Predicting Publication Conference with Neural Networks

Jingyuan Liu
AndrewId: jingyual
jingyual@andrew.cmu.edu

Abstract

In this paper, I formulated the supervised learning framework to predict publication conference. For this task, I explored several ways of encoding features and introduced different classifiers. For features, I explored meta-data feature, topic distributions, and word feature. For classifiers, I introduced how to use Naive Bayes, Decision Tree, SVM, and Neural Networks for this task. I evaluated different combinations of models and features with empirical experiments. The results showed that meta-data would help improve topic modeling features, and the best classifier is Neural Networks with deep learning word features combined with upstream LDA topic distributions.

1 Introduction

When researchers are composing a publication, they will decide which conference it should be submitted to. Submitting a paper to a proper conference is of great importance to researchers. An academic conference with related research topics and good reputation would be likely to make the publication more influential. Besides, conference would gather noted researchers in the filed and bring a great opportunity to discuss about the work. Therefore, in general, submitting publications to proper conferences would make the work more influential and bring researchers opportunites to imporve their work.

However, sometime it could be challenging to decide which conference should a publication be submitted to. First of all, there are too many conferences. For example, there are more than 120 confernces held by ACM every year. For specific filed, like Machine Learning, besides noted ACM conferences like KDD, CIKM, WSDM, there are many influential conferences held by other organizations like IJCAI, ICML, and NIPS. Some regional held conferences also enjoy good reputation like ECML and ICMLDA. Besides the scale of conference, another challenge is that some coferences would have different subtle preferences. For example, compared with ICML, NIPS would prefer neural and kernal related algorithms, while KDD would prefer applications of machine learning methods on data mining. If these challenges are well solved, we could extend the predication model to a conference recommender system, which could be very useful, especially for those new researchers who might get confused when deciding conference choice.

Solving this prediction problem could be challenging. At first, we would need to encode the representative features for the whole text corpus. Good featuress are the fundamentals for machine learning. For publication corpus, not only the words are useful, but also the meta-data like publication authors, organizations, and time contain very important latent information. Another challenge could be how to choose appropriate classifiers for this task. Different classifiers are suitable for different goals with different types of features. Most current related works try to integrate conference information into a general topic model. While this kind of work [11] [9] could be used to model conferences, they do not directly fall into supervised learning framework and hardly be used for predication goal.

In this paper, I tried to solve these challenges and purposed a general supervised learning framework including encoding features and building classifiers. The main contributions are:

1. I explored several different features like meta-data, topic distributions, and word features.
2. I introduced how to do use different classifiers like Naive Bayes, Decision Trees, SVM, and Neural Networks for predicting the publication conference.
3. I presented empirical evluation with real-life data. I also analyzed the results with cross validation and parameter tuning.

The paper is organized as follows. In section 1.1, I would introduce the related work. In section 2, I explored several different features. In section 3, I introduced how to employ several different classifiers for the prediction goal. In section 4, I presented the experiments and analyzed the results. In the last section, I would conclude my work and discuss about future work.

1.1 Related Work

Tang proposed a variant of LDA on modeling publication conference, academic researchers, and publications simultaneously called ACT Model [11]. This work could build conferences over hidden topics distributions, and similar to author-topic models [9]. Besides, there some previous work of topic modeling on integrating meta-data to improve training performances [7] [8]. All these works are variants of LDA, and could be used to help model conference and encode meta-data as features. However, they could not be directly used for supervision tasks.

Neural Networks are quite popular in recent days. Though not able to be well interpreted, Neural Networks could always achieve better performances than other classifiers. While most remarkable applications of Neural Networks are in computer vision filed, there are still influential works used for text mining [1] [4]. Word features are useful for modeling text, and deep learning offers another representative way of encoding word features [6].

Besides neural networks, there are some other important and useful classifiers [12] . Naive Bayes is a probabilistic generative model with assumption that features are conditionally independent based on labels, often used in text classification. Decision Trees and SVM are well-performed classifiers, and always used as baseline methods in classification tasks.

2 Feature Engineering

Features are the fundamentals for machine learning task. For text data, basic features include meta-data (author, publish year, organizations), topic distributions (latent topics of document), and word features. For different type of features, there are different methods to transfer them to numerical values.

2.1 Meta-data

Meta-data is useful for modeling text. I would use index map to encode meta-data. First of all, I would set every meta-data with a index. Then for each publication, if the meta-data appeared, I would set the feature index as 1, otherwise 0. Meta-data include:

a **Authors.** authors are very important attributes of a paper. I can use an indicator vector to encode this attribute to a vector as input feature. At first, I assign every different author a index. Then for each paper, I would put 1 to the position of the index for every author in the paper, otherwise 0.

b **Organizations.** similar to authors.

c **Time.** time is also very import attribute for a paper. The research topics of a conference would change over time. Therefore, when predicting conference for a paper, I need to take the time into consideration. I could use the year as the input of the feature.

2.2 Topic Distributions

When researchers are writing publications, they would first set the hidden topics that publications would cover. If we could well model the hidden topic distributions, then we could naturally model the topic distributions of conferences via modeling publications it contained.

2.2.1 LDA

LDA is commonly used to mine the latent topic distributions of documents [2] [3].

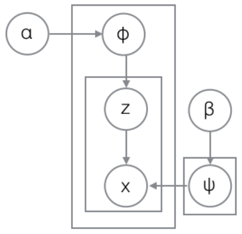


Figure 1: LDA

$$
\begin{aligned}
&\phi_d \sim Dir(\alpha) &&\psi_k \sim Dir(\beta)\\
&z_{dn} \sim Mult(\phi_x) &&x_{dn} \sim Mult(\psi_{z_{dn}})
\end{aligned}
$$

2.2.2 Neural Networks Upstream Conditioning LDA

Besides traditional LDA, we could use Neural Networks Upstream Conditioning model to mine topic distributions [7]. The advantage of upstream conditioning model is this method could integrate meta-data into the latent topic distributions.

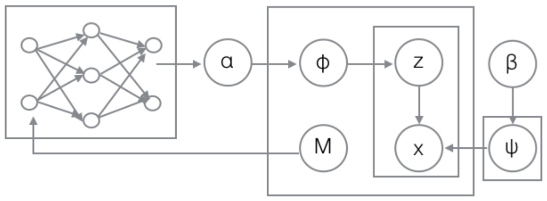


Figure 2: Neural Networks Upstream Conditioning LDA

$$
\begin{aligned}
&\alpha \sim F(M \mid W)\\
&\phi_d \sim Dir(\alpha) &&\psi_k \sim Dir(\beta)\\
&z_{dn} \sim Mult(\phi_x) &&x_{dn} \sim Mult(\psi_{z_{dn}})
\end{aligned}
$$

2.3 Word Features

Word features are the most import feature for modeling documents. We have two common ways to encode word features. The first is just use the word indicator map, which are always used by Naive Bayes. Another way is to use word2vec, which is a deep learning method to encode word features [6].

3 Classification Methods

To predict the publication conference, I formulated the supervised learning framework to do classification. The conferences are treated as labels, and the goal is to predict which conference a publication would belong to. Different classifiers would have different view of this problem. I would introduce how to use several most popular classifiers for this task.

3.1 Scalable Smoothed Naive Bayes

Naive Bayes is one of the most common basic classifiers with good performances. Naive Bayes is suitable for text classification, like email spam detection [5]. Naive Bayes generally assumed the features are conditionally independent over the classification labels. Therefore, with bayes rule, we could derive:

$$logP(y') = \sum_j log\frac{C(X_j = x_j \& Y = y')}{C(X = * \& Y = y')} + log\frac{C(Y = y')}{C(Y = *)}$$

In general, we would add λ smooth to avoid zero count case for Naive Bayes. Besides, in most cases, we do not care the order of word features, which means we would use bag of words model for Naive Bayes. With these assumptions, we could derive:

$$logP(y') = \sum_j log\frac{C(X = x_j \& Y = y') + \lambda}{C(X = * \& Y = y') + \lambda \cdot |X|} + log\frac{C(Y = y') + \lambda}{C(Y = *) + \lambda \cdot |Y|}$$

Naive Bayes is efficient for large scale dataset. We could use the stream and sort pattern, which is also know as map-reduce, to make it scalable:

```
Input  : Documents D with words X, and document labels Y
Output: Trained Smoothed Scalabel Naive Bayes Classifier
foreach publication with label y, and d words $x_1$, $x_2$, ... do
    (a) Print $Y = ANY+ = 1$;
    (b) Print $Y = y+ = 1$;
    (c) foreach word index j in 1,..d do
        Print $Y = y \& X = x_j+ = 1$;
    end
end
Sort the event-counter update message
Scan and add the sorted meesage and get the final trained classifier
```

Algorithm 1: Scalable Smoothed Naive Bayes

3.2 Decision Trees

Systems that construct classifiers are one of the commonly used tools in machine learning. Such systems take as input a collection of cases, each belonging to one of a small number of classes and described by its values for a fixed set of attributes, and output a classifier that can accurately predict the class to which a new case belongs [12].

These algorithms are generally categorized as Decision Trees. Decision Trees always achieve good performances in real-life applications. Complex decision trees can be difficult to understand, for instance because information about one class is usually distributed throughout the tree [12]. Here we used the C4.5 alogirithm which are implemented in Weka. C4.5 introduced an alternative formalism consisting of a list of rules of the form "if A and B and C and ... then class X", where rules for each class are grouped together. A case is classified by finding the first rule whose conditions are satisfied by the case; if no rule is satisfied, the case is assigned to a default class [12].

3.3 SVM

In today's machine learning applications, support vector machines (SVM) are considered a must-try. It offers one of the most robust and accurate methods among all well-known algorithms. It has a sound theoretical foundation, requires only a dozen examples for training, and is insensitive to the number of dimensions. In addition, efficient methods for training SVM are also being developed at a fast pace[12].

SVM are well introduced in Scholkopf and Smola's work [10]. Here we used the Weka SMO implementation for our task. The reason why SVM insists on finding the maximum margin hyperplanes is that it offers the best generalization ability. It allows not only the best classification performance on the training data, but also leaves much room for the correct classification of the future data [12]. Besides, SVM has a good form and allows kernelized methods which could transfer input data into a hyper space.

3.4 Neural Networks

Neural Networks are quite popular in recent days because of its impressively good performances. With Neural Networks, we could both get better features and better classifiers, although not well interpreted. While most remarkable applications of Neural Networks are in computer vision filed, there are still influential works used for text mining [4] [1].

Here we just composed basic two layer neural networks since our training goal is not very complex and the training data is not of very large scale.

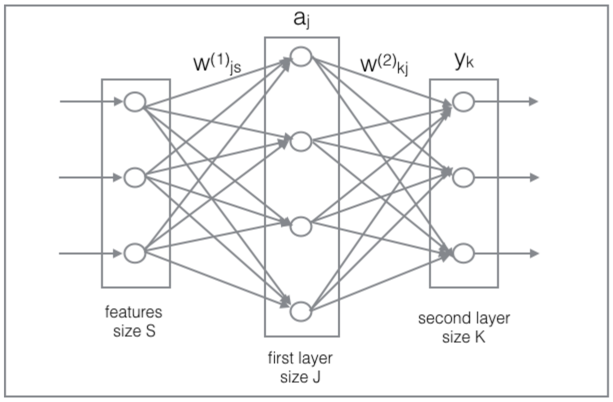


Figure 3: Two Layer Neural Networks

The first layer was size J with j as index (j > k).

$$a_j = \sigma(\sum_s w_{js}^{(1)} c_s), \qquad \sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

The size of second layer was K, with k as index:

$$y_k = \sum_j w_{kj}^{(2)} z_j, \qquad \alpha_k = h(y_k) \tag{2}$$

After we get y_k, we use softmax to get h for multi-label classification:

$$h(y_k) = \frac{e^{y_k}}{\sum_k e^{y_k}} \tag{3}$$

The Neural Networks are implemented via mxnet, an open-source library on github. Mxnet enables symbolic configuration, executor binding, and offers python interfaces, which make the implementation easier.

4 Experiments

Here I presented a series of experiments on real-life dataset to find the best classifiers to predict the publication conference. First I introduce my dataset source and the cleaned data statistics. Then I showed the extracted features. Next I explored several different classifiers to get the best performance classifiers. At last, I would do parameter tuning for optimization.

4.1 Dataset

I used the dataset from AMiner system, which is an academic social network mining system [11]. After cleaning, the whole dataset contains over 1.5m papers. I filtered the papers in the top 10 most common conference. The filtered dataset contains 10k publications from 10 conferences. The dataset contains 31475 words, 5953 organizations, time ranging from 1978 to 2014. The dataset contains 14951 authors. However, most authors are sparse, which means they would only appear once. This kind of author would not help much in building upstream conditioning LDA model but cost much memory, therefore, I filtered those sparse authors, and kept around 2k+ common authors for modeling. I did similar filtering operations to organizations.

4.2 Extracted Features

I first presented our extracted features. Meta-data is just encoded via sparse index matrix, with the document size times the meta-data size. Meta-data is used as the input for Neural Networks Upstream Conditioning LDA.

4.2.1 Topic Distributions

I presented the topic-word distributions (top 10 words) of LDA and Neural Networks Upstream Conditioning LDA. The above columns are the trained Upstream LDA, and below column are the corresponding basic LDA:

Algorithm	**Programing**	**Network**	**Social**	**Machine Learning**
algorithm	code	network	social	data
optimal	test	power	human	svm
paper	development	leakage	link	kernel
set	web	energy	clustering	flow
based	data	link	opinion	algorithm
graph	cost	node	society	research
algorithms	cache	random	network	approach
proposed	programing	matrix	user	simulation
size	software	performance	algorithm	environment
methods	testing	communication	search	performances
algorithms	dynamic	network	user	model
graph	computing	networks	social	models
set	code	proposed	information	modeling
time	resources	sensor	interaction	stucture
tree	programing	nodes	study	terms
rate	implementation	protocal	online	svm
paper	load	delay	future	approach
framework	language	layer	design	research
solution	allows	based	using	study
efficient	words	link	online	global

From the top 10 words in typical Learned topics, we could see that the extracted features are similar. Then I presented the training likelihood over iteration to show the converge speed.

The green line is Upstream LDA and the bule is LDA. We could see that the Neural Networks Upstream Conditioning LDA would achieve higher training likelihood with faster speed. Therefore, it should be better features that is more fitting to the data.

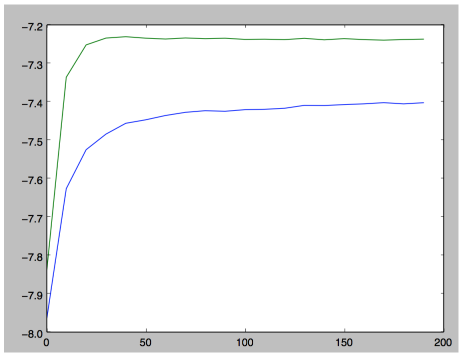


Figure 4: Training LLHW

4.2.2 Deep Learning Word Features

I used the word2vec to get the Deep Learning Word Features [6]. Word2vec is famous for finding the most similar word phrases. Here, I presented several typical similar words with cosine similarity.

Algorithm		**Network**		**Model**	
algorithms	0.732173	networks	0.825327	models	0.751951
pruning	0.715835	topology	0.744842	modeling	0656805
method	0.705341	wireless	0.665959	mathematic	0.625505
greedy	0.700185	gateways	0.657244	markov	0.603707
heuristic	0.692098	link	0.652112	stoachastic	0.603393
proposed	0.687952	node	0.541835	svm	0.595648
technique	0.666948	topologies	0.645444	probabilistic	0.5758
iteratively	0.616023	traffic	0.639832	lingo	0.573474
tabu	0.610261	nodes	0.624277	tree	0.573412
bisection	0.607905	overlap	0.607500	parametric	0.573359

For our prediction goal, I used the doc2vec, a variant of word2vec, which could transfer a doc to a numeric vector as our deep learning word features.

4.3 Classification Performances

After extracting features, we could start testing with classifiers to get the classification performances.

First I just started with baseline methods. For baseline, I would use Scalable Smoothed Naive Bayes, C4.5 Decision Trees, SVM implemented via SMO, and Two Layer Neural Networks. For Naive Bayes, I used the basic word feature. For other classifiers, I used the basic LDA topic distributions.

accuracy	Naive Bayes	Decision Tree	SVM	Neural Network
Training	0.556	0.633	0.626	**0.643**
Tesing	0.515	0.605	0.634	**0.645**

Table 1: Baseline Methods with Basic Features

From the above baseline method performance table, we could see with the basic features, Naive Bayes is obviously worse than other classification methods. Therefore, we no longer consider Naive Bayes for this task. Then I would explored which feature would help most for prediction goal.

The results are showed in the table 2, which recorded the test performances for each classifiers. From the table, we would know that in most cases, NN would be the best choice for prediction goal. Besides, we would find that the best feature is to combine Neural Networks Upstream Conditoning LDA feature, which are trained with meta-data beyond basic topic distributions, and Deep Learning Word Features, the word2vec. Another interesting fact is when combining UpLDA and word2vec as

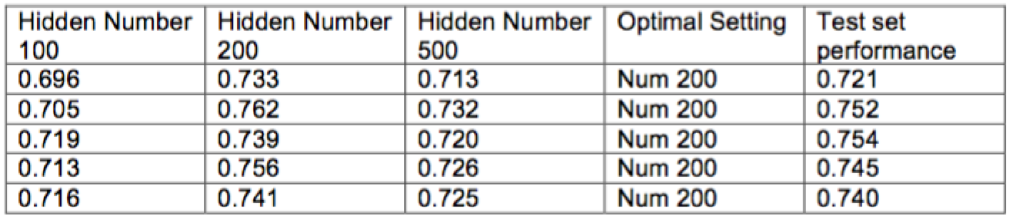
accuracy	LDA	Upstream LDA	word2vec	word2vec+UpLDA
DT	0.605	0.616	0.625	0.595
SVM	0.634	0.647	**0.682**	0.685
NN	**0.643**	**0.655**	0.678	**0.707**

Table 2: Test Performances with Different Features

feature, Decision Trees would get lower performances. This could be that Decision Trees are always not scalable and tend to be overfitting with large scale feature vector.

4.4 Parameter Tuning for Optimization

After finding the best classifier and best feature, then I do parameter tuning for optimization. The basic parameter for NN is the number of hidden points in the hidden layers. Basically, the more hidden points, the complex the model is. For my Neural Networks, there are two layer, the second layer size was set to 10, which is the label size. So I tuned the first hidden number, ranging from 100 to 500, and got the results.



Hidden Number 100	Hidden Number 200	Hidden Number 500	Optimal Setting	Test set performance
0.696	0.733	0.713	Num 200	0.721
0.705	0.762	0.732	Num 200	0.752
0.719	0.739	0.720	Num 200	0.754
0.713	0.756	0.726	Num 200	0.745
0.716	0.741	0.725	Num 200	0.740

Figure 5: Parameter Tuning

From the figure above, we would see that the Hidden Number = 200 would get the best performances. Setting Hidden Number = 500 would be overfitting, which means we are trying to use a too complex model to fit a comparatively simple dataset.

5 Conclusion and Future Work

In this paper, I formulated the supervised learning framework for predicting the publication conference. I explored several ways of encoding the publication information as features. I also discussed about several common and powerful classifiers for classification. After a series of experiments and parameter tuning. I found that the Neural Networks would achieve best performances. The best feature should be the Deep Learning Word Features combining with Neural Networks Upstream Conditioning LDA topic distributions. The best parameters for the model should be 200 hidden nodes for the first layer.

The future work mainly falls into two parts, the first is to extract features in a more elgant way. From current result, I could see that the topic distributions and word features are two different aspects of view on representing dataset and could help each other. However, I just arbitraryly combine these two features. Another point is how to overcome overfitting. We could see the results are sensitive to the number of hidden nodes. And after combining two features, the input feature size could be very big, thus might lead to overfitting for some sensitive classifiers like Decision Trees.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[3] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, pages 27–28, 2006.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[7] David Mimno and Andrew McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. *arXiv preprint arXiv:1206.3278*, 2012.

[8] James Petterson, Wray Buntine, Shravan M Narayanamurthy, Tibério S Caetano, and Alex J Smola. Word features for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, page 1921–1929, 2010.

[9] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, page 487–494. AUAI Press, 2004.

[10] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[11] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[12] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.