again split into training and test sets so that the evaluated metrics could only be made by using a set which was unseen by the model.

Hyperparameter Tuning: However, in a regular LDA model, there is only a single hyperparameter deemed most important and this is the number of topics (n_components). This was decided after preliminary data analysis and because the varied nature of topics within the set data was assumed. For SVM model, other tunable attributes like Regularization parameter (C) and the Kernel type was done. Linear kernel was selected as per requirement and this required a good attempt to optimize the required regularization parameter to overcome the problem of over-fitting.

Model Evaluation Metrics: It was also established that the models were to be assessed by various criteria. In the case of SVM, accuracy, precision, recall, and F1 score were the key measures used in ascertaining how the model was performing with regard to each of the classes. Classification report was useful in partitioning of these metrics to note areas where the model performed well and areas where there was room for improvement. For instance, as seen in the case of the SVM model, the organisation is likely to have higher precision and a higher recall rate for the positive class than in the negative class of sample.

3 Task 3: Analysis and Discussion

3.1 Performance Comparison Insights About the LDA Model: Classification was not predominantly used in the LDA model; it mainly dealt with topic modeling. The main objective was laid in finding the patterns that were not directly observable within the given data. In another way, LDA was helpful in presenting the thematic features in the discussed documents in terms of the groups of relevant words. As much as this gave useful information and background information, it did not assist with the actual classification task. For other models, the LDA model's output included features such as the distribution of topics per document; however, the measure of classification accuracy was not part of the assessment of the LDA model.

Support Vector Machine Model Performance: Among these models, Support Vector Machine (SVM) model was the main classification model tested with validation set. These techniques reached 85% of the total accuracy, showing high discrimination between the classes. A more detailed analysis of the classification metrics revealed several nuances:

Precision, Recall, and F1-Score: Thus, the tables of the "positive" class in terms of the maximized number of true results and minimized number of false results had higher precision, recall, and F1-score than the corresponding figures of the "negative" class. Accuracy is defined as the true positive rate, the total of TP divided by the total of positive predictions has been made, whereas sensitivity is the ratio of accurately identified positive cases by the model out of all the positive cases that exist. F1-score is the precise measure of the common features of precision and recall as it is the harmonic mean of the two. Higher accuracy in the positive class implies the higher capability of the SVM model in classifying positive sentiment as opposed to negative sentiment in the microblogging site.

3.2 Detailed Analysis Examples:

In order to discuss its advantages and potential weaknesses, five instances of the model's specifics out of the validation set were discussed. These included both correctly and incorrectly classified instances:

Correct Classifications: In the case of correctly identified documents the given model probably used strong positive sentiment words that are characteristic of the constructed TF-IDF vectors. Of these, words used were those that could express positive feelings, or recommended the services, or described the services as good, etc. When such

words occur clearly and frequently in the training data set, the model was able to establish their strong link with the positive class.

Misclassifications: More of such misclassifications were recorded in documents that had both positive and negative tone or those with neutral words. For instance, an article containing both positive and negative opinions can mislead the model because the gains derived from use of word's frequency – IDF component in the TF-IDF representation might not be sufficient to depict the sentiment of the language used. Also, if they do not employ negative words and phrases, the negative tones, perhaps, gay or sarcastic might not always be interpreted correctly, thus producing wrong positive labels.

Model Confidence:

Finally, again, the training result of the SVM model: In most cases, it expressed relatively high confident on the predicting result of the positive sample. This may be because positive examples' TF-IDF vectors include greater differences – perhaps particular phrases or words directly linked with positive attitudes. However, the negative class can be hard defined and have fewer features as compared to the positive class making it difficult for the model to determine the difference between the negative class and other classes.

If this issue is to be solved, then the process of feature engineering needs to be expanded here. That is why using more complex techniques of text representation, such as word vectors or contextual characteristics, may better reveal the features of speech. Furthermore, handling class imbalance, for example, through oversampling of the minority class or the use of a different performance measure such as the area under the ROC curve, might enhance the model's performance as well as confidence level when determining negative sentiments.

4 Summary

4.1 Key Findings:

1. Model Selection and Justification **LDA**: Selected due to it use of unsupervised means to find out topics which are buried and useful for discovering the topical distribution of the given set. Indeed, it is especially helpful in analysis of distribution of topics across a large volume of documents.

SVM: Chosen for its stability with regard to such kind of classification problems, particularly when it comes to text data which is often represented via TF-IDF vectors. It was established that the linear kernel of the SVM was efficient for the positive and negative classification of sentiments.

2. Design and Implementation **Preprocessing**: To ensure that the created models would have good quality input text data, their processing included tokenization, stop-word elimination, and lemmatization.

LDA Model: Experienced in recognizing ten topics, which indicate main components in the given dataset.

SVM Model: Implemented a pipeline involving TF-IDF vectorization with linear classification, the accuracy be being 85% on the test set.

3. Analysis and Discussion **Performance**: The performance matrices showed that the SVM model had better F-scores for the positive classes rather the negative, which could mean that there was problem such as class imbalance or less separable features with the negative class samples.

F1 Score: The weighted average F1-score, accordingly was 0.86 here is the sum of the results that indicated the ever reliability of the model and also illuminated certain of the issues for the management to minimize the consideration of negative sentiments.

Detailed Analysis: The model has shown ability in detecting the highly positive sentiments while showed weakness in case of the mixed or ambivalent statements. Fluctuations in confidence mean more work on the features either or solving the imbalance of classes.

Conclusion The project successfully accomplished LDA and SVM algorithms for the text analysis along with results concerning content of the dataset and distribution of sentiments. The findings therefore highlight the need of stringent data preprocessing techniques, optimum choice of the model and acknowledged values for reiteration of feature construction and model assessment. Further work can be conducted in several directions: better models can be considered, for example, architectures based on transformers, and the identified shortcomings can be eliminated to improve the classification quality.

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan), 993-1022.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., & Blei, D. M. (2009). Reading Tea Leaves: How Humans Interpret Topic Models. In *Advances in Neural Information Processing Systems* (pp. 288-296).

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning* (pp. 137-142). Springer, Berlin, Heidelberg.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.

Salakhutdinov, R., & Hinton, G. E. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, 50(7), 969-978.

Wang, S., & Manning, C. D. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* (Vol. 1, pp. 90-94).

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6), 391-407.

Rosen-Zvi, M., Griffiths, T., Steyvers, M., & Smyth, P. (2004). The Author-topic Model for Authors and Documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (pp. 487-494).

Joachims, T. (1999). Making large-scale SVM learning practical. In *Advances in Kernel Methods: Support Vector Learning* (pp. 169-184). MIT Press.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1), 5228-5235.

Qi, X., & Davison, B. D. (2009). Web page classification: Features and algorithms. *ACM Computing Surveys (CSUR)*, 41(2), 12.

Yuan, Y., & Lu, Z. (2008). Text classification based on support vector machine and information gain. In *2008 International Conference on Computer Science and Software Engineering* (Vol. 1, pp. 688-691). IEEE.