CS 175, Project in Artificial Intelligence

Lecture 6:

1. Projects and Proposals
2. Text Generation

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

• Project Proposals
  – Due Friday 6pm
  – Detailed information on the class Website
  – If you are not on a team: please see me at the end of class
  – If you are a team looking for another member: please also see me

• Office Hours:
  – Eric: Thursday 1 to 3
  – Me: Friday 9:30 to 11:30

• Today’s Lecture:
  – Projects and proposals
  – Discussion of text simulation algorithms
Guidelines for Projects

• 2 or 3 students per project
  – Submit only one project proposal per team (include team member names on the proposal)
  – All team members will get the same grade for the proposal.

• Use of external code is allowed and encouraged
  – Such code needs to be acknowledged in your reports

• You must write at least some functionality on your own
  – What you implement is up to you
  – You will need to submit your code at the end of the quarter
Ideas for Finding Project Topics

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

• Browse and read articles in *Links to Tutorial Articles*... on the class Web page

• Browse paper titles and abstracts from research conferences, use search engines (e.g., Google Scholar) 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?”
Project-Related Reference Material for CS 175

CS 175, Winter 2017
Below are links to suggested reading organized by topic. If you are doing a project on any of these topics (or interested in potentially doing a project on these topics) then these online resources should be helpful.

**Text Classification**
Chapter on text classification and naive Bayes from Manning et al
Chapter on vector-based classification for text from Manning et al
Chapter on logistic regression for document classification from Jurafsky and Martin
Comprehensive survey paper on text classification algorithms by Aggarwal and Zhai (2012)
Overview of general principles in machine learning from Goodfellow et al (2016)
Tutorial paper on multi-label classification methods by de Carvalho and Freitas

**Sentiment Analysis**
Chapter on naive Bayes and sentiment classification from Jurafsky and Martin
Survey papers on opinion mining and sentiment analysis: by Pang and Lee (2008) and by Liu and Zhang (2012)

**Sequential Classifiers**
Chapter on recurrent and recursive neural networks from Goodfellow et al (2016)

**Document Clustering**
Chapters on flat clustering algorithms and hierarchical clustering algorithms for text documents, from Manning et al

**Topic Modeling**
Tutorial paper on topic modeling by Steyvers and Griffiths
David Blei’s page on topic modeling

**Vector Embeddings**
Chapter on dense vector representations for words from Jurafsky and Martin
Chapter on latent semantic indexing from Manning et al
Open domain event extraction from twitter
A Ritter, O Etzioni, S Clark - Proceedings of the 18th ACM SIGKDD ..., 2012 - dl.acm.org
... networking sites such as Facebook and Twitter have become an important complementary source of such information. While status messages contain a wealth of useful information, they are very disorganized motivating the need for automatic extraction, aggregation and ...
Cited by 290  Related articles  All 17 versions  Cite  Save  More

Topical keyphrase extraction from twitter
WX Zhao, J Jiang, J He, Y Song... - Proceedings of the 49th ..., 2011 - dl.acm.org
... Since there is no existing test collection for topical keyphrase extraction from Twitter, we manually ...
In order to achieve high phraseness, we first computed the minimum value of pointwise mutual information for all bigrams in one combination, and we removed combinations ...
Cited by 189  Related articles  All 18 versions  Cite  Save

Traffic condition information extraction & visualization from social media twitter for android mobile application
SK Endamoro, S Pradipra, AS Nugroho... - Electrical Engineering ..., 2011 - ieeexplore.ieee.org
Abstract: Traffic jam in Jakarta, Indonesia has become a crucial problem for the society. A Traffic Management Center has been built by the police, in this case Polda Metro Jaya to help people to get the latest information regarding traffic jam. Twitter has been used by TMC
Cited by 47  Related articles  All 9 versions  Cite  Save

Ontology-based information extraction from twitter
K Nehiri - 2012 - archive-ouverte.unige.ch
The popular microblogging service Twitter provides a vast amount of short messages that contains interesting information for Information Extraction tasks. This paper presents a rule-based system for the recognition and semantic disambiguation of named entities in tweets.
Cited by 14  Related articles  All 12 versions  Cite  Save

[TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text.](http://academia.edu/)
K Bontcheva, I Dermanyanski, A Funk, MA Greenwood... - RANLP, 2013 - acadamia.edu
... To combat these problems, research has focused on microblog-specific information extraction algorithms (e.g. named entity recognition for Twitter using CRFs (Ritter et al., 2011), Wikipedia-based topic and entity disambiguation (van Erp et al., 2013), ...) Cited by 105  Related articles  All 16 versions  Cite  Save  More
Examples of Data Sets for Text Analysis

CS 175, Winter 2017

The links below point to just a few 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 use for your projects.

Data Sets with Classification Labels or Ratings
- 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 (can be used to build classifiers using category labels or to provide additional information for other models such as n-gram statistics)
- 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).
- Document classification data sets (a large collection of different data sets used in text classification research)

Other 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 2015, 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 ended in 2016)
- The BioASQ data sets and challenge competitions on question answering for the biomedical domain
Yelp Dataset Challenge

Discover what insights lie hidden in our data.

The Challenge

We challenge students to use our data in innovative ways and break ground in research. Here are some examples of topics we find interesting, but remember these are only to get you thinking and we welcome novel approaches!

Photo Classification

Maybe you’ve heard of our ability to identify hot dogs (and other foods) in photos. Or how we can tell you if your photo will be beautiful or not. Can you do better?

Natural Language Processing & Sentiment Analysis

What’s in a review? Is it positive or negative? Our reviews contain a lot of metadata that can be mined and used to infer meaning, business attributes, and sentiment.

Graph Mining

We recently launched our Local Graph but can you take the graph further? How do user’s relationships define their usage patterns? Where are the trend setters eating before it becomes popular?

Round 11

Our dataset has been updated for this iteration of the challenge - we’re sure there are plenty of interesting insights waiting there for you. This set includes information about local businesses in 11 metropolitan areas across 4 countries. Round 11 has kicked off starting January 18, 2018 and will run through June 30, 2018.

Download Dataset
IMDb Datasets

Subsets of IMDb data are available for access to customers for personal and non-commercial use. You can hold local copies of this data, and it is subject to our terms and conditions. Please refer to the Non-Commercial Licensing and copyright/license and verify compliance.

Data Location

The dataset files can be accessed and downloaded from https://datasets.imdbws.com/. The data is refreshed daily.

IMDb Dataset Details

Each dataset is contained in a gzipped, tab-separated-values (TSV) formatted file in the UTF-8 character set. The first line in each file contains headers that describe what is in each column. A \"N\" is used to denote that a particular field is missing or null for that title/name. The available datasets are as follows:

title basics.tsv.gz - Contains the following information for titles:
  - tconst (string) - alphanumeric unique identifier of the title
  - titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc)
  - primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release
  - originalTitle (string) - original title, in the original language
  - isAdult (boolean) - 0: non-adult title; 1: adult title.
  - startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year.
  - endYear (YYYY) – TV Series end year. \"N\" for all other title types
Completed • Knowledge • 861 teams

Sentiment Analysis on Movie Reviews
Fri 28 Feb 2014 – Sat 28 Feb 2015 (23 months ago)

Data Files

File Name           | Available Formats
--------------------|------------------
test.tsv             | .zip (470.84 kb)
sampleSubmission     | .csv (582.06 kb)
train.tsv            | .zip (1.21 mb)

The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a Phraseld. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data.

- train.tsv contains the phrases and their associated sentiment labels. We have additionally provided a Sentenceld so that you can track which phrases belong to a single sentence.
- test.tsv contains just phrases. You must assign a sentiment label to each phrase.

The sentiment labels are:

0 - negative
1 - somewhat negative
2 - neutral
3 - somewhat positive
4 - positive
Amazon product data

Julian McAuley, UCSD

New - Q/A data!

See our newly-released Q/A data (described in our WWW 2016 paper)!

Description

This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014.

This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).

Files

"Small" subsets for experimentation

If you're using this data for a class project (or similar) please consider using one of these smaller datasets below before requesting the larger files. To obtain the larger files you will need to contact me to obtain access.

K-cores (i.e., dense subsets): These data have been reduced to extract the k-core, such that each of the remaining users and items have k reviews each.

Ratings only: These datasets include no metadata or reviews, but only (user,item,rating,timestamp) tuples. Thus they are suitable for use with mymedialite (or similar) packages.

<table>
<thead>
<tr>
<th>Category</th>
<th>k-core (reviews)</th>
<th>ratings only (ratings)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Books</td>
<td>5-core (8,898,041)</td>
<td>ratings only (22,507,155)</td>
</tr>
<tr>
<td>Electronics</td>
<td>5-core (1,689,188)</td>
<td>ratings only (7,824,482)</td>
</tr>
<tr>
<td>Movies and TV</td>
<td>5-core (1,697,533)</td>
<td>ratings only (4,607,047)</td>
</tr>
<tr>
<td>CDs and Vinyl</td>
<td>5-core (1,097,592)</td>
<td>ratings only (3,749,004)</td>
</tr>
<tr>
<td>Clothing, Shoes and Jewelry</td>
<td>5-core (278,677)</td>
<td>ratings only (5,748,920)</td>
</tr>
<tr>
<td>Home and Kitchen</td>
<td>5-core (551,682)</td>
<td>ratings only (4,253,926)</td>
</tr>
</tbody>
</table>
Different Types of Projects with Review Data

• Predict the review score from the text of the review
  – e.g., sentiment classification: classify as positive/{4,5} versus negative/{1,2}

• Automatically summarize a set of reviews
  – Extract phrases or sentences that mention specific aspects of an item/business, e.g., “price”, “quality”
  – For a given item/business, for each specific aspect, summarize how negative or positive the reviews are for that aspect

• Detect and predict trends in reviews over time
  – For a given restaurant, movie, product....
  – Measure positive/negative sentiment of reviews over time (e.g., per week)
  – Use time-series prediction methods to predict future “direction” of reviews
    • e.g., initially positive reviews that turn negative over time
Examples of Projects from past CS 175 Classes

• Automatically matching resumes to jobs

• Predicting what subreddit a Reddit post should go to or its popularity

• Predict whether a restaurant will close or not from review data

• Identifying the genre of a movie or song

• Simulating text using neural networks

• Rating the quality of answers to questions on StackOverflow

• Generating spatial maps of user happiness from Twitter
Different Types of Projects in General

- **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
Planning a Project

• Topic: select a general type of problem you are interested in, e.g.,
  – Sentiment analysis, information extraction, simulation, chatbots, etc

• Do some background reading to learn more about the topic
  – e.g., from chapters in the books I recommended
  – e.g., from links on class Web site
  – e.g., from Google searches (e.g., research papers, blogs)

• Define your problem as precisely as you can

• Determine at least 1 or 2 data sets you can use for your project

• Figure out how you will evaluate your results, e.g., to compare A v B
  – Experiments: classification accuracy, precision/recall, etc
  – User Studies: human users compare results from A and B
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 part of speech tagger
  – A neural network classifier
  (you could then compare your system to existing system(s))
Questions from Office Hours

Question:
How accurate does our system need to be (e.g., if it’s a classification algorithm) to ensure we get a good grade?

Answer:
Not necessarily accurate at all.
Team A has a project with 99% accuracy
Team B has a project with 60% accuracy
Who gets a better grade?.....it depends on multiple factors, e.g.,

....complexity of the project (“degree of difficulty”) 

....insights (“what was learned”, e.g., what types of errors does the system make? Are these the same types of errors that humans make? What would it take to get a really accurate system, e.g., more data?)
Questions from Office Hours

Question:
*We need to develop a crawler and parser to gather data for our project (e.g., crawling Web sites for resumes and job ads). Is the work on developing a crawler and parser be considered part of our project?*

Answer:

Yes, this can be counted as part of your project work.

But it should not take too large a fraction of your time (e.g., 1 week or so at most)

Otherwise it means you are spending too much time just on collecting data rather than developing algorithms.
Questions from Office Hours

Question:
We are working on a challenging text classification problem where there are 10 different classes (e.g., different types of sentiment, different genres of movies or music, etc). We know from reading research papers on this topic that it can be difficult classify data into all 10 classes accurately. What do you suggest?

Answer:
It is fine (particularly when starting out) to select a subset of the easier classes to work with, rather than all of the classes. For example, in movies you could pick Horror versus Romance, and ignore all other classes.

This will allow you to test your classifier on a problem where you expect to be able to accuracy better than random

You can then work up to more complex problems, e.g.,
- 3 classes, e.g., the 2 original classes plus a 3rd class consisting of all of the other classes
Questions from Office Hours

Question:
*We are working with 500,000 Yelp reviews with a complicated neural network classification problem and it takes 8 hours to just train the model. This is too slow to do experiments. What do you suggest?*

Answer:

You don’t need to work with full-sized data sets, particularly at the beginning.

In this case it would make sense to scale down to maybe 10,000 or 20,000 reviews instead of working with all 500,000. You can randomly select the smaller set, or select based on some criterion (e.g., from a particular city, particular category of restaurant, etc).

If there’s time you can experiment (or at least make predictions on) the full data set towards the end of the quarter.

Key point: the goal is not to build a state-of-the-art system but to understand the key ideas in terms of technical approaches, challenges, evaluation, etc.
Questions from Office Hours

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

Answer:
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.
# Project Deliverables and Deadlines

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Grade Percentage</th>
<th>Date Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Proposal</td>
<td>20%</td>
<td>Friday Feb 2nd</td>
</tr>
<tr>
<td>Progress Report</td>
<td>20%</td>
<td>Friday Feb 23rd</td>
</tr>
<tr>
<td>Presentation/Demo</td>
<td>10%</td>
<td>March 5&lt;sup&gt;th&lt;/sup&gt;/7th</td>
</tr>
<tr>
<td>Final Report</td>
<td>30%</td>
<td>Early Finals Week</td>
</tr>
</tbody>
</table>
General Criteria for how Projects will be Graded

• 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? E.g., what types of errors is the system making? What would it take to improve it?

• Writing and communication:
  – are you able to explain your work clearly?
Project Proposals
Project Proposals

• Your proposal should be 2 to 3 pages long
  – Required to use project proposal template (see class Website)

• 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.
Contents of Project Proposal

1. Project Summary
2. Problem Definition
3. Proposed Technical Approach
4. Data Sets
5. Experiments and Evaluation
6. Software
7. Milestones
8. Individual Student Responsibilities
1. Project Summary

A clear description (2 or 3 sentences) that summarizes your project: e.g., “This project will use XX methods to predict YY using the Z1 and Z2 data sets, with evaluation using classification accuracy and user studies.”

Examples:

- The goal of this project is to explore spam filtering by classifying SMS messages as spam or not spam using various machine learning techniques.

- Application and evaluation of multiple sentiment analysis classification algorithms using multiple datasets. Evaluation of correctness, uncertainty, and differences between the algorithms.

- Our project will be a poem generator that takes in a dictionary and outputs a poem that looks as if a human created the poem. We will be using various algorithms and libraries, such as NLTK and NodeBox, to identify different grammars and apply constraint satisfaction to make a poem that is clear and readable.
2. Problem Definition

Write a few sentences that clearly defines what problem you will be trying to solve. One way to describe this is to think about your project in terms of inputs and outputs: what will the inputs to your system be and what will it produce as output?

Example:

*Sentiment analysis (or opinion mining) is the technique used to correctly classify subjective information through natural language processing. Our specific project will address the problem of performing sentiment analysis on Rotten Tomatoes movie reviews. Based on the phrases located in the movie reviews, we will attempt to assign a sentiment class label between 0 and 4, which represents how negative or positive their review is about the movie. Popular methods for sentiment analysis include support vector machines, “bag of words,” and neural networks and deep learning.*
3. Proposed Technical Approach

- Write a paragraph with a clear description of the methods and algorithms you plan to 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.
3. Proposed Technical Approach (Example)

We are going to divide the whole project into several stages.

- The first stage is to preprocess the movie review. We plan to use stopwords list from NLTK to remove stopwords, punctuations and non-alphabetic words. Secondly, we plan to obtain the base part of the word by using Snowball as a stemmer to remove morphological endings. The Porter Stemming Algorithm will be implemented in this step. Then, we plan to extract opinioned words by applying the positive words list from NLTK. ....

- The second stage is to classify the opinioned words list obtained from the first stage.

- The third stage is training our classifier to recognize the attitude of the reviews. We plan to try logistic regression, support-vector machines, and Multinominal Naive Bayes models for supervised training.
3. Proposed Technical Approach (Example)

**Our Algorithm**

For each unique food product ID, we look at every sentence of individual reviews and remove all the stopwords (Python NLTK stopwords and our custom set of stopwords). This stopword removal step ensures that irrelevant, unhelpful key phrases do not get added to our set of key phrases. We then extract all n-grams that begin with an adjective and end with a noun (and vice versa) from the filtered reviews. We rank these key phrases based on their values of PF-IRF (phrase frequency - inverse review frequency), which is a variation of TF-IDF (term frequency - inverse document frequency). Finally, we can generate new sentences using only the key phrases with high rankings, along with a Markov model simulating common sentence structures.
4. 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 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.

• You can change your data sets during the project if you need to, but you should have identified at least one data set to work with by the time you submit the proposal.
4. Data Sets (Example)

- We plan to work with the Rotten Tomatoes movie review dataset publicly provided on Kaggle.com as the basis for a machine learning competition. The data set is a collection of sentences from reviews that are parsed into phrases by the Stanford Parser. The data set is preprocessed with a predefined vocabulary that simply removes repeated common or short phrases. We plan to improve this simple vocabulary through our own method of preprocessing as discussed in the next section.
4. Data Sets (Example)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Length</th>
<th>Type</th>
<th>Classified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yelp RSS feed <a href="http://www.yelp.com/rss">http://www.yelp.com/rss</a></td>
<td>25 reviews for 5 cities per day (125) Need to collect (1000+)</td>
<td>Short: around 300 characters</td>
<td>Review</td>
<td>Yes: each has a rating (N/5)</td>
</tr>
<tr>
<td>Song Lyrics <a href="http://www.songlyrics.com/">http://www.songlyrics.com/</a></td>
<td>Need to collect</td>
<td>Medium: Around 2,500 characters</td>
<td>Song, Poetry</td>
<td>No</td>
</tr>
<tr>
<td>Twitter Tweets <a href="http://www.sananalytics.com/lab/twitter-sentiment/">http://www.sananalytics.com/lab/twitter-sentiment/</a></td>
<td>5513 tweets Pre-packaged</td>
<td>Short: max 140 characters</td>
<td>Message</td>
<td>Yes: hand classified</td>
</tr>
<tr>
<td>Movie Reviews <a href="http://www.cs.cornell.edu/people/pabo/movie-review-data/">http://www.cs.cornell.edu/people/pabo/movie-review-data/</a></td>
<td>polarity_dataset_v2.0 1000 positive, 1000 negative reviews Pre-Packaged</td>
<td>Medium-Long</td>
<td>Review</td>
<td>Yes: positive/negative</td>
</tr>
</tbody>
</table>
4. Data Sets (Example)
4. Data Sets (Example)
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.

• Aspects to consider
  – Single metrics: classification accuracy
  – Curves: precision-recall
  – Test sets and cross-validation
  – User studies
5. Experiments and Evaluation (Examples)

- The data set comes in a set of two files: one train.tsv file and one test.tsv file. We will essentially use cross validation to split our training data set into a validation set and a testing set in order to evaluate our models. We could also use our models on the test.tsv file and upload a submission file onto Kaggle, which will evaluate our models and give us a corresponding score on the leaderboard. The Kaggle leaderboard currently has about 700 teams, who are ranked by how well their model performs on the test data provided in test.tsv - this can give us an accurate indication on how well our model performs in a more realistic setting.
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
6. Software (Example)

Publicly-available code:

- **NLTK**: provides a list of stop words and build-in naive Bayes classifier.
- **PyEnchant**: provides spell checking and spelling suggestions.
- More later.

Code to be written ourselves in Python:

- **Tokenizer** to parse SMS message.
- **Generate a feature list using tokenized message**.
- **Group misspelled words using PyEnchant suggestions**.
- **Track metadata such as number of misspellings in a message**.
- **Bernoulli naive Bayes** to analyze features and classify messages.
## 6. Software (Example)

<table>
<thead>
<tr>
<th>Publicly-Available Code</th>
<th>Code We Will Write/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programming Languages:</strong></td>
<td><strong>Ranking Algorithms:</strong></td>
</tr>
<tr>
<td>- Python 3.5 and libraries such as NLTK, NumPy, SciPy, and Matplotlib</td>
<td>- Determine the phrase rank for a product by computing the value of <strong>PF-IRF</strong> measure for each phrase. Phrases with high PF-IRF ranks are selected and included in our summary.</td>
</tr>
<tr>
<td>- SQLite3</td>
<td>- <strong>Comparison algorithm</strong> to evaluate similarity between RAKE’s phrases and our algorithm’s phrases.</td>
</tr>
<tr>
<td><strong>Keywords Extraction:</strong></td>
<td><strong>Phrase Extraction Algorithm:</strong></td>
</tr>
<tr>
<td>- RAKE (Rapid Automatic Keyword Extraction) using NLTK</td>
<td>- Extract frequent n-grams from all reviews of a product.</td>
</tr>
<tr>
<td><strong>Evaluation Software:</strong></td>
<td>- Extract key phrases from the frequent n-grams, phrases that contain adjectives followed by nouns (and vice versa) with stopwords eliminated.</td>
</tr>
<tr>
<td>- ROUGE software package to automate the evaluation of our results.</td>
<td><strong>Sentence Generation Algorithm:</strong></td>
</tr>
<tr>
<td></td>
<td>- Uses sentence structure and a <strong>Markov chain</strong> to generate readable sentences containing descriptive phrases.</td>
</tr>
</tbody>
</table>
7. Milestones

• Provide a brief list of milestones. For example, since the project will span 6 weeks of the class (weeks 5 to 10), you could 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 end of week 7.
7. Milestones (Example)

Weeks 5 and 6:
- Search for additional data sets
- Write basic tokenizer and generate feature list using most common words.
- Test accuracy of scikit-learn Bernoulli naive Bayes and multinomial naive Bayes using the basic tokenizer. (If multinomial turns out to be more accurate, then we will implement multinomial instead).

Weeks 7 and 8:
- Write Bernoulli naive Bayes classifier, and test accuracy of basic algorithm
- Compare accuracy with scikit-learn Bernoulli naive Bayes and Multinomial naive Bayes.
- Improve tokenizer and feature list generation.
- Use PyEnchant to group misspelled words.

Weeks 9 and 10:
- Improve tokenizer and feature list generation.
- Explore tracking different metadata features.
8. Individual Student Responsibilities

Summarize briefly what each student will be primarily responsible for in the project. For example, you might write something like this

- **Name 1:** will write and test the code for Algorithms 1 and 2, will integrate components A and B in the pipeline, will assist in doing experiments and interpreting results, will assist in writing project reports

- **Name 2:** will acquire the data sets to test the algorithms, will preprocess the text data (e.g., define the vocabulary for the algorithms), will implement Algorithm 3 and integrate all the components into a pipeline, 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.....]
8. Individual Student Responsibilities (Example)

Student 1:

will write and test the code for our PF-IRF Ranking Algorithm, contribute the rest of his time generating grammatically correct and logical sentences for our summaries, and assist in writing project reports.

Student 2:

will acquire the relevant data from our dataset to feed into our algorithms, preprocess the text data, write the scripts for evaluating the accuracy of the algorithms, write and test the code for our Sentence Generator using ROUGE, and assist in writing project reports.
9. References & Links


Example of a Project Topic: Natural Language Generation
### Markov Chain Models for Text

#### Transition Probabilities
(bigram probabilities, can be learned from text)

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>big</th>
<th>dog</th>
<th>cat</th>
<th>ran</th>
<th>fell</th>
<th>at</th>
<th>wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.0</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>big</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
</tr>
<tr>
<td>dog</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>ran</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>fell</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>at</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>wall</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.5</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

Note that each row sums to 1
## Markov Chain Models for Text

<table>
<thead>
<tr>
<th>Initial Probabilities</th>
<th>the</th>
<th>big</th>
<th>dog</th>
<th>cat</th>
<th>ran</th>
<th>fell</th>
<th>at</th>
<th>wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.7</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>big</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>dog</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>cat</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>ran</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>fell</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>at</td>
<td>0.2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wall</td>
<td>0.1</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>
### Markov Chain Models for Text

<table>
<thead>
<tr>
<th>Initial Probabilities</th>
<th>the</th>
<th>big</th>
<th>dog</th>
<th>cat</th>
<th>ran</th>
<th>fell</th>
<th>at</th>
<th>wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.7</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>big</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>dog</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>cat</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>ran</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>fell</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>at</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wall</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>
SideNote: How do we simulate 1 of K items with K probabilities?

- Example:  
  - K = 6 and probability distribution is [0.1, 0.1, 0.4, 0.0, 0.3, 0.1]

- How can we sample from this?
  - Create the cumulative distribution: [0, 0.1, 0.2, 0.6, 0.6, 0.9, 1]
  - Sample a uniform random number between 0 and 1
    - If it falls between the CDF for item k and item k + 1, then item k is our sample
Simulated Examples

*The big dog at the wall ran at the wall at the dog*......

*The cat at the dog fell at the big wall at the cat*......

*At the wall at the dog ran at the big cat at the cat*......

What problems do we see with these sentences?
Simulated Examples

The big dog at the wall ran at the wall at the dog......

The cat at the dog fell at the big wall at the cat......

At the wall at the dog ran at the big cat at the cat......

What problems do we see with these sentences?

1. Sentences keep going: no punctuation
2. Sentences are not necessarily grammatically correct
3. Lack of semantic coherence: sentences don’t always make sense
Adding Punctuation

- **Simple approach:**
  - We can add punctuation to our vocabulary, e.g., add period, comma, semicolon, etc, and learn Markov transition probabilities from data
  - e.g., in English a sentence will often end with a noun or adjective

- **Potential issues:**
  - Markov approach doesn’t give us direct control of sentence length, e.g., for song lyrics or poetry this might be important
  - Another issue is that sentences could be unrealistically short or long
  - **Alternative approach:**
    - Estimate the distribution of sentence lengths in the corpus and sample a “sentence length” for each new sentence that is simulated
Adding Grammar

- One approach
  - Parse all the sentences in the corpus and keep templates, e.g.,
    - S1: article -> adjective -> noun -> verb ->
    - S2: proper noun -> verb -> article -> adjective -> noun
    - Etc
  - To simulate a new sentence, select sentence template at random from S1, S2, ..
    - Let POS_1, POS_2, POS_3,..... POS_K be the parts of speech in the sentence
  - Given POS_1, randomly pick a word from the initial distribution, restricting to that part of speech
  - For every pair of words, POS_k/POS_k+1, select next word randomly (using bigrams/Markov transition matrix) restricted to POS_k+1

- Will generally be grammatically correct...but maybe not enough variety
**Example:**

<table>
<thead>
<tr>
<th>POS_2 in the sample sentence is a noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First word</th>
<th>POS_2 in the sample sentence is a noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>the</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Better Models

• Instead of bigrams (word->word transitions) could also try to use higher-order ngrams: trigrams, 4grams, etc
  – Can generalize to \( p(\text{next word} \mid \text{previous words}) \)
  – Potential problem: can end up repeating large parts of the original text
  – Might be more straightforward to just include n-grams in the vocabulary
    ....... but this might suffer from sparsity in the training data (few transitions)
  – (See also context-free grammars, next 2 slides)

• Recurrent neural networks (RNNs) or LSTMs
  – Potentially much more powerful than ngrams
  – Can represent history of a sequence with embedded state, rather than explicitly via ngram counts
  – If there’s enough data, RNNS typically outperform ngrams
    • But require a lot of data, a lot of computation time, a lot of experimentation...
  – ....and simulated sentences still might lack semantic coherence
    • e.g., see simulated sequences in Andrej Karpathy’s blog
SC igen - An Automatic CS Paper Generator

About
SC igen is a program that generates random Computer Science research papers, including graphs, figures, and citations. It uses a hand-written context-free grammar to form all elements of the papers. Our aim here is to maximize amusement, rather than coherence.

One useful purpose for such a program is to auto-generate submissions to conferences that you suspect might have very low submission standards. A prime example, which you may recognize from spam in your inbox, is SCI/IIS and its dozens of co-located conferences (check out the very broad conference description on the WMSCI 2005 website). There’s also a list of known bogus conferences. Using SC igen to generate submissions for conferences like this gives us pleasure to no end. In fact, one of our papers was accepted to SCI 2005! See Examples for more details.

We went to WMSCI 2005. Check out the talks and video. You can find more details in our blog.

Also, check out our 10th anniversary celebration project: SCipher!

Generate a Random Paper
Want to generate a random CS paper of your own? Type in some options and click "Generate".

Author 1:
Author 2:
Author 3:
Author 4:
Author 5:

Generate Reset

SC igen currently supports Latin-1 characters, but not the full Unicode character set.

MIT graduate student project from 2005
Generates papers based on hand-generated context-free grammar
Generated papers that were accepted and published by “conferences of ill-repute”

From: https://pdos.csail.mit.edu/archive/scigen/#code
Towards the Simulation of E-Commerce

Herbert Schlangemann

ABSTRACT

Recent advances in cooperative technology and classical communication are based entirely on the assumption that the Internet and active networks are not in conflict with object-oriented languages. In fact, few information theorists would disagree with the visualization of DHTs that made refining and possibly simulating 8 bit architectures a reality, which embodies the compelling principles of electrical engineering [19]. In this work we better understand how digital-to-analog converters can be applied to the development of e-commerce.

I. INTRODUCTION

The synthesis of fiber-optic cables is a natural quagmire. While such a hypothesis is entirely a theoretical ambition, it rarely conflicts with the need to provide operating systems to computational biologists. Similarly, for example, many methodologies measure vacuum tubes. The notion that hackers worldwide interfere with context-free grammar is largely bad. The synthesis of checksums would tremendously improve mobile information.

From: https://en.wikipedia.org/wiki/SCIgen
“The Malfoys!” said Hermione.

Harry was watching him. He looked like Madame Maxime. When she strode up the wrong staircase to visit himself.

“I’m afraid I’ve definitely been suspended from power, no chance—indeed?” said Snape. He put his head back behind them and read groups as they crossed a corner and fluttered down onto their ink lamp, and picked up his spoon. The doorbell rang. It was a lot cleaner down in London.

Evaluation of Text Simulators

• This is a challenge.....there is no “ground truth” to compare to

• One option: User studies
  – Probably better to compare output of Algorithm A with Algorithm B rather than output of A and human-generated text

• General approach
  – Have A and B each generate K pieces of text from the same initial conditions, e.g., given the first 2 or 3 words of an actual sentence
  – Generate at least 10 or 20 such sentences from A and from B
  – Place each pair side by side on the screen (or page), with left and right position randomly selected between A and B
  – For each pair ask the user to select “which text is more human-like, A or B?”
    – Repeat with at least 5 to 10 different users
    – Do a statistical analysis of the results: is there a statistical difference?
Announcements

• Project Proposals
  – Due Friday 6pm
  – Detailed information on the class Website
  – If you are not on a team: please see me at the end of class
  – If you are a team looking for another member: please also see me

• Office Hours:
  – Eric: Thursday 1 to 3
  – Me: Friday 9:30 to 11:30