CENG 463 - Introduction to Natural Language Processing
Fall 2023-2024
Assignment 1

KARADAYI, Meriç
e2448553

December 17, 2023

Task 1

1. Implementation details

1. Firstly algorithm reads all train, dev, and test data. Then applies the preprocessing operations which will be discussed in Answer 2. Then creates mega documents separately for train, dev, and test sets.

2. Then, it creates features from the training set which will be detailed in Answer 3. These features are unigrams, bigrams and trigrams. Then it extracts features of all megadocuments.

3. After the training set is created in the proper format with features, it trains the data with classifier algorithms such as Naive Bayes and SVC.

4. When training is completed, the algorithm tests our model by using the dev set and provides evaluation metrics such as accuracy, recall, precision, f1, and confusion matrix.

5. If the results are satisfactory, finally it tests the model by using test set.

2. Preprocessing Operations

The dataset consists of each book's header and description; these headers and descriptions are separate lines in the data file, so I have processed the lines two by two. Firstly I have converted headers and descriptions into a lower format. The reason for such a lowering is that if we do not do this operation, the same words at the beginning of the sentences and in the middle of sentences will be evaluated as different words. I think this is not desired in this case. Then by using *nltk.tokenize.word_tokenize()* function, I converted my string into a list of tokens. These tokens consist of all digits, words, and punctuation. Then I have discarded punctuations and stop words that probably will not be informative for classifying genres. Then I tried two processes which are lemmatization with *nltk.stem.WordNetLemmatizer* and stemming with *nltk.stem.PorterStemmer*. However, I observed that the lemmatization process contributed more than stemming. I think this is because WordNetLemmatizer is converting words into meaningful tokens, however, stemmer is only clipping the words which can result in different tokens for the same words with different suffixes. Then, I joined the header and description. As the last step, after this join operation, my tokens in the header and the descriptions do not make any difference. Therefore, I also added the words in the header with a tag prefix, "HEAD_" additionally.

3. Feature Selection

I have tried various features for the training model. Firstly I have used all unique unigrams as they exist or not. However, the number of unique words was too large and there were many unnecessary words in them. Therefore, I believe that I should filter them in some way. So my next approach was to use the most frequent n words in the mega document of training. This was performing better results but the accuracy/precision/recall were still quite low. Then, I thought, since we are trying to classify the genres, genre-specific words would be better features for model to distinguish them. For this reason, instead of using the most frequent words in all mega documents, I have chosen the most frequent words separately for each genre and then combined them. By doing that also the possible data-imbalance problem would be solved in some way. The model trained by these features resulted in a much better result. Therefore I decided to use this approach for extracting features. I also used the same approach with bigrams and trigrams. Because even the existence of a word is important, I think words frequently existing nearby is a different information than only existing. The results of bigram and trigrams themselves were not performing well as unigram, but they were using features unigram does not have such names of people, places and organizations. Finally, I have combined all of these unigram, bigram, and trigram features for training the model.

4. Evaluating Performance

The Model that gave the best result was using n-gram combinations.

1. When the results of Naive Bayes are observed, the recall of religion is the lowest score with 0.54. However, the recall of religion is 0.79 which is a high value for our model. We can make such an inference that religion genre books have also a high probability of belonging to some other class and this is leading to such low precision. The second and next lowest score is the precision of the horror. This class has a higher recall score. So, we can interpret that our model is labeling many books as horror and this can also be the reason why its recall is higher. Our general accuracy is approximately 0.70 but I think improving our features and model by focusing on horror and religion can increase this score.

2. If we inspect the SVC results. We can also see that horror labeling is performing significantly less than others as in NB. I think we can conclude that features that are added from horror do not represent that class well.

Tag	Prec.	Recall	F-measure
horror	0.5661	0.7137	0.6314
mystery	0.8072	0.7500	0.7775
philosophy	0.6310	0.7500	0.6854
religion	0.7925	0.5478	0.6478
romance	0.6030	0.7061	0.6505
science	0.8232	0.7087	0.7617
science-fiction	0.6614	0.6917	0.6762
sports	0.8706	0.7447	0.8028

Accuracy: 0.7018766756032172

Figure 1: Naive Bayes test set results

Tag	Prec.	Recall	F-measure
horror	0.5679	0.6966	0.6257
mystery	0.7578	0.7042	0.7300
philosophy	0.6545	0.7061	0.6793
religion	0.7604	0.6348	0.6919
romance	0.6296	0.6711	0.6497
science	0.7972	0.7348	0.7647
science-fiction	0.6518	0.6708	0.6612
sports	0.8744	0.8000	0.8356

Accuracy: 0.7024128686327078

Figure 2: SVC test set results

5. Comparison of Naive Bayes and SVC

When performances of SVC and Naive Bayes are compared, I do not think there are significant changes in the results of evaluation metrics. There are differences class-wise, however, these differences compensate for each other. For example, while the f1 score of mystery in Naive Bayes is higher than SVC, it is vice versa for religion. I think none of them outperformed the other. However, I was expecting SVC to outperform the Naive Bayes classifier. Because Naive Bayes depends on only the probabilities, SVC is fitting weights for features and I believed these weights would be more robust than the probabilities of occurrences.

6.Confusion Matrix of Naive Bayes

When we analyze the confusion matrix of Naive Bayes, we are more eligible to interpret the reasons for the precision and recall results of the classes that are discussed in Answer 4. Our model can not distinguish the difference between religion and philosophy books. 56 religion book is classified as philosophy. Also, our religion predictor is working accurately except for philosophy books. The reason for these misclassifications can be the similarity of their contexts. Also, our model was having difficulties classifying the horror genre. We can see that the model is predicting so many books as horror. This can be the result of a bias to horror in our model.

	horror	mystery	philosophy	religion	romance	science	science-fiction	sports
horror	<167>	21	3	3	17	.	23	.
mystery	29	<180>	.	2	21	1	7	.
philosophy	5	1	<171>	21	2	20	8	.
religion	17	3	56	<126>	8	8	11	1
romance	17	8	1	1	<161>	.	21	19
science	8	2	35	6	1	<163>	13	2
science-fiction	39	8	4	.	15	4	<166>	4
sports	13	.	1	.	42	2	2	<175>

(row = reference; col = test)

Figure 3: Naive Bayes Confusion Matrix

Task 2

1. Input Data Analysis

The second column is Coarse-Grained POS Tags: 'NOUN', 'PRON', 'PROPN', 'SCONJ', 'ADV', 'INTJ', 'CCONJ', 'ADP', 'ADJ', 'PART', 'SYM', 'VERB', 'DET', 'X', 'NUM', 'AUX', 'PUNCT'. which are more general tags.

Third column is Fine-Grained POS Tags: 'CC', 'VBP', 'JJ', 'TO', 'DT', 'NN', 'RP', 'XX', 'PRP$', 'EX', 'FW', 'JJR', 'RB', 'RBS', 'MD', 'AFX', 'WP', 'IN', 'NNPS', 'HYPH', 'LS', 'VBN', 'VB', 'VBD', 'GW', 'POS', 'NNP', 'ADD', '“', ',', 'VBG', 'WDT', 'WP$', 'NFP', 'UH', '.', 'PRP', 'JJS', 'NNS', 'RBR', 'WRB', 'VBZ', '-RRB-', ':', 'SYM', '-LRB-', 'CD', 'PDT', '$', """ which are more detailed, specific tags.

We are using the first(token) and third(Fine-Grained POS tags) columns in this task. We are using the second column only for punctuations.

2. Results of Feature Sets

	Accuracy	Precision	Recall	F1
SET1	0.914	0.914	0.914	0.914
SET2	0.93	0.93	0.93	0.93
SET3	0.938	0.938	0.938	0.938

Table 1: Scores

New features of Set2:

- word itself
- root of the word coming from stemmer
- suffixes of words which stemmer discarded
- is it a stop word or not
- is it a punctuation
- is it first word in the sentence
- is it last word in the sentence
- previous word
- next word

New features of Set3: All the features on Set2 except the previous and next words, additionally all those features for the next and the previous words.

3. Comparison of Logistic Regression and CRF

CRF is more suitable for this task. POS tags of the words are not limited to their own features, they also depend on the sentences and the place where they occur. CRF uses sequences of data for tagging our inputs so the importance of neighbors and sentences is included in this model. However, Logical Regression considers each input independently, and their places in the sentence do not make any change. If we have used Logical Regression instead of CRF, I think the performances of our all pos-taggers models will be badly affected since CRF is a more proper choice. However, in feature set 3 I also additionally added features of the neighbor words, therefore I think it will be affected badly but this effect will be less than the others.