1.5em 0pt

COMS W4705–Spring, 2021
Name: Weiyao Xie
Email: wx2251@columbia.edu

HW1

1. **Performence**

   1.1

Features	Macro f1
Ngram	0.4652
Ngram+Lex	0.4991
Ngram+Lex+Enc	0.4990
Customized	0.5241

Table 1. The table shows macro-averaged f1 scores for models with different features

Features	Negative	Positive	Neutral
Ngram	0.1829	0.5055	0.7073
Ngram+Lex	0.4991	0.5331	0.6420
Ngram+Lex+Enc	0.3220	0.5331	0.6419
Customized	0.3745	0.5266	0.6712

Table 2. The table shows f1 scores for each class of the model

2. **Error Analysis**

- (a) The two encoding I use are: number of hashtags and number of words with all characters in upper case. First, I think people tend to show emotions when they are discussing some hot topics which usually follow a in twitter. I think when someone includes multiple hashtags in his/her tweets, he/her is trying to express some feelings. Second, people use all capital letters when they are trying to express very strong emotions. If all the words are in capital letters, it can be easy to identify whether someone is angry or happy. I think both features can be very helpful in identifying sentiment polarity.

  The reason I did not choose the other two is that elongated words can sometimes be ambiguous. It can show that someone is depressed or it happens when people just don't want to type everything. The POS tags may tell use the sentence structure, but it does not directly encode the sentiment in the sentence structure. People can use same sentence structures to express different emotions.

- (b) Among all three models, Ngram+Lex has the best performance. The lexicon improves the f1 score by 0.03. I think the polarity score helps the model to identify the sentiment much more effectively. The polarity score can be a very intuitive way to identify sentiments. When people using more positive words, they will have a higher total score. When people using negtive words, the score will be very low. Since my model is SVM, I do not need to use absolute value for these polarity scores. I think this is the reason why lexicon is so effective in this model.

- (c) My additional feature is based on Ngram+Lex+Enc. I think the reason that the model uses the encoding features did not meet expectation is that there is no complementary features to support the encodings. For example, when there are a lot of hashtags in a tweet, it is difficult to identify whether a user is positive or negative towards the topic. To better capture users' emotions, I think number of exclamation marks is a good choice. It is easier to identify someone's intention when there are many exclamation marks which show a very strong tone.

3. **Understanding Features**

- (a) **Hashtags**: In twitter, people always use followed by words to set topics to attract people. A Unigram model can be very effective to capture the feature. For example, if a twitter is "I'm happy fabulous fabulous" The countVectorizer will split the words using space, and create features for each words. Therefor the count for fabulous is two here. fabulous is a positive word with very high score in the HS text file. This hashtag feature is succefully captured by Unigram.
  Emoticon: Emoticon can be effectively captured by Bigram model. For a tweet "I'm sad *_*" The last word represents a crying face. Using a bigram model, we can get "sad *_*"=1. This feature can help the model to identify that this tweet has a negative sentiment.
  Elongated Words : The Elongated Words means certain words have repeating characters. A unigram model can effectively capture Elongated Words features. For example, a a tweet "I am so sooooo sad." The tweet has a negative sentiment. The unigram model will seperate "so" and "sooooo" which can help the model to identify the sentiment.

- (b) **All caps**: All caps words make it easier to identify strong tone. "He is COOL" has stronger positive sentiment than "He is cool".
  POS tags : POS tags help to identify the sentence structure. Some part of speech can occur much more often than others in certain sentiment. When describing some object with positive sentiment, adjectives can be used more frequently than pronouns.
  hashtags : In twitter, people always emphasize their emotions using hashtags followed by words. "I had ice-cream happy delicious" These two hashtag words can be the key in determining the sentiment.
  elongated words : Words with repeating characters are usually used when people trying to express strong feelings/emotions. It is easy to that "She looks sooooo beautiful" has a more positive sentiment than "She looks so beautiful"

- (c) First there is a lot of available data to train. Since using char ngram, the number of features will be huge. We need more data to fit our model. Second, people communicating on twitter tend to use short sentences to express emotions/feelings. Under this circumstance, context is not as important. For example, sometimes it is easy to tell someone's sentiment from just one hashtag "happy". However, when analyzing other texts, context becomes more important than just a single word.

- (d) Using non-consecutive ngram can help the model to capture some more general features. Some words do not always appear in fixed combinations, but they can show the similar meanings when partnered with different words. For example, "too" can show up with many different words, but "too" always have the similar meaning. It is better to capture then general meaning of "too" by using non-contiguous ngram.

- (e) Hashtag Sentiment Lexicon is useful because people use +words to express themselves

on twitter. For example, when someone include positive words tags in twitter "amazing". It is highly possible that this twitter has positive sentiment. However, sometimes, the can mean something else. Like number of certain things. 5 can stand for the fifth thing, which does not count as a sentiment feature.

Emoticon can be a very abstract feature to capture sentiment. Sometimes people add a crying face after "I am sad" to indicate a negative sentiment. However, sometimes people also add the same emoticon after "I am happy" to show a positive sentiment because he/she was so happy that he/she cried.

4. **Smoothing**

For the backoff model to work, we need to discount the higher order n-grams to save some probability mass for the lower order n-grams. Otherwise, the probability will exceed 1.

To do this, I will use Kneser-Ney Smoothing to discount the higher order ngram by subtracting a small amount from the count. The advantage of this smoothing method is that by subtracting a small amount from the large count, the large count will not be affected a lot, and since we do not trust the small counts, it is fine to deduct them.

We make the following modifications: When there is neither Bigram nor Trigram:

$$P_3 = \frac{C(w_i)}{N} \tag{1}$$

Thus we need to discount the Bigram to make room for unigram:

$$P_2 = \frac{C(w_{i-1}wi) - d_2}{\sum_{i=1}^{n} C(w_i)} + \lambda_{w_{i-1}} P_3 \tag{2}$$

We subtract a small amount from Bigram, and the second term is the linear interpolation of the unigram we computed in (2). This is the probability we return when there is no trigram feature but there is bigram.

Then since we backed off to bigram, we need to discount from

$$P_1 = \frac{C(w_{i-2}w_{i-1}wi) - d_3}{\sum_{i=1}^{n} C(w_{i-2})} + \lambda_{w_{i-2}} P_2 \tag{3}$$

At the end we return discounted P_1 when there is trigram feature.

Here we can pick d as a small amount. Normally we use 0.75 but when the count is 1, we can use 0.5 instead. The second term shows the bigram with an interpolation weight λ. The concept of linear interpolation is illustrated in 3.28 in the textbook.

$$\begin{aligned} P_{(w_n|w_{n-2}w_{n-1])}} &= \lambda_1 w_{n-2:n-1} P(w_n|w_{n-2}w_{n-1}) \\ &+ \lambda_2 w_{n-2:n-1} P(w_n|w_{n-1}) \\ &+ \lambda_3 w_{n-2:n-1} P(w_n) \end{aligned} \tag{4}$$

Where the sum of λ is 1. The λ is determined by count of each ngram.