Inductive Learning Algorithms and Representations for Text Categorization

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. (1998)

In Proceedings of the seventh international conference on Information and knowledge management. ACM.

Seminar Data Analytics I

International Masters Program in Data Analytics

University of Hildesheim

Summer Semester 2018

Olasubomi Saheed KASALI

Outline

- Introduction
- Motivation
- Objective
- Inductive Learning Methods
- Reuters Data Set
- Results
- Other Experiments
- Conclusion and Future Work

Introduction

What is Text Categorization?

- The assignment of natural language texts to one or more predefined categories based on their contents.
- Also known as Text Classification
- Helps to find, filter and manage information
- For assigning subject categories to documents
- Automated text categorization technologies supports general, consistent, and relatively static structures.

Introduction

Text Categorization

Automatic text categorization plays an important role in a wide variety of information management tasks like;

- Real-time sorting of email or files into folder hierarchies
- Topic identification to support topic-specific processing operations
- Structured search and/or browsing
- Finding documents that match long-term standing interests or more dynamic task-based interests

Introduction

Inductive Learning Algorithms

What is Inductive Learning?

- It enables the system to recognize patterns and regularities in previous knowledge or training data and extract the general rules from them.
- It helps in inducing general rules and predicting future activities.
- The extracted generalized rules helps in reasoning and problem solving

Introduction

Inductive Learning Algorithms

Inductive learning Algorithms that were considered are;

- Find Similar
- Decision Trees
- Naïve Bayes
- Bayes Nets
- Support Vector Machines

Motivation

As the volume of information available on the Internet and corporate intranets continues to increase, there is growing interest in helping people better find, filter, and manage these resources.

Objectives

To compare the effectiveness of five different automatic learning algorithms for text categorization in terms of;

- Learning Speed
- Real-Time Classification Speed
- Classification Accuracy

on a collection of hand-tagged financial newswire stories from Reuters

Inductive Learning Method

Classifiers

Algorithm that implements classification, especially in a concrete implementation, is known as a classifier.

The term "classifier" also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

Inductive Learning Method

Example

- Reuters categories include *acquisitions*, *earnings*, *interest*

- Classifiers for the Reuters category *interest* may include

if (interest AND rate) OR (quarterly), then *confidence(interest category)* = 0.9

confidence(interest category) = 0.3*interest + 0.4*rate + 0.7*quarterly

Inductive Learning Method

Text Representation and Feature Selection

For the Find Similar algorithm, tf*idf term weights are computed and all features are used.

For the others, only binary feature values are used (i.e. a word either occurs or does not occur in a document)

The Mutual Information Measure was used for Feature Selection

$$MI(x_i, c) = \sum_{x_i \in \{0,1\}} \sum_{c \in \{0,1\}} P(x_i, c) \log \frac{P(x_i, c)}{P(x_i)P(c)} \qquad \text{(I)}$$

The k features for which mutual information is largest for each category was selected.

Inductive Learning Method

Inductive Learning Algorithms

1.) Find Similar((a variant of Rocchio's method for relevance feedback)

In Rocchio's formulation, the weight assigned to a term is a combination of its weight in an original query, and judged relevant and irrelevant documents.

$$x_j = \alpha \cdot x_{q,j} + \beta \cdot \frac{\sum_{i \in rel} x_{i,j}}{n_r} + \gamma \cdot \frac{\sum_{i \in non-rel} x_{i,j}}{N - n_r} \qquad \text{(ii)}$$

- The parameters α, β, and γ control the relative importance of the original query vector.
- No initial query, so α=0. We also set γ=0 so the available code could easily be used.
- There is no explicit error minimization involved in computing the Find Similar weights.
- Test instances are classified by comparing them to the category centroids using the Jaccard similarity measure. If the score exceeds a threshold, the item is classified as belonging to the category.

Inductive Learning Method

Inductive Learning Algorithms

2.) Decision Trees

- A decision tree was constructed for each category using the approach described by Chickering et al. (1997).
- The decision trees were grown by recursive greedy splitting, and splits were chosen using the Bayesian posterior probability of model structure.
- A *structure prior* that penalized each additional parameter with probability 0.1, and derived parameter priors from a prior network as described in Chickering et al. (1997) with an equivalent sample size of 10.
- A class probability rather than a binary decision is retained at each node.

Inductive Learning Method

Inductive Learning Algorithms

3.) *Naïve Bayes*

- A naïve-Bayes classifier is constructed by using the training data to estimate the probability of each category given the document feature values of a new instance.
- For the Naïve Bayes classifier, we assume that the features *x1,…,xn* are conditionally independent , given the category variable *C*.

$$P(\vec{x} \mid C = c_k) = \prod_i P(x_i \mid C = c_k) \qquad \text{(iii)}$$

- The assumption of conditional independence is generally not true for word appearance in documents, still the Naïve Bayes classifier is surprisingly effective.

Inductive Learning Method

Inductive Learning Algorithms

4.) Bayes Nets

- Allows for a limited form of dependence between feature variables, thus relaxing the very restrictive assumptions of the Naïve Bayes classifier.

- A 2-dependence Bayesian classifier was used that allows the probability of each feature *xi* to be directly influenced by the appearance/non-appearance of at most two other features.

Inductive Learning Method

Inductive Learning Algorithms

5.) Support Vector Machines (SVMs)

- An SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin

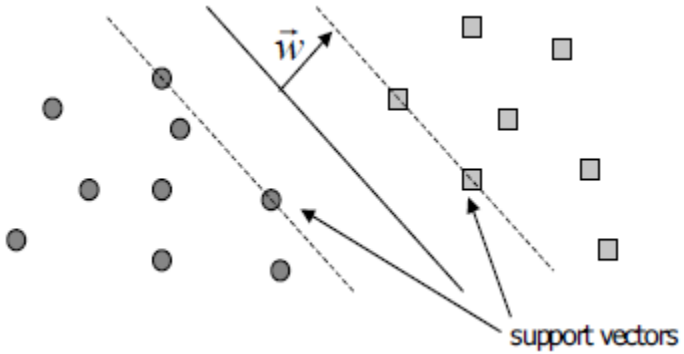


Fig 1: Linear Support Vector Machine

The formula for the output of a linear SVM is

$$u = \vec{w} \cdot \vec{x} - b$$

Inductive Learning Method

Inductive Learning Algorithms-*Support Vector Machines (SVMs)*

Where $\vec{w}$ is the normal vector to the hyperplane, and $\vec{x}$ is the input vector.

- In the linear case, the margin is defined by the distance of the hyperplane to the nearest of the positive and negative examples.

- The method developed by Platt (1998) which breaks large Quadratic Programming (QP) problem down into a series of small QP problems that can be solved analytically was used.
- Once the weights are learned, new items are classified by computing $\vec{w}.\vec{x}$
- A sigmoid to the output of the SVM was fit, so that the SVM can produce posterior probabilities that are directly comparable between categories.

Reuters Data Set

Reuters-21578 (ModApte split)

- Reuters- 21578 collection was used.
- 12,902 classified into 118 categories (e.g., corporate acquisitions, earnings, money market, grain, and interest).
- The stories average about 200 words in length.
- The ModApte split was used

Reuters Data Set

- 75% of the stories are used to build classifiers (9603 stories)
- 25% to test the accuracy of the resulting models (3299 stories)
- The mean number of categories assigned to a story is 1.2
- Many stories are not assigned to any of the 118 categories
- Some stories are assigned to 12 categories.
- The number of stories in each category varied widely (from “earnings” which contains 3964 documents to “castor-oil” which contains only one test document).

Reuters Data Set

- Table 1 shows the ten most frequent categories along with the number of training and test examples in each.
- These 10 categories account for 75% of the training instances, with the remainder distributed among the other 108 categories.

Category Name	Num Train	Num Test
Earn	2877	1087
Acquisitions	1650	719
Money-fx	538	179
Grain	433	149
Crude	389	189
Trade	369	118
Interest	347	131
Ship	197	89
Wheat	212	71
Corn	182	56

Table 1: Number of Training/Test Items

Reuters Data Set

Summary of Inductive Learning Process for Reuters

- Fig 2 summarizes the process for testing the various learning algorithms.
- Text files are processed using Microsoft's Index Server.
- All features are saved along with their tf*idf weights.
- For the Find Similar method, similarity is computed between test examples and category centroids using all these features.
- For all other methods, we reduce the feature space by eliminating words that appear in only a single document, then selecting the k words with highest mutual information with each category.
- These k-element binary feature vectors are used as input to four different learning algorithms. For SVMs and decision trees k=300, and for the other methods, k=50.

Reuters Data Set

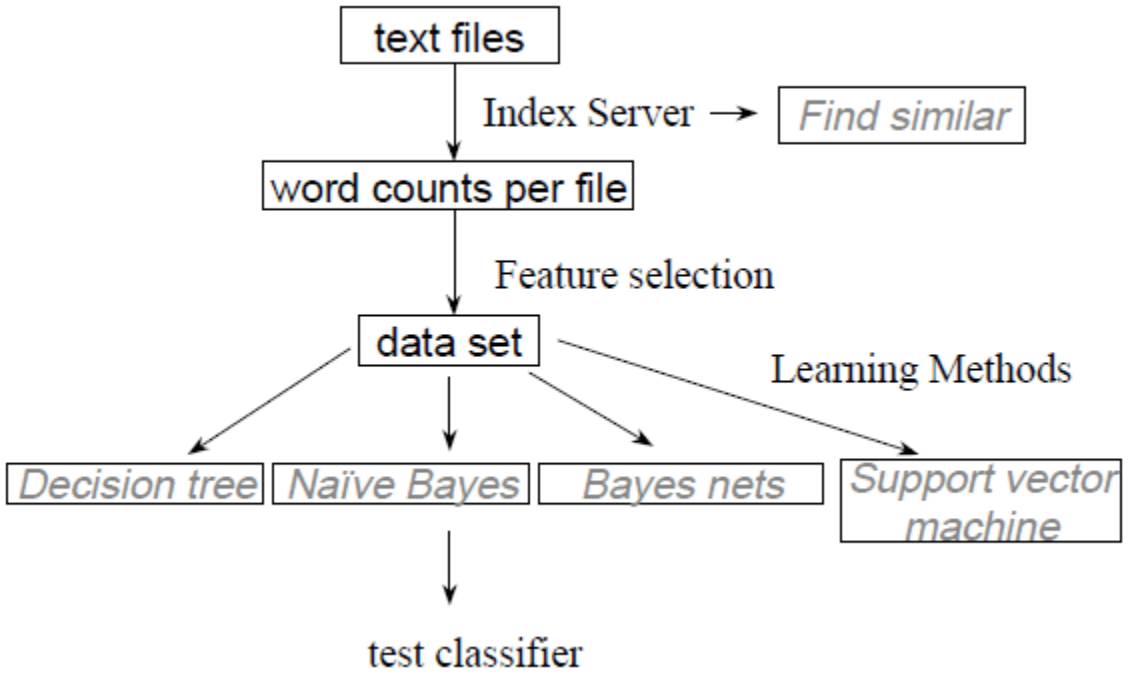


Fig 2: Schematic of Learning Process

Results

Training Time

- Training times for the 9603 training examples vary substantially across methods.
- The algorithms was tested on a 266MHz Pentium II running Windows NT.
- Using the mutual-information feature-extraction step takes much more time than any of the inductive learning algorithms

	Find Similar	**Support Vector Machines**	**Naïve Bayes**	**Decision Trees**	**Bayes Nets**
Training Time(for the 10 largest categories)	<1 CPU sec/category	<2 CPU secs/category	8 CPU secs/category	~70 CPU secs/category	~145 CPU secs/category

Table 2: Comparing Training Time

Results

Classification Speed for New Instances

- All of the five classifiers are very fast
- They require less than 2 msec to determine if a new document should be assigned to a particular category.
- Much time is spent in pre-processing the text than in categorization.

Results

Classification Accuracy

- Measures for classification are based on Precision and Recall.

- Precision is the proportion of items placed in the category that are really in the category.
- Recall is the proportion of items in the category that are actually placed in the category.

- The average of precision and recall (the so-called breakeven point) was reported for comparability to earlier results in text classification.

Results

Classification Accuracy

	Findsim	NBayes	BayesNets	Trees	LinearSVM
earn	92.9%	95.9%	95.8%	97.8%	98.0%
acq	64.7%	87.8%	88.3%	89.7%	93.6%
money-fx	46.7%	56.6%	58.8%	66.2%	74.5%
grain	67.5%	78.8%	81.4%	85.0%	94.6%
crude	70.1%	79.5%	79.6%	85.0%	88.9%
trade	65.1%	63.9%	69.0%	72.5%	75.9%
interest	63.4%	64.9%	71.3%	67.1%	77.7%
ship	49.2%	85.4%	84.4%	74.2%	85.6%
wheat	68.9%	69.7%	82.7%	92.5%	91.8%
corn	48.2%	65.3%	76.4%	91.8%	90.3%
Avg Top 10	64.6%	81.5%	85.0%	88.4%	92.0%
Avg All Cat	61.7%	75.2%	80.0%	N/A	87.0%

Table 3: Breakeven Performance for 10 Largest Categories, and over all 118 Categories.

Results

Classification Accuracy

- Support Vector Machines were the most accurate method
- Accuracy for Decision Trees was 3.6% lower.
- Bayes Nets provided some performance improvement over Naïve Bayes as expected.
- All the more advanced learning algorithms increase performance by 15-20% compared with Find Similar

Other Experiments

- For other applications, training data may be much harder to come by

- For good performance generalization, performance for the 10 most frequent categories was looked at.

Varying the number of positive instances but keeping the negative data constant

- Using only 10% of the training sets data performance is 89.6%, with a 5% sample 86.2%, and with a 1% sample 72.6%

- In general, having 20 or more training instances provides stable generalization performance.

Other Experiments

- Can NLP analyses (e.g. using “interest rate” for the category “interest” rather than “rate” or “interest”) improve classification accuracy?

- For the SVM, the NLP features actually reduced performance on the 118 categories by 0.2%.

- Will moving to a richer representation than binary features improve categorization accuracy?

- Initial results using representation that encoded words as appearing 0,1, or >=2 times in each document with Decision Tree classifiers did not yield improved performance.

Conclusion

- SVMs are the most accurate classifier and the fastest to train.
- The real-time classification speed of the five classifiers are very fast.
- The accuracy of the simple linear SVM is among the best reported for the Reuters-21578 collection.
- Representing documents as binary vectors of words, was as good as finer-grained coding.
- Inductive learning methods described can be used to support flexible, dynamic, and personalized information access and management in a wide variety of tasks.

Future Work

- Extend the text representation models to include;

i.) additional structural information about documents

ii.) knowledge-based features to provide substantial improvements in classification accuracy.

- Extending this work to automatically classify items into hierarchical category structures.

Reference

A.M. AlMana and M. S. Aksoy, “An Overview of Inductive Learning Algorithms,” Int. J. Comput. Appl., vol. 88, no. 4, pp. 20–28, 2014.

F. Sebastiani, “Text categorization”, Alessandro Zanasi (ed.) Text Mining and its Applications, WIT Press, Southampton, UK, pp. 109-129, 2005.

Chickering D., Heckerman D., and Meek, C. A Bayesian approach for learning Bayesian networks with local structure. In Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence, 1997

Cortes, C., and Vapnik, V., Support vector networks. *Machine Learning, 20*, 273-297, 1995.

Korde V and Mahender C. Text classification and classifiers: A survey. International Journal of Artificial Intelligence & Applications (IJAIA) 2012 3(2): 85-99.

Platt, J. Fast training of SVMs using sequential minimal optimization. To appear in: B. Scholkopf, C. Burges, and A. Smola (Eds.) Advances in Kernel Methods – Support Vector Learning, MIT Press, 1998

Rocchio, J.J. Jr. Relevance feedback in information retrieval. In G.Salton (Ed.), The SMART Retrieval System: Experiments in Automatic Document Processing, 313-323. Prentice Hall, 1971.

Z. Zhang and E. R. Hancock, Hypergraph based information-theoretic feature selection, Pattern Recognition Letters, vol. 33, no. 15, pp. 1991 1999, 2012

Yang, Y. and Pedersen, J.O. A comparative study on feature selection in text categorization. In Machine Learning: Proceedings of the Fourteenth International Conference (ICML’97), 412-420, 1997.

Sebastiani F. Machine learning in automated text categorization. ACM Computing Surveys. 2002;34(1):1–47

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire N´edellec and C´eline Rouveirol, editors, Proceedings of the European Conference on Machine Learning, pages 137–142, Berlin, 1998. Springer

An Effective Approach to Enhance Centroid Classifier for Text Categorization

by Songbo Tan and Xueqi Cheng

Information Security Center, Institute of Computing Technology, China

Seminar Data Analytics I

International Masters Program in Data Analytics

University of Hildesheim

Summer Semester 2018

Nurbakyt Kulbatshayeva

Outline

- Introduction
- Centroid Classifier
- Motivation
- Objective
- Model Adjustment algorithm
- Comparison with Other Methods
- Performance plots
- Conclusion

Introduction

Text Categorisation methods:

- Centroid Classifier
- Naive Bayes
- Winnow or Perceptron
- Voting
- Support Vector Machines (SVM)

Centroid Classifier

1. Compute the weighted representation of each training document using normalized TFIDF;

2. Calculate the centroid vector Ci for each training class ci:

$$C_i = \frac{1}{|c_i|} \sum_{d \in c_i} d$$

Centroid Classifier

3. Calculate the similarity of one document d to each centroid by inner product measure:

$$Sim(d, C_i) = d \cdot C_i$$

4. Based on these similarities, assign d the class label corresponding to the most similar centroid.

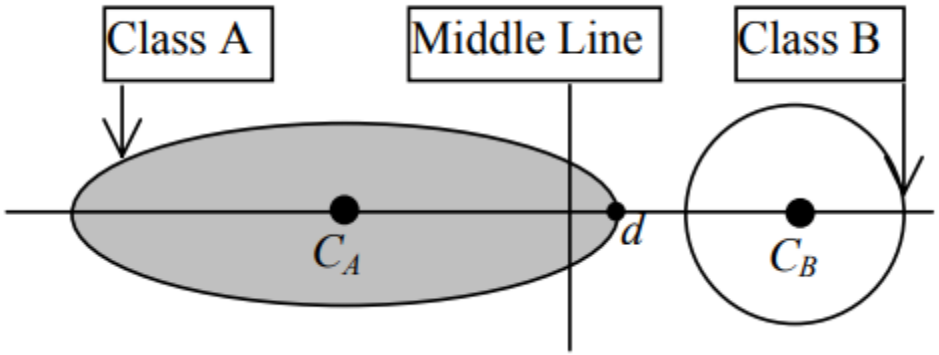
Motivation

Centroid Classifier has been shown to be a simple and yet effective method for text categorization. However, it is often plagued with model misfit (or inductive bias) incurred by its assumption.

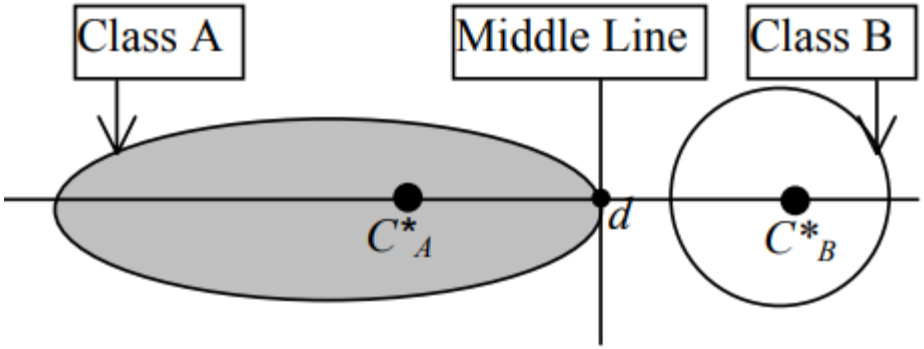
Objectives

To make use of some criteria to adjust Centroid Classifier model, which includes training-set errors as well as training-set margins.

Model Adjustment



Original Centroids



Refined Centroids

Model Adjustment

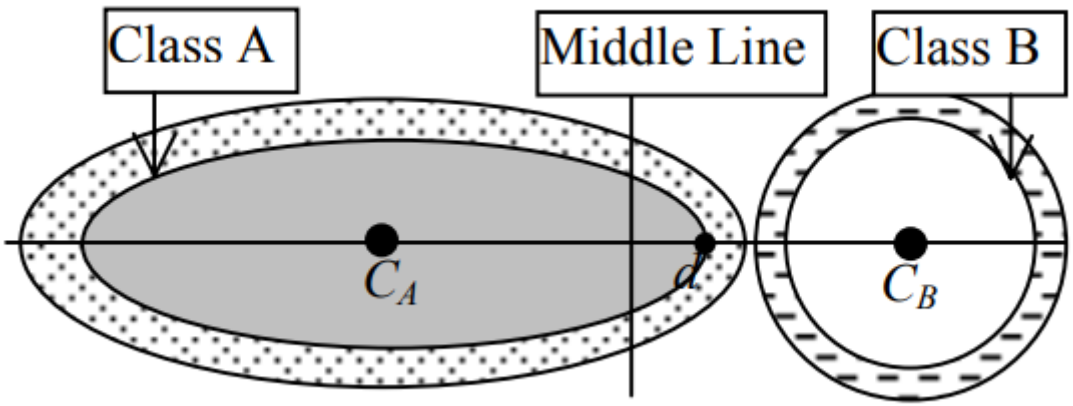
Formulas to calculate new Centroids:

$$C_A^* = C_A + \eta \cdot d$$

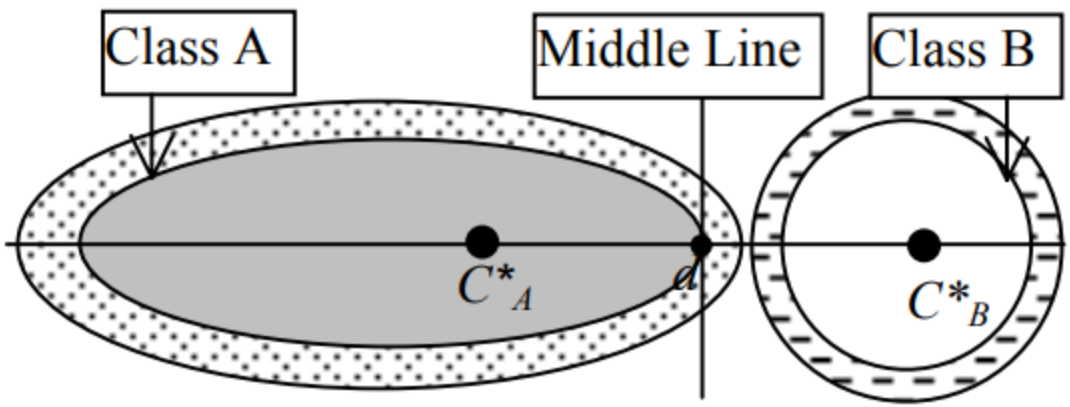
$$C_B^* = C_B - \eta \cdot d$$

- C^*_A– new Centroid
- C_A – old Centroid
- η - Learn Rate
- d - document

Model Adjustment: unseen examples

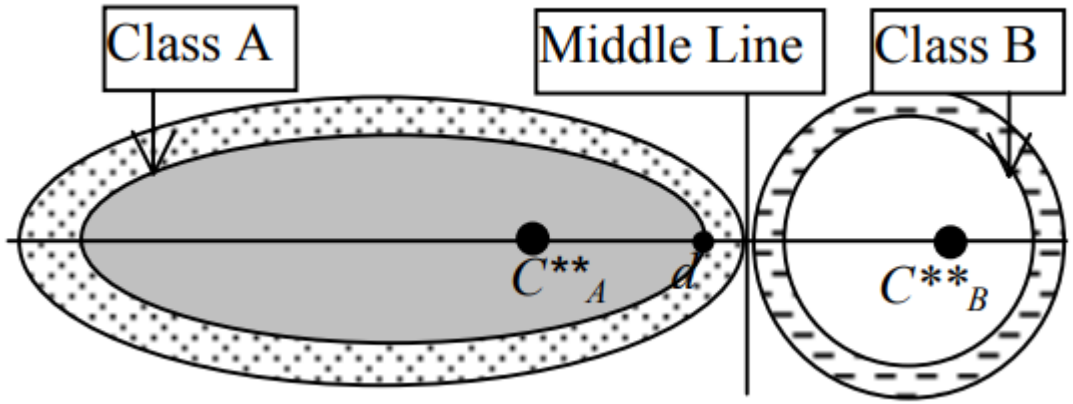


The distribution of unseen examples of Class A and Class B



Refining the centroids by training examples

Model Adjustment: unseen examples



Refining the centroids by training examples and unseen examples

Algorithm

```
1. Load training data and parameters;
2. Calculate Ci for each class ci;
3. For iter=1 to MaxIteration Do
   3.1 Classify all training documents;
   3.2 Update centroids by formula:
```

$$C_A^* = C_A + \eta \times \left\{ \sum_{\substack{d \in c_A \\ \rho(d) < 0}} d - \sum_{\substack{d \notin c_A \\ \rho(d) < 0}} d + \omega \times \left(\sum_{\substack{d \in c_A \\ 0 < \rho(d) < \theta}} d - \sum_{\substack{d \notin c_A \\ 0 < \rho(d) < \theta}} d \right) \right\}$$

Formula for Centroid updating

$$C_A^* = C_A + \eta \times \left\{ \sum_{\substack{d \in c_A \\ \rho(d) < 0}} d - \sum_{\substack{d \notin c_A \\ \rho(d) < 0}} d + \omega \times \left(\sum_{\substack{d \in c_A \\ 0 < \rho(d) < \theta}} d - \sum_{\substack{d \notin c_A \\ 0 < \rho(d) < \theta}} d \right) \right\}$$

- C^*_A – new Centroid
- C_A – old Centroid
- η - Learn Rate
- d - document
- ω – weight
- ρ - kind of margin
- θ - MinMargin

Experimental Design

- **Reuters-21578**: subset, consisting of 92 categories and 10,346 documents;
- **20NewsGroup**: 13 categories and 19,446 documents;
- **Industry Sector**: Sector-48 subset, consisting of 48 categories and 4,581 documents;
- **OHSUMED:** 11,162 documents and 10 categories;
- **RCV1:** 56 categories and 41,320 documents.

2/3 of documents for training and 1/3 for testing

Experimental Design

For MA:

- MaxIteration = 10
- η - Learn Rate = 0.5
- ω – weight = 0.2
- θ – MinMargin = 0.1

$$C_A^* = C_A + \eta \times \left\{ \sum_{\substack{d \in c_A \\ \rho(d) < 0}} d - \sum_{\substack{d \notin c_A \\ \rho(d) < 0}} d + \omega \times \left(\sum_{\substack{d \in c_A \\ 0 < \rho(d) < \theta}} d - \sum_{\substack{d \notin c_A \\ 0 < \rho(d) < \theta}} d \right) \right\}$$

Comparison with Other Methods

Table: The performance of different methods

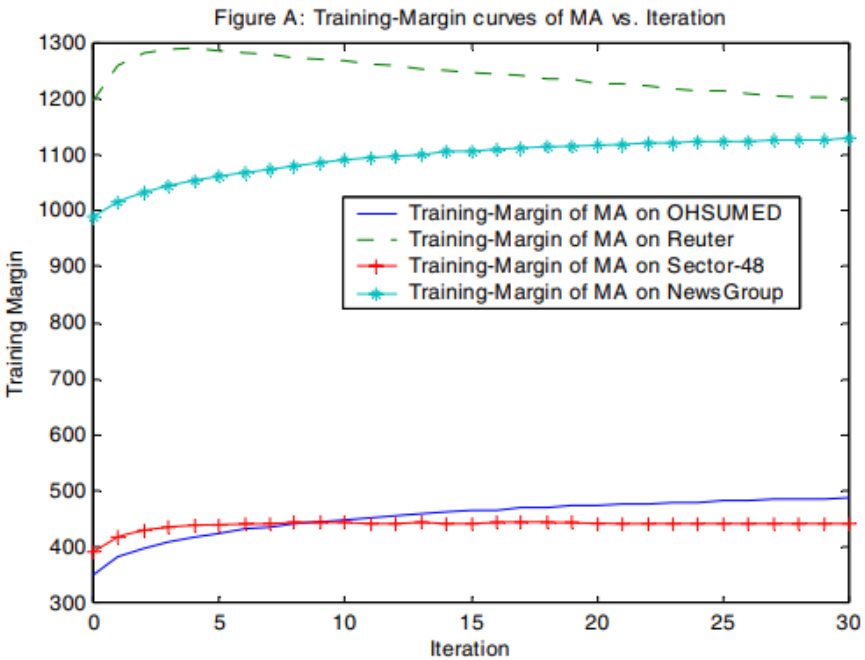
	MA		Centroid		Winnow		Perceptron		LibSvm	
	MicroF1	**MacroF1**	**MicroF1**	**MacroF1**	**MicroF1**	**MacroF1**	**MicroF1**	**MacroF1**	**MicroF1**	**MacroF1**
OHSUMED	**0.8049**	**0.7940**	0.7676	0.7600	0.7193	0.7110	0.7100	0.6996	0.7906	0.7800
Reuters	0.8565	**0.6061**	0.7820	0.5617	0.8263	0.4891	0.7918	0.4553	**0.8694**	*0.4875*
Sector-48	**0.8970**	**0.9000**	0.8055	0.8152	0.8003	0.8389	0.7845	0.7943	0.8732	0.8780
NewsGroup	0.8892	0.8859	0.8429	0.8389	0.8105	0.8161	0.8089	0.8081	**0.9040**	**0.9029**
RCV1	0.7166	**0.4912**	0.6778	0.4883	0.6142	0.4138	0.6335	0.3379	**0.7213**	0.4126

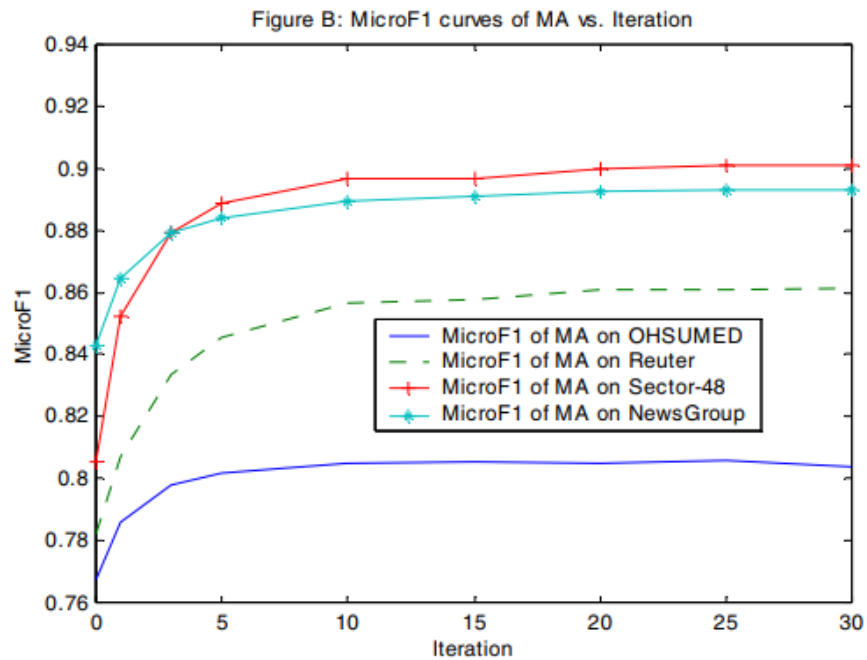
Comparison with Other Methods

Table: Training Time in seconds

	MA	Centroid	Winnow	Perceptron	LibSvm
OHSUMED	1.39	0.40	1.72	1.95	62.28
Reuters	18.41	0.40	7.75	11.01	80.77
Sector-48	11.91	0.50	4.92	9.51	38.31
NewsGroup	7.56	0.48	4.90	6.34	160.11
RCV1	10.46	0.42	4.64	6.79	38.54

Performance plots of MA





Training margin and prediction performance of MA vs. MaxIteration

Conclusion

- **Model Adjustment** (MA) algorithm was proposed to deal with model bias problem of Centroid Classifier.
- two measures for Model Adjustment: training-set **errors** and training-set **margins**.
- **misclassified** examples and small-margin examples are picked out to update the classifier model.
- Model Adjustment could make a significant difference on the **performance** of Centroid Classifier.

Criticism

1. An effective approach to enhance centroid classifier for text categorization
S Tan, X Cheng - European Conference on Principles of Data Mining ..., 2007 - Springer
Centroid Classifier has been shown to be a simple and yet effective method for text categorization. However, it is often plagued with model misfit (or inductive bias) incurred by its assumption. To address this issue, a novel Model Adjustment algorithm was proposed. The basic idea is to make use of some criteria to adjust Centroid Classifier model. In this work, the criteria include training-set errors as well as training-set margins. The empirical assessment indicates that proposed method performs slightly better than SVM classifier in ...
☆ ❝❝ Cited by 11 Related articles All 9 versions

2. Requires more time than Centroid Classifier:

	MA	Centroid
OHSUMED	1.39	0.40
Reuters	18.41	0.40
Sector-48	11.91	0.50
NewsGroup	7.56	0.48
RCV1	10.46	0.42

References

1. Shankar, S., Karypis, G.: Weight adjustment schemes for a centroid based classifier. Technical report, Dept. of Computer Science, University of Minnesota
2. McCallum, A., Nigam, K.: A Comparison of Event Models for Naive Bayes Text Classification. In: AAAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41– 48 (1998)
3. Mun, P.P.T.M.: Text Classification in Information Retrieval using Winnow, http://citeseer. ist.psu.edu/cs
4. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
5. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: ECML, pp. 137–142 (1998)
6. Liu, Y., Yang, Y., Carbonell, J.: Boosting to Correct Inductive Bias in Text Classification. In: CIKM, pp. 348–355 (2002)
7. Wu, H., Phang, T.H., Liu, B., Li, X.: A Refinement Approach to Handling Model Misfit in Text Categorization. In: SIGKDD, pp. 207–216 (2002)
8. Tan, S., Cheng, X., Ghanem, M.M., Wang, B., Xu, H.: A Novel Refinement Approach for Text Categorization. In: ACM CIKM, Bremen, Germany, pp. 469–476 (2005)

Character-level Convolutional Networks for Text Classification

Xiang Zhang, Junbo Zhao and Yann LeCun
Courant Institute of Mathematical Sciences, New York University
2015

Seminar Data Analytics I
International Masters Program in Data Analytics
University of Hildesheim
Jia-Jen Yang
29. 5. 2018

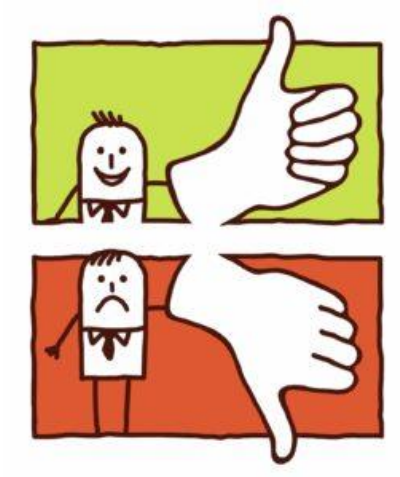
Outline

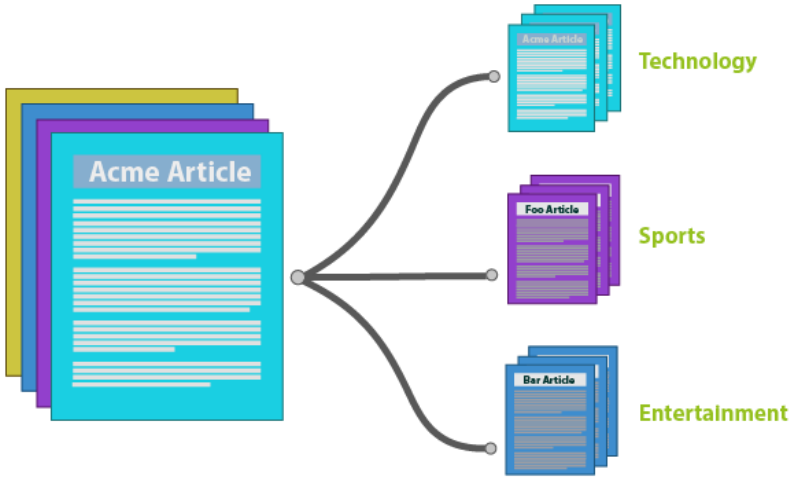
- Introduction
  - Text Classification
  - Convolutional Neural Network (CNN)
- Character-level Convolutional Networks
- Comparison Models and Large-scale Datasets
- Results and Discussion
- Conclusion

Introduction

Text Classification

- A classic topic for natural language processing
- Assign predefined categories to free-text documents
- Application includes email spam filtering (spam vs. ham), sentiment analysis (positive vs. negative) and news categorization

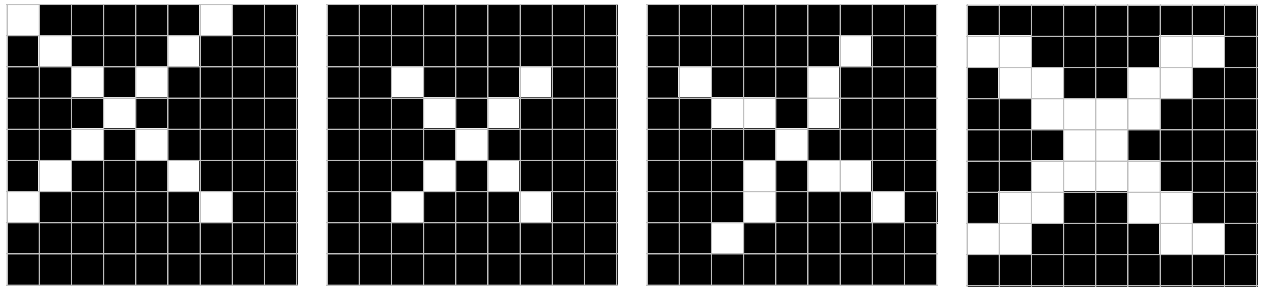




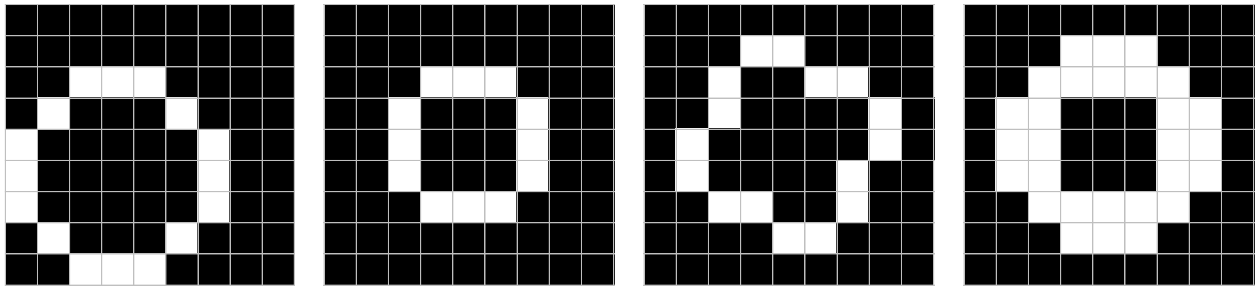
Picture : http://resonatecompanies.com/are-you-positive-or-negative-on-the-u-s-economy-for-2017/
https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

Convolutional Neural Network (CNN)

- Influential model in Deep Learning
- Inspired by the organization of the animal visual cortex
- Extracting information from raw signals, ranging from computer vision applications , video analysis, drug discovery to natural language processing

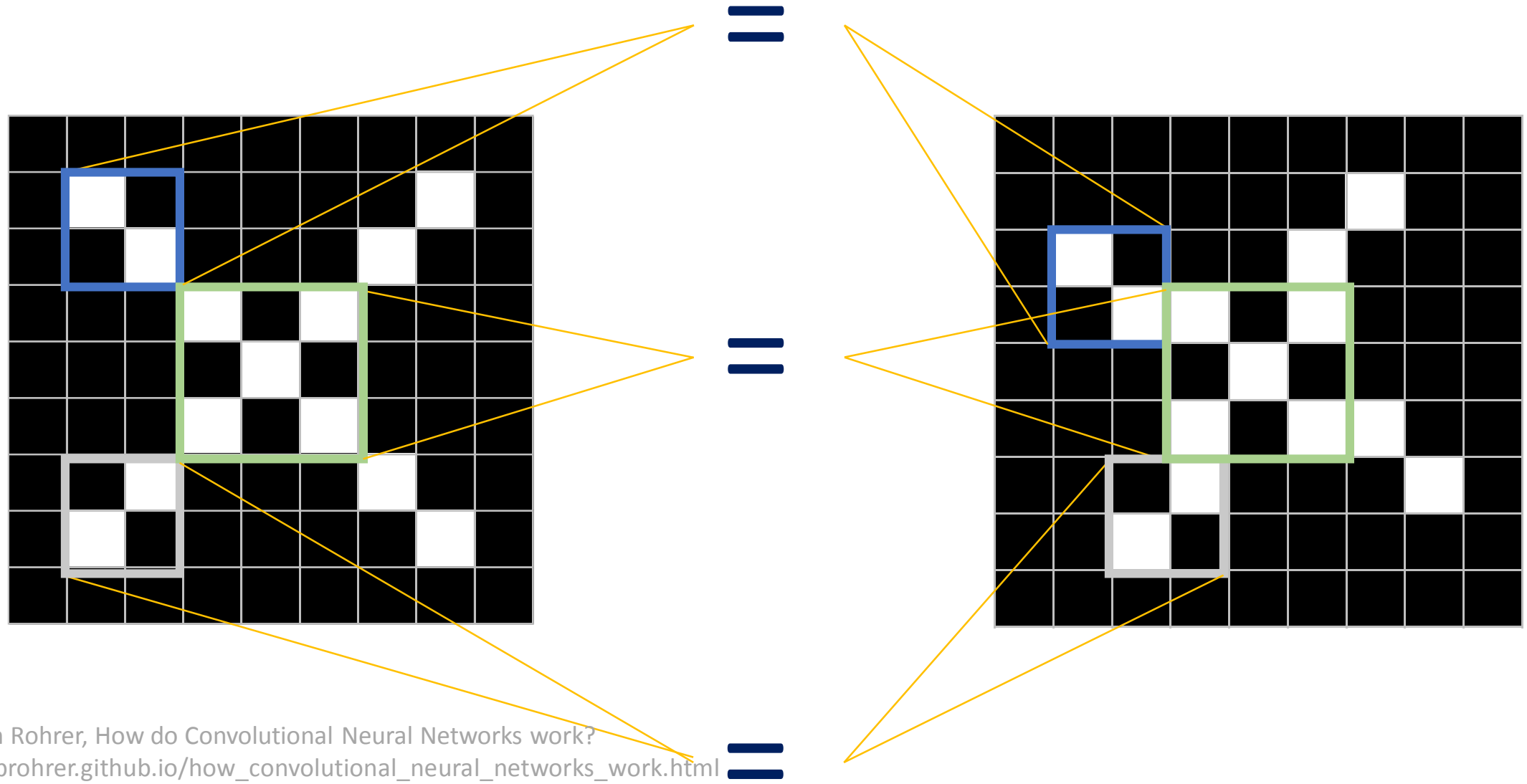


X



O

Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html



Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Imagine the picture to be a matrix

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

A feature of X, also called filter

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

1. Multiply each image pixel by the corresponding feature pixel.
2. Sum up all the multiplcation.
3. Divide by the total number of pixels in the feature
4. Move the filter to next patch and do again

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

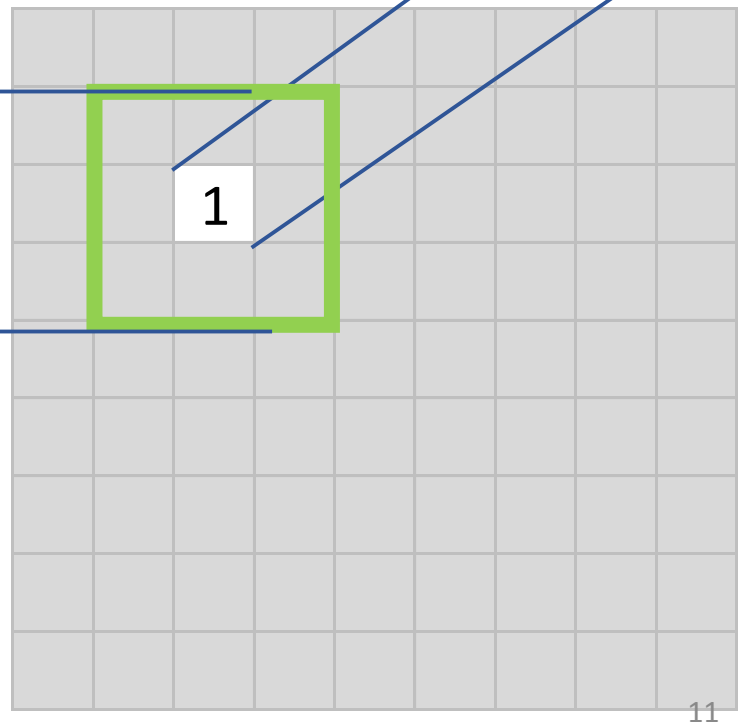
Convolution

1	-1	-1
-1	1	-1
-1	-1	1

1	1	1
1	1	1
1	1	1

$$\frac{1+1+1+1+1+1+1+1+1}{9} = \boxed{1}$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Convolution

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$\otimes$

1	-1	-1
-1	1	-1
-1	-1	1

$=$

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Pooling & Rectified Linear Units (ReLUs)

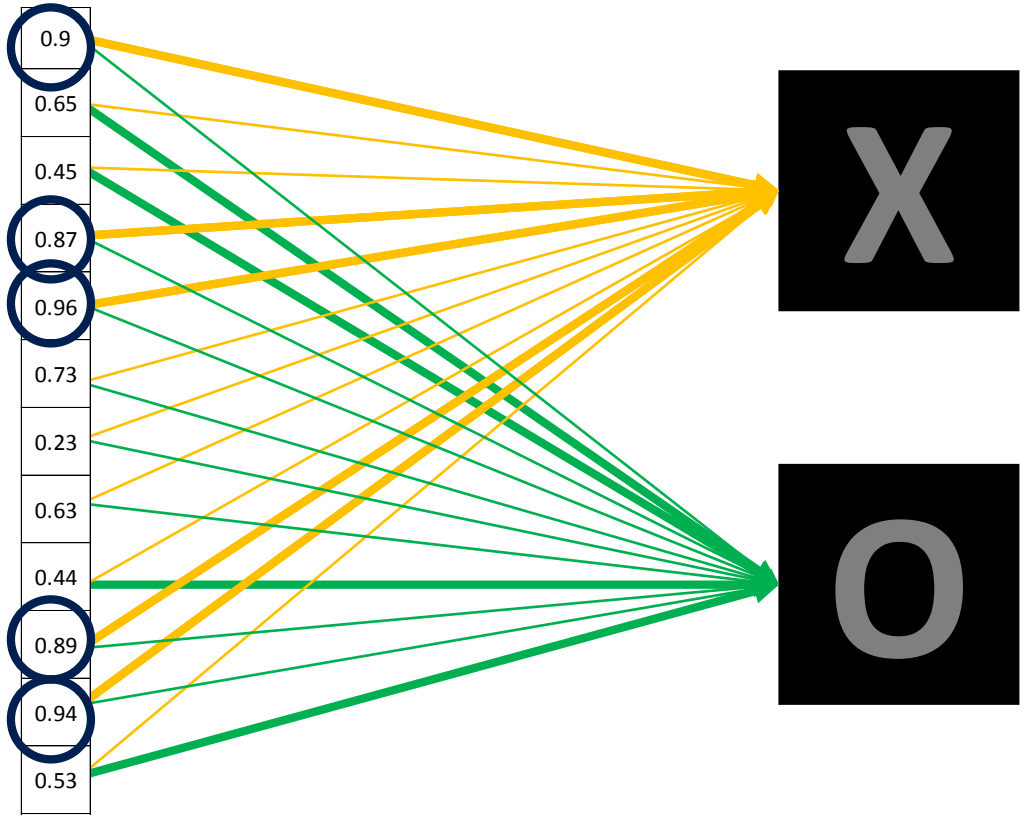
1. Take the maximum value in the filter. Move the filter to next patch and do again
2. Change every negative value to zero (If exists)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00			

Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Fully-connected



Brandon Rohrer, How do Convolutional Neural Networks work?
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Character-level Convolutional Networks

Characters and input format

abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:'''/\|_@#$%^&*~`+-=<>()[]{}

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	s	.....
h								1											.....
i									1										.....
l												1							.....
d				1															.....
e					1														.....
s																		1	.....
h								1											.....
e					1														.....
i									1										.....
m													1						.....
	.....																		

< ----------7 0 ---------->

- 2 ConvNets – one large and one small. They are both 9 layers deep with 6 convolutional layers and 3 fully-connected layers.
- The input have number of features equal to 70 due to our character quantization method, and the input feature length is 1014
- It seems that 1014 characters could already capture most of the texts of interest. We also insert 2 dropout modules in between the 3 fully-connected layers to regularize. They have dropout probability of 0.5.

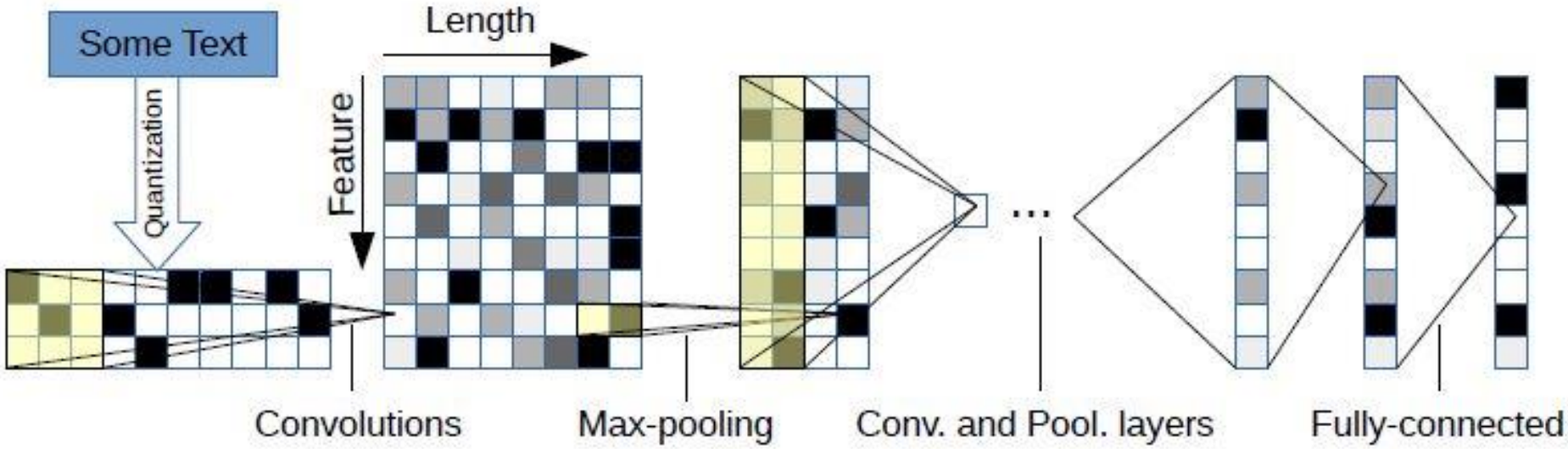


Figure 1: Illustration of our model

- configurations for convolutional layers

Table 1: Convolutional layers used in our experiments. The convolutional layers have stride 1 and pooling layers are all non-overlapping ones, so we omit the description of their strides.

Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

- configurations for fully-connected (linear) layers

Table 2: Fully-connected layers used in our experiments. The number of output units for the last layer is determined by the problem. For example, for a 10-class classification problem it will be 10.

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

Comparison Models and Large-scale Datasets

Comparison Models

- Traditional Methods
  - Bag-of-words and its TFIDF
  - Bag-of-ngrams and its TFIDF
  - Bag-of-means on word embedding
- Deep Learning Methods
  - Word-based ConvNets
  - Long-short term memory

Large-scale Datasets

Table 3: Statistics of our large-scale datasets. Epoch size is the number of minibatches in one epoch

Dataset	Classes	Train Samples	Test Samples	Epoch Size
AG's News	4	120,000	7,600	5,000
Sogou News	5	450,000	60,000	5,000
DBPedia	14	560,000	70,000	5,000
Yelp Review Polarity	2	560,000	38,000	5,000
Yelp Review Full	5	650,000	50,000	5,000
Yahoo! Answers	10	1,400,000	60,000	10,000
Amazon Review Full	5	3,000,000	650,000	30,000
Amazon Review Polarity	2	3,600,000	400,000	30,000

Results and Discussion

Testing errors of all the models. Numbers are in percentage

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

Plot of relative error

- Each of these plots is computed by taking the difference between errors on comparison model and our character-level ConvNet model, then divided by the comparison model error.

$$\frac{\text{(Error in comparison model – Error in Caracter-level CNN model)}}{\text{Error in comparison model}}$$

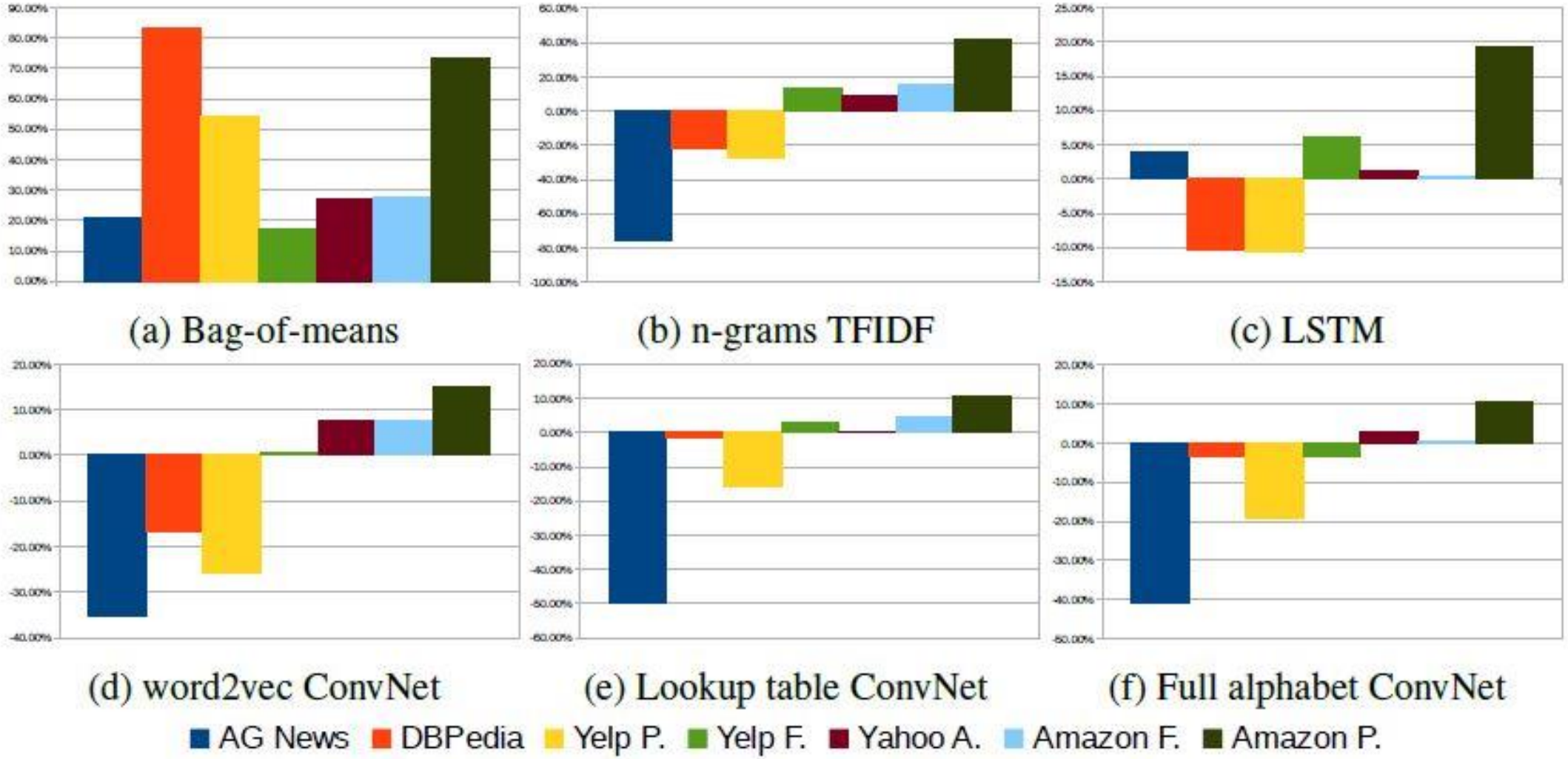


Figure 3: Relative errors with comparison models

Plot of relative error

1. Character-level CNN is an effective method.
2. Larger datasets tend to perform better.
3. CNN may work well for user-generated data.
4. Choice of alphabet makes a difference.

Conclusion

- This article shows that character-level ConvNet is an effective method. It performs better. However, how well our model performs in comparisons depends on many factors, such as dataset size, whether the texts are curated and choice of alphabet.