CS 175, Project in Artificial Intelligence

Lecture 5: Discussion of Projects

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine
Today’s Lecture

• 11 to 11:45: Discussion of Project Proposals
  – General guidelines
  – Examples of projects
  – Tips and suggestions
    [Will discuss detailed format of project proposals next Monday]

• 11:45 to 12:15
  – Review and Q&A for Assignment 2 by Jihyun Park (TA)
Upcoming Schedule

- Assignment 2: Due this Friday at 10pm
  - Jihyun’s office hours today: 3 to 5pm, DBH 4013
  - Kevin: additional office hours today, 1 to 2pm, DBH 4059
  - Discussion section on Friday, 1 to 2pm here
  - Piazza (online)

- Next Assignment: Project Proposals
  - Will be available by end of day on Friday
  - Due the following Friday
  - Will discuss project ideas today
    ..........and details of the proposal next Monday
Assignment 1

- Grades and comments returned via EEE
- See Kevin at office hours today (1 to 2) or next Monday with any questions
Project Proposals
Project Proposals

• Your proposal should be 2 to 3 pages long
  – Required to use project proposal template (available by this weekend)

• Project proposals will be graded like a homework assignment and receive a weight of 20% of your overall grade.

• Proposals will primarily be graded on
  – (a) clarity (is it clear what will be done in this project?) and
  – (b) completeness (does the proposal address all of the important aspects of the proposed project?)

• Note: if a project is too simple (or too complex!), or missing important details, it may be returned to you and a revision requested.

(Assignment for Project Proposals will be available on Webpage by end of day Friday)
Project Teams

• Students will work in 3-person project teams
  – Use Piazza, come to discussion, email TAs if either
    1. you are 2 people and need a 3rd person
    2. you are 1 person and need to form a team

• Only one project proposal should be uploaded per project (include all team member names on the proposal)

• All team members will get the same grade for the proposal.
Ideas for Project Topics

• Refer to past lecture slides and the rest of these slides

• Look at Web pages under *background reading and links* on the class Web page

• Browse paper titles and abstracts from research conferences, use search engines to search on specific topics

• Post privately or publicly on Piazza to get some general initial feedback, e.g., “would this idea be sufficient for a project?”

• Come to office hours next Tuesday
Possible Topics for Projects

- Document classification
  - Document categories, sentiment analysis, spam detection

- Document clustering and topic modeling
  - Develop a tool to help explore and visualize large text corpora

- Word prediction/auto-complete

- Text generation/simulation/synthesis

- Other ideas
  - Question-answering systems
  - Detecting trends in text over time
  - Identifying authors from the text they write
  - Predicting a company’s stock value from newspaper reports
Question:

How can we plan out the details of our project when we haven’t yet done any work on this topic?

Your project proposal should be your best estimate at this time of what your project will be about - with the realization that there may need to be changes and adjustments along the way.

– For example, you might find that the initial data set you chose is not really suitable and need to change data sets.
– Or that the algorithm you planned to investigate is not really suitable to the task and need to change to another algorithm.
– Or that the problem you are addressing is too difficult and that you need to pick a simpler one.
Example: Document Classification.....Many Possible Projects

Different Preprocessing Methods
- Bag of Words
- Stemming
- Ngrams/Phrases
- Word embedding

Different Classification Models
- Naïve Bayes
- K-nearest neighbor
- Logistic regression
- Neural network
- Deep neural network
Example: Document Classification.....Many Possible Projects

<table>
<thead>
<tr>
<th>Different Preprocessing Methods</th>
<th>Different Classification Models</th>
<th>Types of Classification Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bag of Words</td>
<td>Naïve Bayes</td>
<td>Binary Classification</td>
</tr>
<tr>
<td>Stemming</td>
<td>K-nearest neighbor</td>
<td>Multiclass Classification</td>
</tr>
<tr>
<td>Ngrams/Phrases</td>
<td>Logistic regression</td>
<td>Multi-label, Multiclass</td>
</tr>
<tr>
<td>Word embedding</td>
<td>Neural network</td>
<td>Ranking</td>
</tr>
<tr>
<td></td>
<td>Deep neural network</td>
<td></td>
</tr>
</tbody>
</table>
Example: Document Classification.....Many Possible Projects

Different Preprocessing Methods
- Bag of Words
- Stemming
- Ngrams/Phrases
- Word embedding

Different Classification Models
- Naïve Bayes
- K-nearest neighbor
- Logistic regression
- Neural network
- Deep neural network

Types of Classification Problems
- Binary Classification
- Multiclass Classification
- Multi-label, Multiclass
- Ranking

Different Training Algorithms
- Gradient Descent
- Stochastic Gradient
- Regularization
- Dropout
Example: Document Classification.....Many Possible Projects

<table>
<thead>
<tr>
<th>Different Preprocessing Methods</th>
<th>Different Classification Models</th>
<th>Types of Classification Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bag of Words</td>
<td>Naïve Bayes</td>
<td>Binary Classification</td>
</tr>
<tr>
<td>Stemming</td>
<td>K-nearest neighbor</td>
<td>Multiclass Classification</td>
</tr>
<tr>
<td>Ngrams/Phrases</td>
<td>Logistic regression</td>
<td>Multi-label, Multiclass</td>
</tr>
<tr>
<td>Word embedding</td>
<td>Neural network</td>
<td>Ranking</td>
</tr>
<tr>
<td></td>
<td>Deep neural network</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Different Training Algorithms</th>
<th>Different Application Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient Descent</td>
<td>Sentiment Classification</td>
</tr>
<tr>
<td>Stochastic Gradient</td>
<td>News Article Classification</td>
</tr>
<tr>
<td>Regularization</td>
<td>Wikipedia Page Classification</td>
</tr>
<tr>
<td>Dropout</td>
<td>....and more</td>
</tr>
</tbody>
</table>
Different Types of Projects

• Method-focused
  – Comparing different algorithms (e.g., for document classification)
    • E.g., compare naïve Bayes, logistic regression, and neural networks
    • Evaluate on multiple different data sets, extensive experiments
  – Compare different text representation methods
    • Different methods for extracting n-grams
    • The effect of stop words
  – Test the sensitivity of classifiers to (e.g.,)
    • Vocabulary size, document length, number of documents, etc

• Application-focused
  – Focus on a particular problem, evaluate different methods
    • Sentiment classification for Twitter or other social media
    • Classifying utterances in conversation into dialog acts
    • Spam detection
  – Typically trying to see how well an automated system can do on a task
Software Development

• You will likely use both
  – code that your team writes for the project (required)
  – publicly-available code, e.g., from scikit-learn or NLTK (optional but encouraged)

• A typical pipeline will contain a mix of code
  – If you did not write the code, you need to acknowledge the source

• Part of your project could be implementing an algorithm or technique that already exists as publicly-available code, e.g.,
  – A parser
  – A neural network classifier
General Criteria for Grading of Projects

• Technical competence
  – Were algorithms and methods used correctly and appropriately?
  – Did students understand the methods they were using?
  – Were systematic experiments conducted and results interpreted?

• Effort
  – How much work was done (e.g., coding, experiments, background reading, etc)
  – Note that effort alone is necessary but not sufficient for a high grade!

• Creativity and insight:
  – Did you demonstrate creativity in your project?
  – What did insights did you gain from the project?

• Writing and communication:
  – are you able to explain your work clearly?
Examples of Public Data Sets
Data Sets

• For most of your projects you will want to use at least one large data set (or "corpus") for your project

• There are many real-world publicly-available data sets available for research purposes that you can use, e.g,
  – Yelp Dataset Challenge
  – 20 newsgroups data
  – ....and many more
  – See links to Data Sets on Class Web page:
    http://www.ics.uci.edu/~smyth/courses/cs175/reading/text_data_sets.html

• You can also gather your own data
  – E.g., by crawling Web sites or using APIs, e.g., the Twitter API
Examples of Data Sets for Text Analysis

CS 175, Winter 2016

The links below point to just a small number of the many data sets for text analysis that you can find on the Web, and should help you in terms of finding data sets to work on for your projects.

Data Sets with Classification Labels or Ratings

- Document classification data sets (a large collection of different data sets used in text classification research)
- Yelp Data Set Challenge (2.2M reviews of businesses from over 500k users in 10 cities)
  (and here’s a pointer to work from our own group at UCI that recently won the Round 5 Challenge)
- Kaggle Data Sets. Contains multiple data sets with text content. Kaggle is a company that hosts data mining/prediction competitions
- Movie review data for sentiment analysis, from Pang and Lee, Cornell
- Product review data from Johns Hopkins University (goal is to predict ratings on scale of 1 to 5)
- A variety of different text data sets from the UCI Machine Learning Repository (many already in the "bag of words" format)
- Data Sets on "learning to rank" (for Web search)
- All of Wikipedia
- Various text and Web-related data sets from Yahoo! Labs

(note that these data sets can also be used for unsupervised learning, such as clustering or topic modeling, by ignoring the class labels during training).
Interesting Text Data Sets (often used for Clustering and other Exploratory Methods)

- Enron email data set, from CMU (note that there are other "cleaner" versions available on the Web if you search...)
- Python code for downloading IMDB (Internet Movie Database), with 425k titles and 1.7 million filmographies of cast and crew
- A survey of data sets available for building data-driven dialogue systems
- Book Summaries Corpus
- Full text of US patents from 1980 to the present, from the USPTO (US Patent and Trademark Office), hosted by Google
- Very large data set of all Reddit submissions between 2006 and 2015

Data Sets used to build Language Models and Auto-complete Algorithms

- Ngram data from Peter Norvig (Google), with an accompanying tutorial book chapter
- Google ngrams, and Google syntactic ngrams over time, from Google books

Question-Answering Data Sets

- WikiQA, a data set for "open-domain" question answering, from Microsoft Research
- Question-Answering Data Sets from TREC (funding by the National Institute of Standards and Technology, NIST)
- Question Answering Corpus from DeepMind (part of Google)
- The Allen AI Science Challenge on Kaggle (competition ends Feb 13 2016)
- The BioASQ data sets and challenge competitions on question answering for the biomedical domain with an accompanying tutorial book chapter

Ontologies/Structured Data (useful for Information Extraction/Annotation)

- The DBpedia Data Set
Welcome to Kaggle Datasets
The best place to discover and seamlessly analyze publicly available data.

Dig In
Explore a dataset with our in-browser analytics tool, Kaggle Scripts. You can also download it in an easy to read format.

Build
Create your data science portfolio. Publish insights and code with Kaggle Scripts and it will be saved to your profile.

Connect
Engage with other data scientists. Share feedback on other Kagglers' scripts, or ask a question in a dataset's forum.

Hillary Clinton’s Emails
357 Scripts • 19 Topics

UCI Iris
46 Scripts • 4 Topics

Ocean Ship Logbooks (1750–1860)
117 Scripts • 24 Topics
### UCI Machine Learning Repository

#### Browse Through: 30 Data Sets

<table>
<thead>
<tr>
<th>Default Task</th>
<th>Name</th>
<th>Data Types</th>
<th>Default Task</th>
<th>Attribute Types</th>
<th># Instances</th>
<th># Attributes</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classification</strong> (21)</td>
<td><strong>UCI 3D Road Network (North Jutland, Denmark)</strong></td>
<td>Sequential, Text</td>
<td>Regression, Clustering</td>
<td>Real</td>
<td>434874</td>
<td>4</td>
<td>2013</td>
</tr>
<tr>
<td><strong>Regression</strong> (4)</td>
<td><strong>UCI Amazon Commerce reviews set</strong></td>
<td>Multivariate, Text, Domain-Theory</td>
<td>Classification</td>
<td>Real</td>
<td>1500</td>
<td>10000</td>
<td>2011</td>
</tr>
<tr>
<td><strong>Clustering</strong> (9)</td>
<td><strong>UCI Badges</strong></td>
<td>Univariate, Text</td>
<td>Classification</td>
<td>Integer</td>
<td>264</td>
<td>1</td>
<td>1994</td>
</tr>
<tr>
<td><strong>Other</strong> (6)</td>
<td><strong>UCI Bag of Words</strong></td>
<td>Text</td>
<td>Clustering</td>
<td>Integer</td>
<td>8000000</td>
<td>1000000</td>
<td>2008</td>
</tr>
<tr>
<td><strong>CNAE-9</strong></td>
<td><strong>UCI CNAE-9</strong></td>
<td>Multivariate, Text</td>
<td>Classification</td>
<td>Integer</td>
<td>1080</td>
<td>857</td>
<td>2012</td>
</tr>
<tr>
<td><strong>DBWorld e-mails</strong></td>
<td><strong>UCI DBWorld e-mails</strong></td>
<td>Text</td>
<td>Classification</td>
<td>Integer</td>
<td>64</td>
<td>4702</td>
<td>2011</td>
</tr>
<tr>
<td><strong>Dresses_Attribute_Sales</strong></td>
<td><strong>UCI Dresses_Attribute_Sales</strong></td>
<td>Text</td>
<td>Classification, Clustering</td>
<td></td>
<td>501</td>
<td>13</td>
<td>2014</td>
</tr>
<tr>
<td><strong>Farm Ads</strong></td>
<td><strong>UCI Farm Ads</strong></td>
<td>Text</td>
<td>Classification</td>
<td>Integer</td>
<td>4143</td>
<td>54877</td>
<td>2011</td>
</tr>
</tbody>
</table>
The Switchboard Dialog Act Corpus

1. Overview
2. Getting and using the corpus
   2.1 Downloads
   2.2 Python classes (preferred)
      2.2.1 Transcript objects
      2.2.2 Utterance objects
      2.2.3 CorpusReader objects
   2.3 Working directly with the CSV file (dispreferred but okay)
3. Annotations
   3.1 Dialog act annotations
   3.2 Penn Discourse Treebank 3 POS
   3.3 Penn Discourse Treebank 3 Trees
4. Exercises

1 Overview

The Switchboard Dialog Act Corpus (SwDA) extends the Switchboard-1 Telephone Speech Corpus, Release 2, with the associated turn. The SwDA project was undertaken at UC Boulder in the late 1990s.

Recommended reading:

- Dialog Act Coders' Manual
- Stolcke et al. 2000

Code and data:

- swda.zip: our distribution of the data
- swda.py: Python classes for working with swda
- swda_functions.py: auxiliary functions for using swda.py
## Examples of Dialog Acts (Labels for Utterances)

<table>
<thead>
<tr>
<th>name</th>
<th>act_tag</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement-non-opinion</td>
<td>sd</td>
<td>Me, I'm in the legal department.</td>
</tr>
<tr>
<td>Acknowledge (Backchannel)</td>
<td>b</td>
<td>Uh-huh.</td>
</tr>
<tr>
<td>Statement-opinion</td>
<td>sv</td>
<td>I think it's great</td>
</tr>
<tr>
<td>Agree/Accept</td>
<td>aa</td>
<td>That's exactly it.</td>
</tr>
<tr>
<td>Abandoned or Turn-Exit</td>
<td>%</td>
<td>So, -</td>
</tr>
<tr>
<td>Appreciation</td>
<td>ba</td>
<td>I can imagine.</td>
</tr>
<tr>
<td>Yes-No-Question</td>
<td>qy</td>
<td>Do you have to have any special training?</td>
</tr>
<tr>
<td>Non-verbal</td>
<td>x</td>
<td>[Laughter], [Throat_clearing]</td>
</tr>
<tr>
<td>Yes answers</td>
<td>ny</td>
<td>Yes.</td>
</tr>
<tr>
<td>Conventional-closing</td>
<td>fc</td>
<td>Well, it's been nice talking to you.</td>
</tr>
<tr>
<td>Uninterpretable</td>
<td>%</td>
<td>But, uh, yeah</td>
</tr>
<tr>
<td>Wh-Question</td>
<td>qw</td>
<td>Well, how old are you?</td>
</tr>
<tr>
<td>No answers</td>
<td>nn</td>
<td>No.</td>
</tr>
<tr>
<td>Response Acknowledgement</td>
<td>bk</td>
<td>Oh, okay.</td>
</tr>
<tr>
<td>Hedge</td>
<td>h</td>
<td>I don't know if I'm making any sense or not.</td>
</tr>
<tr>
<td>Declarative Yes-No-Question</td>
<td>qy\d</td>
<td>So you can afford to get a house?</td>
</tr>
</tbody>
</table>
WikiQA: A Challenge Dataset for Open-Domain Question Answering

Yi Yang, Wen-tau Yih, and Christopher Meek
21 September 2015

Abstract

We describe the WikiQA dataset, a new publicly available set of question and sentence pairs, collected and annotated for research on open-domain question answering. Most previous work on answer sentence selection focuses on a dataset created using the TREC-QA data, which includes editor-generated questions and candidate answer sentences selected by matching content words in the question. WikiQA is constructed using a more natural process and is more than an order of magnitude larger than the previous dataset. In addition, the WikiQA dataset also includes questions for which there are no correct sentences, enabling researchers to work on answer triggering, a critical component in any QA system. We compare several systems on the task of answer sentence selection on both datasets and also describe the performance of a system on the problem of answer triggering using the WikiQA dataset.

Details

Publication type: Inproceedings
Published in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing
Publisher: ACL – Association for Computational Linguistics
Examples of Projects from Past Years
# Projects on Document Classification

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Data Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis of financial journalism for market prediction</td>
<td>Reuters articles</td>
</tr>
<tr>
<td>Evaluating the accuracy of three distinct single-label text</td>
<td>Reuters articles, and possibly Wikipedia</td>
</tr>
<tr>
<td>classification models</td>
<td></td>
</tr>
<tr>
<td>Reddit miner</td>
<td>Reddit posts</td>
</tr>
<tr>
<td>Document reading level classification</td>
<td>Reddit posts</td>
</tr>
<tr>
<td>Wikipedia pages and what pages are about cats</td>
<td>Wikipedia pages</td>
</tr>
<tr>
<td>Classification of documents by author</td>
<td>Gutenberg Books</td>
</tr>
</tbody>
</table>
## Projects on Text Generation/Simulation

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Data Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>The botinator chatbot system</td>
<td>IMDB</td>
</tr>
<tr>
<td>Author simulator 2015</td>
<td>General text</td>
</tr>
<tr>
<td>Poetry generation</td>
<td>General text</td>
</tr>
<tr>
<td>Text simulation</td>
<td>General text</td>
</tr>
<tr>
<td>Grammar-learning and sentence-generating AI</td>
<td>General text</td>
</tr>
</tbody>
</table>
# Projects on Product/Movie Reviews

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Data Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotten or fresh? An exploration on the freshness of movies</td>
<td>Rotten Tomatoes movie reviews</td>
</tr>
<tr>
<td>Rotten tomatoes movie review classification with machine learning and NLP</td>
<td>Rotten Tomatoes movie reviews</td>
</tr>
<tr>
<td>Analysis of different algorithms in classifying reviews</td>
<td>Product reviews</td>
</tr>
<tr>
<td>Reach for the stars: prediction of product review star ratings</td>
<td>Product reviews</td>
</tr>
<tr>
<td>Foodiecity</td>
<td>Facebook, Yelp, Twitter</td>
</tr>
</tbody>
</table>
### Projects on Sentiment Analysis

<table>
<thead>
<tr>
<th>Project Title</th>
<th>Data Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidence and sentiment analysis algorithms</td>
<td>Rotten Tomatoes movie reviews</td>
</tr>
<tr>
<td>News and stock price relationship via sentiment analysis and linear regression</td>
<td>News articles</td>
</tr>
<tr>
<td>Technology stock market predictions based on twitter trends</td>
<td>Twitter</td>
</tr>
<tr>
<td>Tonal analysis of tweets</td>
<td>Twitter</td>
</tr>
<tr>
<td>Sentiment analysis</td>
<td>Twitter</td>
</tr>
<tr>
<td>Sentiment analysis of trends on microblogs</td>
<td>Twitter and/or Facebook</td>
</tr>
<tr>
<td>BladeRunner, sentiment analysis on Twitter tweets</td>
<td>Twitter</td>
</tr>
</tbody>
</table>
# Projects on Other Topics

<table>
<thead>
<tr>
<th>Project Title</th>
<th>Data Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter trend detection</td>
<td>Twitter</td>
</tr>
<tr>
<td>Exploring spam filtering by applying various classifiers with SMS spam data sets</td>
<td>SMS texts</td>
</tr>
<tr>
<td>AutoCorrect</td>
<td>Any text</td>
</tr>
<tr>
<td>AlgorithmicQuestGenerator</td>
<td>Gutenberg Books</td>
</tr>
</tbody>
</table>
Slides from Students about Projects in Previous CS 175 Classes
Twitter Sentiment Analysis

Used Sentiment140 tweets DB

Preprocessing:
- Removed and replaced URLs, usernames, and hashtags with `<URL>`, `<USER>`, `<HASHTAG>`
- Omitted neutral sentiment tweets from test sets

We found that the SVM classifier had the most growth potential in accuracy.

The SVM classifier without the data preprocessed had an accuracy of 79%. After preprocessing the data, it became 4% more accurate. We believe this is because of the reduction of features, which made the classifier easier to classify the data.

We plan to further investigate and experiment with more feature sets.
It's interesting that training data with stop words predicts test data better training data without stop words. The reason maybe stop words connect sentences to form the meaning of the whole review.
Accuracy with other categories of tweets as negative examples for sarcasm.

<table>
<thead>
<tr>
<th>Category</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Happy</td>
<td>0.8594313175810974</td>
</tr>
<tr>
<td>Sad</td>
<td>0.7597116539847817</td>
</tr>
<tr>
<td>Fearful</td>
<td>0.8157366519470093</td>
</tr>
<tr>
<td>Courageous</td>
<td>0.912745545911375</td>
</tr>
<tr>
<td>Sincere</td>
<td>0.7590909090909090</td>
</tr>
<tr>
<td>Relaxed</td>
<td>0.8740629685157422</td>
</tr>
<tr>
<td>Stressed</td>
<td>0.8192771084337349</td>
</tr>
</tbody>
</table>

Table Legend:
Left: Label of the tweets that are mixed in with sarcastic tweets

Right: Accuracy of the classifier with this particular mix of tweets.

Not bad for sarcastic!

Classifier Accuracy
mystery suspense

kingsman

bag of words

bag of words

bag of words

bag of words

bag of words

bag of words

bag of words

bag of words

bag of words

bag of words

128 minutes

matthew vaughn

2/13/15

55 word synopsis

5 person abridged cast

20th century fox

fresh

fresh

r

1 feature

1 feature

1 feature
Rotten Tomatoes Movie Review Classification With Machine Learning and Natural Language Processing

<table>
<thead>
<tr>
<th></th>
<th>Overall Error</th>
<th>False Positive Error</th>
<th>False Negative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>12.38%</td>
<td>16.58%</td>
<td>9.54%</td>
</tr>
<tr>
<td>Multinomial Naive Bayes</td>
<td>13.62%</td>
<td>18.55%</td>
<td>9.59%</td>
</tr>
</tbody>
</table>

Figure: Error Rate Results - Binary Classification of Positive vs Negative Phrases

- How we conduct the binary classification:
  We removed phrases labeled 2, which is neutral. Then, we combined 0 and 1 into one class “Negative”, which we relabeled as 0, and 4 and 5 into one class “Positive”, which we relabeled as 1.

- We were surprised at how well the classifier was able to distinguish between positive and negative phrases. We are looking into creating a multi-level classifier that would try to classify whether a phrase is neutral or not, and then using this to determine whether that phrase is actually positive or negative.
Lessons learned along the way...

Preprocessing works:

The Logistic Regression classifier correctly classified 70.19% of the Reuters test set in a bag of words representation, and 73.61% on the lemmatized version. Lemmatizing provided a 3.42% increase in accuracy, which lead to 112 more documents being properly classified.

Cross-validate first:

After weeks of testing the Support Vector Machine classifier on only the standard train/test splits, we finally ran cross-validation and were surprised to find that the SVM was significantly less accurate than we thought. For example, it is producing 68.1% accuracy on the Ohsumed 5-category standard split, but only 59.74% in cross-validation on the same data. It is actually being outperformed by LR!

Experiment a little:

Using the Multinomial Naive Bayes classifier, an attempt was made to reduce the number of categories used to train the classifier for each document. The process produced less accuracy but this only confirmed that our baselines for each experiment were in fact the greatest lower bounds.
Exploring Spam Filtering by Applying Various Classifiers with SMS Spam Data Sets

Unexpected result:

By implementing all the heuristics and experimenting on them, we found out that checking the misspelled words provides more information in determining spam SMS.

Reason:

By looking at that data, we found that the spam messages are constructed in a more “formal” format than regular message, so they have less incorrect words.
“aww! im so happy cuz you are happy 8D i have to go goodnight girl!! you are AMAZING! have a nice night! loveya?”

• Naïve bayes probability .93
• SVM distance 1.75

“I hate the way pills make my stomach feel like it's boiling. This is worse than being sick. Also”

• Naïve bayes probability .97
• SVM distance -2.10

The above are two tweets that both classifiers were very confident about. Both were mislabeled by the user. The first was incorrectly labeled by the user as negative and the second was incorrectly labeled as positive. The tweets sentiment however is obviously the opposite. We have found at least a few tweets with this problem
Twitter Trend Detection

- While many claim social media to be negative (even toxic), the top words are often positive!
- “love”, “like”, “good”, “please”, “best”
- Negative words and obscenities are further down.
- Twitter isn’t as bad as you think ;)

![Graph showing word counts](image.png)
 Parsing Difficulties

- We are using the Stanford parser to generate parse trees for our PCFG. For complex sentences, the parser often finds sentences within sentences. As a result, we generate sentences with strange punctuation (periods in the middle, comma at the end, etc.).

- Also, some complex sentences are parsed as NP (noun phrase) rather than S (sentence). As a result, some of our generated sentences are just single noun phrases, and some noun phrases become complex sentences.

Bad Generated Tree Example:

```
ROOT
  ↓
  S
   ↓
   NP  VP
       ↓
       NP  SBAR
           ↓
           ...
           ↓
           S
```

- Solution: sample the generated RHS using both the symbol and its direct parent. This lets us tell the difference between a sentence/NP at the root vs. in the middle of the tree.

- Similar to using a higher-order Markov chain

- Examples of valid expansions:
  - (ROOT -> S) -> NP VP .
  - (SBAR -> S) -> NP VP
  - (ROOT -> NP) -> NP : S .
  - (S -> NP) -> DT NN
Poetry Generator - Interesting Find

- We are using a Markov generator to produce our initial sentences (sentences that have not gone through our poetry formatting yet).
- When using a Dr. Seuss data set, the following was yielded:
  
  Congratulations!
  Today is your day!
  Your mountain is waiting.
  So... get on your own.
  And when you're in a Lurch.
  You'll come down from the Lurch with an unpleasant bump.
  And then things start to happen, don't worry.
  Don't stew.

- It's pretty short, but this is to be expected due to the end character delimiter paring with the short sentences found in Dr. Seuss...
But What We Didn’t Expect...

- The adverse effect is the potential of a few large sentences causing sentences to become freakishly long more often than we want.
- For example, this is from the text Jeeves:
  
  Jeeves, my man, you know what I subsequently learned was Madison Square Garden, where Mr. Pepper?" "Yes.
  
  I proposed to her at lunch one Sunday before I knew squads of chappies down Washington Square way who started the evening papers as the rest of the zest with which she had lost a bit of lunch with me to broach the subject.
  
  George’s uncle was in your way, Mr. Sturgis has offered his services to his expressed intention of remaining in the drawing-room in my autobiography.

- So, need a word count MAX and MIN, otherwise too much stress on poetry generator to find syllable count and delete words
Jimmy Fallon tweet Simulation

wine for cats. Because every girl
dreams of hearing her man say 'nice
!!', for already getting

Thank you guys like having @
StephenAtHome and climbing a guy
raised $ 20 min: Man never heard
even realize they love you to the
photobomb with @

# WhyImSingle tweets helped #
LateNight # FunShow http://t . I spent
the return of The Year . # LateNight #
VladdyPootPoot ', b ' l . Ha !!! London
# funtimes # BIYM # FallonTonight ', b

getting old, but whee did it again ! #
BurgerSummit # nomayo ", b ' Arnold @
Schwarzenegger showed me some of his other
reviews were bad .

need a little help Obama \xe2 \x9d l ( ft . Any opportunity to have no that guy
for the show about this many people
wearing a werewolf

he was arrested for stopping by the
day ? Keep tweeting the man say it
was throwing shade last night at 10
minutes . co / Ux3WCDbJG2 ', b ' 
Thanks Jon every post - Aime with a
baby carrots
Projects on Text Simulation
How could we Simulate Realistic-Looking Sentences?

Markov model approach

- Store a large list of counts of frequently occurring bigrams
- Sample the next word using Probability(next word | current word)
- Include punctuation in our vocabulary
How could we Simulate Realistic-Looking Sentences?

Markov model approach

– Store a large list of counts of frequently occurring bigrams
– Sample the next word using Probability(next word | current word)
– Include punctuation in our vocabulary

– Example:
  • This is very good .
How could we Simulate Realistic-Looking Sentences?

Markov model approach
- Store a large list of counts of frequently occurring bigrams
- Sample the next word using Probability(next word | current word)
- Include punctuation in our vocabulary

- Example:
  - This is very good .

- Another Example:
  - This is not a very good .

What is the source of this problem?
Grammar has long range dependencies.....beyond pairs of words
A more complex example of a parse tree....

From https://www.safaribooksonline.com/library/view/python-3-text/9781782167853/ch06s10.html
A Project on Text Synthesis?

• Build a model that can generate text in a particular style, e.g.,
  – Training data = text documents from a particular author
  – Algorithm: learns a model of the style and vocabulary for that author
  – Simulator: can generate new text that the author has not spoken before

• Could investigate different Markov modeling or Ngram methods and grammar-based techniques

• An important question is how you can evaluate your model
  – Typically want to compare the quality of Method A versus Method B
    • E.g., Method A is a simple baseline, and Method B is the method you developed
    • This is known as an A/B test (widely used in industry and research)
  – You give samples of text generated by each method to a set of human evaluators
    • The evaluators are not told which text came from Method A or B
    • They rank which of the two pieces of text they think is highest quality
    • Can repeat this multiple times with the same evaluator
    • Can then do a statistical test to see if Method A or B is consistently better
Resources for Text Simulation/Generation

Using “Natural Language Generation” as a query:

http://en.wikipedia.org/wiki/Natural_language_generation
http://swizec.com/blog/natural-language-generation-system-architectures/swizec/4535
https://code.google.com/p/simplenlg/wiki/Section1
http://en.wikipedia.org/wiki/Markov_chain#Markov_text_generators
https://code.google.com/p/simplenlg/
https://inlg2014.wordpress.com/
http://doc.utwente.nl/65551/1/templates-squib.pdf
http://www.gilesthanomas.com/2010/05/generating-political-news-using-nltk/
Example (in Python) of Classifying Yelp Reviews

(slides provides by Dimitris Kotzias, PhD student, Computer Science Department, UCI)
Real Example from Yelp Data

Simple pipeline for classification of Yelp Reviews

- Extract the restaurant reviews
- Convert them to a tf*idf array
- Split data into training and testing
- Train on training data, and Test

```python
if __name__ == '__main__':
    extract_restaurant_reviews()
    X, Y = convert_to_array()
    X_train, X_test, Y_train, Y_test = split_data(X, Y)
    train_and_test(X_train, X_test, Y_train, Y_test)
```
### Yelp Dataset

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Reviews</td>
<td>706,693</td>
</tr>
<tr>
<td>Number of Reviews w/o Neutral</td>
<td>595,468</td>
</tr>
<tr>
<td>Number of Tokens</td>
<td>85,392,376</td>
</tr>
<tr>
<td>Vocabulary Size w/o Stopwords</td>
<td>176,114</td>
</tr>
<tr>
<td>Array Dimensions</td>
<td>(595468, 176114)</td>
</tr>
<tr>
<td>Non-zero entries</td>
<td>28,357,001</td>
</tr>
<tr>
<td>Density</td>
<td>0.000027027</td>
</tr>
</tbody>
</table>
Histogram of Review Lengths
Real Example from Yelp Data

```python
def extract_restaurant_reviews():
    # get all the ids of restaurants
    ids = set()
    with open('./data/yelp/restaurants.json', 'r') as jfile:
        for line in jfile:
            data_point = json.loads(line)
            ids.add(data_point['business_id'])
    print('Total restaurants: ', len(ids))

    # get all the reviews
    reviews = []
    with open('./data/yelp/yelp_academic_dataset_review.json', 'r') as jfile:
        for line in jfile:
            r = json.loads(line)
            id = r['business_id']  # if business is a restaurant
            if id in ids:
                reviews.append(r)

    # save the reviews
    with open('./data/yelp/restaurant_reviews.json', 'w') as output_file:
        json.dump(reviews, output_file)
        output_file.write("\n")
    print('A total of ', len(reviews), ' reviews')

# Total restaurants: 14,308
# A total of 706,693 reviews
```
Real Example from Yelp Data

```python
# Grab only the text, then convert it to a tfidf matrix
def convert_to_array(min_pos=4, max_neg=2):
    dir = './data/yelp/
    name = dir + 'restaurant_reviews.json'  # load data
    with open(name, 'r') as jfile:
        data = json.load(jfile)

    text = []
    Y = []
    for d in data:  # keep only the text and label
        review = d['text']
        stars = int(d['stars'])
        if stars >= min_pos:  # translate number of stars to binary
            score = 1
        elif stars <= max_neg:
            score = 0
        else:
            continue  # do not consider neutral

        text.append(BeautifulSoup(review).get_text())
        Y.append(score)

    # parameters should change depending on problem
    vectorizer = TfidfVectorizer(stop_words='english', max_df=1.0, min_df=0.0)  # this is awesomest.
    X = vectorizer.fit_transform(text)

    print 'data shape: ', X.shape
    return X, Y
```

data shape: (595468, 176114)
Real Example from Yelp Data

```python
# split to train and test

def split_data(X, Y, test_size=0.5):
    data_train, data_test, labels_train, labels_test = train_test_split(X, Y, test_size=test_size, random_state=42)
    # important to be random, but have same results across different runs ;)

    print 'training size: ', data_train.shape[0],
    print 'testing size: ', data_test.shape[0]  # careful these are sparse matrices

    return data_train, data_test, labels_train, labels_test

training size: 297734
testing size: 297734
```
Real Example from Yelp Data

```python
def train_and_test(X_train, X_test, Y_train, Y_test):

    # Specify the model. Again parameters should change
    logreg = linear_model.LogisticRegression(penalty='l2', fit_intercept=True)  # fit_intercept= bias

    # Train....
    logreg.fit(X_train, Y_train)
    pickle.dump(logreg, open('./data/yelp.log_model.pkl', 'w'))  # save in case we need later

    print('Training: '),
    predicted = logreg.predict(X_train)  # Test
    print('acc:', metrics.accuracy_score(Y_train, predicted))

    print('Testing: '),
    predicted = logreg.predict(X_test)  # Test
    probs = logreg.predict_proba(X_test)
    print('acc:', metrics.accuracy_score(Y_test, predicted))
    print('auc:', metrics.roc_auc_score(Y_test, probs[:, 1]))  # this is easy to plot as well

Training: acc: 0.95586
Testing: acc: 0.94812
auc: 0.98233
Overall takes about 15-20 mins to run (may produce some warnings)
```
Planning and Organization of Projects
Project Tips: Goals

• Be clear in your goals
  – e.g., “will systematically evaluate the accuracy of logistic regression and neural network classifiers on the Reuters data set and 2 other data sets”
  – Ok to not to have all the details of how you will get there, but important to know what the goal is
  – Ok if goals are updated/changed as you learn more about the problem

• Team members should agree on the goals
  – To be effective the team needs to be clear about the goals
  – If there is any doubt about what the goal is, ask questions and discuss
Project Tips: Plan in Stages

Plan your project in stages so that the overall project is not dependent on the riskier elements working.

Example:

PHASE 1
- Original Documents
- Standard Bag of Words
- Standard Logistic Regression
- Cross-Validation Experiments
Project Tips: Plan in Stages

Plan your project in stages so that the overall project is not dependent on the riskier elements working.

Example:

**PHASE 1**
- Original Documents
- Standard Bag of Words
- Standard Logistic Regression
- Cross-Validation Experiments

**PHASE 2**
- Bag of Phrases (ngrams)
Project Tips: Plan in Stages

Plan your project in stages so that the overall project is not overly dependent on the riskier elements.

Example:

**PHASE 1**
- Original Documents
- Standard Bag of Words
- Bag of Phrases (ngrams)

**PHASE 2**
- Standard Logistic Regression

**PHASE 3**
- Cross-Validation Experiments
- Deep Neural Network
**Project Tips: Evaluation Methods**

- Very important to have a clear idea of how you will evaluate your system

- For some tasks, such as document classification, there are well-defined metrics that are straightforward
  - E.g., cross-validated classification accuracy

- For other tasks, such as clustering, you will have to do some research to figure out what metrics are appropriate
  - For some projects, some user evaluation may be necessary

- Always include a baseline method in your experiments
  - E.g., for classification your baseline could be a Naïve Bayes classifier
Project Tips: Revision/Source Control for Code

- Each team should use a revision control system
  - e.g., Github system (freely available)
  - If you are not familiar with these systems, this is a good time to learn

- Revision control
  - Provides a systematic way for a team to develop code, scripts, documents, etc
  - Individuals can “check out” code, work on it, and then “commit”
  - Earlier versions of code can be recovered
    - Useful when you want to go back to an earlier version without a bug
Project Tips: Revision Control for Experiments

• You are likely to conduct many experiments over the course of the project, comparing versions of preprocessing, parameter settings, algorithms
  – You want to be able to keep track of your experiments and results in a systematic way

• Recommendations:
  – Organize and document your code/scripts for experiments
  – Use time-stamps, give your scripts interpretable names, use comments

• Consider using tools such as IPython Notebook, with github, for documentation and collaboration
  – just like a “lab notebook” in a science lab

• Reproducibility is important
  – Scripts will allow you to regenerate results from earlier in the project
  – Also allows team members to share results efficiently
General Project Tips: Writing

• Write clearly
  – Try to put yourself in the mind of the person reading it
  – Make sure you don’t leave out important details and concepts
  – Use spell-checkers, grammar-checkers, etc
  – Target audience = a random student in the class

• Work collaboratively
  – Teams can use a shared document with version control, e.g., Google Docs
  – Have each member of the team edit, read, comment on the document
  – Work on your documents collaboratively: will lead to better documents

• Use figures and tables when you can
  – “A picture is worth a 1000 words”