Predicting Research Domains from Titles and Abstracts Using ArXiv Data

Muchang Bahng*
Department of Computer Science
Duke University
Durham, NC 27708
muchang.bahng@duke.edu

Alice Hu*
Department of Computer Science
Duke University
Durham, NC 27708
alice.hu@duke.edu

Changmin Shin*
Department of Computer Science
Duke University
Durham, NC 27708
changmin.shin@duke.edu

Patrick Wang
Department of Electrical Engineering
Duke University
Durham, NC 27708
patrick.wang@duke.edu

Abstract

The rapid expansion of scientific publishing, which is exemplified by repositories such as ArXiv, has made it increasingly difficult to efficiently organize and retrieve research outputs. As the volume and diversity of submissions continue to grow, manually assigning papers to subject categories has become extremely tedious and prone to inconsistency, particularly in interdisciplinary or rapidly evolving areas. This study investigates automated approaches to classify ArXiv research papers using natural language processing techniques applied to titles and abstracts. Focusing on subfields within computer science, we evaluate a spectrum of models from probabilistic methods to modern neural architectures. These approaches include a Multinomial Naive Bayes classifier using bag-of-words features, a Long Short-Term Memory (LSTM) network, and a Doc2Vec document embedding model paired with a multilayer perceptron. Using a large-scale metadata subset from the ArXiv repository, we examine how well each model captures semantic cues and domain-specific terminology. The results highlight the strengths and limitations of different modeling strategies. Overall, the study provides insight into methods for supporting automated classification in large scientific repositories. Our codebase is available in our Github repository.[1]

*Equal contribution

[1] https://github.com/aliceyh7/ECE684-Research-Abstract-Classifier/tree/main

1 Introduction

In recent years, the explosive growth of scientific publishing—catalyzed by the development of large language models (LLMs) that can efficiently implement codebases from scratch—has made it increasingly difficult to organize, retrieve, and interpret the progress of research. While peer-reviewed publications and journals such as ICML, CVPR, and ICLR, represent the "gold-standards" of research, open access archives such as ArXiv, maintained by Cornell University, have played an even more essential role in this paradigm shift. From its founding in 1991, ArXiv has been a public platform for institutional and independent researchers to share their works online across a diverse range of fields, such as mathematics, physics, computer science, and statistics. In fact, it has been common practice for researchers to post their reports on ArXiv even before said paper is accepted into a conference.

The scale of open access research far exceeds the throughput of peer-reviewed publication venues. Top machine learning conferences receive about 10,000 papers annually, which must be assigned to proper reviewers to undergo a reviewing process. On the other hand, ArXiv receives on average of 24,000 new submissions per month, which are manually assigned to subject categories by authors and moderators. While this system helps with maintaining order, manually identifying each paper is time-consuming and error-prone, especially where research areas overlap or evolve quickly. Automating this classification of research papers is an essential workflow that can alleviate these efforts on large scales. Moreover, individual researchers who wish to keep personal catalogs can also benefit from this automation. The two most representative subtexts of a paper are its title and the abstract, indicating the need for natural language processing (NLP). Given that most practitioners who wish to skim over a paper reads through these pieces of text, we demonstrate that it is possible to train classifiers that can mimic the behavior of specialists when organizing papers.

This project aims to explore and compare several NLP techniques to automatically predict the research category of ArXiv papers based on their titles and abstracts. More specifically, our goal is to develop two neural architectures—a bidirectional LSTM and a Doc2Vec MLP classifier—and compare their performance against the multinomial naive Bayes, a standard benchmark in NLP tasks. We further aim to evaluate how each model captures the semantic structure of scientific writing. The study focuses on a subset of computer science subfields, where overlaps in vocabulary and style present a realistic challenge for automatic classification. Our results show how linguistic features and contextual cues can be used to support scalable document organization and retrieval in large research repositories.

2 Related Works

ArXiv and Scientific Document Classification. Automated classification of scientific papers has been widely studied as research repositories such as ArXiv continue to grow rapidly. Classical approaches relied on TF–IDF features paired with supervised models like Naive Bayes or SVMs Kim and Gil [2019], Schubotz and et al. [2020]. These methods demonstrated reliable performance when topic boundaries are

well defined, but they often struggle in domains with overlapping terminology—such as computer science subfields. More recent work has explored deep learning–based architectures, using RNNs or CNNs to capture sequential and semantic structure in paper abstracts Romanov et al. [2017]. Several studies have benchmarked classification models specifically on ArXiv metadata Rahman and et al. [2025], showing that neural models can outperform classical baselines, although gains vary across granular subfields.

Document Embeddings and Neural Representations. Beyond surface-level lexical features, document embeddings have become central to scientific text categorization. Doc2Vec Le and Mikolov [2014] provides dense representations that encode global semantic information and has been applied successfully to academic corpora. More recent embedding models such as SciBERT Beltagy et al. [2019] and domain-specific transformers have improved downstream classification accuracy by capturing scientific vocabulary and phrasing more effectively. Unsupervised approaches combining pretrained embeddings with clustering Turrisi [2023] have also been explored for automatically organizing large repositories. These works collectively highlight that representation choice—bag-of-words, learned embeddings, or pretrained language models—plays a substantial role in classification performance.

Hierarchical and Multilabel Research Taxonomies. Because scientific knowledge is often organized hierarchically, several works study classification within structured taxonomies such as the Mathematics Subject Classification (MSC) or biomedical ontologies. Yang and et al. [2023]. While these methods achieve promising results, they typically require explicit taxonomies or domain-specific ontologies that are not readily available or well-maintained for many subfields of computer science. In contrast, ArXiv categories are relatively flat, author-assigned, and noisy, making the classification problem closer to an open-domain, single-label setting.

Challenges in Large-Scale Research Organization. A recurring challenge highlighted across prior work is the ambiguity and overlap between related research areas. For example, distinctions between machine learning (cs.LG), artificial intelligence (cs.AI), and computer and language (cs.CL) can be subtle, and authors frequently cross-list their papers across multiple categories. However, these methods require extensive feature engineering or metadata not always available at scale. The present work follows a simpler, fully text-based framework while systematically comparing classical and neural methods on realistic overlapping categories.

3 Methodology

3.1 Data

The data used in this project come from arXiv Dataset, which is publicly available on Kaggle. This data set is a mirror of the official ArXiv repository and includes metadata for roughly 1.7 million submissions across a wide range of STEM disciplines. Each

record contains key information such as the title, abstract, authors, comments, journal reference, digital object identifier(DOI), categories, and version history. The data set is automatically updated weekly by Cornell University.

Because the full dataset is very large, this project focuses on the computer science portion, selecting a subset of 30 subfields that had more than one hundred papers each. These categories represent a diverse set of topics within computer science that share overlapping terminology and methodological language, making the classification task significant. Some of the notable categories include cs.AI(Artificial Intelligence), cs.CL(Computation and Language), cs.LG(Machine Learning), cs.CV(Computer Vision), cs.DS(Data Structures and Algorithms), cs.CR(Cryptography and Security), and cs.SE(Software Engineering).

The Kaggle dataset provides metadata in JSON format, where each entry corresponds to a single ArXiv paper. For each record, the title and abstract fields are combined into a single text sample to serve as input for classification. The metadata can be accessed directly through Kaggle or through Google Cloud Storage buckets that host the full dataset. Only the metadata file is used in this study as the PDFs and source files available in the full ArXiv collection are excluded due to their size and processing complexity.

3.2 Naive Bayes Baseline

As a baseline, we consider a Multinomial Naive Bayes classifier operating on k-gram frequency vectors. Given a sequence $\mathbf{s}$, we extract a vector of k-gram counts $\mathbf{x} \in \mathbb{R}^d$ and assume conditional independence of features:

$$P(y \mid \mathbf{x}) \propto P(y) \prod_{i=1}^{d} P(x_i \mid y).$$

The class prediction is obtained via

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \left[\log P(y) + \sum_{i=1}^{d} x_i \log P(x_i \mid y) \right].$$

Naive Bayes is fast, highly interpretable, and provides a strong baseline for sparse and high-dimensional data. The simplicity of this model allows robust performances, even with a small number of training samples. However, the conditional interdependence is often violated in real sequences, which results in limited ability to capture contextual or structural dependencies.

3.3 Doc2Vec Embedding Model

To obtain dense representations of genetic sequences, we embed each sequence using a Doc2Vec encoder trained in the Distributed Bag-of-Words (DBOW) style. Let $E(\mathbf{s}) \in \mathbb{R}^m$ denote the resulting embedding. A linear classifier then produces logits via

$$\mathbf{z} = WE(\mathbf{s}) + \mathbf{b},$$

and the predicted class is

$$\hat{y} = \arg\max_c z_c.$$

An advantage of the Doc2Vec model is that it yields a compact, parsimonious representation of an entire document, allowing it to skip over the heavy sequential processing done in usual RNNs. As a successor of the Word2Vec and GloVe embedding models, our document embedder allows to capture sementically meaningful information beyond simple k-gram frequencies. This allows the model to recognize non-local dependencies and can subsequentily compare sequence-level similarities. However, Doc2Vec is insensitive to exact positional information and may struggle when classification depends on sequence order.

3.4 LSTM Sequence Model

We further consider a recurrent neural model that directly consumes raw nucleotide sequences. Let $\mathbf{s} = (s_1, \ldots, s_T)$ denote a sequence and let $v(s_t)$ be the embedding of nucleotide s_t. The LSTM computes hidden states by

$$h_t = \mathrm{LSTM}(v(s_t), h_{t-1}),$$

and uses the final hidden state to produce logits:

$$\mathbf{z} = W h_T + \mathbf{b}, \qquad \hat{y} = \arg\max_c z_c.$$

The LSTM models a sequential structure. It captures long-range dependencies and also some order-sensitive patterns, which simpler models cannot do. This makes it a good fit for tasks where the arrangement of subsequences is meaningful. However, LSTMs require significantly more data and computation, are slower to train, and they may overfit without careful regularization or sequence-level augmentation.

3.5 Evaluation Framework

All models share a consistent preprocessing pipeline and are evaluated using the same train, validation, and test splits. While the original plan was to measure performance in terms of overall accuracy and macro-averaged F1 score, the heavily skewed nature of the dataset revealed potential flaws in the performance metrics. For example, computer vision category had more than 100,000 samples, while software engineering had a few hundred.

This imbalance leads to an inflation in accuracy score in certain categories, as the models can reach high accuracy simply by predicting the majority class. To address this issue, we adopted a balanced accuracy that takes an average of accuracies computed for predicting each category. Adding this metric accounts for the imbalances and provides a more representative measure of model performance. Confusion matrices are used to identify patterns of misclassification and to interpret which research categories are most easily or most frequently confused.

4 Results

4.1 Experimental Setup

We begin by describing the preprocessing of our data. Let $\mathcal{D} = \{(t_i, a_i, c_i\}_{i=1}^{N}$ describe the title t, abstract a, and class i of a given paper. We concatenate the title and abstract (with a space token in between) to prepare our input

$$x_i = t_i \oplus \texttt{' '} \oplus a_i \tag{1}$$

We then one hot encode each class c to get

$$y = \text{onehot}(c) \in \mathbb{R}^{30} \tag{2}$$

To ensure consistency and to prepare the text for modeling, all text is lowercased and tokenized, and punctuation, digits, and special characters are removed. A vocabulary is constructed from the processed training data to define the set of unique tokens used by each model. Then the dataset is split into training, validation, and test subsets using stratified sampling to preserve class distributions. The training data is used to fit the models, the validation set to tune hyperparameters, and and the test set to provide an unbiased measure of performance. This procedure ensures consistent comparison across all models.

4.2 Quantitative Analysis

Across the three modeling approaches—Multinomial Naive Bayes, Doc2Vec with an MLP classifier, and an LSTM sequence model—we observe a gradual improvement in classification accuracy as the models incorporate increasingly rich representations of linguistic structure.

Model	**Accuracy**	**Balanced Accuracy**
Naive Bayes	72.85	52.21
Doc2VecMLP	64.39	53.62
LSTM	63.01	**56.54**

Table 1: Comparison of accuracy and balanced accuracy on Naive Bayes, Doc2VecMLP, and LSTM.

As shown in Table 1, the multinomial Naive Bayes model achieves a raw accuracy of 72.85%. However, we found that this was due to the major performance gains for the "computer vision" and "game theory" categories. When looking at the balanced accuracy, we observed a 20% decrease in accuracy, highlighting the challenges in classifying data from a heavy-tailed distribution. These results suggest that simple frequency-based models thrive in classifying papers from fields either having iconic keywords or from fields with an adequately large number of papers to construct an accurate frequency distribution.

In sparser categories, the Doc2Vec + MLP model, outperforms the Naive Bayes marginally at 54.62%. This similarity suggests that although Doc2Vec captures global

semantic information, the additional representational power does not substantially improve classification for research papers. We believe that this was due to training limitations where we could not train the Doc2Vec and the MLP classification models in an end-to-end fashion due to each model being loaded by different Python libraries. Thus, we had to train using block gradient descent, which may have led to slower convergence. Another point of improvement was in the lack of computational resources to do an comprehensive grid search, as Doc2Vec embeddings can be sensitive to hyperparameter choices and may underperform without extensive tuning.

The LSTM model achieves the highest balanced accuracy at 56.54%, indicating that a sequential approach where the models processes words one-by-one may capture higher contextual relationships in the title + abstract string. Due to the long and short term memory gates of the LSTM, we postulate that it was able to better distinguish between categories with overlapping vocabularies—such as `cs.AI`, `cs.LG`, and `cs.CL`, where subtle differences in phrasing or methodological emphasis are important. However, this model was by far the most expensive to train due to the higher complexity of the model, and we observed that it required more aggressive regularzation techniques (using weight decay and a dropout rate of 0.4) to prevent it from overfitting.

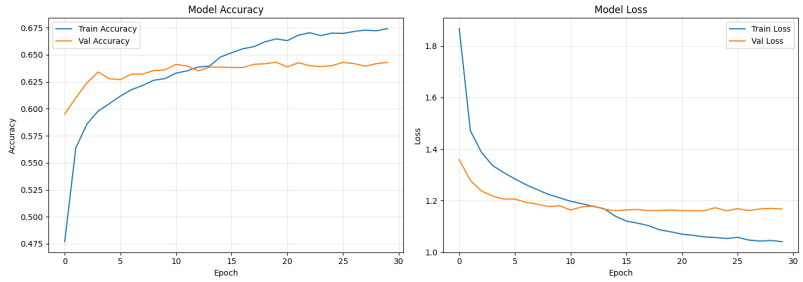

Figure 1: Doc2VecMLP: Training and validation curves for loss and accuracy over time.

As shown in Figure 1, the Doc2VecMLP model's training accuracy steadily increases from roughly 0.47 to around 0.67, while the validation accuracy follows closely, rising to about 0.64 before plateauing. The training and validation losses both decrease sharply over the first few epochs and then flatten out, with only a small gap between the two curves. This behaviour suggests that the Doc2Vec embeddings are expressive enough for the classifier to learn a useful decision boundary without severe overfitting, although the early saturation of the validation curve indicates that additional capacity or regularization is unlikely to yield large gains.

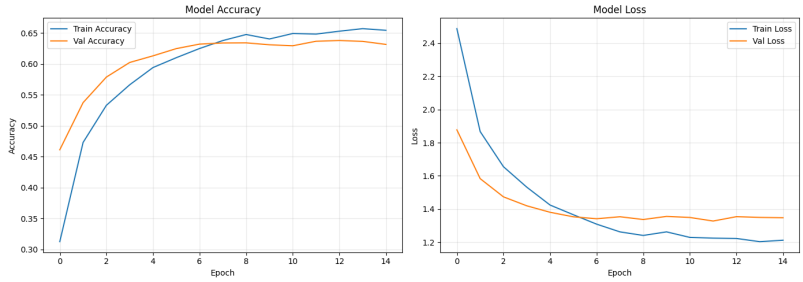

Figure 2: LSTM: Training and validation curves for loss and accuracy over time.

For a further analysis of the comparisons between the three models, we refer the reader to the appendix.

4.3 Qualitative Analysis

We augment our results with a qualtitative analysis of the Doc2Vec model. While neural networks like the LSTM are inherent black-box in nature, our hybrid embedding + MLP approach for the Doc2VecMLP model may allow for some interpretability. For example, in Figure 3 and 4, we show the latent output embeddings of the Doc2Vec model before it is forward propagated into the MLP. We observe that each point—representing a sample in a 256-dimensional space—seems to lie on a lower-dimensional manifold.

Overall, these results show that while classical methods remain competitive for scientific text classification, models that incorporate richer linguistic structure can yield meaningful improvements—particularly in domains with overlapping terminologies and nuanced contextual cues.

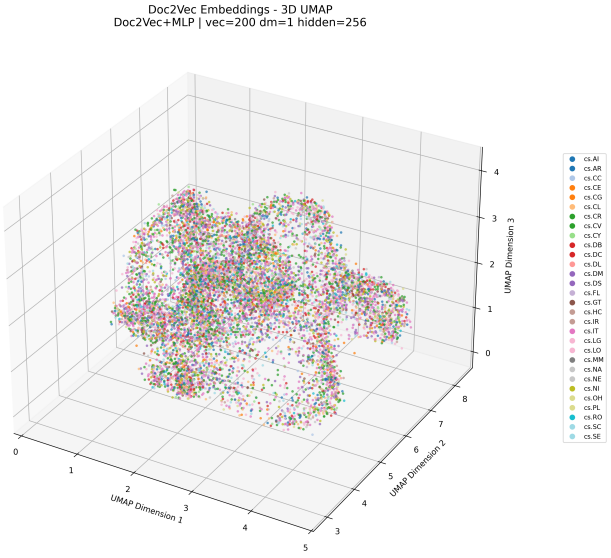

Figure 3: Embeddings of document vectors in $\mathbb{R}^2$.

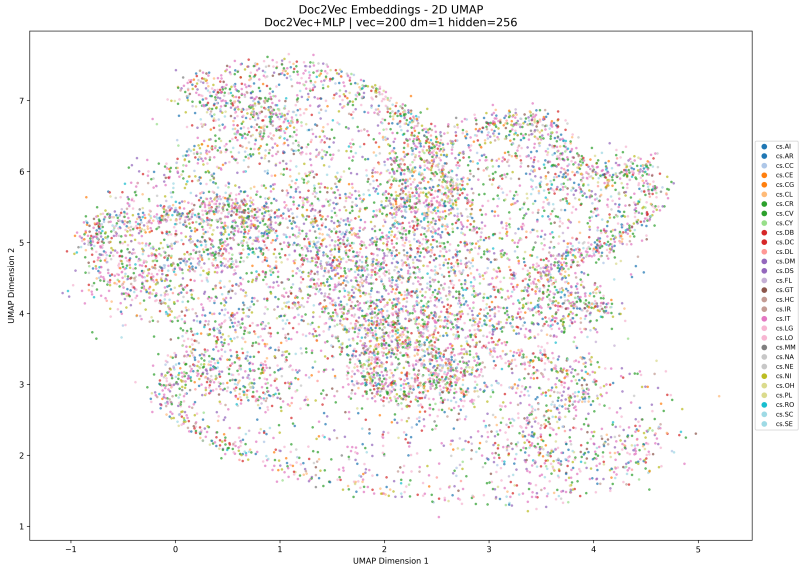

Figure 4: Embeddings of document vectors in $\mathbb{R}^3$.

5 Conclusion

Due to the ever-growing size of technical literature, it becomes increasingly difficult to keep up with the progress of each report, necessitating the need for such streamlined workflows. In this study, we have presented two neural approaches to automate classification of research papers in scientific domains: a bidirectional LSTM classifier, and a neural classifier appended to a fine-tuned Doc2Vec feature extractor. We have successfully demonstrated that extending the architecture to adapt to the structure of title + abstract summaries—either through sequential processing or one-shot embedding—improves the balanced accuracy of these models compared to the baseline.

There are several factors limiting the scope of this project. First, we consider only a small subset of arXiv's computer science categories and limit ourselves to the title and abstract, which may not capture the technical depth or cross-disciplinary nature of many papers. Second, the labels themselves are noisy: arXiv categories are author-assigned and frequently cross-listed, so different but related areas - such as cs.AI, cs.LG, cs.CL - can be difficult to separate even for human experts. Third, class imbalance and limited computational resources constrain the range of hyperparameters and model sizes that we can explore, especially in the case of the bidirectional LSTM. Finally, we evaluate on a single snapshot of the dataset and do not explicitly study how these models behave under distribution shift as arXiv content and writing styles evolve over time.

A future extension of this work may investigate more expressive architectures, such as transformer-based encoders like BERT or SciBERT, applied to the same title and abstract inputs for an improved neural baseline. Another natural direction is to incorporate richer context-such as full-text sections, author information, or citation signals-to do a better job of capturing subtle topic boundaries between closely related subfields. Since many arXiv papers are cross-listed, reframing the task as multi-label classification rather than single-label prediction would more closely match the real structure of the data. Beyond raw accuracy, future work could also examine model interpretability and calibration-for example, identifying which phrases drive a prediction and how confident scores behave-with an eye toward downstream applications like search, recommendation, or automatic routing of submissions to appropriate moderators.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

Sang-Woon Kim and Joon-Min Gil. Research paper classification systems based on tf-idf and lda schemes. *Human-centric Computing and Information Sciences*, 2019.

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, 2014.

Shadikur Rahman and et al. Automated research article classification and recommendation using nlp and ml. *arXiv preprint arXiv:2510.05495v1*, 2025.

Aleksandr Romanov, Konstantin Lomotin, and Ekaterina Kozlova. Automatization of scientific articles classification. In *CEUR Workshop Proceedings*, 2017.

Moritz Schubotz and et al. Automsc: Automatic assignment of mathematics subject classification labels. *arXiv preprint arXiv:2005.12099*, 2020.

Rosanna Turrisi. Beyond original research articles categorization via nlp. *CEUR Workshop Proceedings*, 2023.

Heng Yang and et al. Research on the automatic subject-indexing method of academic papers based on climate change domain ontology. *Sustainability*, 2023.

6 Appendix

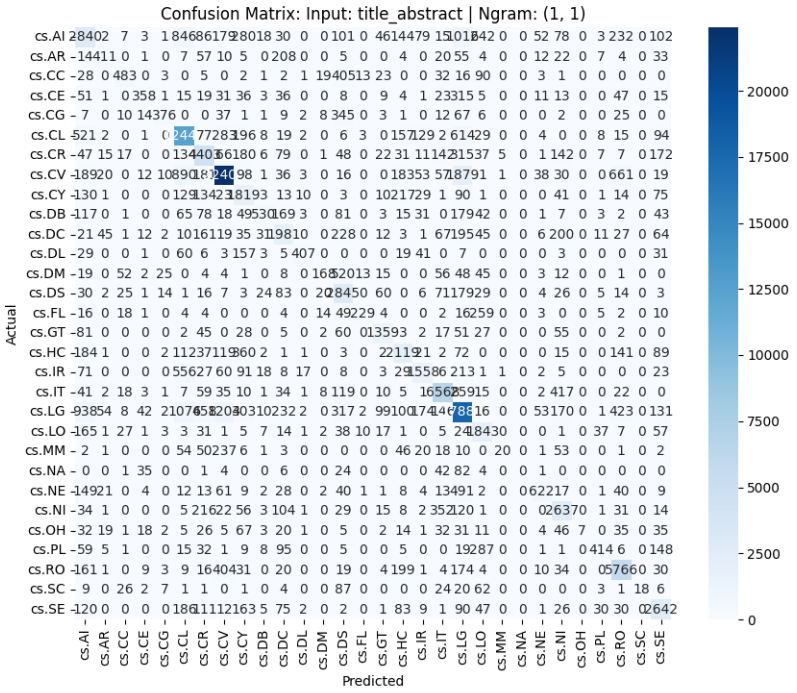

Figure 5: Confusion matrix for Naive Bayes model.

The confusion matrix for Naive Bayes model shows that it was able to achieve high accuracy in certain classes such as cs.CV(Computer Vision), cs.CL(Computation and

Language), and cs.LG(Machine Learning). However, it tends to confuse between different subfields more than the other two neural architecture modes, which suggest that it is less capable of analyzing the contextual overlap between closely related topics.

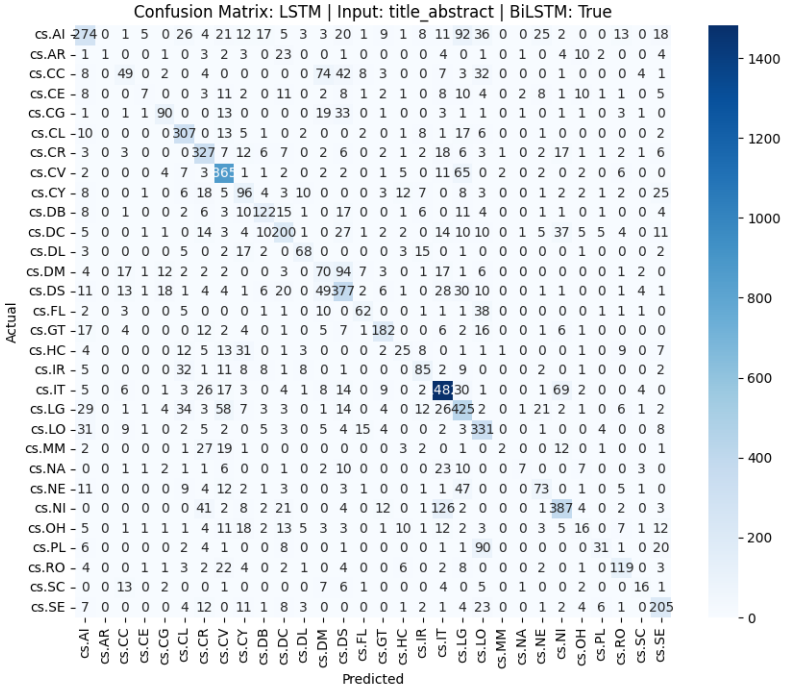

Figure 6: Confusion matrix for LSTM model.

In case of LSTM model, we can see that it mostly confuses subfields such as cs.DS(Data Structure) and cs.DM(Discrete Mathematics) and cs.IT(Information Theory) and cs.LG (Machine Learning), but overall, it shows a clear diagonal concentration.

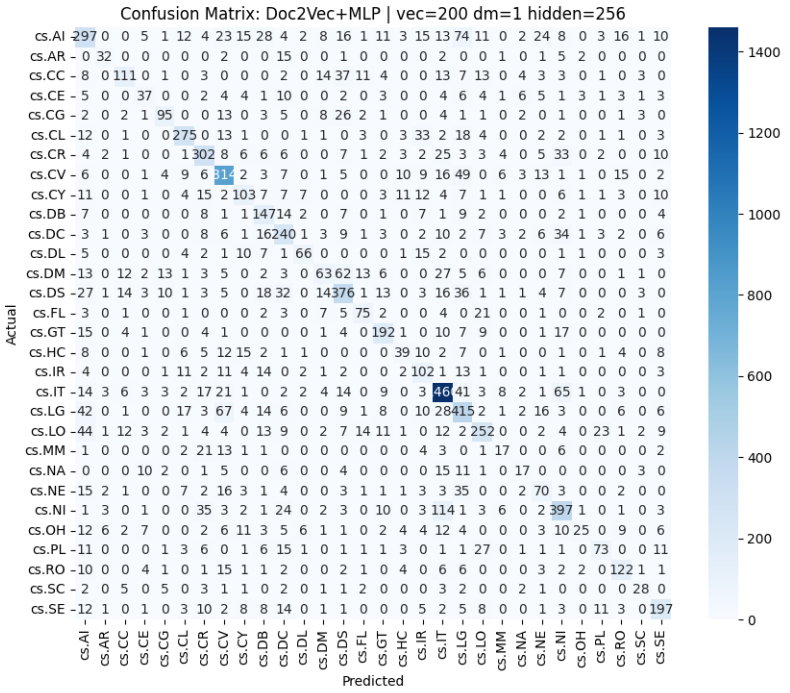

Figure 7: Confusion matrix for Doc2Vec with MLP model.

For Doc2Vec with MLP model, the confusion reflected in the LSTM model is also visible, but again, there is a clear diagonal concentration. This suggests that when the boundaries between the subfields are vague, this model tend to be more robust.