Week 7: Propaganda Detection

This week we will be looking at the propaganda detection task (Da San Martino 2019) and developing a baseline model for technique classification.

We will be working with an adapted version of the dataset from the paper. In particular, I have

- reduced the number of propaganda techniques
- randomly sampled sentences labelled with a particular technique
- randomly sampled sentences (from the same original articles) which do not contain propaganda
- reformatted the data so that the snippets can be "read" in the context of the sentence by inserting \<BOS> and \<EOS> tags.

Let's load it up in a pandas dataframe so that we can look at some examples to illustrate that last point.

In [ ]:
```
import os
parentdir = "/Users/juliewe/Dropbox/teaching/AdvancedNLP/2024/week7/lab7/lab
train_file= "propaganda_train.tsv"
train_path=os.path.join(parentdir,train_file)
```

In [ ]:
```
import pandas as pd
train_df=pd.read_csv(train_path,delimiter="\t",quotechar='|')
train_df.head(20)
```

We can see in the cell above that there are 2 columns. The first column is the label i.e., the propaganda technique or the label "not_propaganda". The 2nd column contains the text. Within the text we can see the special \<BOS> and \<EOS> tags which indicate the propaganda snippet.

Let's have a look at some examples from the **loaded_language** class.

In [ ]:
```
train_df[train_df["label"]=="loaded_language"]
```

In the first loaded_language example (row 5 of the original dataframe), we can see that there is a single word *annihilated* which is between \<BOS> (**beginning of span**) and \<EOS> (**end of span**).

Exercise 1: Exploratory Data Analysis

Write code and plot appropriate graphs to visualise each of the following questions.

a) How many samples are there for each class?

b) What is the average length of sentence for each class?

c) What is the average length of propaganda snippet for each class?

In [ ]:

In [ ]:

In [ ]:

Exercise 2: Sentence Level Binary classification

Build a simple classifier (e.g., Naïve Bayes or Logistic Regression) which can take a sentence and predict whether it contains propaganda or not. Things you will need to think about

- making a binary "propaganda" or "not_propaganda" label
- splitting the data into training and validation
- making a bag-of-words representation of each sentence. This could be a dictionary where the keys are the words and the values are the frequencies within the sentence.
- the implementation of the classifier itself. You are not expected to build this yourself. A good one to use would be the multinomialNB classifier in scikit-learn

It's worth thinking about the input format expected by the classifier before pre-processing the data. Code to import and use the scikit-learn multinomialNB classifier is below for consideration

In [ ]:
```
#This gives us some random toy data.
#In this toy data there are 10 data points (e.g., sentences)
# each sentence is represented as a vector of 100 values
# each value could be the frequency of a particular word in the vocab.  The
# these are stored in X
# there are 2 possible labels (0 and 1) which are stored in Y

import numpy as np
rng=np.random.RandomState()
X = rng.randint(6,size=(10,100))
Y = rng.randint(2, size=10)

print(X)
print(Y)
```

In [ ]:
```
#we can give this as input to the MultinomialNB classifier using the fit met
from sklearn.naive_bayes import MultinomialNB

classifier=MultinomialNB()
classifier.fit(X,Y)
```

In [ ]:
```
#we can predict the value for any datapoint
#here we are making up some more random points with random labels so I would

X = rng.randint(6,size=(5,100))
Y = rng.randint(2, size=5)
print(Y)
classifier.predict(X)
```

So imagine you have sentence representations and labels as follows. We need to generate vectors for them where each column corresponds to a particular word in the

vocabulary

In [ ]:
```
toy_training_data=[({"everyone":1,"hates":1,"vectors":1},1),({"vectors":1,"a
Xdicts,Y=zip(*toy_training_data)
```

We could write some code to turn the Xdicts into vectors (you first need to work out what the vocab is and assign an index to each vocab item). Or we can use scikit-learn's CountVectorizer directly on the texts.

In [ ]:
```
toy_training_data=[("everyone hates vectors",1),("vectors are useful",0)]
Xsents,Y=zip(*toy_training_data)
```

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer()
vectorizer.fit(Xsents)

# Printing the identified Unique words along with their indices
print("Vocabulary: ", vectorizer.vocabulary_)

# Encode the sents
Xvectors = vectorizer.transform(Xsents)

print(Xvectors)
```

In [ ]:
```
classifier=MultinomialNB()
classifier.fit(Xvectors,Y)
```

In [ ]:
```
classifier.predict(Xvectors[1])
```

What happens if some new sentences have words not in the vocabulary? We can see here that they are just ignored by the vectorization process - this is fine as they are not going to help the classifier as they are unknown

In [ ]:
```
toy_test_data=[("everyone hates useful vectors",1),("vectors are really use
Xsents,Y=zip(*toy_test_data)
testVectors=vectorizer.transform(Xsents)

print(testVectors)
```

In [ ]:
```
classifier.predict(testVectors)
```

Lets go back to a modified version of the exercise which assumes we are going to use CountVectorizer and MultinomialNB

Build a MultinomialNB classifier which can take a sentence and predict whether it contains propaganda or not. Things you will need to think about

- making a binary "propaganda" or "not_propaganda" label
- splitting the data into training and validation
- making a bag-of-words representation of each sentence using CountVectorizer
- Training a MultinomialNB classifier on the training data and evaluating it on the validation data

In [ ]:
```

```

In [ ]:

In [ ]:

Exercise 3: Snippet level binary classification

Repeat exercise 2 but train and test on propaganda snippets rather than whole sentences.

In [ ]:

Exercise 4: Mixing snippets and sentence

- How well does your sentence level classifier work on snippets?
- How well does snippet level classifier work on sentences?
- What about if you train on both sentences and snippets?

In [ ]:

Extensions

1) Can you use cross-validation to evaluate your classifiers rather than a single training/development split

2) Can you use a different type of classifier and / or feature representation. E.g., logistic regression where the feature values are tf-idf values rather than frequencies?

3) Can you carry out multi-class classification to identify the propaganda technique used?

4) Does it help to first use your binary classifier to decide whether there is any propaganda or not?

In [ ]: