Instructions for L90 Practical: Sentiment Detection of Reviews

Simone Teufel (Lead demonstrator Paula Czarnowska)
`sht25@cl.cam.ac.uk; pjc211@cl.cam.ac.uk`

Michaelmas 2018/19

This practical concerns sentiment classification of movie reviews. You will use two different machine learning approaches based on bag-of-word features. For this task, you are not allowed to use any other packages than those described below. Your second task is to improve over the two baseline systems using document embeddings and perform an error analysis on the strengths and weaknesses of the new approach. You can use whichever machine you want for development, but your final system(s) must run on the MPhil machines (on your own personal VMs provided for you), and you must include a pointer to your working code on the Mphil machines (your account).

You will find 1000 positive and 1000 negative movie reviews in `/usr/groups/mphil/L90/data/{POS,NEG}/*.txt`. You will write code that decides whether a random unseen movie review is positive or negative, and two reports in the form of a scientific article that describe the results you achieved in the two tasks. To prepare yourself for this practical, you should have a look at a few of these reviews to understand the difficulties of the task, and think about how one might go about classifying them.

In particular, you will reimplement key aspects of the following paper:

> Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan (2002). *Thumbs up? Sentiment Classification using Machine Learning Techniques.* Proceedings of EMNLP.

Bo Pang et al. were the "inventors" of the movie review sentiment classification task, and the above paper was one of the first papers on the topic.

Advice: Please read through the entire instruction sheet and familiarise yourself with all requirements before you start coding or otherwise solving the tasks. Writing clean, modular code can make the difference between solving the assignment in a matter of hours, and taking days to run all experiments.

A quick note on installing packages in the MPhil machines

You can install packages by downloading the .tar file in your *home folder* and then installing the packages from there (while setting your path variables as needed). An alternative would be to do the following:

1. Go to `https://pip.pypa.io/en/stable/installing/` and download `get-pip.py`
2. Run `python get-pip.py --user`
3. Then to install a package (e.g., `scipy`) run `python -m pip install --user scipy`

Part One: Naive Bayes Baseline

How could one automatically classify movie reviews according to their sentiment? Your task in Part One is to establish two commonly used baselines.

Machine Learning using Bags of Words representations

You will implement a Machine Learning approach that operates on a simple Bag-of-Words (BoW) representation of the text data, as described in Pang et al. (2002). In this approach, the only features we will consider are the words in the text themselves (or short sequences of these words), without any other sources of information. The BoW model is a popular way of representing text information as vectors (or points in space), making it easy to apply classical Machine Learning algorithms on NLP tasks. However, the BoW representation is also very crude, since it discards all information related to context (other than ngram information).

Write your own code to implement the Naive Bayes (NB) classifier.[1] As a reminder, the Naive Bayes classifier works according to the following equation:

$$\hat{c} = \arg\max_{c \in C} P(c|\bar{f}) = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(f_i|c)$$

where $C = \{\text{POS}, \text{NEG}\}$ is the set of possible classes, $\hat{c} \in C$ is the most probable class, and $\bar{f}$ is the feature vector. Remember that we use the log of these probabilities when making a prediction:

$$\hat{c} = \arg\max_{c \in C} \{logP(c) + \sum_{i=1}^{n} logP(f_i|c)\}$$

You can find more details about Naive Bayes here:

> https://web.stanford.edu/∼jurafsky/slp3/6.pdf

and pseudocode here:

> https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

As features for classification, please use unigrams and bigrams (separately and jointly), as described in Pang et al. (2002). Reimplement their frequency cutoff (infrequent unigrams and bigrams are discarded).

You may use whichever programming language you prefer (C++, Python, or Java being the most popular ones), but you must write the Naive Bayes training and prediction code from scratch. You will not be given credit for using off-the-shelf Machine Learning libraries such as **mlpack** (C++), **scikit** (Python), **Weka** (Java), etc.

The data in `/usr/groups/mphil/L90/data-tagged/{POS,NEG}/*.tag` contains the text of the reviews, where each document is one review. You will find the text has already been tokenised for you. Your algorithm should read in the text, store the words and their frequencies in an appropriate data structure that allows for easy computation of the probabilities used in the Naive Bayes algorithm, and then make predictions for new instances.

> Train your classifier on files cv000–cv899 from both the `/POS` and the `/NEG` directories, and test it on the remaining files cv900–cv999. Report results using simple classification accuracy as your evaluation metric[2].

Smoothing

> The presence of words in the test dataset that haven't been seen during training can cause probabilities in the Naive Bayes classifier to be 0, thus making that particular test instance undecidable. The standard way to mitigate this effect (as well as to

[1]This section and the next aim to put you a position to replicate Pang et al., Naive Bayes results. However, the numerical results will differ from theirs, as they used different data.

[2]**Voluntary thought experiment**: Is accuracy an equally appropriate evaluation metric in a situation where 90% of your data instances are of positive movie reviews? If you are curious about this, you can simulate this scenario by keeping the positive reviews data unchanged, but only using negative reviews cv000–cv089 for training, and cv900–cv909 for testing. Calculate the classification accuracy, and explain what changed. (This is a voluntary exercise; if you decide to do it, please don't report it)

give more clout to rare words) is to use smoothing, in which the probability fraction $\frac{count(w_i,c)}{\sum\limits_{w \in V} count(w,c)}$ for a word w_i becomes $\frac{count(w_i,c)+smoothing(w_i)}{\sum\limits_{w \in V} count(w,c)+\sum\limits_{w \in V} smoothing(w)}$

Implement Laplace feature smoothing ($smoothing(\cdot) = \kappa$, constant for all words) in your Naive Bayes classifier's code, and report the impact on performance.

Statistical significance testing

When comparing two versions of a system, significance testing is used to determine whether their difference in performance (with respect to a particular performance metric) is statistically significant.[3]

What would it mean for a difference not to be statistically significant? The sample you train and test on are always just a tiny sample of an infinite set of all possible instances. Thus, it is possible that observed differences in the reported performance are really just noise (random), despite *looking* as if there was a real difference.

We perform a *paired difference test*, as the two systems can be run on the identical data, which creates *paired samples* – the score obtained by one system for a particular data item is paired to that of the other system for the same item. *Paired difference test* are designed to assess whether the population mean (in terms of performance) of the two runs is different.

Formally, a paired difference test is carried out in the form of a two-tailed paired hypothesis test with two disjoint hypotheses:

H_0: $\mu_1 = \mu_2$ or equivalently $\mu_1 - \mu_2 = 0$
there is no difference in statistic μ between the two samples
H_1: $\mu_1 \neq \mu_2$ or equivalently $\mu_1 - \mu_2 \neq 0$
there is a difference in statistic μ between the two samples

There are two outcomes of a statistical test, Reject H_0 (difference found) and Do Not Reject H_0 (no difference found).

	H_0 **True**	H_1 **True**
Reject H_0	Type I Error	Correct (Difference)
Do Not Reject H_0	Correct (No Difference)	Type II Error

One of the simplest statistical tests is the **sign test**. The sign test is described in Siegel and Castellan (1986)[4], page 80 (scans of the relevant pages are available in the L90 directory `/usr/groups/mphil/L90/resources/`). As presented in the slides, the sign test is based on the binomial distribution.

You will implement this test and apply it to find out whether the difference between smoothed and unsmoothed NB systems is significant. (Each change that you apply can be thought of as a new "system"). In short, you will count all cases when System A is better than System B, when System B is better than System A, and when they are the same. Call these numbers $Plus$, $Minus$ and $Null$ respectively. The sign test returns the probability that the null hypothesis is true. This probability is called the p-value and it can be calculated for the two-sided sign test using the following formula (we multiply by two because this is a two-sided sign test and tests for the significance of differences in either direction):

$$2 \sum_{i=0}^{k} \binom{N}{i} q^i (1-q)^{N-i}$$

where $N = 2\lceil \frac{Null}{2} \rceil + Plus + Minus$ is the total number of cases, and $k = \lceil \frac{Null}{2} \rceil + \min\{Plus, Minus\}$ is the number of cases with the less common sign. In this experiment, $q = 0.5$. Here, we treat

[3]You should always attempt to perform a significance test with any results. If it's not possible due to the properties of data or metric, then you should explain in the text why it's not possible. If you cannot find significance a the chosen level, you can also report "marginal" significance (typically if $0.5 < p < 0.6$).

[4]Siegel and Castellan, Nonparametric Statistics for the behavioural sciences, McGraw-Hill.

ties by adding half a point to either side, rounding up to the nearest integer if necessary. You can quickly verify the correctness of your sign test code using a free online tool.[5].

From now on, report all differences between systems using the sign test. Be careful with your exact language when reporting any result in your report. Using words such as "better" or "outperform" when discussing the differences between two metrics implies that you have run a test and found the difference to be significant. Writing "better" and not even having **tested** for significance is a typical rookie mistake.[6] If significance was not established, you are strictly speaking even making a false (or at least unsupported) claim, as results that might be due to chance are not in fact "better". The strongest statement you can make in this situation is "despite not significantly different, the results of system A are at least numerically higher than those of system B" (but as nobody cares about arbitrary results that could just as easily have been created by chance, you may as well not bother).

If you are comparing more than two different methods (i.e., systems), tests have to be performed in a pair-wise manner. This creates a triangular matrix of test results in the general case. To avoid redundancy, try to see if you can bundle and summarise trends. In text, it's enough to mention the keyword "significantly better" or "statistically significantly better" the first time you make a direct comparison of results. Ideally, you should **only** talk about significant difference and ignore all others. At the first point of using "significant", give details, maybe in a footnote (which test you use, at which significance level, and whether two- or one-tailed). From then onward in the paper, your reader will understand that you know what's going on and play by the rules, and you don't need to mention significance again each time. After having written a section on results, make it a habit to check each "better" (because it often sneaks in). If you cannot say "significantly better" for some legitimate reason (e.g., because no known test exists for a metric you use), you can use the vague term "considerably better", which doesn't have a technical meaning like "significantly better" does.

Cross-validation

A serious danger in using Machine Learning on small datasets, with many iterations of slightly different versions of the algorithms, is that we end up with Type III errors, also called the "testing hypotheses suggested by the data" errors. This type of error occurs when we make repeated improvements to our classifiers by playing with features and their processing, but we don't get a fresh, never-before seen test dataset every time. Thus, we risk developing a classifier that's better and better on our data, but worse and worse at generalizing to new, never-before seen data.

A simple method to guard against Type III errors is to use cross-validation. In N-fold cross-validation, we divide the data into N distinct chunks/folds. Then, we repeat the experiment N times, each time holding out one of the chunks for testing, training our classifier on the remaining N - 1 data chunks, and reporting performance on the held-out chunk. We can use different strategies for dividing the data:

- Consecutive splitting:

  cv000–cv099 = Split 1
  cv100–cv199 = Split 2
  ...

- Round-robin splitting (mod 10):

  cv000, cv010, cv020,... = Split 1
  cv001, cv011, cv021,... = Split 2
  ...

- Random sampling/splitting: Not used here (but you may choose to split this way in a non-educational situation)

[5]For example `https://www.graphpad.com/quickcalcs/binomial1.cfm`

[6]An even more embarrassing mistake is to write "significant better" and not know that this is a technical term with a keyword function. In day-to-day language use, "significant" might be used equivalently to "noticeable" or "a lot". This is not how the word is used in science.

Write the code to implement 10-fold cross-validation for your Naive Bayes classifier from and compute the 10 accuracies. Report the final performance, which is the average of the performances per fold.

If all splits perform equally well, this is a good sign.

Write code to calculate and report variance, in addition to the final performance.

Please report all future results using 10-fold Round-Robin cross-validation now (unless told to use the held-out test set).

How can you use statistical significance testing on cross-validated data?

Write Report for Part One

Up to 1,000 words, excluding references. Due on **Friday 16 November 2017 at 12:00 noon** (Note: mention of word limit always means that you should state the actual number of words you used. Not doing so will incur loss of some points in the marking scheme). Style your report as a paper that describes replication of (part of) Pang et al. 2002.