L90 Practical: Sentiment Detection of Reviews

Simone Teufel

November 14, 2014

This practical concerns sentiment classification of movie reviews using a machine learning approach based on bag-of-word features, using a sentiment lexicon, a POS-tagger and a syntactic parser. You must use the MPhil machines for this task. Do not use any other packages than those described below.

You will find 1000 positive and 1000 negative movie reviews in `/usr/groups/mphil/L90/data/{pos,neg}`. To prepare yourself for this practical, you should have a look at a few of these texts; you will write a program that decides whether a random unseen movie review is positive or negative, and an assessed report in the form of a scientific article that describes the results you achieved.

1 Part One: Baseline and Essentials

How could one automatically classify movie reviews according to their sentiment? Your task in Part One is to establish two commonly used baselines – by implementing and evaluating several NLP methods on this task. Part One will be mainly covered in the session on 14/11.

1.1 Sentiment lexicon

If a sentiment lexicon exists, machine learning is not necessary. One might simply look up open-class words in the lexicon, and count how many positive or negative words there are in a lexicon. If the sentiment lexicon also has information about the magnitude of sentiment, we could add up all sentiment scores and decide the polarity of the movie review using the sign of the score.

Your first task is to implement this approach, using the sentiment lexicon in `/usr/groups/mphil/l90/resources/sent\_lexicon`. That lexicon records the magnitude of sentiment, so you can implement both solutions (please use a switch in your program).

1.2 Statistical test

Does the magnitude improve results? Statistically significantly? Apply the sign test on each item you consider (each document) to check. From now on, report all differences between systems using the sign test. Find out how statistical test results are reported, and use that format in your final report. The sign test is described in Siegel and Castellan (1986), page . Scans of relevant pages in L90 directory. An implementation of the sign test in perl is also in the L90 directory.

1.3 Machine-learning of bags of words/ngram/POS

Your second task is to contrast the symbolic approach (one approach, with two versions, magnitude vs non-magnitude) with a simple bag-of-words machine learning approach, as in Pang et al. (2002), using only the words as features. You will use the WEKA machine learning system for this. (Instructions on separate sheet).

The data in `/usr/groups/mphil/L90/data` is already tokenized and sentence segmented. Transform each document into a vector of features (words and n-grams) and feed it to WEKA.

Replicate Pang et al.'s bag-of-word results as closely as you can, on unigrams, bigrams and 3grams, with different machine learning algorithms. The numerical results will differ from Pang et al.'s, as they used different data (700+700 instead of your 1000+1000). Report results by cross-validation. (Explain in your paper what that is.) As the data are balanced, we can use simple classification accuracy as our evaluation metric.

Now install the Stanford parser (instructions on separate sheet). It offers POS-tagging as a pre-step to parsing. Use its POS tagger on your texts. Replicate what Pang et al. were doing:

- Replace your features by word+POS features (why could this help?)
- Add word+POS features to your word-only features
- For each POS, test whether training ONLY on features with that POS improves results.

As always, record differences and whether they are statistically significant.

1.4 Sentiment Lexicon information as features

How could you use the information from the lexicon with the bag-of-word/ngram features? Implement and experiment; report differences to your best system.

1.5 Negation

Think of a way to incorporate a simple treatment of negation in your system. Report differences to your best system.

2 Part Two: extension

Part Two will be mainly treated in the second Practical Class on 28/11. Now your task is to improve your sentiment recogniser with a treatment of your choice. You might perform an error analysis to decide which phenomenon you want to address. Here are suggestions:

- Use dependencies from the Stanford parser (how? why would it help?)
- Use lexical information from WordNet (which information? how could it help?)
- Perform anaphora resolution (why would this help?)
- Try to model discourse effects in reviews

3 The report

Write up your results in 4000 words, using the ACL formatting styles found at `acl2014.org/CallforPapers.hml` (about $\frac{2}{3}$ down the page). This should compile out to about 8 pages. Use scientific language. Do not assume the reader knows anything about the task or the data. Do not describe the work you did as an assignment; pretend that you did this work in a self-driven way. You can refer to data that we gave you as "given", or you can simply say "we use the sentiment lexicon from [Citation]". Write in the 1st person plural. Define the technical terms you use. Write an abstract that makes clear what you have achieved. Follow the Introduction–Methods–Results–Conclusion model. You can leave out the related work section, unless you had the time to find other work to relate to yours (not required). Concentrate on getting methods and results sections right. Under methods, describe the machine learning algorithms used, using formulae. (Therefore, if you possibly can, it is much better to compose the report in Latex.) Cite relevant papers using the Harvard citation style. Use Bibtex if possible. (Obviously, read the papers before citing them – never cite anything you haven't read.) Report numerical results in tables. Use statistical

significance tests. For conclusions, summarise what you have done and think of future work one could do based on your work, and describe it.

Read other scientific papers and try to mimic their style.

- Maximum 4000 words incl. references
- Submission: January 14, 2015 12noon