Paragraph Topic Classification

Eugene Nho
Graduate School of Business
Stanford University
Stanford, CA 94305
enho@stanford.edu

Edward Ng
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
edjng@stanford.edu

1 Introduction

The project attempts to predict the topics of paragraph-length texts. We posit that machine learning algorithms can extend a person's capability to consume information through the automated categorization of text and through an increased capability to locate relevant information.

We formally define the problem as constructing a model that, given an input of text, outputs a vector of relevant topics. The project uses approximately 400 000 Wikipedia article summaries categorized by Wikipedia users a priori. A number of approaches were taken including Naive Bayes, One-vs-Rest Classifier (OvR) with GloVe Vectors, Latent Dirichlet Allocation (LDA)/OvR, GloVe Vectors/LDA/OvR, Convolution Neural Networks (CNN), and Long Short Term Memory networks (LSTM).

2 Related Work

We found a number of papers associated with topic classification, which we will segment into traditional, topic modeling, and finally the use of neural networks approaches to topic classification.

2.1 Traditional Approaches

The use of Naive Bayes in topic classification has been preeminent given its dual properties of being accurate and quick to train (Ting et. al (2011)). However, there are variations of Naive Bayes that would have likely improved our performance, specifically an algorithm developed by Wang et. al (2012) which used Naive Bayes log-count ratios as features into an SVM model (known as NB-SVM), and using bigrams rather than tf-idf. This combined model (with a small modification, known as NB-LM) is considered to be one of the state-of-the-art supervised learning algorithms for topic classification. Further research indicated the use of bigrams (and n-grams in general) does not reveal any improved performance (Tan et. al, 2002), leaving the implementation of NB-LM as the primary focus on any future implementation using traditional learning algorithms.

2.2 Topic Modeling

Another approach to topic classification is to presume that for a given dataset of corpora, there are n number of topics distributed, and that for each given document there is some probability that they belong to a given topic. We implement Latent Dirichlet Allocation

2.3 Neural Networks

The recent insurgent interest of neural networks have also pushed the application of neural networks to include topic classification. Convolution neural networks, initially targeted for image classification applications, have been modified to also perform natural language processing with a high rate of success and considered to be state-of-the-art for topic classification. We implement a rudimentary convolution neural network based on Kim et. al (2014) (with the slight modification of using GloVe vectors rather than word2vec) which provided the highest rate of accuracy out of all our algorithms.

However, there are a couple of variations that may have improved our performance on this task, including the use of individual characters as inputs (Zhang et. al (2015)) and training embeddings through the use parallel convolution layers with differing region sizes (Johnson et. al (2014)).

The increased complexity of neural network topology is often associated with the need for larger training data sets, with Zhang et. al (2015) explicitly mentioning that a dataset of several millions needs to be provided for character-level convolutional neural nets to outperform more traditional approaches. In order for these other topologies to function well, our dataset of Wikipedia articles would have been required to be much larger. However, using parallel convolution layers may have been an area of future exploration.

3 Dataset

We used Wikipedia articles summaries as our data. Each article's first summary paragraph served as our input, and its topic assignments as our labels. We scraped Wikipedia for total $\approx 400,000$ inputs, all belonging to one or more of the following five topics: mathematics, politics, computer science, film, and music.

We limited our scope of topics to five to test the efficacy of our models without requiring an unwieldy amount of data, and thus computational resources. Our training and test sets were 275 000 and 40 000 entries respectively, with an average word count of 93 for each summary.

We used Wikipedia because it is one of the few available sources of topically labeled, multilabel dataset. We insisted on solving a multilabel problem because, in real-world applications, a piece of text is rarely about one topic only. An example entry is:

> In mathematics, the minimum k-cut, is a combinatorial optimization problem that requires finding a set of edges whose removal would partition the graph to k connected components. These edges are referred to as k-cut. The goal is to find the minimum-weight k-cut. This partitioning can have applications in VLSI design, data-mining, finite elements and communication in parallel computing.

4 Features

We used term frequency-inverse document frequency (tf-idf) and GloVe as our main features.

4.1 Term Frequency-Inverse Document Frequency

We used tf-idf because it captures how important a word is to a document in a corpus by calculating the frequency of a word in a given document offset by its frequency in the overall corpus, de-emphasizing common words like "is" and "the." It is also a common baseline feature for information retrieval tasks. The weighting scheme used for tf-idf is the default provided by sklearn.

4.2 Global Vectors for Word Representation (GloVe)

Later in the project, we used pre-trained GloVe to feed richer representation of input text into our models. GloVe represents words as vectors, with the vector representing the co-occurrence of the word with other words over a global set. The cost function which GloVe minimizes is

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

where P_{ij} is the probability of word i occurring in the context of j, W represents the entire vocabulary, f provides is a given weighting function, and u and v are the vector representations of i and j respectively.

Our hypothesis for using GloVe vectors was that they would more effectively capture the underlying relationships between words characterizing the same topic because related words are likely represented by similar vectors. We used pre-trained GloVe vectors from Stanford's NLP group[1] (trained on Wikipedia 2014 corpus, 100-dimensions).

5 Methods

Our overall approach to text classification was starting with a simple baseline model and gradually adding more sophisticated features or model architecture to see if and how the results improve. Table 1 summarizes our approaches and rationale.

Table 1: Summary of approaches and rationale

Model Approach [features]	Rationale
Naive Bayes [tf-idf]	Common baseline model for text classification
OvR [GloVe]	One-vs-Rest supports multilabel learning; richer feature (GloVe)
LDA + OvR [tf]	To capture latent topics more effectively
LDA + GloVE + OvR [tf]	Complementary behavior (word embedding + topic distribution)
CNN [GloVe]	Fast; generalizes well; local word order is not important

5.1 Baseline: Naive Bayes

We used Naive Bayes for our baseline implementation. Naive Bayes is often a go-to strategy for baseline given its speed and ease of its training. We trained Naive Bayes from sklearn for multiclass topic classification, where paragraphs that fit into multiple topics were classified to be under a single compound topic. Since we limit ourselves to 5 topics, there are $2^5 = 32$ possible labels to choose from. Input feature was a tf-idf matrix.

5.2 GloVe + One-vs-Rest SVM classifier

To test our hypothesis that feeding richer representation of our input text would improve the results, we used GloVe as our feature and fed it into a linear Support Vector Machine (SVM) classifier using One-vs-Rest strategy (OvR). We chose OvR because it enables us to do multilabel classification by constructing k SVMs, where k is the number of classes. Given m training data ($x^{(1)}$, $y^{(1)}$),..., ($x^{(m)}$, $y^{(m)}$), where $x^{(i)} \in \mathbb{R}^n$ and $y^{(i)} \in \mathbb{R}^k$ indicates classes $x^{(i)}$ belongs to, the lth SVM solves the following problem:

$$\min_{\vec{w}_l, \vec{b}_l, \xi_l} \frac{1}{2}\vec{w}_l^T \vec{w}_l + c\sum_{i=1}^{m} \xi_l^{(i)}$$
$$\text{s.t. } \vec{w}_l^T x^{(i)} + b_l \geq 1 - \xi_l^{(i)}, \text{if } \vec{y}_l^{(i)} = 1$$
$$\vec{w}_l^T x^{(i)} + b_l \leq -1 + \xi_l^{(i)}, \text{if } \vec{y}_l^{(i)} = 0$$

where $\xi_i \geq 0$ for all i and c is a penalty parameter. In our implementation, $x^{(i)}$ was constructed by first vectorizing each word of the ith example into a $\mathbb{R}^{100}$ vector using the pre-trained GloVe weights, and then collapsing the matrix into a single vector using tf-idf as the weighting factor.

For a new example x, OvR conducts the following for each l: $\vec{y}_l = \text{sgn}(\vec{w}_l^T x + b_l)$

5.3 LDA + One-vs-Rest SVM classifier

LDA is a 3-level hierarchical Bayesian model often used for topic modeling in natural language processing. Fully explaining the model is outside the scope of this paper, but one can get an intuition by observing how a single document is modeled:

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta$$

where α and β are parameters, θ~Dirichlet(α) is a random variable representing topic mixture, **w** is the document, z_n is a topic for n^{th} word of the document, and w_n is $n_t h$ word. Intuition is that the term inside parenthesis, representing each document, is modeled as $p(z_{zn})p(w_{dn}|z_{dn})$. In other words, each document is modeled as random mixtures over latent topics, where each topic is characterized by a distribution over words. Our hypothesis for choosing LDA was that modeling

[1] http://nlp.stanford.edu/projects/glove/

documents this way would help us addresses head-on a core challenge we were facing:"How can we classify documents based on underlying themes, rather than presence or absence of certain key words?"

Since LDA is an unsupervised learning model, we used its output, document topic matrix (number of inputs $\times$ number of topics), as our design matrix X that gets fed into the OvR classifier.

5.4 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNN) apply layers of convolving filters to features. Traditionally CNNs have been to computer vision, but recently they are increasingly applied to natural language processing problems. Instead of applying filters to image pixels, CNNs in NLP apply the filters to the matrices representing paragraphs, typically consisting of rows of word embeddings such as word2vec or GloVe.

We will not define CNNs in mathematical formalism in this paper, but the essence of CNNs can be seen at the convolution layers. Given the document matrix $x_1 : n \in \mathbb{R}_{n\times k}$, where x_i's (word embedding vectors like GloVe) for all n words are stacked together, convolution occurs when a filter $\vec{w} \in \mathbb{R}^{hk}$ is applied to a window of h words to produce a new feature. A feature c_i is generated from window of words $x_i : i + h - 1$ by

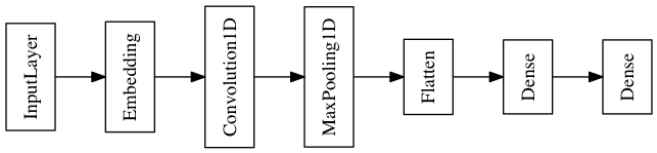
$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$$

where $b \in \mathbb{R}$ is a bias term and f is a non-linear activation function. This filter is applied to each possible window of words in the paragraph, producing $n - h + 1$ of c_i's above.

We chose CNN for two main reasons. First, we believe CNNs do what LDA does–capturing latent topics–but in a much more powerful way. By applying convolving filters and abstracting away from the raw word embeddings, CNNs can capture the "shapes of topics" the same way CNNs applied to image recognition capture the shapes of objects. Second, CNNs are fast and relatively easy to train with limited GPU resources, which enabled us to experiment with various hyperparameter settings without extensive infrastructure setup.

The model architecture is a multi-label variant of the CNN architecture proposed in Kim et. al (2014), using a static embedding layer, and an extra fully-connected layer. In order to do multi-label classification we used a binary cross entropy loss function and sigmoid activation.

Figure 1: Graph of CNN Model Architecture



5.5 Long Short Term Memory (LSTM)

LSTM is a Recurrent Neural Network (RNN) architecture that is well-suited to learn from sequential experience when long time lags of unknown size exist between important events. It accomplishes this by having gates: computing the following

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1}\right), f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1}\right), o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1}\right)$$

$$\widetilde{c}_t = \tanh\left(W^{(c)}x_t + U^{(c)}h_{t-1}\right), c_t = f_t \cdot c_{t-1} + i_t \cdot \widetilde{c}+_t, h_t = o_t \cdot \tanh(c_t)$$

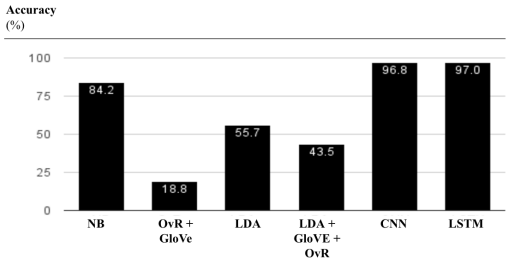
Intuition is that at each time t, given input x_t and hidden state from $t - 1$, LSTM cell has the power to decide whether to input, forget, or output the data based on function σ, thereby enabling the overall network to learn when "important events" may be close or far apart in a given sequence of data.

We tried LSTM mainly as a benchmark against our CNN results. RNN architecture is widely viewed as the go-to strategy for natural language processing (NLP) tasks because language is a sequential type of data. Our model architecture is defined by an embedding layer, a single layer of 200 LSTM cells, and a feed-forward layer using sigmoid activation. As with CNN, it is multi-label variant using a binary cross entropy loss function.

6 Results and Discussion

The overall results for our various approaches at paragraph topic classification are summarized in Figure 2 below. There are a few points worth noting about these results.

Figure 2: Metrics for all Approaches



	NB	OvR + GloVe	LDA	LDA + GloVE + OvR	CNN	LSTM
Precision	90	91	68	67	95	95
Recall	84	39	80	93	89	89
F1 score	86	53	73	76	92	92

First, Naive Bayes was very robust despite its simplicity, and given its preeminence in topic classification, this is not at all surprising.

Second, the GloVe and One-vs-Rest model performed very poorly on accuracy. We believe this has to do with the fact that we collapsed the GloVe document matrix into a vector to feed into the classifier. We suspect they began to converge toward similar, overlapping pattern the more data we used. This could be because GloVe synthesizes the relationship between words with vector distances, and with more data, different types of differences all get reflected with the similar distances. As a data point, GloVe + OvR achieved 40% accuracy when trained over only 10 000 samples.

Third, contrary to the initial hypothesis, LDA did not work well in topic prediction. We believe the principal reason is the difficulty to control the way an unsupervised algorithm learns. The output of LDA we used for classification–document topic matrix – is LDA's best guess about the likelihood of a given sample belonging to each of the five clusters it learned to differentiate between. A corpus with m documents could be clustered into five groups in 5^m different ways, and it is nearly impossible to exactly align the way an unsupervised learning algorithm clusters them with the way humans did a priori.

Fourth, both neural networks far outperformed other approaches with 97% accuracy. It is noteworthy that CNN and LSTM both ended up with very similar metrics, but CNN was much faster and robust. CNN's training time for 354 419 samples was on average 30 minutes on a laptop CPU, whereas LSTM took 3 hours for half the number of samples. Lastly, as the next section on hyperparameter tuning will discuss, CNN was very robust, showing consistently effective results across various hyperparameter settings. In contrast, LSTM's results varied widely depending on hyperparameters.

6.0.1 Hyperparameter Tuning

Our CNN model proved relatively stable over various hyperparameter settings. For example, deeper networks performed only marginally better, but at much higher train and test time cost. Going from 1 layer to 3, we saw an increase of ≈ 0.5 percentage points for both recall and precision. Changing dropout layers, number of fully-connected layers, number and size of filters, and activation function between ReLU and tanh did not yield more than 5% difference in accuracy.

On the other hand, LSTM was very sensitive to the number of epochs, going from 40% to 86% as we increased the epochs from 1 to 5. The results also varied depending on whether we allowed embeddings to be continuously updated (10% higher when allowed), as well as the number of LSTM cells per layer (80% for 100 cells vs 86% 200 cells). In the end, however, LSTM performed marginally better than CNN.

7 Future Work

We would like to further perform hyperparamter tuning of the models implemented, perhaps with grid search. In addition, given additional time, we would have ideally explored the weights assigned to our CNN and LSTM model to make sense of how a given corpus is weighted. We would also re-establish our baseline using the NB-SVM model proposed by Wang et. al (2012).

References

[1] Ting, S. L., Ip, W. H., Tsang, A. H. (2011). Is Naive Bayes a good classifier for document classification?. International Journal of Software Engineering and Its Applications, 5(3), 37-46.

[2] Yoo, J. Y., Yang, D. (2015). Classification Scheme of Unstructured Text Document using TF-IDF and Naive Bayes Classifier.

[3] Wang, S., Manning, C. D. (2012, July). Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2 (pp. 90-94). Association for Computational Linguistics.

[4] Blei, D. M., Ng, A. Y., Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

[5] Ramage, D., Hall, D., Nallapati, R., Manning, C. D. (2009, August). Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1 (pp. 248-256). Association for Computational Linguistics.

[6] Tan, C. M., Wang, Y. F., Lee, C. D. (2002). The use of bigrams to enhance text categorization. Information processing management, 38(4), 529-546.

[7] Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[8] Zhang, X., Zhao, J., LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in Neural Information Processing Systems (pp. 649-657).

[9] Johnson, R., Zhang, T. (2014). Effective use of word order for text categorization with convolutional neural networks. arXiv preprint arXiv:1412.1058.