Inferring the Topic(s) of Wikipedia articles

Marina Zavalina
mz2476@nyu.edu
New York University

Peeyush Jain
pj891@nyu.edu
New York University

Sarthak Agarwal
sa5154@nyu.edu
New York University

Chinmay Singhal
cs5597@nyu.edu
New York University

Isaac Johnson
isaac@wikimedia.org
The Wikimedia Foundation

ABSTRACT

Thousands of new articles are added to Wikipedia everyday in more than 300 languages that Wikipedia supports. Once added, it is useful to classify these articles into a much smaller set of general categories for applications such as inferring user interests, recommending articles to read or edit, etc. Any manual or rule-based classification approach is tough and prone to errors given the size and diversity of Wikipedia articles. Currently, Wikipedia deploys a gradient boosting based machine learning model that does this classification automatically. The model has decent performance on English articles but does not do well when extended to other languages. In this paper we present a neural network based topic classification model that performs significantly better than this existing model. Our results show that upon fine-tuning, this model is also able to generalize well to articles in different languages and articles belonging to newly added categories. Additionally, we explore a variety of approaches consisting of language dependent methods such as LSTMs & self-attention based transformer networks that have achieved state of the are results in various language processing tasks. We further evaluate language independent methods such as Graph neural networks and article metadata based models, that do not use any article text but rather provide efficient and scalable results.

1 INTRODUCTION

Around 20 million unique Wikipedia articles exist across almost 300 different languages. These articles cover an incredibly rich spectrum of topics from cities in Greenland to the Universe to Harry Potter Influences and Analogues. It is incredibly useful to be able to classify these millions of articles into a much smaller set of general categories for applications such as inferring user interest [19], recommending articles to read or edit [1], or summarizing statistics about what types of articles readers view[1].

Currently, Wikipedia uses a Gradient Boosting based model called Drafttopic [1] for this topic classification task. It does so by taking an average of the word embeddings associated with each word in an article and learns a classifier on top of that article representation. While this model is simple and effective, it has the drawback that it is currently implemented only for English and is difficult and resource-intensive to scale it to more languages.

The goal of this project is to improve upon this existing classifier and also explore a variety of alternative approaches that better scale to more languages. The questions we try to answer are - (a) Can we improve the topic classification performance by using specialized deep learning approaches? (b) Can we transfer the models trained on English articles to classify articles in other languages? (c) Can we incorporate Wikipedia metadata to achieve better classification results? (d) Given a new topic with a very few articles associated with it, can we classify those articles accurately as well?

However, there are certain challenges that we face. The distribution of classes is imbalanced i.e for some topics we have around 30,000 articles while for others there may be only 30 articles. This class imbalance makes the multilabel classification more difficult as the model is then not able to learn equally well for all classes. Also, we have ground truth labels only for English articles and correspondingly this means that we have labels for only those articles in other languages that are aligned with these English articles. We present our approaches keeping these challenges in mind.

In this paper, we first compare different approaches for multilabel classification on English articles. In Section 3.1 we present a simple Bag-of-Words model. In Section 3.2 we experiment with other neural network architectures like LSTMs [8], self attention [12] and transformers [18] to check if using such advanced models leads to improves performance. Secondly, we explore if machine learning models trained on just English articles can be expanded to other languages. In Section 3.3 we evaluate a model that simply replaces English-only word embeddings with multilingual word embeddings in place. Thirdly, we leverage domain-specific knowledge of Wikipedia (as opposed to using all Wikipedia article text) to improve model performance and make the models more scalable. In Section 3.4 we use Wikipedia section heading & Wiki Links (in-links & out-links) to build machine learning models that leverage this information and compare model performance to standard natural language models. Finally, in Section 3.5 we explore transfer learning to deal with the occurrence of a new topic. We try to fine tune the last layer of our pre-trained model to classify articles belonging to the new category. We also explore cosine similarity based one-shot learning methods that are efficient and give decent results.

2 RELATED WORK

A lot of prior work has been done on topic classification using machine learning on Wikipedia datasets and also in general. Asthana and Halfakar [1] build a topic model, named *Drafttopic* using folksonomy developed and maintained by groups at Wikipedia (WikiProjects), that focus on specific subjects. The authors are able to create a mapping from articles to categories using this folksonomy and a training set of tagged articles. They use word2vec word embeddings

[1] https://analytics.wikimedia.org/published/datasets/one-off/English%20Wikipedia%20Page%20Views%20by%20Topics.html#Top-50-articles-read-in-March-2019-on-English-Wikipedia

for topic related feature extraction and explore Random Forests, Gradient Boosting classifiers and SVMs algorithms. The gradient boosting model, which outperforms other approaches, is currently implemented by Wikipedia to perform a multilabel classification over 44 mid-level categories. This model forms the baseline for our exploration.

Deep Learning approaches to Text Classification: Recurrent Neural Networks (RNNs) have shown promising results in text classification problems and have achieved state-of-the-art results in a variety of language tasks [13]. They treat the input as a sequence of tokens and use a hidden vector that acts as a memory, keeping previous input representations in mind while solving any task. The major advantage of RNNs is that they can process inputs having arbitrary lengths. But they are vulnerable to the problems of vanishing and exploding gradients [3] and thus can hinder learning. Long Short-Term Memory networks (LSTMs) [8] solve this issue by using multiple gates that regulate information flow in each node state. Although, LSTMs can take care of the sequence structure, they may lose the ability to give higher weights to more important words in a sentence. Vaswani et al. [18] use an attention mechanism to extract words that are most important to the meaning of a sentence. Lin et al. [12] propose a new model for extracting sentence embeddings by introducing self-attention. These embeddings can be used for downstream tasks like sentiment analysis, text classification etc. and show significant performance gains compared to other approaches.

Multilingual Classification: Classifying articles in just one language is a relatively common task but to be able to classify them in several languages, which also forms the basis of our analysis, multilingual embeddings are needed. Joulin *et al.* [4] offer a method for aligning word embeddings learnt separately for each language and publish trained embeddings for 44 languages [10]. The language-specific word embeddings are first trained on Wikipedia articles. These word embeddings (for each pair of language) are then aligned using a linear transformation matrix learned by minimizing a Relaxed CSLS (cross-domain similarity local scaling) loss which is explained in the paper.

Graph Based Topic Classification: Recently Graph Neural Networks (GNN) have attracted wide attention [2]. GNNs were first proposed by Scarselli et al. [17] and have been used in many natural language processing tasks. [6]. Defferrard et al. employed Graph Convolutional Neural Networks (GCN) for text classification and outperformed the traditional CNN models [6]. Kipf and Welling [11] propose a model for learning suitable node features $H_L \in R^{n \times d}$ (where n is number of nodes, d is dimension of node feature vector and L is the number of layers), such that H_L captures graph topology along with individual node features for improved latent representations. The layer-wise propagation rule is based on a first-order approximation of spectral convolutions on graphs. Hamilton *et al.* [7] propose a framework called GraphSAGE (SAmple and aggreGatE) that generalizes the GCN approach. The model can generate embeddings for unseen nodes based on features such as article text, extending GCN to the task of inductive unsupervised learning. The idea is to train a set of aggregator functions that learn to aggregate feature information from a node's neighbors and the node itself. Considering Wikipedia as a huge graph where each article is a node and the links between pages are edges, these graph based approaches can definitely assist our topic classification task.

Transfer learning. Apart from these novel model based approaches, being able to leverage the knowledge from one model, which is trained on a specific task, and using it to fine-tune another model, which may focus on a similar or different task, is an extremely useful scenario. Transfer learning aims to achieve this exactly. Pan and Yang [15] provide a detailed review of classic transfer learning techniques for various machine learning tasks such as classification, regression etc. A lot of work has also been done on natural language processing and especially on training one language model and then using these pre-trained weights to solve other downstream tasks. Ruder et al. [16] provide a comprehensive overview of the current state of transfer learning in the field of understanding language and text, which is crucial for our problem.

3 METHODOLOGY

3.1 Bag-of-Words (BoW) NN Model

Our first task was to improve upon the performance of Wikipedia's current Drafttopic model. As we don't have access to the deployed Drafttopic model, we recreate the model in PyTorch as a baseline and verify that performance is similar. We implement a simple and efficient Bag-of-words (BoW) neural network (NN) model with 1 layer (in other words, BoW logistic regression or NN with no hidden layers) for our multilabel topic classification problem. We denote this model as *Drafttopic-NN*.

The architecture of our Drafttopic-NN model is inspired by the original Drafttopic model but is not exactly the same. Similar to Drafttopic, we use pre-trained word embeddings, but here we use Facebook's fastText word embeddings [10] instead of word2vec embeddings trained on Googles News [14]. There are two reasons why we use fastText. Firstly, these embeddings are trained on Wikipedia articles, which is an advantage as we are working with the same data. And secondly, we explore multilingual approaches later in our work, and fastText provides pre-trained aligned word embeddings for different languages, which means similar words in different languages have similar vector representations. Also, we use the average of the word embeddings for all the words in an article to calculate an article's embedding and then pass it through a classification layer, similar to what is used in Drafttopic. Figure 1 shows the architecture of our Drafttopic-NN model.

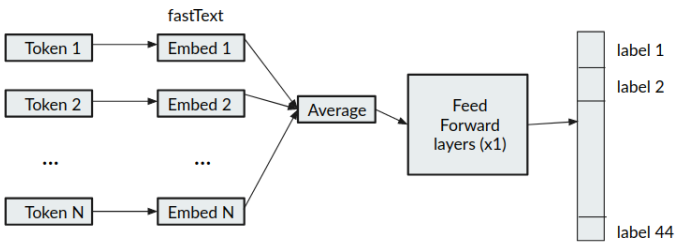

Figure 1: Drafttopic-NN model architecture

The main difference between these two models is that Drafttopic is a gradient boosting model, while for our implementation we use

a neural network with fully-connected layers. This choice is made so that we can explore other advanced NN models, such as BoW NN with various hidden layers and recurrent neural network (RNN) models.

We compare our baseline Drafttopic-NN model, with the existing Drafttopic model, as well as with deeper BoW NN models. We try different numbers of layers, as well as try out different hyperparameters such as optimizer, the dimension of the hidden layers, dropout and learning rate. We perform this analysis only on English articles in order to compare model performance.

3.2 Alternative NLP Models

3.2.1 Basic LSTM Model: In the Bag-of-Words model, we lose information about the sentence structure as it treats an input sequence as an unordered collection of words. Recurrent Neural Networks (RNNs) solve this issue by updating their hidden representation sequentially for each word in the sequence. LSTM networks remedy some shortcomings associated with learning long sequences by adding gated functions and thus are more efficient compared to RNNs. As a first alternative multilabel classification model, we try a one-directional (left to right) LSTM architecture. The network uses 300-dimensional pre-trained fastText word embeddings which are passed through an LSTM network with a 64 hidden size followed by a linear layer to transform the final hidden layer into class logits.

3.2.2 Self Attention Based Model. The next architecture we experiment with is an LSTM network with self attention. The model is based on the work of Linet al. [12]. It consists of two parts. The first part is a bidirectional LSTM, and the second part is the self-attention mechanism, which provides a set of summation weight vectors for the LSTM hidden states. The weighted sum of hidden states is used to create a sentence embedding which is used further in the classification task. Figure 2 shows the architecture of our self-attention based LSTM model.

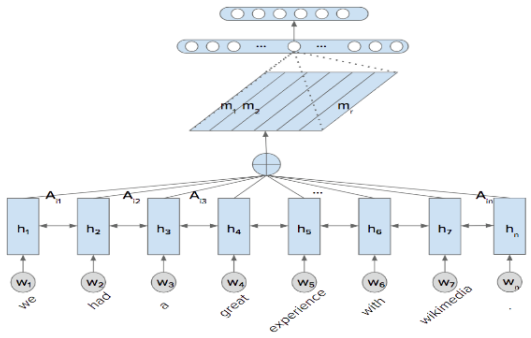

Figure 2: Self-attention LSTM architecture

A sentence S with n tokens is represented as a sequence of word embeddings: $S = (w_1, w_2, ..., w_n)$ (we limit n to 1,000). To gain dependency between adjacent words, we pass it through a one-directional LSTM, as opposed to bidirectional LSTM as used in the paper, to reduce model complexity. The LSTM outputs n hidden states, one for each time step. For convenience, let H be the matrix of all hidden states. The dimensions of H are $(n \times u)$ where u is the hidden size for the LSTM (64 in our model). The attention weights are calculated using -

$$a = \text{softmax}(w_1 * \tanh(W_2 \cdot H^T))$$

Here W_2 is a weight matrix of size $d \times u$ and w_1 is a vector of size d, where d is a hyperparameter. a is of size n with entries summing to one and act as the weights for the hidden states.

3.2.3 Inverse Document Frequency Based Model. In this approach, we use a similar architecture as the previous case but instead of the self attention weights, we use softmaxed inverse document frequencies of the words as weights. Article embeddings are formed using these weights and this approach runs faster than the previous approach because of the simplicity of this model.

3.2.4 Transformer Based Model. Finally, we explore a transformer architecture (hidden size 256) with two encoder layers, each with 4 self attention heads, followed by linear layers with ReLU, and an output layer to transform the input into class logits.

3.3 Multilingual Embeddings

Another challenge we tackle in this work is to expand the topic classification model to languages other than English. We test two approaches of extending the model from English to Russian and Hindi: (a) Fine-tuning model learned on English articles to articles in other languages, (b) Training one multilingual model for all three languages at the same time.

A logical step is to transfer topic classification models discussed in previous sections to articles in languages other than English. For both BoW NN and LSTM, the only part of the model that significantly depends on the language is the word embeddings matrix. Other parts of the model can be shared as long as the article representations are aligned among different languages. So we use fastText aligned word embeddings to create these aligned article representations. FastText provides pre-trained aligned word embeddings for 44 languages including English, Russian and Hindi.

First, we test both our best BoW NN and Self-Attention based LSTM models trained using English articles on articles in Russian. In the experiment, we use plugin aligned embeddings for Russian language (instead of English embeddings). Then we also fine-tune the model on Russian articles and compare the performance of this setting with a setting where the model is trained on Russian articles from scratch.

Second, we train our best BoW NN model on a mix of articles in several languages. For example, we train a multilingual model on a mix of articles in English and Russian and see how well it performs on these languages compared to monolingual models. Additionally, we test how well our multilingual model extrapolates to a new language, e.g. Hindi.

3.4 Metadata based Models

Wikipedia articles contain several other components besides the article text which can be explored for topic classification. In this approach, we identify these alternate data sources and run experiments to analyze their effect on model performance. Wikipedia articles are organized in sections. Sections improve the readability of articles and provide a natural pathway for editors to break

down the task of expanding a Wikipedia article into smaller pieces. Our first metadata-based approach uses section headings instead of using all of the text of a Wikipedia article. The model used is the same as described in section 3.1. It uses fastText word embeddings to obtain a representation for the section heading which is then fed to a feedforward neural network for performing multilabel text classification.

Another approach that we explore makes use of inlinks and outlinks corresponding to each Wikipedia article. Inlinks refer to the connections coming to the article from other articles within the Wikipedia database. Outlinks refer to the links present within an article that lead to other articles. We create a task-specific dataset that consists of article identifiers (IDs) along with two columns containing the list of inlink and outlink articles. We remove the entries which do not have either inlink or outlink references and then find the representations for these features separately using a neural network. These learned embeddings are then concatenated together to get a consolidated representation for an article, which is finally passed through feedforward layers.

Apart from this BoW NN model, we explore graph neural networks (GNNs) as well. Graph approaches can be useful in a multilingual classification setting as basic graph features do not depend on language. In our case, articles and links between the articles form a directed graph structure naturally. We consider articles as vertices V, outlinks and inlinks as directed edges E and together they form a graph $G = \{V, E\}$. Each article v has a feature vector x_v, which we can for each node in a matrix X. Our goal is then to infer $p(y_v^i|X, G)$ for each article v and topic i.

Graph methods usually rely on the assumption that connected nodes are similar in some sense. In our case, links within articles can be used for topic classification because connected articles are more likely to have similar topics. For example, the article about Antarctica has an outlink to the article about the South Pole, both of which are related to the same topic - Geography.Antarctica.

One way to model message passing between nodes is by using Graph Convolution Networks (GCNs) [11]. We use a model called GraphSAGE [7] which has a better-optimized training algorithm as opposed to simple GCNs, as well as the ability to generate feature vectors for unseen nodes. We use the node ID as a node feature and try out the following hyperparameters: the dimension of node embedding, the dimension of hidden layers, number of hidden layers, aggregator functions and dropout.

3.5 Transfer Learning

We explore transfer learning in two ways for the problem in hand: (a) Given a trained classification model and sufficient labeled examples for a new category (which the model has not seen before) we fine-tune by changing the last layer of the model and classify articles belonging to this new category accurately (b) Given just a few, 10 articles, for the new category (in the case of insufficient labeled examples for a new task), we explore one-shot learning approaches to classify articles belonging to the new category accurately. We work with 5000 articles belonging to the "Human Rights" category which is not among the original 44 topics that our multilabel classification model has been trained on. These articles pass through the exact data pre-processing pipeline as other articles.

For scenario (a), we fine-tune our best performing model by changing the last layer to predict a binary output 1 (article belongs to "Human rights") or 0 (articles do not belong to "Human rights"). In order to get a balanced dataset, 5000 labeled examples for the negative class are randomly sampled from articles belonging to the "History and society" and the "Politics and Government" categories, both of which are among the 44 original categories. The choice of these two classes lies in the fact that history, society, politics, the government are all related to the domain of human rights and the model will thus have to learn better representations to be able to distinguish these categories. We try both variants of transfer learning: "Feature extraction", where we keep our best model weights frozen and just update the weights of the final binary classification layer and "Fine-tuning", where we update the weights of all layers in the network but initialize the weights using our best model. The optimizer we use is Adam with a learning rate of 0.001 and the *BCEwithLogitsLoss* in PyTorch.

For scenario (b), we experiment with a simple category embedding model for a start, i.e. we take the category name such as "Human rights" and create a reference embedding for this class by averaging the individual word embeddings for each word in the category named - "Human", "rights". We do the same for the negatively sampled category. Once we have this, we create a similar document embedding for the articles by averaging all the word embeddings for that article and then apply cosine similarity to see whether the model is able to classify articles into the "Human rights" category or not just based on the category names.

Furthermore, we tried a Siamese network [5] style approach for textual data. Consider we have only 10 articles for our new "Human rights" category. Our aim is still to correctly predict the articles belonging to this new class. As a reference image is used in Siamese networks, here we create a reference document for our new category by taking an average of the 10 document embeddings for the 10 articles and thus obtain a single reference document embedding for the category. We similarly obtain a reference embedding for the negative class and then use cosine similarity to see if we are able to classify accurately or not. The problem for this approach is that in images, it is very easy to obtain a reference image that accurately depicts one class like "dog" but, natural language being a subjective domain, it is extremely difficult to get one perfect reference representation that summarizes any category. We choose the category with the maximum cosine similarity value for prediction in both cases.

The goal from this transfer learning exploration is that once these methods work for English articles, we can extend the methodology to more languages using our models from the other multilingual approaches that we discussed before.

3.6 Data

We train our BoW model in section 3.1 and alternative NLP models in section 3.2 on a dataset consisting of 2% of all the English Wikipedia articles which is comprised of ~103,000 articles in total. For multilingual embedding based methods in section 3.3, we use ~33,000 aligned articles in three different languages - English, Russian & Hindi. Each of the articles in one language has a corresponding article present in the other two languages. The articles

are not exact translations of each other but are related to the same topic.

For metadata-based models in Section 3.4, we extract the information (section text or links) from our 2% English Wikipedia dataset and create our own task-specific datasets. For graph models, since articles in one language usually link to articles in the same language, we can work with them independently. Only 0.6% of the articles don't have any connections within the graph for the 33K English articles, so the articles are well connected. In Figure 3, we plot the word distribution across the English Wikipedia articles. The plot, as expected, has a long tail with articles having a low word count.

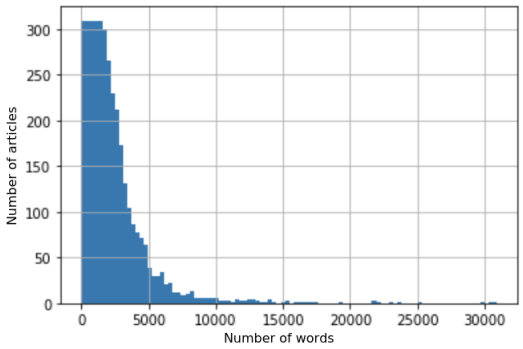

Figure 3: Word count distribution for English articles.

While performing data exploration, we also observe that the target labels do not have equal representation in the dataset. There are 44 different topics. Figure 4 shows the category distribution across all English articles in our entire dataset. For example, category *Culture.Language and Literature* has around 30,000 articles, while *Geography.Maps* has only 30.

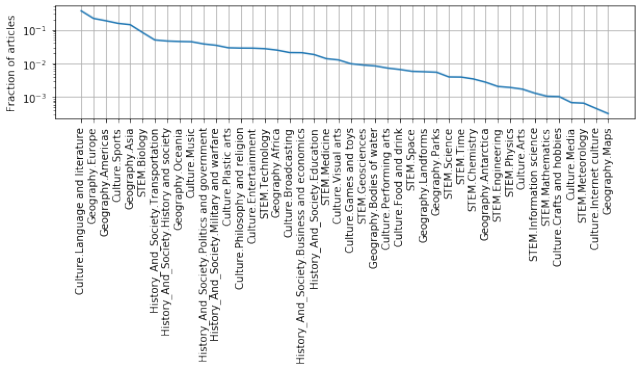

Figure 4: Distribution of topics.

3.6.1 Dataset preprocessing. Before evaluating our methods, we clean the datasets to remove irrelevant features that we do not want to include in our training process. The preprocessing step involves extracting text content from the raw JSON file. For this purpose, we use a modified version of the *wikifil.pl script*[2] from Matt Mahoney. We convert all the text to lowercase, then convert numeric characters to their equivalent text representations depending on the language of the article. We tokenize the text by creating a vocabulary of words from the training data and filter the stop words which do not represent a relevant signal for classification. For English & Russian articles, we use NLTK's inbuilt stop-word list whereas for Hindi, as there is no list available we use the list published by Jha et al. [9].

3.6.2 Dataset split: . We split the processed articles set into three parts: we use 80% of the articles for training the different methods, 10% for testing, and 10% as a validation set for hyperparameter tuning. For aligned articles in English, Russian and Hindi we make sure that each triplet goes together in the same split.

3.7 Evaluation Metrics

The classification problem in hand has 44 classes and the articles can belong to more than one of these classes. This corresponds to a multilabel classification problem. In order to make things simpler, we treat this as a one-vs-rest classification problem for each class and make use of the binary cross-entropy loss (*BCEWithLogitsLoss*) in Pytorch.

The input to the loss function is a matrix of size Batch Size x Number of Classes. A sigmoid activation layer is applied to every element individually and the binary cross-entropy loss is calculated. The loss outputs the mean of all these class-wise BCE values. *BCEWithLogitsLoss* combines a sigmoid layer and the *BCELoss* in one single python class, making it more numerically stable than using a sigmoid layer followed by a *BCELoss* as then we can take advantage of the log-sum-exp trick for numerical stability.

To get predictions, we apply a sigmoid function to the final logits and use a threshold of 0.5. We then calculate per class metrics: precision, recall, F1 score, and aggregated metrics: (micro/macro) precision, recall, F1 score. We use the micro F1 score as the evaluation metric for assessing model performance and for hyperparameter tuning, as our dataset contains imbalanced classes.

4 RESULTS

We present here the results for all our explorations mentioned in the methodology section.

In order to improve upon the baseline set by our Draftopic-NN model, we try the following architecture and hyperparameter settings for the BoW NN model: (a) Number of hidden layers: 2 or 3 (b) Optimizers: Adam or Stochastic Weighted Averaging (SWA), (c) Hidden layer dimension: 40, 80, 120, (d) Dropout probability: 0 (no dropout), 0.1, 0.2, (e) Learning rate: 0.1, 0.01, 0.001. All these models use fastText word embeddings for mapping words to a 300-dimensional numeric vector representation and are run for 10 epochs. Sigmoid activation and a threshold of 0.5 were applied on top of the model outputs (logits) to get the final predictions.

Our best model, which we call Drafttopic-NN++, was able to achieve the best micro F1 score of 0.816. It uses two hidden layers, a hidden dimension of 120, a dropout probability of 0.1 and the Adam optimizer with a learning rate of 0.1. A comparison of results of

[2] http://mattmahoney.net/dc/textdata.html.

Specification	Micro Precision	Micro Recall	Micro F1 Score
Frozen Drafttopic-NN++ trained on English articles	0.768	0.269	0.340
Finetuned Drafttopic-NN++ trained on English articles	0.819	0.747	**0.781**
Drafttopic-NN++ trained from scratch	0.835	0.671	0.744
Frozen Self-Attention LSTM trained on English articles	0.0654	0.742	0.120
Finetuned Self-Attention LSTM trained on English articles	0.828	0.711	**0.765**
Self-Attention LSTM trained from scratch	0.795	0.657	0.719

Table 1: Performance on articles in Russian language.

this best model with baselines is shown in Table 2. Results for the per-class metrics of our best Drafttopic-NN++ model are present in Appendix A.

Model	Micro Precision	Micro Recall	Micro F1 Score
Drafttopic	0.826	0.576	0.668
Drafttopic-NN	**0.835**	0.736	0.783
Draftopic-NN++	0.830	**0.802**	**0.816**
3 layer BoW NN model	0.829	0.791	0.810

Table 2: Comparison of BoW NN models

Our more advanced language-dependent approaches, such as LSTM, self-attention based LSTM, Transformer neural networks, don't further improve the scores achieved by our BoW models. The results for this comparison between the advanced approaches and our best Draftopic-NN++ model are shown in Table 3. We found that the Inverse document frequency (IDF) LSTM model performs almost similar to the self-attention based LSTM model and thus IDF values can be used as a replacement for attention weights, thereby reducing the number of parameters in the model and thus leading to a decrease in the training time.

Model	Micro Precision	Micro Recall	Micro F1 Score
Drafttopic	0.826	0.576	0.668
Drafttopic-NN++	0.830	**0.802**	**0.816**
Basic LSTM	**0.867**	0.754	0.807
Self Attention LSTM	0.861	0.774	**0.816**
IDF LSTM	0.833	0.777	0.804
Transformer	0.841	0.766	0.801

Table 3: Comparison of Language dependent Models

Using the best classification models obtained, we were also able to achieve promising results for our main task which was developing a multilingual model. Micro F1 score for a model trained on a new language improves when the weights are fine-tuned upon initialization from a pre-trained model for English articles. But the frozen model trained on English articles does not perform well on a new language (Russian). One possible reason can be a poor alignment of the multilingual fastText embeddings. These pre-trained embeddings are usually generated to solve a particular problem and may not generalize well to all tasks within the same umbrella of language processing. Overfitting of the model to a particular language (English), from which we initialize the weights, may also be a possible reason for the low performance. We try both the Drafttopic-NN++ and the best self-attention based LSTM models for this analysis, the results of which are shown in Table 1. Frozen Drafttopic-NN++ model has high micro-precision and low micro recall, while self-attention based LSTM, on the contrary, has low micro-precision and high micro recall.

For a given language, models that are trained on a mixture of aligned English, Russian and Hindi articles perform comparably to the monolingual models mentioned above i.e models that were only trained on articles in a given language. The training size for monolingual models is 10,000 articles and for the multilingual models, we have 10,000 aligned articles for each of English, Russian, Hindi languages in the training set. Figure 3 depicts a visualization showing the comparison of the mixed and the monolingual models.

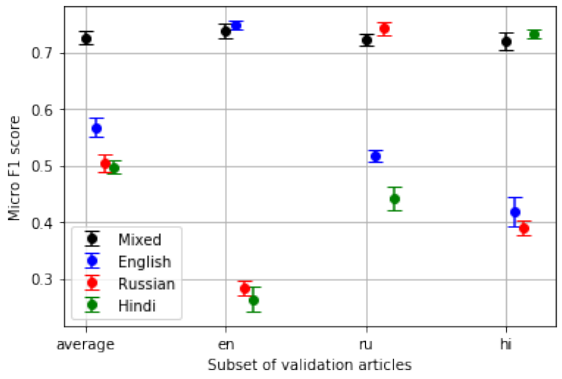

Figure 5: Monolingual vs Multilingual model comparison

For our language-independent model approaches, we explored a few metadata-based and graph-based approaches as mentioned in the methodology section. We observe that the links and section-header based approaches don't perform well in comparison to the language-based approaches that use the whole article text for prediction. GNN model that uses just connections between articles performs surprisingly well, GraphSAGE achieves a 0.642 micro F1

score. The results for these text-independent models are presented in Tabel 4.

Metadata	Micro Precision	Micro Recall	Micro F1 Score
Section Text	0.762	0.403	0.531
Wiki Links (BoW)	0.550	0.513	0.534
Wiki Links (GraphSAGE)	0.649	0.636	**0.642**

Table 4: Comparison of Language Independent Models

The transfer learning exploration also gives us a positive direction to move ahead in. These experiments are only performed for English articles and can be further extended to solve the multilingual problem. For our first scenario where we have sufficient labeled examples a fine-tuned Drafttopic-NN++ network gives us an accuracy of 90.37% for predicting articles belonging to a new category. Also for the second scenario where we have insufficient labeled examples, simple cosine similarity approaches also give us reasonably well accuracies as shown in Table 5. Although these results are fascinating, they need to be taken with a grain of salt. As our experiments only use examples from one new category "Human rights", it may be possible that our training set has articles that are quite similar to articles under this category and thus our models perform well. Further experiments with larger datasets are required to give a concrete estimate of how favorable transfer learning is for the problem in hand.

Model	Accuracy
Feature extractor model	84.98%
Fine-tuned model	**90.37%**
Class embedding model	**78.84**%
Reference document model	77.95%

Table 5: Transfer learning accuracies

5 CONCLUSION AND FUTURE WORK

In this paper, we explore different approaches for topic classification on Wikipedia articles. Wikipedia has an existing classification model that works on English articles but they are finding it difficult to get scale it to other languages. We replace the existing *Drafttopic* model by a Bag-of-Words based neural network model. The hyperparameter tuned BoW NN model achieves a much higher F1 score compared to the *Drafttopic* model. This BoW NN model also generalizes well on languages other than English and achieves good performance for Russian & Hindi articles after fine-tuning.

We evaluate basic and self-attention based LSTM models and get similar F1 score as obtained by the BoW NN model. We conclude that the BoW NN model outperforms these advanced language models as it trains much faster and has fewer parameters. We next train the models with feature representation built using section header text instead of the entire article text. We achieve a lower F1 score using this curtailed article representation.

We further explore two language-independent approaches (a) Creating BoW based vector embeddings for inlinks & outlinks for each article in our dataset and then using the concatenated representation for label prediction. The results obtained by this approach are not encouraging and we conclude that such features do not have enough discriminatory signals for text classification. (b) Graph-based approaches imply that the Graph NN model shows great potential as it achieves a decent micro F1 score using only link data as input. The advantage of this approach is its easier scalability to many languages.

A particularly interesting direction for future work is to explore Graph NN models further and combine text-based models with these graph models, e.g. by including node features such as text embeddings for articles. Also, weighted loss functions can be evaluated as the classification labels are not equally distributed. Trying out transfer learning on articles from newer sets of categories is also a possible future direction.

ACKNOWLEDGMENTS

We would like to thank our amazing mentor Isaac Johnson, Wikimedia Foundation for all his help and support throughout the project and also for all the effort he put in getting us quality datasets to work with. His ideas helped a lot in giving a proper direction to our project. We would also like to thank Anastasios Noulas, New York University for his constructive feedback and all his suggestions about graph models.

REFERENCES

[1] Sumit Asthana and Aaron Halfaker. 2018. With Few Eyes, All Hoaxes Are Deep. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 21 (Nov. 2018), 18 pages. https://doi.org/10.1145/3274290

[2] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR* abs/1806.01261 (2018). arXiv:1806.01261 http://arxiv.org/abs/1806.01261

[3] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (March 1994), 157–166. https://doi.org/10.1109/72.279181

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS'93)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 737–744. http://dl.acm.org/citation.cfm?id=2987189.2987282

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *CoRR* abs/1606.09375 (2016). arXiv:1606.09375 http://arxiv.org/abs/1606.09375

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *CoRR* abs/1706.02216 (2017). arXiv:1706.02216 http://arxiv.org/abs/1706.02216

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[9] Manjunath; Shenoy P Deepa; K R Venugopal Jha, Vandana; N. 2018. Hindi Language Stop Words List. *Mendeley Data* 1 (2018). https://data.mendeley.com/datasets/bsr3frvvjc/1

[10] Armand Joulin, Piotr Bojanowski, Tomas Mikolov, and Edouard Grave. 2018. Improving Supervised Bilingual Mapping of Word Embeddings. *CoRR* abs/1804.07745 (2018). arXiv:1804.07745 http://arxiv.org/abs/1804.07745

[11] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016). arXiv:1609.02907

http://arxiv.org/abs/1609.02907

[12] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. *CoRR* abs/1703.03130 (2017). arXiv:1703.03130 http://arxiv.org/abs/1703.03130

[13] Danilo P. Mandic and Jonathon Chambers. 2001. *Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability*. John Wiley & Sons, Inc., New York, NY, USA.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. (2013). arXiv:cs.CL/1301.3781

[15] S. J. Pan and Q. Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (Oct 2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[16] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer Learning in Natural Language Processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. 15–18.

[17] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *Trans. Neur. Netw.* 20, 1 (Jan. 2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[19] Ramtin Yazdanian, Leila Zia, Jonathan Morgan, Bahodir Mansurov, and Robert West. 2019. Eliciting New Wikipedia Users' Interests via Automatically Mined Questionnaires: For a Warm Welcome, Not a Cold Start. *CoRR* abs/1904.03889 (2019). arXiv:1904.03889 http://arxiv.org/abs/1904.03889

A APPENDIX

Table 6 shows the confusion matrix, Precision, F1 and recall for each label generated by the Drafttopic-NN++ model.

Category	Count	True Negative	False Negative	True Positive	False Positive	Precision	Recall	F1 Score
Culture.Arts	19.0	9976	11	8	1	0.888	0.421	0.571
Culture.Broadcasting	217.0	9716	56	161	63	0.718	0.741	0.730
Culture.Crafts and hobbies	14.0	9982	11	3	0	1.0	0.214	0.352
Culture.Entertainment	295.0	9598	57	238	103	0.697	0.806	0.748
Culture.Food and drink	67.0	9919	24	43	10	0.811	0.641	0.716
Culture.Games and toys	109.0	9879	19	90	8	0.918	0.825	0.869
Culture.Internet culture	6.0	9990	6	0	0	0.0	0.0	0.0
Culture.Language and literature	3631.0	6078	239	3392	287	0.921	0.934	0.928
Culture.Media	3.0	9993	3	0	0	0.0	0.0	0.0
Culture.Music	435.0	9489	90	345	72	0.827	0.793	0.809
Culture.Performing arts	72.0	9916	31	41	8	0.836	0.569	0.677
Culture.Philosophy and religion	274.0	9638	102	172	84	0.671	0.627	0.649
Culture.Plastic arts	302.0	9634	93	209	60	0.776	0.692	0.732
Culture.Sports	1567.0	8237	70	1497	192	0.886	0.955	0.919
Culture.Visual arts	139.0	9799	58	81	58	0.582	0.582	0.582
Geography.Africa	225.0	9741	72	153	30	0.836	0.68	0.750
Geography.Americas	1884.0	7655	436	1448	457	0.760	0.768	0.764
Geography.Antarctica	27.0	9969	2	25	0	1.0	0.925	0.961
Geography.Asia	1399.0	8353	191	1208	244	0.831	0.863	0.847
Geography.Bodies of water	81.0	9896	26	55	19	0.743	0.679	0.709
Geography.Europe	2168.0	7403	522	1646	425	0.794	0.759	0.776
Geography.Landforms	57.0	9919	26	31	20	0.607	0.543	0.574
Geography.Maps	1.0	9995	1	0	0	0.0	0.0	0.0
Geography.Oceania	468.0	9487	57	411	41	0.909	0.878	0.893
Geography.Parks	60.0	9922	34	26	14	0.65	0.433	0.519
History_And_Society.Business and economics	207.0	9723	116	91	66	0.579	0.439	0.5
History_And_Society.Education	181.0	9791	79	102	24	0.809	0.563	0.664
History_And_Society.History and society	449.0	9459	360	89	88	0.502	0.198	0.284
History_And_Society.Military and warfare	331.0	9614	73	258	51	0.834	0.779	0.806
History_And_Society.Politics and government	343.0	9557	140	203	96	0.678	0.591	0.632
History_And_Society.Transportation	551.0	9374	79	472	71	0.869	0.856	0.862
STEM.Biology	771.0	9171	83	688	54	0.927	0.892	0.909
STEM.Chemistry	36.0	9956	18	18	4	0.818	0.5	0.620
STEM.Engineering	22.0	9972	16	6	2	0.75	0.272	0.399
STEM.Geosciences	90.0	9892	33	57	14	0.802	0.633	0.708
STEM.Information science	10.0	9986	8	2	0	1.0	0.2	0.333
STEM.Mathematics	11.0	9985	9	2	0	1.0	0.181	0.307
STEM.Medicine	154.0	9824	59	95	18	0.840	0.616	0.711
STEM.Meteorology	5.0	9990	3	2	1	0.666	0.4	0.5
STEM.Physics	18.0	9977	14	4	1	0.8	0.222	0.347
STEM.Science	43.0	9951	43	0	2	0.0	0.0	0.0
STEM.Space	67.0	9924	9	58	5	0.920	0.865	0.892
STEM.Technology	275.0	9680	83	192	41	0.824	0.698	0.755
STEM.Time	29.0	9966	29	0	1	0.0	0.0	0.0

Table 6: Category-wise Result Matrix