TTIC 31190 HW2

Lynn Zheng

October 28, 2020

1 Unigram Binary Features with Perceptron Loss

Best DEV accuracy: 0.6509
Using the weights that achieved the best DEV accuracy,
DEVTEST accuracy: 0.5880

2 Unigram Binary Features with Hinge Loss

Best DEV accuracy: 0.6655
DEV accuracy: 0.6080
(See Section 5. Feature Engineering for the complete table)

3 Feature Weight Analysis

Label	Feature/Word	Weight
0	listless	1.8500
	disingenuous	1.8000
	pity	1.7600
	Lacks	1.7400
	lacking	1.7300
	elsewhere	1.7300
	hardly	1.7000
	lacks	1.6900
	tiresome	1.6600
	poorly	1.6500
1	1950s	0.7800
	Warren	0.7800
	Troopers	0.7700
	Pete	0.7600
	anarchic	0.7300
	Bartlett	0.7200
	versus	0.6900
	Greek	0.6900
	stake	0.6800
	Deuces	0.6800
2	vividly	2.2300
	pleasant	2.0300
	thought-provoking	1.8400
	wonderfully	1.8300
	treat	1.6500
	gorgeously	1.6300
	touching	1.5900
	miracle	1.5700
	moved	1.5600
	delicious	1.5500

Table 1: Unigram binary feature weights, trained with hinge loss. Labels: 0 - negative sentiment, 1 - neutral, 2 - positive

For positive and negative labels, the words with large weights are usually words with corresponding sentiments, for example, “pleasant”, “gorgeously” for positive labels and “lacking”, “tiresome” for negative labels. For neutral labels, the words with large weights don’t appear too distinct but tend to be proper nouns.
It’s noteworthy the top 10 weights for positive or negative weights are significantly larger

than the top 10 for neutral labels. This may be the case that the notion of "strongly" positive or negative words is valid, but less so the notion of "strongly" neutral words (i.e., words strongly-indicatively of neutral sentiments).

Also note that both "Lacks" and "lacks" appear in the top 10 weighted-words for negative labels. This may be indicative of duplicate features which might lead to overfitting. Motivated by this observation, Section 6. Discussion and Future Improvement discusses my experiment with training on a corpus with capitalization removed.

4 Error Analysis

Error category	**Gold standard**	**Predicted**	**Sentence**
incorrect/questionable gold standard	1	0	The ga-zillionth airhead movie about a wife in distress who resorts to desperate measures .
	1	0	It 's mighty tedious for the viewer who has to contend with unpleasant characters , hit-and-miss performances and awkwardly staged scenes .
	1	0	Though Catch Me If You Can is n't badly made , the fun slowly leaks out of the movie .
	1	0	More maudlin than sharp .
	1	0	It 's hard to know whether or not to recommend this film because for every thing it does right there 's at least one and occasionally two things it gets ever so wrong .
	1	0	Full of witless jokes , dealing in broad stereotypes and outrageously unbelievable scenarios , and saddled with a general air of misogyny
	1	0	Nasty , ugly , pointless and depressing , even if you hate clowns .
negation (in **boldface**)	0	2	A great ensemble cast ca **n't** lift this heartfelt enterprise out of the familiar .
	0	2	Impostor has a handful of thrilling moments and a couple of good performances , but the movie does **n't** quite fly .
	0	2	Determined to be fun , and bouncy , with energetic musicals , the humor did **n't** quite engage this adult .
	2	0	A difficult , absorbing film that manages to convey more substance despite its repetitions and inconsistencies than do most films than are far more pointed and clear .

	2	0	A taut psychological thriller that does **n't** waste a moment of its two-hour running time .
	2	0	A marvel like **none** you 've seen .
	2	0	I sympathize with the plight of these families , but the movie does n't do a very good job conveying the issue at hand .
words not seen in TRAIN (in **boldface**)	2	0	Filmmakers who can deftly change moods are **treasures** and even **marvels** .
	2	0	**Slick** piece of **cross-promotion** .
	2	0	**Hilariously** inept and ridiculous .
context/sarcasm	0	2	The only excitement comes when the credits finally roll and you get to leave the theater .
	0	2	I 'll bet the video game is a lot more fun than the film .
	0	2	The lower your expectations , the more you 'll enjoy it .

Table 2: Error analysis for unigram binary features + hinge loss

For the **incorrect/questionable gold standard** category, a lot of movie reviews annotated as neutral actually appear pretty negative and the model tends to classify them as negative. This points to the issue of subjectivity in text sentiment classification.

For the **negation** category, the model makes two types of mistakes: (1) mistaking negative ones as positive, and (2) mistaking positive ones as negative. This reveals the unigram model's weakness and may be better addressed using longer n-grams.

For the **words not seen in train** category, the cause of error may be that words strongly indicative of positive or negative sentiments are missing from the training corpus. When inspecting the difference between the training vocabulary and the DEVTEST vocabulary, I found that words like "Which" and "Famous" weren't in the training corpus despite the fact their lowercase counterparts are. This indicates that case sensitivity might have caused our model to lose information: it may assign zero weights to capitalized words when in fact it has learned the weights for their lowercase counterparts.

Finally, the **context/sarcasm** category contains the errors that may be caused by the model's lack of common-sense knowledge or the context of the particular piece of movie review, or, the model's failure to interpret sarcasm.

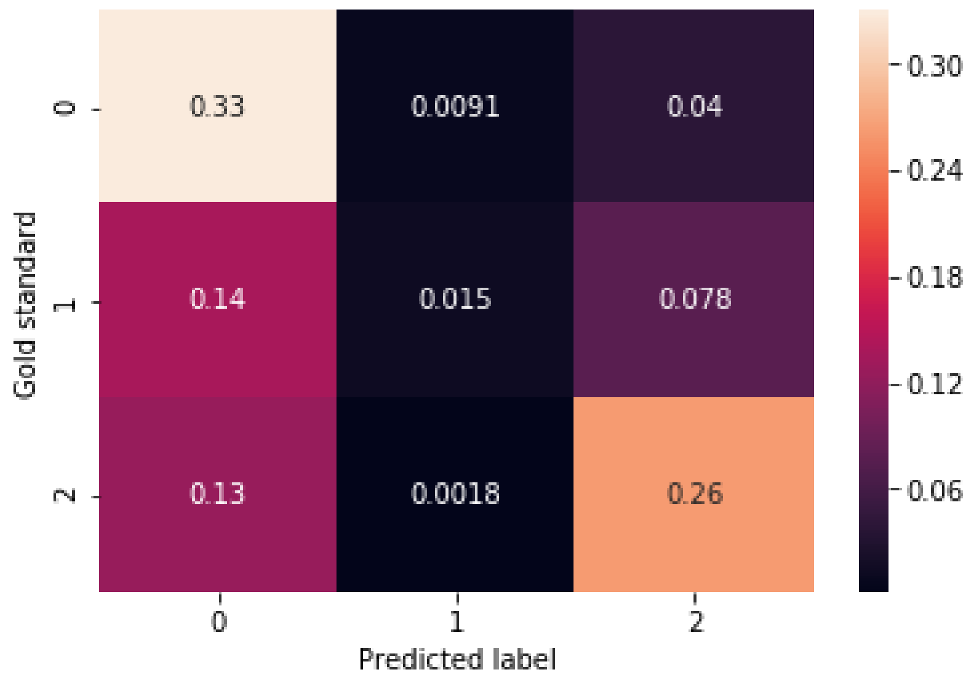

Figure 1: Confusion matrix for unigram binary features trained with hinge loss on DEVTEST

From the confusion matrix above, it appears that the model makes the most errors mistaking neutral or positive labels as negative ones. The model also seems to favor the negative class, classifying 50% of the examples from DEVTEST as negative.

5 Feature Engineering

Both of my functions, `get_binary_features, get_count_idf_features` support n-grams for any specified n. I've implemented a utility for enumerating n-grams in a sentence called `get_ngram_counters(token_lists, n)`.

5.1 Bigram Binary Features

This feature checks for the presence of a bigram in a sentence. The design is motivated by errors the model made when the sentence contains negation, such as "'t great". Formally,

$$f^{bigram,binary}(\text{x, y}) = \mathbb{I}[\text{y = label}] \wedge \mathbb{I}[\text{x contains "word1 word2"}]$$

5.2 Unigram Count Features

This feature assigns feature values based on the number of times the unigram appeared in the sentence. The motivation is that, although each training example is relatively short and unlikely to have multiple repeated words, multiple occurrences of the same word may indicate a stronger sentiment. People tend to repeat words for emphasis, for example, I found in `sst3.train-full-sentences` an example with a positive label:

```
Good fun , good action , good acting , good dialogue , good pace , good
    cinematography .
```

Formally,

$$f^{unigram,count}(\text{x, y}) = \begin{cases} \sum_{i=1}^{|\text{x}|} \mathbb{I}[\text{x}_i\text{= word}] & \text{if } \mathbb{I}[\text{y = label}] \\ 0 \quad \text{otherwise} \end{cases}$$

5.3 Unigram TF-IDF Features

This feature extends the count feature and assigns more weight to rare words. The motivation is that people may use rare or self-crafted hyphenated words for emphasis of stronger sentiments. An example labeled positive and one labeled negative from `sst3.train-full-sentences` are shown below:

```
As averse as I usually am to feel-good , follow-your-dream Hollywood fantasies ,
    this one got to me .

Made me unintentionally famous -- as the queasy-stomached critic who staggered
    from the theater and blacked out in the lobby .
```

Formally,

$$f^{unigram,tf-idf}(\texttt{x, y}) = f^{unigram,count}(\texttt{x, y}) \times \frac{|S|}{|\{s \in S : s\text{contains word}\}|}$$

5.4 DEV and DEVTEST Accuracy

Feature + loss	**Best DEV accuracy**	**Corresponding DEVTEST accuracy**
Unigram binary + perceptron	0.6345	0.5626
Unigram binary + hinge	0.6655	**0.6098**
Bigram binary + hinge	0.6400	0.5499
Unigram count + hinge	**0.6727**	0.5880
Unigram TF-IDF + hinge	0.5436	0.5281

Table 3: Best DEV accuracy and DEVTEST accuracy evaluated using the weights that performed the best on DEV, for all five different features

We observe that unigram binary with hinge loss achieved the best performance. Bigram binary performed relatively well on DEV but not so well on DEVSET. A possible explanation is that the highly-weighted bigrams it captured from train aren't very well-represented in `devtest`. Also note that there are sentence segments in train and only full sentences in DEV and DEVSET, which may be a cause for the difference in bigram representations. Unigram count performed slightly better than unigram binary on DEV but slightly worse on DEVTEST. I did expect similar performance for unigram binary and count features as the feature values on each training example won't differ too much when the sentence is short, with few repeated words. Unigram TF-IDF achieved the lowest accuracy, but also had the smallest difference between DEV and DEVTEST accuracy. This may be because any rare word seen in train occuring from DEV and DEVTEST will be well-captured and assigned a large feature score by TF-IDF features. In all, unigram binary features are pretty good baseline features that are also very easy to implement.

5.5 Feature Weight Analysis

	Bigram binary	**Unigram count**	**Unigram TF-IDF**
0	lacks the	lacks	direct-to-void
	never quite	lacking	Independence
	lacking in	tiresome	off-puttingly
	's hardly	listless	Mitch
	pretty mediocre	lousy	undermined
	most offensive	sloppy	deficit
	a failure	uneven	riding
	terrible movie	unfunny	simplify
	never rises	worst	rope
	time stinker	depressing	confining
1	most part	means	coastal
	elements .	While	Debate
	an exploration	nor	Might
	Bartlett 's	Big	Bardem
	anything else	Cletis	Sychowski
	left field	entire	Russians
	one word	word	Quelle
	the deadpan	If	R-rated
	's time	Barry	friggin
	bizarre sort	York	whirls
2	**n't disappoint**	pleasant	cunning
	very best	thought-provoking	tenderly
	to behold	treat	exude
	funny stuff	wonderfully	follow-your-dream
	of laughs	touching	expanded
	very funny	vividly	streaks
	to resist	delight	belly-dancing
	, amusing	wonderful	Juan
	never dull	feel-good	sexiness
	entertaining and	funniest	Easily

Table 4: Top 10 weighted features for the three new feature templates. The numeric weights are reported in the code file.

For bigram binary features, as I expected, a lot of the highly-weighted features are bigrams heavily indicative of the corresponding sentiment label. The results also validated my design goal in how they captured some negation bigrams like **n't disappoint** and **never null**.

For unigram count features, the highly-weighted features are very similar to those of unigram binary features. We also observe the similar trend with highly-weighed words for the neutral label: most are generic words, words that serve grammatical function, or proper nouns.

Finally, highly-weighted unigram TF-IDF features consist of a lot of rare words, especially hyphenated words. The association between the word sentiment and the label category is less clear than that of the other features, but we still see the trend of proper nouns appearing in the neutral label list.

6 Discussion and Future Improvements

Some possible improvements include setting cutoff values greater than one to eliminate features that are too rare to be informative. For better computation precision, we could use numpy floats with more bytes.

I also tried removing capitalization from TRAIN, DEV, DEVTEST and run the experiment with unigram binary loss.
Best DEV accuracy: 0.6727
DEVTEST accuracy: 0.6007

The results weren't very different from training on the raw corpus. This might mean that, in our corpus, capitalization didn't provide much information for word sense disambiguation (otherwise the accuracy would drop significantly). This also means that the bottleneck to our model performance is more about feature engineering rather than the lack of weight-sharing between capitalized words and their lowercase counterparts.