CS 175, Project in Artificial Intelligence

Lecture 6: Discussion of Projects (continued)

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine
Upcoming Schedule

• Project Proposals: Due this Friday at 10pm

• Office hours
  – My office hours: 10 to 11:30 tomorrow (Tuesday)
  – Jihyun: Wednesday, 3 to 5pm, DBH 4013
  – Kevin: today (Monday), 2 to 4pm, DBH 4059

• No discussion section this Friday

• Feel free to ask project-related questions via Piazza
Project Proposals
Project Proposals

• Your proposal should be 2 to 3 pages long
  – Use the project proposal template on the Assignment Web page

• 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.
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

• Get feedback, iterate
  – Target audience: someone who is a junior or senior in computer science
  – Give drafts of your proposal to other students/friends and ask for comments

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

• 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
1. Project Summary

A clear description (2 or 3 sentences) that summarizes

(a) the goal of the project (what is the primary aim of the project)

and

(b) how you plan to achieve the goal (mention the methods, algorithms, data sets, you plan to use).
2. Problem Description and Background
Write about 1 paragraph defining (in more detail than in the summary) what problem your project will address.

For example if your project is multi-label document classification then you would clearly define what multi-label document classification is.

Mention if you can what methods/algorithms (that you know of) that have been used in the past to address this problem.

Add one or more references to a text or a paper that discusses the problem if you can (see the class Web site for suggestions, or do a search in Google Scholar using appropriate keywords).
3. Data Sets

Briefly describe what data set(s) you plan to use in the project. Include references to the data (e.g., a URL) if you can.

If for example you are doing document classification, you can describe for example how many documents are in the data set, average document length, how many classification labels.

If you are using multiple data sets you could put this type of information in a table.

Mention for example whether you plan to work with data that already has a predefined vocabulary or whether you plan to define your own vocabulary.

If you are able to access and take an initial look at your data, feel free to also include a figure or two in this section, e.g., a histogram of document lengths.
4. Proposed Technical Approach
Provide a description of the methods and algorithms you will use on the project.

If the system you are building can be thought of as a pipeline with multiple components feel free to provide a figure that illustrates the pipeline with blocks for different components and brief descriptions of each component.

If your project involves comparing different algorithms (e.g., for document classification) then in this section you would list and briefly mention the algorithms you plan to use in your project (e.g., naïve Bayes, logistic regression, support-vector machines, neural networks, etc).
5. Experiments and Evaluation
Provide a brief and clear description of how you will evaluate the results of your project, e.g., accuracy for classification, precision-recall for document ranking.

Will you use cross-validation or does your data set(s) come with a fixed train-test partition?

For tasks like clustering or topic modeling you may have to do some research to see how evaluation is done on these tasks.

For some projects you may have to do some user studies for evaluation, e.g., present users with results from Algorithm A and Algorithm B, using the same input data for each algorithm, without telling the user which algorithm is which, and have them select the one they prefer.
6. Software
Provide a list of the major pieces of project software that you expect to use, divided into 2 sets:

   (1) publicly-available code, and
   (2) code will write yourself.

This list will probably be incomplete at this point (which is fine) since you may not know yet about all of the publicly-available software that might be relevant to your project.

My expectation is that most students will use Python, given that we have been using Python in class and there are many useful publicly-available tools for text analysis in Python. However, if you prefer to use a language such as Java that is ok too - please indicate this clearly in this section.
7. Milestones
Provide a brief list of milestones, e.g., since the project will span 6 weeks of the class (weeks 5 to 10), you may want to break your milestones into a list of 3 intermediate phases:

- Weeks 5 and 6
- Weeks 7 and 8
- Weeks 9 and 10

For example, much of the data gathering and preprocessing and coding (development and test) could happen in the earlier weeks, and much of the experimentation and evaluation in the later weeks.

Note that you have a progress report due at the beginning of week 8 (on Monday Feb 22nd).
<table>
<thead>
<tr>
<th>Week</th>
<th>Monday</th>
<th>Wednesday</th>
<th>Deadlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 4th</td>
<td>Introduction and course outline</td>
<td>Basic concepts in automated text analysis</td>
<td></td>
</tr>
<tr>
<td>January 11th</td>
<td>Review of classification algorithms</td>
<td>Text classification methods</td>
<td>Assignment 1 (Monday, 10pm)</td>
</tr>
<tr>
<td>January 18th</td>
<td>No class (University holiday)</td>
<td>Ideas for class projects</td>
<td>Assignment 2 (Friday)</td>
</tr>
<tr>
<td>January 25th</td>
<td>Discussion of project proposals</td>
<td>Text clustering methods</td>
<td>Project Proposal (Friday)</td>
</tr>
<tr>
<td>February 1st</td>
<td>Evaluation methods</td>
<td>Word prediction methods</td>
<td></td>
</tr>
<tr>
<td>February 8th</td>
<td>Topic modeling for text</td>
<td>Office hours (no lecture)</td>
<td></td>
</tr>
<tr>
<td>February 15th</td>
<td>No class (University holiday)</td>
<td>Office hours (no lecture)</td>
<td></td>
</tr>
<tr>
<td>February 22nd</td>
<td>Office hours (no lecture)</td>
<td>Office hours (no lecture)</td>
<td>Progress report (due Monday)</td>
</tr>
<tr>
<td>March 1st</td>
<td>Student presentations (in class)</td>
<td>Office hours (no lecture)</td>
<td></td>
</tr>
<tr>
<td>March 8th</td>
<td>Student presentations (in class)</td>
<td>Student presentations (in class)</td>
<td></td>
</tr>
<tr>
<td>March 15th</td>
<td>Final Report (due Monday)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8. Individual Student Responsibilities
Summarize briefly what each student will be primarily responsible for in the project.

For example, if you are implementing 2 algorithms and there are 3 students, then one arrangement could be:

**Name 1:** will write and test the code for Algorithm 1, will assist in doing experiments and interpreting results, will assist in writing project reports

**Name 2:** will write and test the code for Algorithm 2, will assist in doing experiments and interpreting results, will assist in writing project reports

**Name 3:** will acquire the data sets to test the algorithms, will preprocess the text data (e.g., define the vocabulary for the algorithms), will write the scripts for evaluating the accuracy of the algorithms, will assist in writing project reports.

[Note these are just suggestions – you can and should organize responsibilities in whatever way makes sense – and inevitably as the project progresses these responsibilities may need to be changed as some tasks may take much more time (or much less time) than originally expected.]
Project Teams

- Students will work in 3-person project teams
  - Contact me via Piazza (e.g., privately) if
    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.
Example (in Python) of Classifying Yelp Reviews

Slides provided by Dimitris Kotzias, PhD student, Computer Science Department, UCI

Questions about the code: email Dimitris at jimkotz2 at gmail.com
The fifth round of the Yelp Dataset Challenge ran throughout the first half of 2015 and we were quite impressed with the projects and concepts that came out of the challenge. Today, we are proud to announce the grand prize winner of the $5,000 award: “From Group to Individual Labels Using Deep Features” by Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth (from the University of California, Irvine, the University of Oxford, and the Canadian Institute for Advanced Research). This paper proposes a novel approach to using group-level labels (e.g. the category of an entire review) to learn instance-level classification (e.g. the category of specific sentences inside this review). The authors designed a new objective (cost) function for training a model which uses features from a deep-learning convolutional neural network. This trained neural network can, in turn, be used as a classifier predicting which category a specific instance belongs to. Their innovative research has broad implications for a variety of fields, and not just text classification.

This entry was selected from many submissions for its technical and academic merit. A PDF copy of the winning paper can be found at this location. This paper was published in the Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. For a full list of all previous winners of the Yelp Dataset Challenge, head over to the challenge site. Thanks to all who participated!
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)
```
Real Example from Yelp Data

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

Number of Reviews

Review Length
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('
')
    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 tf*idf 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 awes
    X = vectorizer.fit_transform(text)

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

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
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 accuracy: 0.95586
Testing accuracy: 0.94812
Area under the Curve (AUC): 0.98233

Overall takes about 15-20 mins to run (may produce some warnings)
Woohoo! Keeping it Fresh: Predict Restaurant Inspections has come to a close!

Many thanks to the participants for all of their hard work and commitment to using data for good!

Submissions Close:
July 7, 2015, 11:59 p.m.

Evaluation Period Ends:
Aug. 19, 2015, 11:59 p.m.

<table>
<thead>
<tr>
<th>Place</th>
<th>Prize Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>$3,000</td>
</tr>
<tr>
<td>2nd</td>
<td>$1,000</td>
</tr>
<tr>
<td>3rd</td>
<td>$1,000</td>
</tr>
</tbody>
</table>
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.gilesthomas.com/2010/05/generating-political-news-using-nltk/
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)
      - Deep Neural Network
  - Standard Logistic Regression
    - Cross-Validation Experiments

PHASE 2

PHASE 3
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 to you to regenerate results from earlier in the project
  – Also allows team members to share results efficiently
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*
Upcoming Schedule

• Project proposals: Due this Friday at 10pm

• Office hours
  – My office hours: 10 to 11:30 tomorrow (Tuesday)
  – Jihyun: Wednesday, 3 to 5pm, DBH 4013
  – Kevin: today (Monday), 2 to 4pm, DBH 4059

• No discussion section this Friday

• Feel free to ask project-related questions via Piazza