Naive Bayes versus BERT: Jupyter notebook assignments for an introductory NLP course

Jennifer Foster and Joachim Wagner
School of Computing
Dublin City University
jfoster|jwagner@dcu.ie

Abstract

We describe two Jupyter notebooks that form the basis of two assignments in an introductory Natural Language Processing (NLP) module taught to final year undergraduate students at Dublin City University. The notebooks show the students how to train a bag-of-words polarity classifier using multinomial Naive Bayes, and how to fine-tune a polarity classifier using BERT. The students take the code as a starting point for their own experiments.

1 Introduction

We describe two Jupyter[1] notebooks that form the basis of two assignments in a new introductory Natural Language Processing (NLP) module taught to final year students on the B.Sc. in Data Science programme at Dublin City University. As part of a prior module on this programme, the students have some experience with the NLP problem of quality estimation for machine translation. They have also studied machine learning and are competent Python programmers. Since this is the first Data Science cohort, there are only seven students. Four graduate students are also taking the module.

The course textbook is the draft 3rd edition of (Jurafsky and Martin, 2009).[2] It is impossible to teach the entire book in a twelve week module and so we concentrate on the first ten chapters. The following topics are covered:

1. Pre-processing
2. N-gram Language Modelling
3. Text Classification using Naive Bayes and Logistic Regression
4. Sequence Labelling using Hidden Markov Models and Conditional Random Fields
5. Word Vectors
6. Neural Net Architectures (feed-forward, recurrent, transformer)
7. Ethical Issues in NLP

The course is fully online for the 2020/2021 academic year. Lectures are pre-recorded and there are weekly live sessions where students anonymously answer comprehension questions via zoom polls and spend 20-30 minutes in breakout rooms working on exercises. These involve working out toy examples, or using online tools such as the AllenNLP online demo[3] (Gardner et al., 2018) to examine the behaviour of neural NLP systems.

Assessment takes the form of an online end-of-semester open-book exam worth 60% and three assignments worth 40%. The first assignment is worth 10% and involves coding a bigram language model from scratch. The second and third assignments are worth 15% each and involve experimentation, using Google Colab[4] as a platform. For both assignments, a Jupyter notebook is provided to the students which they are invited to use as a basis for their experiments. We describe both of these in turn.

2 Notebooks

We describe the assignment objectives, the notebooks we provide to the students[5] and the experiments they carried out.

2.1 Notebook One: Sentiment Polarity with Naive Bayes

The assignment The aim of this assignment is to help students feel comfortable carrying out text classification experiments using *scikit-learn* (Pedregosa et al., 2011). Sentiment analysis of movie reviews is chosen as the application since it is a

[1] https://jupyter.org/

[2] https://web.stanford.edu/~jurafsky/slp3/

[3] https://demo.allennlp.org/

[4] https://colab.research.google.com

[5] An updated version of the notebooks will be made available in the materials repository of the Teaching-NLP 2021 workshop.

Proceedings of the Fifth Workshop on Teaching NLP, pages 112–114

familiar and easily understood task and domain, requiring little linguistic expertise. We use the dataset of Pang and Lee (2004) because its relatively small size (2,000 documents) makes it quicker to train on. The documents are provided in tokenised form and have been split into ten cross-validation folds. We provide a Jupyter notebook implementing a baseline bag-of-words Naive Bayes classifier which assigns a label *positive* or *negative* to a review. The students are asked to experiment with this baseline model and to attempt to improve its accuracy by experimenting with

1. different learning algorithms, e.g. logistic regression, decision trees, support vector machines, etc.

2. different feature sets, such as handling negation, including bigrams and trigrams, using sentiment lexicons and performing linguistic analysis of the input

They are asked to use the same cross-validation set-up as the baseline system. Marks are awarded for the breadth of experimentation, the experiment descriptions, code clarity, average 10-fold cross-validation accuracy and accuracy on a 'hidden' test set (also movie reviews).

The notebook We implement document-level sentiment polarity prediction for movie reviews with multinomial Naive Bayes and bag-of-words features. We first build and test the functionality to load the dataset into a nested list of documents, sentences and tokens, each document annotated with its polarity label. Then we show code to collect the training data vocabulary and assign a unique ID to each entry. Documents are then encoded as bag-of-word feature vectors in NumPy (Harris et al., 2020), optionally clipped at frequency one to produce binary vectors. Finally, we show how to train a multinomial Naive Bayes model with scikit-learn, obtain a confusion matrix, measure accuracy and report cross-validation results. The functionality is demonstrated using a series of increasingly specific Python classes.

What the students did Most students carried out an extensive range of experiments, for the most part following the suggestions we provided at the assignment briefing and the strategies outlined in the lectures. The baseline accuracy of 83% was improved in most projects by about 3-5 points. The algorithm which gave the best results was logistic regression, whose default hyper-parameters worked well. The students who reported the highest accuracy scores used a combination of token unigrams, bigrams and trigrams, whereas most students directly compared each n-gram order. The students were free to change the code structure, and indeed some of them took the opportunity to refactor the code to a style that better suited them.

2.2 Notebook Two: Sentiment Polarity with BERT

The assignment The aim of this second assignment is to help students feel comfortable using BERT (Devlin et al., 2019). We provide a sample notebook which shows how to fine-tune BERT on the same task and dataset as in the previous assignment. The students are asked to do one of three things:

1. Perform a comparative error analysis of the output of the BERT system(s) and the systems from the previous assignment. The aim here is to get the students thinking about interpreting system output.

2. Using the code in this notebook and online resources as examples, fine-tune BERT on a different task. The aim here is to 1) allow the students to experiment with something other than movie review polarity classification and explore their own interests, and 2) test their research and problem-solving skills.

3. Attempt to improve the BERT-based system provided in the notebook by experimenting with different ways of overcoming the input length restriction.

The notebook We exemplify how to fine-tune BERT on the (Pang and Lee, 2004) dataset, using Hugging Face Transformers (Wolf et al., 2020) and PyTorch Lightning (Falcon, 2019). We introduce the concept of subword units, showing BERT's token IDs for sample text input, the matching vocabulary entries, the mapping to the original input tokens and BERT's special `[CLS]` and `[SEP]` tokens. Then, we show the length distribution of documents in the data set and sketch strategies to address the limited sequence length of BERT. We implement taking 1) a slice from the start or 2) the end of each document, or 3) combining a slice from the start with a slice from the end of each document. In doing so, we show the students how a dataset

can be read from a custom file format into the data loader objects expected by the framework.

What the students did Of the ten students who completed the assignment, three chose the first option of analysing system output and seven chose the second option of fine-tuning BERT on a task of their choosing. These included detection of hate speech in tweets, sentence-level acceptability judgements, document-level human rights violation detection, and sentiment polarity classification applied to tweets instead of movie reviews. No student opted for the third option of examining ways to overcome the input length limit in BERT for the (Pang and Lee, 2004) dataset.

3 Future Improvements

We surveyed the students to see what they thought of the assignments. On the positive side, they found them challenging and interesting, and they appreciated the flexibility provided in the third assignment. On the negative side, they felt that they involved more effort than the marks warranted, and they found the code in the notebooks to be unnecessarily complicated. The object-oriented nature of the code was also highlighted as a negative by some. For next year, we plan to 1) streamline the code, hiding some of the messy details, 2) reduce the scope of the assignments, and 3) provide more BERT fine-tuning example notebooks.

Acknowledgements

The second author's contribution to this work was funded by Science Foundation Ireland through the SFI Frontiers for the Future programme (19/FFP/6942). We thank the reviewers for their helpful suggestions, and the DCU CA4023 students for their hard work and patience!

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

WA et al. Falcon. 2019. Pytorch lightning. GitHub repository.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing*. Pearson Prentice Hall, Upper Saddle River, N.J.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, Barcelona, Spain.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.