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

A presentation on “Inductive Learning Algorithms and Representations for Text Categorization” by Olasubomi Saheed KASALI
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

  \[
  \text{if (interest AND rate) OR (quarterly), then } \text{confidence(interest category)} = 0.9
  \]

  \[
  \text{confidence(interest category)} = 0.3\times\text{interest} + 0.4\times\text{rate} + 0.7\times\text{quarterly}
  \]
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

\[ MI(x_i, c) = \sum_{x_i \in \{0,1\}} \sum_{c \in \{0,1\}} P(x_i, c) \log \frac{P(x_i, c)}{P(x_i)P(c)} \]  \hspace{1em} (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.

   
\[
x_j = \alpha \cdot x_{q,j} + \beta \cdot \frac{\sum_{i\in\text{rel}} x_{i,j}}{n_r} + \gamma \cdot \frac{\sum_{i\in\text{non-rel}} x_{i,j}}{N-n_r}
\]

   (ii)

• The parameters \( \alpha, \beta, \) and \( \gamma \) control the relative importance of the original query vector.
• No initial query, so \( \alpha=0 \). We also set \( \gamma=0 \) 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 \( x_1, \ldots, x_n \) are conditionally independent, given the category variable \( C \).

\[
P(\tilde{x} | C = c_k) = \prod_i P(x_i | C = c_k) \tag{iii}
\]

- 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 $x_i$ 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

$$u = \hat{w} \cdot \hat{x} - b.$$
Inductive Learning Method

Inductive Learning Algorithms—Support Vector Machines (SVMs)

Where $\mathbf{w}$ is the normal vector to the hyperplane, and $\mathbf{x}$ 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 $\mathbf{w} \cdot \mathbf{x}$

- 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.

<table>
<thead>
<tr>
<th>Category Name</th>
<th>Num Train</th>
<th>Num Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ear</td>
<td>2877</td>
<td>1087</td>
</tr>
<tr>
<td>Acquisitions</td>
<td>1650</td>
<td>719</td>
</tr>
<tr>
<td>Money-fx</td>
<td>538</td>
<td>179</td>
</tr>
<tr>
<td>Grain</td>
<td>433</td>
<td>149</td>
</tr>
<tr>
<td>Crude</td>
<td>389</td>
<td>189</td>
</tr>
<tr>
<td>Trade</td>
<td>369</td>
<td>118</td>
</tr>
<tr>
<td>Interest</td>
<td>347</td>
<td>131</td>
</tr>
<tr>
<td>Ship</td>
<td>197</td>
<td>89</td>
</tr>
<tr>
<td>Wheat</td>
<td>212</td>
<td>71</td>
</tr>
<tr>
<td>Corn</td>
<td>182</td>
<td>56</td>
</tr>
</tbody>
</table>

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.
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

<table>
<thead>
<tr>
<th></th>
<th>Find Similar</th>
<th>Support Vector Machines</th>
<th>Naïve Bayes</th>
<th>Decision Trees</th>
<th>Bayes Nets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time</td>
<td>&lt;1 CPU sec/category</td>
<td>&lt;2 CPU secs/category</td>
<td>8 CPU secs/category</td>
<td>~70 CPU secs/category</td>
<td>~145 CPU secs/category</td>
</tr>
<tr>
<td></td>
<td>Training Time(for the 10 largest categories)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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


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

"An Effective Approach to Enhance Centroid Classifier for Text Categorization" by Nurbakyt Kulbatshayeva
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 $C_i$ for each training class $c_i$:

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

"An Effective Approach to Enhance Centroid Classifier for Text Categorization"
by Nurbakyt Kulbatshayeva
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
- \( \eta \) - 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

"An Effective Approach to Enhance Centroid Classifier for Text Categorization" by Nurbakyt Kulbatshayeva
Model Adjustment: unseen examples

Refining the centroids by training examples and unseen examples
Algorithm

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

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

"An Effective Approach to Enhance Centroid Classifier for Text Categorization"
by Nurbakyt Kulbatshayeva
Formulas for Centroid updating

\[ C^*_A = C_A + \eta \times \left\{ \sum_{d \in C_A} d - \sum_{d \notin C_A} d + \omega \times \left\{ \sum_{d \in C_A} d - \sum_{d \notin C_A} d \right\} \right\} \]

- \( C^*_A \) – new Centroid
- \( C_A \) – old Centroid
- \( \eta \) - Learn Rate
- \( d \) - document
- \( \omega \) – weight
- \( \rho \) - kind of margin
- \( \theta \) - MinMargin

"An Effective Approach to Enhance Centroid Classifier for Text Categorization" by Nurbakyt Kulbatshayeva
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
- $\eta$ - Learn Rate = 0.5
- $\omega$ – weight = 0.2
- $\theta$ – MinMargin = 0.1

$$C^*_A = C_A + \eta \times \left\{ \sum_{d \in c_A} d - \sum_{d \in c_A} d + \omega \times \left( \sum_{d \in c_A} d - \sum_{d \in c_A} d \right) \right\}$$
Comparison with Other Methods

Table: The performance of different methods

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

"An Effective Approach to Enhance Centroid Classifier for Text Categorization" by Nurbakyt Kulbatshayeva
Comparison with Other Methods

Table: Training Time in seconds

<table>
<thead>
<tr>
<th></th>
<th>MA</th>
<th>Centroid</th>
<th>Winnow</th>
<th>Perceptron</th>
<th>LibSvm</th>
</tr>
</thead>
<tbody>
<tr>
<td>OHSUMED</td>
<td>1.39</td>
<td>0.40</td>
<td>1.72</td>
<td>1.95</td>
<td>62.28</td>
</tr>
<tr>
<td>Reuters</td>
<td>18.41</td>
<td>0.40</td>
<td>7.75</td>
<td>11.01</td>
<td>80.77</td>
</tr>
<tr>
<td>Sector-48</td>
<td>11.91</td>
<td>0.50</td>
<td>4.92</td>
<td>9.51</td>
<td>38.31</td>
</tr>
<tr>
<td>NewsGroup</td>
<td>7.56</td>
<td>0.48</td>
<td>4.90</td>
<td>6.34</td>
<td>160.11</td>
</tr>
<tr>
<td>RCV1</td>
<td>10.46</td>
<td>0.42</td>
<td>4.64</td>
<td>6.79</td>
<td>38.54</td>
</tr>
</tbody>
</table>
Performance plots of MA

Figure A: Training-Margin curves of MA vs. Iteration

Figure B: MicroF1 curves of MA vs. Iteration

Training margin and prediction performance of MA vs. MaxIteration

"An Effective Approach to Enhance Centroid Classifier for Text Categorization" by Nurbakyt Kulbatshayeva
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:

<table>
<thead>
<tr>
<th></th>
<th>MA</th>
<th>Centroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>OHSUMED</td>
<td>1.39</td>
<td>0.40</td>
</tr>
<tr>
<td>Reuters</td>
<td>18.41</td>
<td>0.40</td>
</tr>
<tr>
<td>Sector-48</td>
<td>11.91</td>
<td>0.50</td>
</tr>
<tr>
<td>NewsGroup</td>
<td>7.56</td>
<td>0.48</td>
</tr>
<tr>
<td>RCV1</td>
<td>10.46</td>
<td>0.42</td>
</tr>
</tbody>
</table>
References


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
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
Introduction

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

Convolution  Pooling  Relu  Fully-connected
Brandon Rohrer, How do Convolutional Neural Networks work?
Imagine the picture to be a matrix

Brandon Rohrer, How do Convolutional Neural Networks work?
Convolution

A feature of X, also called filter

Brandon Rohrer, How do Convolutional Neural Networks work?
Convolution

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

Brandon Rohrer, How do Convolutional Neural Networks work?
Convolution

Brandon Rohrer, How do Convolutional Neural Networks work? 
### Convolution

Brandon Rohrer, How do Convolutional Neural Networks work?  
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)

Brandon Rohrer, How do Convolutional Neural Networks work?
Fully-connected

Brandon Rohrer, How do Convolutional Neural Networks work?
Character-level Convolutional Networks
Characters and input format

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

<------------------------------------------------------------------------------------------------------------------70------------------------------------------------------------------------------------------------------------------>
• 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.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Large Feature</th>
<th>Small Feature</th>
<th>Kernel</th>
<th>Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1024</td>
<td>256</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1024</td>
<td>256</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1024</td>
<td>256</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
<td>256</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>1024</td>
<td>256</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>6</td>
<td>1024</td>
<td>256</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
• 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.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Output Units Large</th>
<th>Output Units Small</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2048</td>
<td>1024</td>
</tr>
<tr>
<td>8</td>
<td>2048</td>
<td>1024</td>
</tr>
<tr>
<td>9</td>
<td>Depends on the problem</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Train Samples</th>
<th>Test Samples</th>
<th>Epoch Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG’s News</td>
<td>4</td>
<td>120,000</td>
<td>7,600</td>
<td>5,000</td>
</tr>
<tr>
<td>Sogou News</td>
<td>5</td>
<td>450,000</td>
<td>60,000</td>
<td>5,000</td>
</tr>
<tr>
<td>DBPedia</td>
<td>14</td>
<td>560,000</td>
<td>70,000</td>
<td>5,000</td>
</tr>
<tr>
<td>Yelp Review Polarity</td>
<td>2</td>
<td>560,000</td>
<td>38,000</td>
<td>5,000</td>
</tr>
<tr>
<td>Yelp Review Full</td>
<td>5</td>
<td>650,000</td>
<td>50,000</td>
<td>5,000</td>
</tr>
<tr>
<td>Yahoo! Answers</td>
<td>10</td>
<td>1,400,000</td>
<td>60,000</td>
<td>10,000</td>
</tr>
<tr>
<td>Amazon Review Full</td>
<td>5</td>
<td>3,000,000</td>
<td>650,000</td>
<td>30,000</td>
</tr>
<tr>
<td>Amazon Review Polarity</td>
<td>2</td>
<td>3,600,000</td>
<td>400,000</td>
<td>30,000</td>
</tr>
</tbody>
</table>
Results and Discussion
Testing errors of all the models. Numbers are in percentage

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

\[
\frac{\text{Error in comparison model} - \text{Error in Caracter-level CNN model}}{\text{Error in comparison model}}
\]
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.