Homework 1: Supervised Text Classification (100 points)

Kathleen McKeown, Spring 2021
COMS W4705: Natural Language Processing

Due 2/3/21 at 11:59pm ET

Please post all clarification questions about this homework on the class EdStem under the “hw1” folder. You may post your question privately to the instructors if you wish.

Overview

The goal of Homework 1 is to gain experience with supervised text classification using hand-crafted features. To do this, you will be 1) implementing classifiers for sentiment analysis (§1) and 2) analyzing features and n-gram smoothing methods (§2).

The classifiers you implement will be trained on a sentiment analysis dataset of Tweets from 2013 [1]. Your goal is to predict whether a tweet has a “positive”, “negative”, or “neutral” sentiment. You will be provided with two data files: for training and for development. We will run your model on a hidden test set. Skeleton code and the data files are provided to you on CourseWorks.

1 Programming portion (40 points)

1.1 Tasks

In this assignment, you will implement **4** models using different feature sets and evaluate each model using $F1$. Your models will have
Input features you have selected from a tweet.
Output a label $y \in \{0, 1, 2\}$, indicating the sentiment polarity of the tweet: “negative” (0), “positive” (1), “neutral” (2).

The **4 models** you must implement are (see §1.1.1 for feature details):

1. *Ngram* model: this model should use **only** word ngrams
2. *Ngram+Lex* model: this model should use
   - word ngrams
   - lexicon-based features: for **1** of the lexicons in §1.1.1 2.a), implement **all 4** features in 2.b).
3. *Ngram+Lex+Enc* model: this model should use
   - word ngrams
   - lexicon based features (as in *Ngram+Lex model*)
   - **2** encoding features from §1.1.1, 3
4. Custom model: this model should use
   - the features used in either the *Ngram*, the *Ngram+Lex* or the *Ngram+Lex+Enc* model, whichever performs best
   - **1 additional** feature that you think will improve performance the most, either from §1.1.1 or use your imagination.

For each model above, you should choose the architecture (SVM or Naive Bayes) that gives the highest **macro-averaged F1** on the development set and include only this in your code. If this model has tunable parameters, you should tune them to perform well without overfitting. You may tune parameters you find in the scikit-learn documentation. Some examples are "kernel" and "C" for SVM. In order to achieve good performance, you may experiment with choosing the k best features (e.g., k best n-grams), with different values for k.

Your submission must include all deliverables specified in §3.1 and must run in our environment.

1.1.1 Feature Details

Each of your models will use features detailed here:

1. Ngrams
   (a) word ngrams ($n = 1, 2, 3, 4$)
   (b) character ngrams ($n = 3, 4, 5$)
2. Lexicon-based features.

   (a) Lexicons
      i. Hashtag Sentiment Lexicon
      ii. Sentiment140 Lexicon
   (b) Features per lexicon. Let $score(w, p)$ be the lexicon score for token w with emotion/polarity p.
      i. total count of tokens in the tweet with $score(w, p) > 0$
      ii. total score: $\sum_{w \in tweet} score(w, p)$
      iii. maximal score: $\max_{w \in tweet} score(w, p)$
      iv. score of the last token in the tweet with $score(w, p) > 0$

3. Encoding-based features
   (a) All caps: the number of words with all characters in upper case
   (b) POS tags: the number of occurrences of each part-of-speech tag
   (c) hashtags: the number of hashtags
   (d) elongated words: the number of words with one character repeated more than two times

1.2 Resources

1.2.1 Skeleton Code

You will be provided with a file `hw1.py` that takes 4 command line arguments. Do not modify the command line argument processing or we may not be able to run your code. For mode details see §3.1.

1.2.2 Allowed external resources

For the assignment you are allowed to use scikit-learn implementations of models, feature selection algorithms, and metrics. You may also use standard python packages such as `pandas` and `numpy`. Do not use any other publicly available code (e.g., github repos). If you are unsure about using a package, please ask. Apart from these exceptions, you must write all code yourself. Please refer to the Academic Integrity policy if you have questions in this regard.

As an example, You can look at the text classification example in scikit-learn called “document_classification_20newsgroups.py”.

1.2.3 Data Format

The sentiment data is provided in csv format, with the following 3 columns:

- "tweet_tokens": the tokenized tweet.
- "pos_tags": part-of-speech tags for each token in the tweet.
- "label": the sentiment label (either "'positive", "negative", or "neutral").

The lexicons are provided as zip files with accompanying READMEs. Please read the `readme.txt` for details on processing the files.

2 Written portion (60 points)

Please write all answers in a single file that you will submit as a pdf (see §3.2). Make sure to justify all your answers.

1. **Performance**. (5 points)
   Show the macro-averaged F1 and the class-wise F1 (i.e., the F1 for each individual class) for all 4 models. Show all results in a single table.

   Note: your classifier must achieve a score of at least 0.5 macro-averaged F1 on the hidden test set to receive full credit for this portion (worth 1 point). This threshold should not be difficult to reach.

2. **Error Analysis** (9 points)

   (a) What two encoding features did you implement in the *Ngram+Lex+Enc* model? Why did you choose these and not the others?

   (b) Which of models *Ngram*, *Ngram+Lex*, and *Ngram+Lex+Enc* performed best? Discuss why you think this might be.

   (c) What additional feature did you implement in the *Custom* model? Why? Were you right about it improving performance or wrong? Provide evidence.

3. **Understanding Features** (20 points).
   These questions are based on the features used in [1]. Answer each of the following questions briefly (1-2 sentences).

   (a) Mohammad et al. [1], discuss how features such as hashtags, emoticons, and elongated words could be captured by ngrams . Give one example for each feature (hashtags, emoticons, elongated words) showing how ngrams capture that feature.

(b) For each encoding-based feature (§1.1.1, 3), give an example showing how the feature might capture sentiment.

(c) Why might character ngrams be more important for Twitter data than for news data?

(d) Mohammad et al. [1], section 3.1 mentions non-contiguous ngrams. Why might these be used?

(e) What are the differences between the two lexicons described in Mohammad et al. [1], in sections 2.2.1 and 2.2.2? Provide an example where each lexicon would get the correct sentiment label and where each would not.

4. **Smoothing** (26 points).

Consider the following back-off scheme. First, for some word w we define the sets

$$\begin{aligned}
\mathcal{A}(w_{i-1}) &= \{w : c(w_{i-1}, w) > 0\} \\
\mathcal{B}(w_{i-1}) &= \{w : c(w_{i-1}, w) = 0\} \\
\mathcal{A}(w_{i-2}, w_{i-1}) &= \{w : c(w_{i-2}, w_{i-1}, w) > 0\} \\
\mathcal{B}(w_{i-2}, w_{i-1}) &= \{w : c(w_{i-2}, w_{i-1}, w) = 0\}
\end{aligned}$$

where c is a function that counts n-grams in the training set. For example, if the bigram "smart cat" appears 15 times in the corpus, we will have $c(smart, cat) = 15$.

Now we define a back-off trigram model:

$$p(w_i|w_{i-2}, w_{i-1}) = \begin{cases} p_1(w_i|w_{i-2}, w_{i-1}) & \text{if } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ p_2(w_i|w_{i-2}, w_{i-1}) & \text{if } w_i \in \mathcal{A}(w_{i-1}) \text{and } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \\ p_3(w_i|w_{i-2}, w_{i-1}) & \text{if } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

where

$$\begin{aligned}
p_1(w_i|w_{i-2}, w_{i-1}) &= p_{ML}(w_i|w_{i-2}, w_{i-1}) \\
p_2(w_i|w_{i-2}, w_{i-1}) &= \frac{p_{ML}(w_i|w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} p_{ML}(w|w_{i-1})} \\
p_3(w_i|w_{i-2}, w_{i-1}) &= \frac{p_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} p_{ML}(w)}
\end{aligned}$$

and p_{ML} is the maximum likelihood estimate. For example:

$$p_{ML}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}.$$

Show how to modify the above model to form a valid probability distribution. You may do this by changing p_1, p_2, p_3 and using p_{ML} and/or the count function c. Prove your modification works. Show and explain all your work.

Update: The modifications you propose must give a *reasonable* backoff model. In particular the probabilities of unseen n-grams should not overwhelm the probabilities of seen n-grams (for the same n). Hint: to do this you may need to modify the count function.

3 Deliverables and Submission Instructions

Please read:

- **We WILL NOT accept code that does not run in Python 3.6.***, this includes broken code because of Python 2 print errors.
- **We WILL NOT accept hand-written work** for the written portion.
- File names (including for the zip file) must be **exactly** as specified for full credit

3.1 Programming (40 points)

Submit **one** zip file name **<YOUR-UNI>_hw1.zip** to CourseWorks.
This should have **exactly** the following files. Please DO NOT include the data files in the zip.
NOTE: your zip file should have only the following files:

- **`features.py`**: A file that takes a csv file as input and returns the input features and labels form the data in the csv. Indicate with comments the code for implementing each feature. If we cannot easily find the feature extraction, we will not grade it.
- **`hw1.py`**: A file that does the following
  - Takes 4 command line arguments: 1) the training file full path, 2) the testing file full path, 3) a model name (as a string), and 4) the full path

    to the directory containing the lexica. While working on this homework, you will pass the development file as the test file. NOTE: all this should be already done by the skeleton code provided and should not be modified.

  - Trains the model on the training set
  - Evaluates the model on the input testing file.
  - Prints out the macro-averaged F1 on the input testing file.
  - Prints the class-wise F1 on the input testing file.

- **A README file that must include:**
  - Your name and email address.
  - Homework number.
  - Information on how to train and test your classifier.
  - A description of special features (or limitations) or your classifier.

Your code should be documented in a meaningful way and implemented efficiently. This can mean expressive function/variable names as well as commenting.

Your code should be runnable with the following command (as an example):

```
hw1.py --train <path_to_resources>/data/train.csv
    --test <path_to_resources>/data/dev.csv
    --model "Ngram+Lex"
    --lexicon_path <path_to_resources>/lexica/
```

where resources is the unzipped `resources.zip` file provided to you.

3.2 Written Answers (60 points)

You should submit the following on Gradescope:

- A **hw1-written.pdf** file containing your name, email address, the homework number, and your answers for the written portion.

4 Academic integrity

Copying or paraphrasing someone's work (code included), or permitting your own work to be copied or paraphrased, even if only in part, is not allowed, and will result in an automatic grade of 0 for the entire assignment or exam in which the copying or paraphrasing was done. Your grade should reflect your own work. If you believe you are going to have trouble completing an assignment, please talk to the instructor or TA in advance of the due date.

References

[1] Saif M. Mohammad, Kiritchenko, S., and Zhu, X. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. *SemEval-2013*. `https://www.aclweb.org/anthology/S13-2053.pdf`

[2] Preslav Nakov, Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. 2013. SemEval-2013 Task 2: Sentiment Analysis in Twitter. *SemEval-2013*. `https://arxiv.org/pdf/1912.06806.pdf`