Instructions for NLP Practical (Units of Assessment) SVM-based Sentiment Detection of Reviews (Part 1)

Simone Teufel (Lead demonstrator Guy Aglionby)
`sht25@cl.cam.ac.uk; ga384@cl.cam.ac.uk`

This practical expands on MLRD's simple Naive Bayes BoW-based sentiment classification of movie reviews (but in case you haven't been through the MLRD experience, it's also a stand-alone practical).

In Part 1, you will upgrade machine learning algorithm to Support Vector Machines, a methodology which outperforms NB in many NLP tasks. The features you use will be unigrams and bigrams. Together with an NB system, you now have the key components that enable you to reimplement the famous paper by Pang et al (2002), which you have already come across in MLRD:

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

Please re-read this paper. You will perform a reimplementation of this experiment, and these types of systems will act as your baselines for the experiments in Part 2. A word about reimplementations:

- In general, you should attempt to mimic as closely as possible what the original authors have done. The point of a reimplementation is comparability.
- During reimplementation of somebody's work only from a paper, it's often necessary to interpret some vague statement. Information might be omitted that you would have needed. This is a normal state of affairs, as it is impossible to write every detail down. However, if you feel that something crucial is missing, you should state your interpretation/assumptions so that your readers will know what you have done.
- Reimplementation can also include some experimentation beyond what the original authors' work. Pang et al. (2002) did things in a certain way, and you should repeat that as closely as possible. But if you find a slight variation that works even better, that is certainly worthy of mention (although not necessary).

You will find the well-known 1000 positive and 1000 negative movie reviews on the course's website (under "course materials"), split into `POS` and `NEG` directories. There are a carefully disambiguated version of the data in a Pang follow-on paper[1]. Meta-data such as the film's name, director and author list have been removed. Another clean-up that has been done semi-automatically is that any mention in the text of the number of stars given has been removed. For instance, "The reason I gave this a five-star rating was that I really liked Pitt as the villain" got turned into "I really liked Pitt as the villain". (What effect does this manipulation have on the task?) Note that the 700+700 used in the 2002 Pang et al. paper are a subset of the 1000+1000 from the 2004 paper.

You can use your own machine or Lab machines for development, but your final system(s) must run on the lab machines; you must include a pointer to your working code on the lab

[1] Pang, Bo, and Lillian Lee. "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts." Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2004.

machines (your account). You may use a programming language of your choice (but note that python is the natural choice for NLP applications).

General 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.

1 NB; Bag of Words/ngram representation

As you might recall from MLRD, simple Bag-of-Words (BoW) representation of the text data use a large set of words in the text themselves. You will use these types of features here again (think about whether you should use a frequency threshold and how you would set it[2]), and you will add bigrams (reminder: you collected bigrams of genes for the HMM task of MLRD when compiling stats for the transition frequencies). The BoW model is a popular but crude way of representing text information as vectors (or points in space), making it easy to apply classical Machine Learning algorithms on NLP tasks. (You will replace these representations with document embeddings later, but reuse the SVM classification framework.)

First you need an Naive Bayes (NB) classifier. 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:

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

The reviews are already been tokenised for you. Your algorithm should prepare the data into the format required by your NB classifier. You may use your own previous code from MLRD or any package on the web. Make sure the NB version you use uses at least Laplace smoothing. Please describe clearly which implementation you use. If it's an existing package, cite for initial introduction of the algorithm, and for particular implementation (if different). Prefer well-known implementations (check this, e.g., on Google Scholar).

2 Support Vector Machines; Bag of Words/ngram representation

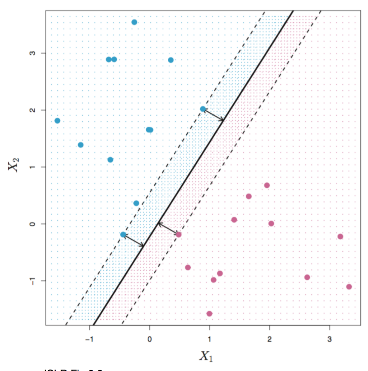
SVM is a generalisation of simple maximal margin classifier and support vector classifier, both of which require that classes are separable by linear boundary. [3] Support Vector Machines are the extension of support vector classifier to non-linear boundaries, but they support only two classes (extensions to multi-class is possible).

A hyperplane in p-dimensions is a flat $p-1$-dimensional affine subspace. In 2 dimensions a hyperplane is a line; in 3 dimensions, a hyperplane is a plane. We can now compute the distance between data points and various hyperplane, out of which we select the one that creates the largest margin (best separation) between the two classes. Support vectors are data points lying on the margin. Classification of a test point depends on which side of this hyperplane it falls

[2]Strictly speaking, **experimenting** with feature cutoffs (i.e., doing a systematic search) is methodologically questionable, as you don't have a separate validation corpus. There is a danger of overtraining. The only thing you are (kind of) allowed to do is to choose a feature cutoff once (e.g. 2 or 3 or 4) before the experiment and then run the experiment only once with this feature cutoff. If you compare that to the full feature set (no cutoff), most people would probably judge that was still OK. But we are moving into a grey area.

[3]Image taken from ISLR = "An Introduction to Statistical Learning with Applications in R" by G. James et. al.

on. The points closest to the margin are the most difficult to classify. The size of the margin is standardly interpreted as the SVM's confidence in the classification.



source: ISLR Fig 9.3

We recommend SVM Light as implementation for the practical.[4]

3 Lemmatisation/Stemming

Pang et al (2002) don't use stemming on the word tokens, but there are good reasons why stemming should improve results. Please use the **Porter Stemmer**, versions of which you can find on the web in any language. For information on stemming, refer to lecture 2 (morphology). What effect could stemming potentially have?

4 Cross-validation

There are different possible strategies for dividing the data:

- Consecutive splitting:

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

- Round-robin splitting (mod N, here: 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)

Follow Pang et al. in their use of crossvalidation, and use Round-Robin splitting (on the first part of the file names) for comparability. It is assumed that you will use **stratified** cross-validation.

5 Statistical significance testing

This entire section is a reminder (if you have been through MLRD). **Executive summary: Use the sign test to report significance.**

[4]Described in detail on `http://svmlight.joachims.org/`

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.[5]

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)[6], page 80 (scans of the relevant pages are available on the NLP materials website). The sign test is based on the binomial distribution.

How to do it: 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 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.[7].

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.[8] If significance was not established, you are strictly speaking even making a false (or at least unsupported) claim, as results that

[5]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 $0.5 < p < 0.6$).

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

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

[8]An even more embarrassing mistake is to write "significantly 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.

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.

6 Mini-report

Describe the results of your replication of Pang et al. in a mini-report, which is due for submission on **14 November, 4pm**[9]. The mini-report for Part 1 is only 300 words[10] long (ticked). That means you will need to compress a lot of material into few words, hopefully resulting in a compact, precise writing style – incidentally not too unlike from what you need for a Part 2 dissertation. More specific writing advice will be given in the form of slides.

The report must also contain a link to working code on the lab machines so that we can check in case of doubt. **Important**: it is the report that is assessed, not your code. If you don't describe clearly what you have done in the report, don't expect the assessor to go into your code fishing for explanations. The code is not routinely read.

7 Note

In the accompanying slides, I mention a few times that you should "replicate Pang as closely as possible". This is not to be misread as an instruction "replicate the entire Pang et al. paper", but as: "replicate Pang as closely as possible, **in those aspects we ask you to do**". These aspects are listed as sections in these instructions. Pang et al. do at the same time more and less than you are being asked to do here.

[9]Early submission by Nov 9 will result in feedback from me by Nov 21.

[10]Note: mention of word limit implies that you should state the actual number of words you actually used.