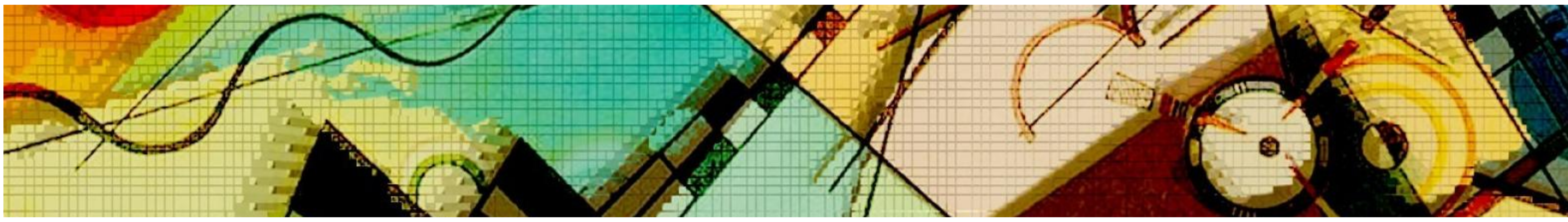


CSC411 Project 3: Supervised and Unsupervised Learning for Sentiment Analysis

For this project, you will build and analyze several algorithms for sentiment analysis. Specifically, you will try to automatically figure out whether a movie review is positive or negative.

The input: movie reviews

You will work with the Cornell Movie Reviews dataset. You will want to process it to convert all the words to lowercase, and to get rid of punctuation. Download the polarity dataset , and randomly split it into a test, a validation, and a training set. (Use e.g. 200 movies for testing, and 200 movies for validation.)

Part 1 (2 pts)

Describe the datasets. You will be predicting whether the review is positive or negative from keywords that appear in the review. Is that feasible? Give 3 examples of specific keywords that may be useful, together with statistics on how often they appear in positive and negative reviews.

Part 2 (15 pts)

Implement the Naive Bayes algorithm for predicting whether the review is positive or negative. Tune the parameter m using the validation set, and report how you did it and what was the result. Report the performance on the training and the test sets that you obtain. Note that compting products of many small numbers leads to underflow. Use the fact that $a_ 1 a_ 2 ... a_ n = \exp(\log a_1 + \log a_2 + ... \log a_n$). In your report, explain how you used that fact.

Part 3 (3 pts)

List the 10 words that most strongly predict that the review is positive, and the 10 words that most strongly predict that the review is negative. State how you obtained those in terms of the the conditional probabilities used in the Naive Bayes algorithm.

Part 4 (15 pts)

Train a Logistic Regression model on the same dataset. For a single movie review, For a single review r the input to the Logistic Regression model will be a k -dimensional vector v , where $v[k]=1$ if the k -th keyword appears in the review r . The set of keywords consists of all the words that appear in all the reviews.

Plot the learning curves (performance vs. iteration) of the Logistic Regression model. Describe how you selected the regularization parameter (and describe the experiments you used to select it).

Part 5 (5 pts)

At test time, both Logistic Regression and Naive Bayes can be formulated as computing

$$\theta_0 + \theta_1 I_1(x) + \theta_2 I_2(x) + ... + \theta_k I_k(x) > \text{thr}$$

in order to decide whether to classify x as 1 or 0 . Explain, in each case, what the θ ‘s and the I ‘s are.

Part 6 (5 pts)

Compare the top 100 θ s that you obtained using Naive Bayes and Logistic Regression.

Part 7 (15 pts)

In this part, you will investigate the use of features obtained using unsupervised learning in order to compute features that are useful for sentiment analysis (though actually using the unsupervised learning for the full task is not part of this project).

The method we will use to compute interesting features is called word2vec. It allows us to map each word in our vocabulary to a k-dimensional vector, in a way that’s more interesting than one-hot encoding. Since word2vec uses unsupervised learning, it does not make use of the fact that we know that some reviews are positive and some reviews are negative.

How word2vec works

Take the sentence “I have never been to Alpha Centauri.” For each word, let’s consider one left and one right word as the context. We could map all the possible contexts to all the possible words. The ([context], target) pairs are thus ([I, never], have), ([have, been], never), ([never, to], been), ([been, Alpha], to), ([to, Centauri], Alpha).

We would like to predict the context from the word. The (input, output) pairs are thus (I, have), (never, have), (have, never), (been, never), (never, been), (never, to), (been, to), (Alpha, to), (to, Alpha), (Centauri, Alpha).

The goal is to compute feature vector $word2vec(w)$ for every word w such that it’s possible to learn a Logistic Regression model that that is able to classify $(word2vec(w), word2vec(t))$ as 1 if t appears near w , and as 0 otherwise.

It's possible to achieve this goal without ever labelling the texts from which the words come.

Task: testing (and getting an idea of how to train) word2vec

We are supplying you with word2vec embeddings for a list of word. Show that the word2vec embeddings work for figuring out whether t appears together with w or not using Logistic Regression. Describe the experiment you performed and its results.

The word2vec embeddings for the ~40k words in the reviews are available in embeddings.npz using

```
load("embeddings.npz")["emb"]
```

The word indices are available using

```
load("embeddings.npz")["word2ind"].flatten()[0]
```

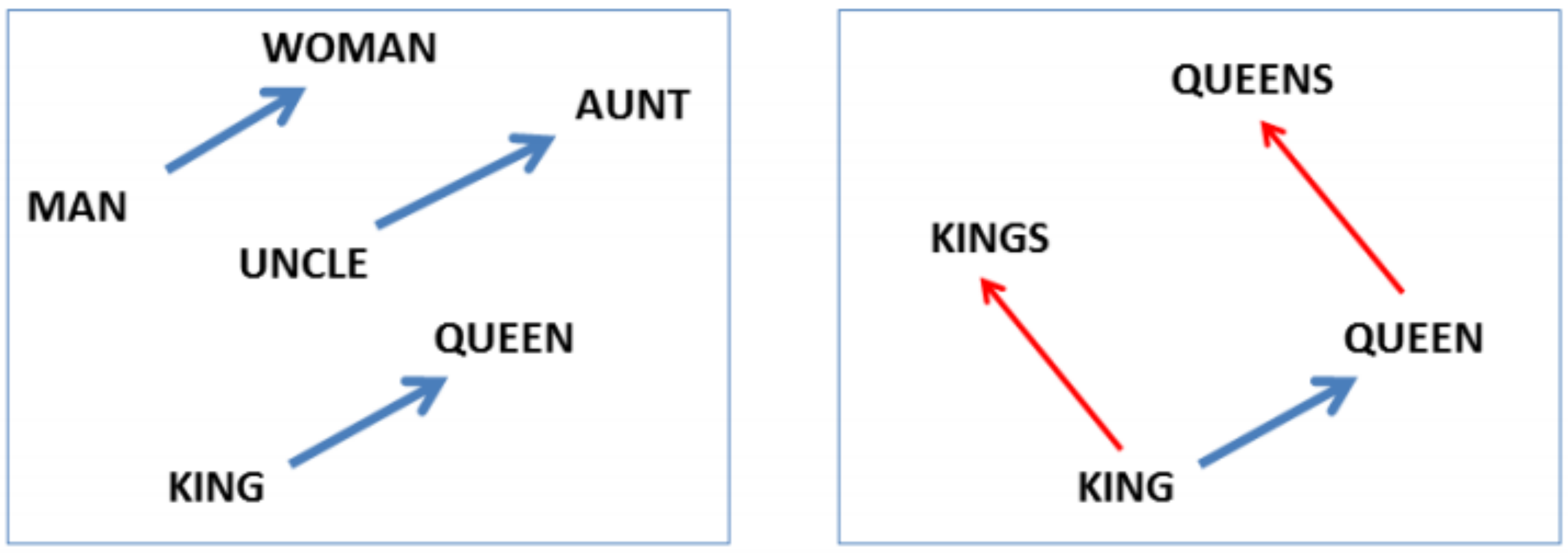
The experiment you are running in this Part is the first step to actually obtaining word2vec embeddings from the training set. In order to obtain them, you would optimize the parameters of the logistic regression and the parameters of the embeddings. This is not part of this project.

Part 8 (10)

The reason word2vec works is that words that appear in similar contexts have similar embeddings (in the sense that the Cosine or Euclidean distance between the embeddings is small). Investigate this by finding the 10 words whose embeddings are closest to the embedding of "story", and the the 10 words whose embeddings are closest to the embedding of "good." Find two more interesting examples like that that demonstrate that word2vec works.

word2vec embeddings trained on large datasets show all sorts of cool properties. For example, sometimes we see something like

$$word2vec(queen) = word2vec(king) + word2vec(man) - word2vec(woman)$$



(Mikolov et al., NAACL HLT, 2013)

Part 9 (Bonus 10 pts)

Generate a dataset on which Logistic Regression works a significantly better than Naive Bayes. Explain how you accomplished this, and include the code you used to generate the dataset, as well as your experimental results, in your report.

What to submit

The project should be implemented using Python 2 or 3, using TensorFlow. Your report should be in PDF format. You should use LaTeX to generate the report, and submit the .tex file as well. A sample template is on the course website. You will submit at least the following files: `naivebayes.py`, `logistic.py`, `compare_nb_log.py`, and `word2vec.py`, as well as the write-up. MORE TO COME. You may submit more files as well.

Reproducibility counts! We should be able to obtain all the graphs and figures in your report by running your code. The only exception is that you may pre-download the images (what and how you did that, including the code you used to download the images, should be included in your submission.) Submissions that are not reproducible will not receive full marks. If your graphs/reported numbers cannot be reproduced by running the code, you may be docked up to 20%. (Of course, if the code is simply incomplete, you may lose even more.) Suggestion: if you are using randomness anywhere, use `numpy.random.seed()`.

You must use LaTeX to generate the report. LaTeX is the tool used to generate virtually all technical reports and research papers in machine learning, and students report that after they get used to writing reports in LaTeX, they start using LaTeX for all their course reports. In addition, using LaTeX facilitates the production of reproducible results.

Using my code

You are free to use any of the code available from the CSC411 course website.

Readability

Readability counts! If your code isn't readable or your report doesn't make sense, they are not that useful. In addition, the TA can't read them. You will lose marks for those things.

Academic integrity

It is perfectly fine to discuss general ideas with other people, if you acknowledge ideas in your report that are not your own. However, you must not look at other people's code, or show your code to other people, and you must not look at other people's reports and derivations, or show your report and derivations to other people. All of those things are academic offences.