Automatic Library of Congress Classification

Seyed Ahmad Abdollahpouri Hossieni, Teerapat Chaiwachirasak, Yujie Chen, Katie Warburton

Abstract

The Library of Congress Classification system is a complex bibliographic classification system that can make manual book classification challenging. To ease the challenge of classification we propose a text classification model that classifies documents based on their title and synopsis. We hypothesized that incorporating synopses and, using word embeddings to vectorize the title and synopses would improve classification. To test out our hypotheses we designed a Naive Bayes classifier, Support Vector Machine, Multi-Layer Perceptron, and LSTM to predict 15 of 21 Library of Congress classes. The LSTM with large BERT embeddings outperformed all other models and was able to classify documents with 76% accuracy when trained on a document's title and synopsis. This is competitive with previous models that classified documents using their Library of Congress Subject Headings.

1. Introduction

The Library of Congress Classification (LCC) is a bibliographic classification framework that organizes texts (physical, digital, or otherwise) into classes based on their knowledge domains. Alphanumeric class identifiers are assigned to individual library resources to indicate both a resource's subject and location within a digital or physical space [1]. The LCC is enumerative and hierarchically constructed [1]. This means that all its classes and subclasses are explicitly defined within the classification scheme and that the classes are arranged in a tree-like structure of more general topics and their subdivisions. Despite being developed specifically for the US Library of Congress, the LCC has been widely adopted by other libraries in the US, especially large academic ones.

The enumerative nature of the LCC means that it is highly descriptive but also very complex. Every class and subclass is explicitly defined to capture the potential content of past, present, and future library resources [1]. This complexity combined with the constant introduction of new resources can make manual classification a time-consuming task. Therefore, there is motivation to develop and perfect techniques that can automate the classification process. Techniques developed for the automatic library of congress classification of library resources can hypothetically be extended to other library classification systems (i.e., the Dewey Decimal System) leading to the eventual development of more general automatic library classification techniques.

Figure 1. The 21 top-level classes of the Library of Congress classification system.

The structure of the LCC is as follows. Resources are divided into 21 main classes corresponding to various general topics such as science, language & literature, technology, etc. These main classes are depicted in figure 1. Each main

class is denoted using a letter from the Latin alphabet. The main classes are recursively divided into subclasses that correspond to increasingly more specific topics within a domain. Subclasses are indicated using either additional letters or numerical ranges [1]. For example, class P has subclass PA and subclass PA is subdivided into subclass PA1-199. Documents classified within the LCC are also assigned Library of Congress Subject Headings (LCSH) which come from a controlled vocabulary of document topics. They are used to index a document based on its main subject(s).

Previous approaches to the automatic classification of documents into the LCC have focused on predicting the fully specified classification for a given text and achieved low accuracies [2, 3, 4]. Larson designed a classifier to predict the LCC for documents in class Z [2]. 30,000 documents were divided into one of the 8,435 subclasses of class Z. Various combinations of the titles and LCSHs for documents within a subclass were used to manually create a vectorized representation of said subclass. New documents were then converted to vectorized representations using the same criteria and assigned to a subclass based on similarity. When predicting a class only using the first LCSH assigned to a document, Larson achieved an accuracy of 46.6%[2]. Ávila-Argüelles et al. implemented a term presence discrimination algorithm that predicted the full LCC for a document in class Q based on the terms in a document's title [4]. They achieved a recall of 36.13% on the test set [4]. Frank and Paynter used hierarchical clustering of a document's LCSHs to predict the full LCC for a document in any Library of Congress class and achieved an accuracy of 55.32% [3].

The low accuracies associated with predicting a document's fully specified LCC can be partially attributed to the fact that the number of target categories is in the thousands. This means that many subcategories are sparsely populated and therefore can be underrepresented in the training data [3,5]. Because of the scarcity of data at lower levels in the hierarchy the goal of our project was to design a text classifier that could predict the top level (main) library of congress class of a document. Generating an accurate main class prediction is important because any error in the prediction of a document's top-level class will mean that the document will not belong in any of its predicted subclasses.

Frank and Paytner classified documents into their top-level class with 80.27% accuracy using LCSH [3] however LCSH are specific to documents classified within the LCC and thus text classification methods trained using LCSH are not extensible to other classification systems. As well, LCSHs must be manually assigned to a new text before classification. Therefore to achieve our goal we created vectorized representations of a text's title and summary (i.e., synopsis or abstract) instead. We hypothesized that incorporating synopses would capture additional important associations that have previously been ignored and could therefore improve the classification accuracy. Once vectorized, the document representations were used to train a Naive Bayes Classifier, Support Vector Machine (SVM), a Multi-Layer Perceptron (MLP), and an LSTM network to output a document's predicted class. The code of our project can be found at https://github.com/ahmad-PH/iml_group_proj.

To the best of our knowledge, another knowledge gap in automatic library classification relates to the use of word embeddings. We believe that it will be highly beneficial to use word embeddings, given that these embeddings encode some semantic meaning for each word, which is much richer information than a naïve one-hot encoding or bag-of-words methods [7]. Therefore we implemented models that use both word2vec [9, 10] and BERT [11] word embeddings. Being able to accurately classify books with an automatic book classification system based upon titles and summaries would facilitate future library practice.

2. Methods

2.1. Dataset

The project was conducted using the OhioLINK Circulation Data, which is made available by OCLC Online Computer Library Center Inc. and OhioLINK under the ODC Attribution License [6]. This database contains nearly 30 million circulation records for approximately 6 million unique documents held by OhioLINK libraries. Only a subset of 125,488 records from the dataset were used. These were records that contained all three of a document's title, synopsis, and Library of Congress Call Number (LCCN), which contains the LCC of a document.

Once the relevant document information was collected, the LCCN of a document was parsed to determine the document's top-level class in the LCC. Documents were then sorted by class and the number of documents per class was computed. The number of documents within each class was extremely unbalanced. For example, class P had 62 448 documents whereas class V only had 146. The dataset was balanced to prevent the classifier from classifying based on the experimental probability of a document falling into a class C.

To balance the dataset, first, all classes with less than 1200 documents were excluded from the dataset. This led to the exclusion of classes A, C, M, U, V, and Z in the text classification model. Next, 2000 documents were randomly selected from each class with more than 2000 documents. Therefore the dataset consisted of 15 classes, each with 1200 to 2000 documents. 200 documents were randomly selected from each class in the dataset to create the test set. The remaining documents in each class were put into the training set. Thus, each model was trained using a set of 25,562 training examples and 3000 testing examples spanning 15 of the 21 Library of Congress top-level classes.

2.2 Preprocessing and Feature Extraction

We have used two types of title and synopsis representations in this work, each of which required its own separate preprocessing.

2.2.1 Term frequency–Inverse document frequency (TF-IDF)

TF-IDF is a scoring measure that can be used to capture each word's importance by looking at its term frequency and inverse document frequency. We assign each word in the titles and synopsis TF-IDF scores to form a TF-IDF vector representation of each record.

As the number of columns for this matrix is equal to the total unique number of vocabulary in the whole corpus, which is slightly over 80,000. This can cause problems in the training process, both in the performance due to the high dimensional space (curse of dimensionality), and in the computational resources needed to complete the training. To combat this, we use one of the dimension reduction techniques called Principal Component Analysis (PCA) to reduce the number of dimensions of the TF-IDF matrix to 500 dimensions.

2.2.2 Word Embeddings

We have two different types of preprocessing for generating the word2vec and BERT word embeddings. For generating the word2vec embeddings, the preprocessing steps include lowercasing, punctuation and stopword removal, and word tokenization provided by the nltk package [11]. The embeddings themselves are 300-dimensional vectors trained on the Google News dataset, provided by the gensim package [12]. For generating the BERT word embeddings, the preprocessing has been done by the BertTokenizer class of the transformers package [13] provided by huggingface. We rely on this class to do lowercasing and some basic tokenization before generating the embeddings.

We tried 3 variants of BERT pre-trained models, which we shall call BERT-tiny, BERT-small, and BERT-large. BERT-tiny and BERT-small are from Bhargava's work [14][15], of which each word embedding size is 128 and 512 respectively. The intention for selecting these two models is to pick embeddings size close to word2vec to see the

effect of the size of BERT embeddings, and whether compact BERT embeddings would be sufficient in the classification. BERT-large has a word embedding dimension of 768 and is taken from the model based on Devlin et al[16]. BERT-large was trained on BookCorpus and English Wikipedia.

The embeddings extracted for the title and synopsis each consists of a list of vectors. For our non-sequential models, including SVC and MLP, we average this list of embeddings into a single embedding because the models require a fixed-length input. For the sequential models, however, the embeddings are directly fed to the model.

2.3 Models

Four machine learning algorithms were used, namely Naive Bayes [17], support vector classifier (SVC) [18], multilayer perceptron [19], and long short-term memory (LSTM) [20]. Each model was developed under three different scenarios, including title alone, synopsis alone, and combining title and synopsis. As a general strategy, we divided the 25,562 training examples into training and validation sets with a 70: 30 ratio. Models were trained on the training set. Hyperparameters were tuned with the validation set by optimizing the validation accuracy. Once the best hyperparameters were selected, a model was refitted using the combined training and validation sets and then applied to the test set to assess the model performances. We compared the four types of classifiers based on the sensitivity (recall), positive predictive value (precision), F1 score, and accuracy.

The outcome of primary interest is the machine learning test performance given different word embeddings involving both title and synopsis. Our secondary interest is the addition of a synopsis to the title on model test accuracy.

2.3.1 SVC

The SVC was implemented using the scikit-learn package [21] in Python. The classifier was built with a radial basis kernel and penalized for misclassification. The optimal hyperparameter values were “C = 100” and “gamma=0.001”.

2.3.2 Multi-Layer Perceptron

We tried both scikit-learn and PyTorch packages for this model, both of which yielded very close results. We will only include the result from the scikit-learn version for simplicity. The architecture of the model consists of two hidden layers of 600 units each, with the ReLU activation function. Using early stopping, the model only trained for 24 epochs with a learning rate of 0.001.

2.3.3 LSTM

Our data consists of book titles and synopsis, which is inherently a sequential type of data. To address this in the previous models, the embeddings were averaged to create a single vector. But, this might lose useful information. Therefore, we wanted to explore some sequential models as well to see if performance could be improved when the restriction to average the vectors was lifted.

Because we have titles and synopsis which are separate sequences of data, the model that we will refer to as “LSTM” actually consists of two separately trained LSTM modules. Each LSTM module has 2 layers of 64 dimensions each and uses trainable initial states of all zeros. Each LSTM is run on its associated input (one on title and one on synopsis), and the output at the last timestep is collected from both and concatenated to form the input to a linear classifier, which then applies a softmax to the final output.

We also experimented with using GRUs to see if the results would be any different. Since the results were almost identical to those of LSTM, we do not include those results separately.

3. Results

Firstly, to visualize our data and have a better feeling of its nature, we used the t-SNE method on both word2vec and BERT_small embeddings generated for record titles. The embeddings for each title were averaged before being fed to a t-SNE with 2 components. The results have been included in figures 2 and 3. These visualizations show a meaningful degree of clustering among the embeddings of each class. The classes “Medicine”, “History of Americas”, and “Social Sciences” in figure 2 are good examples.

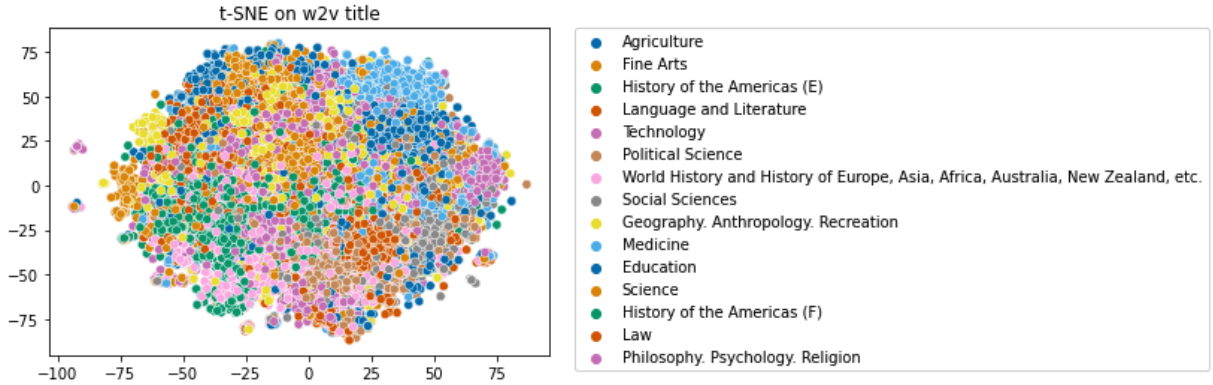

Figure 2. T-SNE results for the word2vec embeddings of record titles.

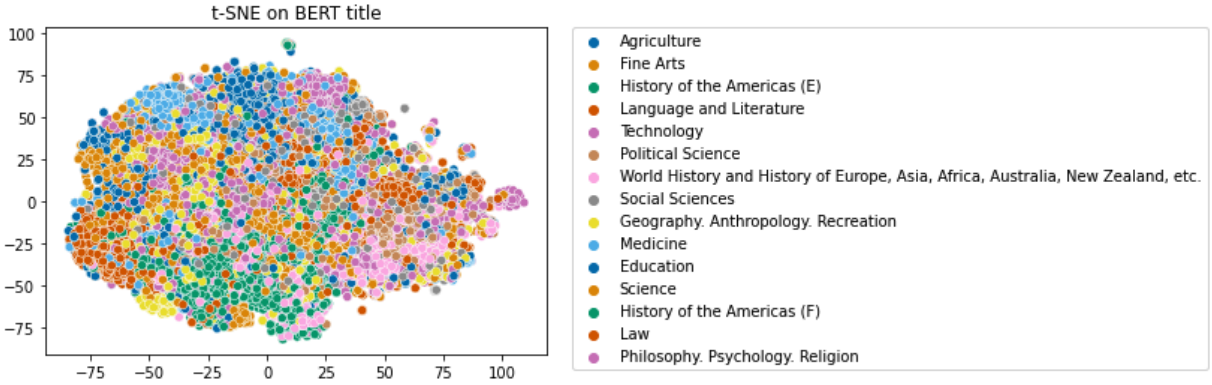

Figure 3. T-SNE results for the BERT_small embeddings of record titles.

Table 0: Summary of dimensions of word embeddings for three prediction scenarios

	Title only	**Synopsis only**	**Title and Synopsis**
word2vec	300	300	600
BERT tiny	128	128	512
BERT small	215	512	512
BERT large	768	768	1536

Table 1 summarizes the algorithm performances based on the choice of features. In general, among the four classifiers, LSTM resulted in the highest model performance while the Naive Bayes model was associated with the lowest performance.

Table 1: Summary of algorithm performances based on the choice of features

	Naive Bayes	SVC	MLP	LSTM
Accuracy				
TF-IDF	33.87%	62.30%	67.93%	NA
word2vec	46.07%	69.03%	69.93%	71.77%
BERT_large	50.67%	75.13%	75.60%	76.03%
Precision				
TF-IDF	42.25%	66.99%	68.50%	NA
word2vec	51.69%	69.60%	71.12%	72.12%
BERT_large	53.57%	75.43%	75.79%	76.10%
Recall				
TF-IDF	33.87%	62.30%	67.93%	NA
word2vec	46.07%	69.03%	69.93%	71.77%
BERT_large	50.67%	75.13%	75.60%	76.03%
F1 Score				
TF-IDF	33.82%	62.59%	67.97%	NA
word2vec	46.45%	69.13%	69.83%	71.75%
BERT_large	50.38%	75.19%	75.51%	**76.01%**

Abbreviations: SVC: support vector classifier, MLP: multilayer perceptron, LSTM: long short-term memory, NA: not available, TF-IDF: term frequency–Inverse document frequency

Table 2 summarizes the contribution of title and synopsis to the model accuracy. As compared to the title, the test accuracy was driven by the synopsis.

Table 2: Summary of the contribution of title and synopsis to overall performance (accuracy)

	Title	**Synopsis**	**Title and Synopsis**
Naive Bayes	39.03%	53.20%	50.67%
SVC	62.37%	74.07%	75.13%
MLP	61.27%	73.30%	75.60%
LSTM	NA	NA	76.03%

Abbreviations: SVC: support vector classifier, MLP: multilayer perceptron, LSTM: long short-term memory

4. Discussion

We trained a Naive Bayes Classifier, SVM, MLP, and LSTM to predict one of 15 Library of Congress classes for a document given its title, synopsis, or a combination of the two. A document's title and synopsis were represented using one of TF-IDF, pre-trained word2vec embeddings, or pre-trained BERT embeddings of various sizes. The models were evaluated using accuracy, precision, recall, and f1 scores The results indicate that, of the proposed models, the best model for predicting a document's LCCs was an LSTM trained on 768-dimensional BERT representation of a document's title and synopsis. It achieved an accuracy of 76% and an f1-score of 0.76. The performance of the SVC and MLP tended to be equivalent whereas Naive Bayes consistently performed the worse with f1-sores never passing 0.51 on the test set. All models performed best on the 768-dimensional BERT representations (BERT-large). The order of performance for tile and synopsis representations was as follows (from best to worst); BERT-large, BERT-small, word2vec, BERT-tiny, TF-IDF. This shows that word embeddings do provide meaningful additional information for classification, as hypothesized.

The results also indicated that the synopsis better represented the distinctions between documents contained in one class versus another. Training on the synopses instead of the titles always led to a significant increase in classification ability. This supported our hypothesis that the synopses would capture important associations between classes and words that the titles would be unable to represent. Including both the title and synopses when training a model improved performance enough for us to justify training on both.

Our best model was not able to achieve the 80% accuracy that Frank and Paynter achieved when predicting the top-level class of a document using a hierarchical clustering algorithm based on document LCSHs [3]. However, the hierarchical clustering algorithm achieved 80% accuracy for top-level classes only when trained on 800,000 documents. Frank and Paynter reported that, when trained on 10,000 documents, their algorithm could only predict the main class of a document with 62.39% accuracy, and when trained on 100,000 it could do it with 74.76% accuracy [3]. Therefore our model, trained on 25,562 documents, outperformed a hierarchical clustering model trained on 100,000 documents. This shows the effectiveness of strong word embeddings in tackling sparse training data. Their effectiveness was also demonstrated by figures 2 and 3 which show meaningful clustering of documents within a class when representing them with word embeddings. Our results also demonstrate that the synopsis has

enough information in it to compensate for the lack of LCSHs, which are carefully chosen manual features. The exclusion of them opens up our model to being used in more general book classification tasks which might lack this feature.

The study is limited due to the size of our dataset. All the models indicated a large degree of overfitting as train. To reduce overfitting we would most likely also have to train the model on a much larger dataset. Each top-level class has a large variance in the type and content of the documents that would be assigned to them. This is because the topics of the top-level classes are very general. Selecting only a 2000 element subset of the documents means that the true distribution of the data is most likely not being captured by the training set. Therefore in an attempt to improve accuracy and reduce overfitting, our future work would involve training the LSTM model on a larger quantity of data. Our model is also limited by the exclusion of 6 Library of Congress classes. Because the amount of data for each of these 6 classes were so sparse, we could not include them without having a very small balanced dataset or having an unbalanced dataset that would risk predicting the more common classes over a document's actual class. In the future, we would like to gather enough data so that the model can be trained to predict every top-level Library of Congress Class.

Overall, our results indicated that a document's titles and synopsis can capture a significant portion of the distinctions between documents placed into one class versus another. Moreover, an LSTM can be trained to predict the top-level LCC of a document with 76% accuracy.

5. Contributions

Yujie: Trained and assessed the performance of SVM models and reported the SVM and general model development approaches and relevant results.

Katie: Researched previous automatic LCC attempts and found the dataset. Wrote the introduction and helped to write the discussion. Researched and understood the MARC 21 bibliographic standard to parse through the dataset and extract documents with an LCC, title, and synopsis. Balanced the dataset and split it into a train and test set. Described data balancing and the dataset in the report.

Teerapat: Wrote the code for generating tf-idf features and BERT embeddings. Trained Naive Bayes and MLP on tf-idf features and BERT embeddings. Wrote training pipelines that take ML models from the whole team and train them together in one same workflow with multiple data settings (title only, synopsis only, and title + synopsis) to get a summarized and unified result. Trained LSTM models on BERT embeddings on (Google Collab).

Ahmad: Wrote the code for generating word2vec embeddings and its corresponding preprocessing and the code for MLP and LSTM models on these embeddings. Came up with the idea of visualizing the averaged embeddings. Wrote the parts of the report corresponding to these sections.

Note that after each member wrote their section in the report, the editing was mostly done together, so all members were involved in the final editing of the report.

6. References

1. Library of Congress Classification.https://www.librarianshipstudies.com/2017/11/library-of-congress-classification.html. 2020. Accessed November 13, 2021.
2. Larson RR. Experiments in automatic library of congress classification. Journal of the American Society for Information Science 1992;43:130-148.
3. Frank E, Paynter GW. Predicting Library of Congress classifications from Library of Congress subject headings. Journal of the American Society for Information Science and Technology 2004;55:214-227.
4. Ávila-Argüelles R, Calvo H, Gelbukh A, Godoy-Calderón S. Assigning Library of Congress Classification codes to books based only on their titles. Informatica 2010;34.
5. Godby CJ, Stuler J. The library of congress classification as a knowledge base for automatic subject categorization. Subject Retrieval in a Networked Environment: KG Saur; 2013:163-169.
6. OhioLINK Collection and Circulation Analysis—Circulation Data. Dublin, OCLC Research; 2011. https://www.oclc.org/research/areas/systemwide-library/ohiolink/circulation.html.
7. Rudkowsky E, Haselmayer M, Wastian M, Jenny M, Emrich Š, Sedlmair M. More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures* 2018;12:140-157.
8. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* 2013.
9. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. Paper presented at: Advances in neural information processing systems2013.
10. Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* 2018.
11. Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.
12. Rehurek, R., & Sojka, P. (2011). Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, *3*(2).
13. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
14. Iulia Turc and Ming-Wei Chang and Kenton Lee and Kristina Toutanova (2019). Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation. CoRR, abs/1908.08962.
15. Prajjwal Bhargava, Aleksandr Drozd, & Anna Rogers. (2021). Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics.
16. Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR, abs/1810.04805.
17. Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning, 2009. "The Naive Bayes Classifier"Page 210-215, Springer, NewYork.
18. Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152).
19. Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning, 2009. "Neural networks"Page 395-401, Springer, NewYork.
20. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
21. Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn: Machine learning in Python." The Journal of machine Learning research 12 (2011): 2825-2830.