Final year Project Design and Implementation

Initial Steps

* Using NLTK train a classifier model (chapter 6) with the spoilers’ dataset to be able to detect spoilers
* Use the ‘pickle’ library in the standard python libraries, or any save features in NLTK, to save the classifier model so that it doesn’t’ have to be retrained every time you use the extension
* Integrate the classifier model with the current chrome extension.
  + Get the text from each paragraph
  + Pass that text into the model
  + If the model thinks it’s a spoiler black it out
  + Otherwise leave it as is

**What your machine learning representation will be? With nltk?**

* Nltk extracts from the text provided in the data set and creates features. These features will then be given probabilities based on the Naïve Bayes formulas. In my case, these probabilities refer to how likely this entry in my data base is likely to be a spoiler. The higher the probability, the higher the chance of the entry being a spoiler. All the words in the particular entry will then be calculated together to give an answer as whether it is a spoiler or not.
* To dive deeper into this, the Naïve Bayes classifier is essentially saying, “given word A in review text B what is the probability that this entry is a spoiler”.
* Nltk reads the paragraph in but it breaks the sentences up into individual words and features internally. However, nltk does not take context into consideration and the context in which everything is said matters. That is tokenization. Different types of tokenizers in within nltk. Choice you make of tokenizer can have an impact. Tokenizers don’t work well for twitter due to hashtag and .
* For data like this pre processing is needed. I plan to break the paragraph into sentences. Then break it up further into words and these words can be looked at as features. Pre-processing will allow us to remove useless words like “the” “and” “it” etc. This will clean our data and make it clearer. Abbreviations can also be removed.
* Spam Ham classifier could be useful
* Text normalisation – remove normalisation
* Stemming – maps words back to the same stem. Wont work for different tenses ie ran.
* Lemmatisation – tries to parse sentence and figure out tense and returns stem.
* <https://towardsdatascience.com/spam-classifier-in-python-from-scratch-27a98ddd8e73>
* <https://kite.com/python/docs/nltk.NaiveBayesClassifier>
* NLtk does a lot of stuff in background

What if we want to change the representation?

Word2vec representation would it be better?

What is the balance of spoiler to non-spoiler?

What is the right performance metric to use? Accuracy or something else?

What tokenizer is nltk using and is it working well.

What is naïve bayes doing in the background??

Learn NLTK

Really understand data

Change representations ie bag of words , word2vec

BAG of words is the easiest representation.

Text processing models called transformers – using bert.

Is it really getting 80%?

Getting naïve bayes to work to word2vec is really difficult . One is a vector one is probability.

Is it naïve bayes classifier and what does it do

Is your data balanced

* Way more non- spoilers than spoilers.
* Have to balance out data , possibly remove some none spoiler entries.
* Spoilers – 150924
* None – Spoilers – 422989
* Not a bad thing 3:1 isnt too bad
* Possibly do under sampling to use small amount of data set to balance the sample
* Having a balanced sample can be a good thing
* Careful of changing Data and make sure to go back to original data.

Questions

What is the distribution of the lengths of document, on average are they long or short?

Distribution of vocabulary, most frequent words? Big vocab small vocab? Unique words?

Stop words, stop word removers.

Nice description of my data set, make sure you understand data set.

Try out different models and see how well they work ?

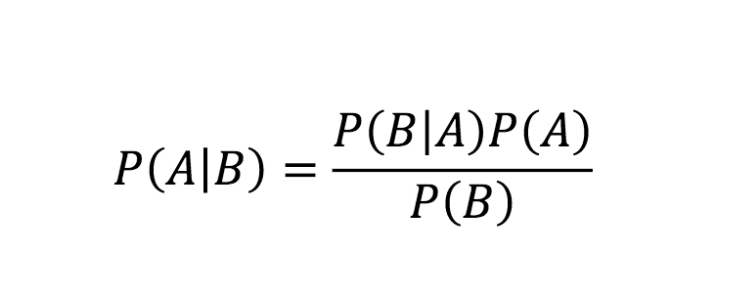
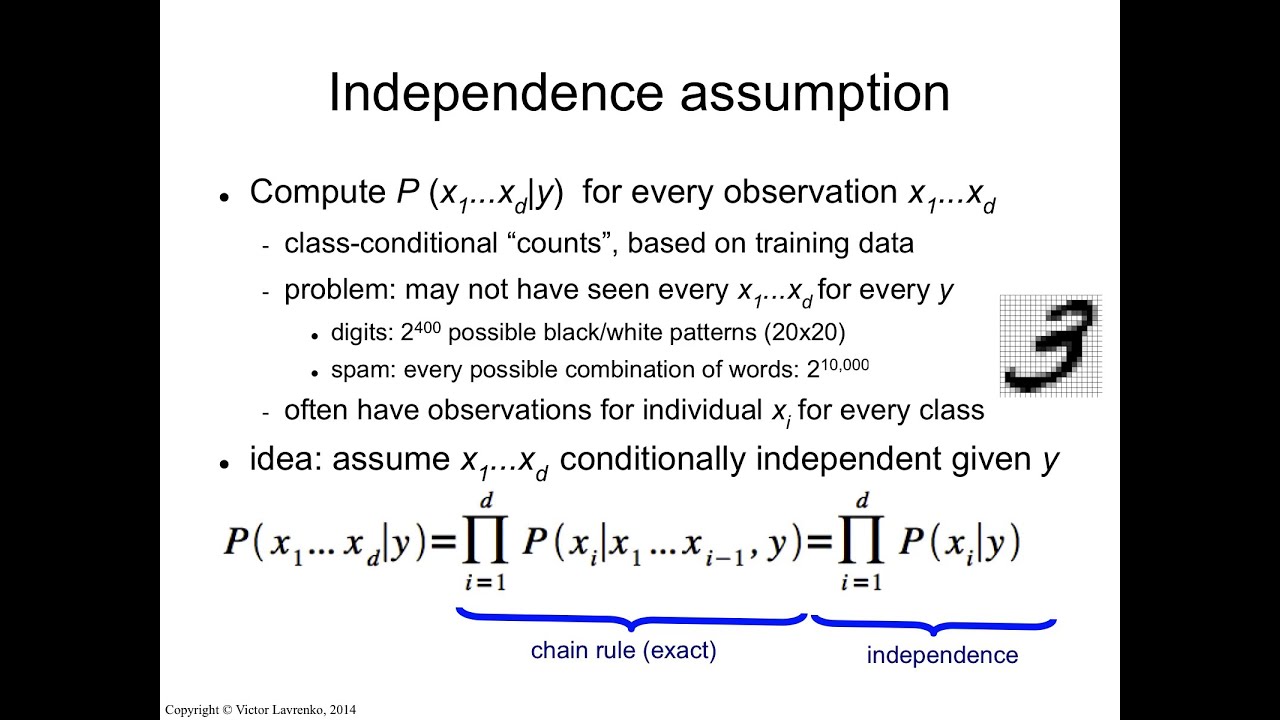
Data Exploration ???

Introduction and Recap

Wikipedia defines a spoiler as “*an element of a disseminated summary or description of any piece of fiction that reveals any plot elements which threaten to give away important details. Typically, the details of the conclusion of the plot, including the climax and ending, are especially regarded as****spoiler****material*” [2]. A forum on quora.com claims that “*A spoiler in a movie is something which ruins the suspense, thrill provocation, imagination, or excitement that you may otherwise experience* [3]. In today’s world, there are many different interpretations of a term. It is important to pinpoint what definition of “spoiler” the project will be focusing in on. My interpretation of spoiler in text Is “a piece of text which gives away important moments of a piece of media”. I aim to create a solution to this - an app that will remove spoilers.

The goal is to develop a piece of software that can hide spoilers on the internet. The idea behind “Spoilers Begone” is to detect a spoiler and to highlight and remove them so consumers can avoid shows being ruined for them. Recognising that a piece of text is a spoiler can be considered a supervised learning problem.

As expressed previously a movie spoiler is “an element of a disseminated summary or description of any piece of fiction that reveals any plot elements which threaten to give away important details”. Movie spoilers are now amongst public enemies’ as many people claim that they ruin the excitement, exhilaration and anticipation of films for them. With an increasingly large number of consumers discussing pieces of media while the event is live there is an imperative need to remove spoilers from the internet. There is no doubt that it is conceivable to remove movie spoilers from the internet. What is more important is how this is done. Removing movie spoilers from the internet or just removing content from the internet in general will include supervised machine learning techniques. Supervised learning is where you have input variables and output variables and you use an algorithm to learn the mapping function from the input to the output. In our case, the spoiler can be considered the input and our database of spoilers is the output. My approach can be considered an amalgamation of supervised machine learning and text classification techniques. We need to be able to classify a statement as a spoiler for this system to work correctly. I believe that one of the most efficient ways to do this is using a Google Chrome extension to remove spoilers from the internet, adding to the functionality of Google Chrome. This way, there is no need for downloading an entirely new platform and integrating it with social networking applications. I was sceptical about choosing this topic as I was unsure about the number of available sources and information surrounding it. Contrarily, it is clear that it is a major contemporary problem and there have been many attempts to solve it. With the constant improvement in technology and the rise of social media usage we are always at risk of stumbling across a spoiler or simply information you would have preferred to avoid. An interesting observation I have made is that people tend to enjoy discussing the piece of media, event or movie online rather than actually enjoying the content. The age of what has been dubbed the “second screen” is truly here.

As mentioned, I planned to use the nltk python libraries which can help me segregate spoilers from non-spoilers. The classifier I am using is the Naïve Bayes classifier. This classifier is based on the Naïve Bayes algorithm. The algorithm uses the Bayes rule to find the probability for a label:

The Naïve Bayes algorithm then makes the “naïve “assumption that all features are independent:

Instead of actually clearly calculating the features, the algorithm calculates the numerator for each label and normalizes them so they sum to one.

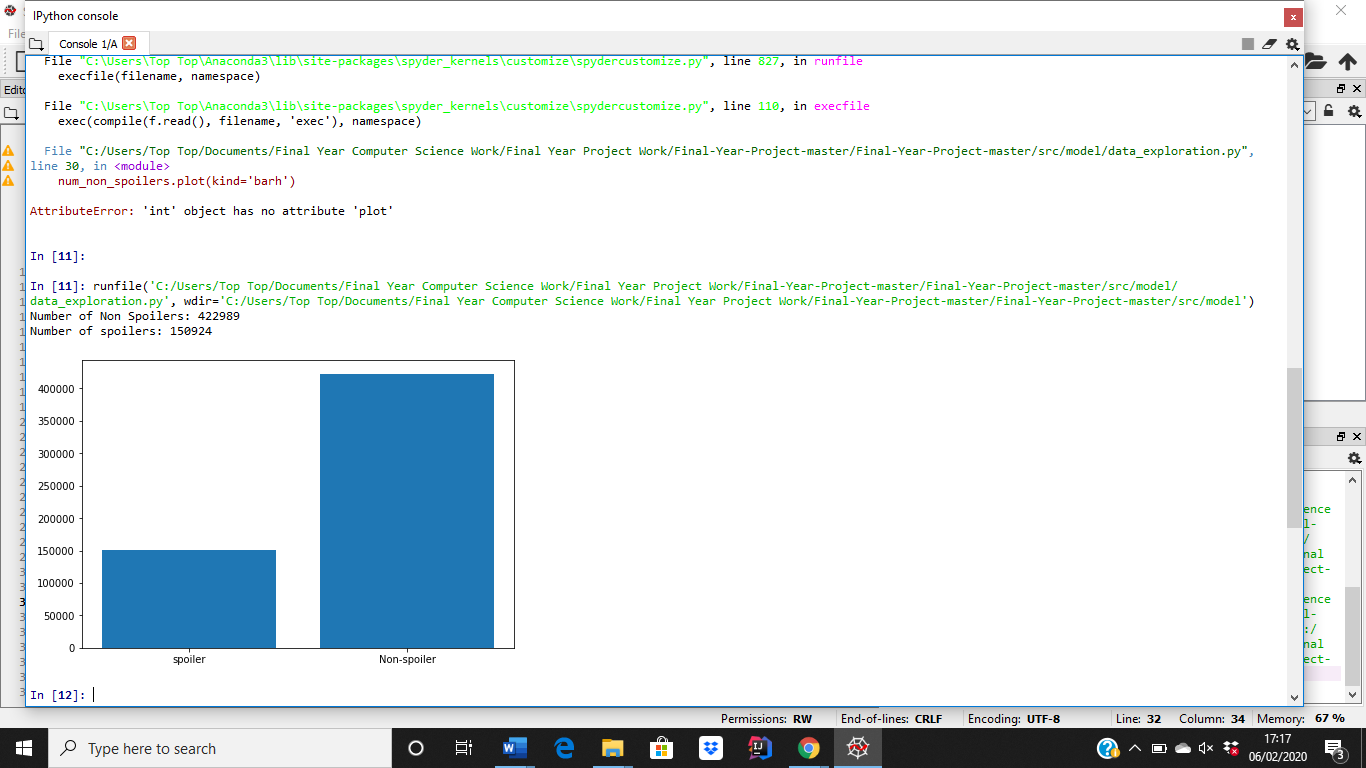
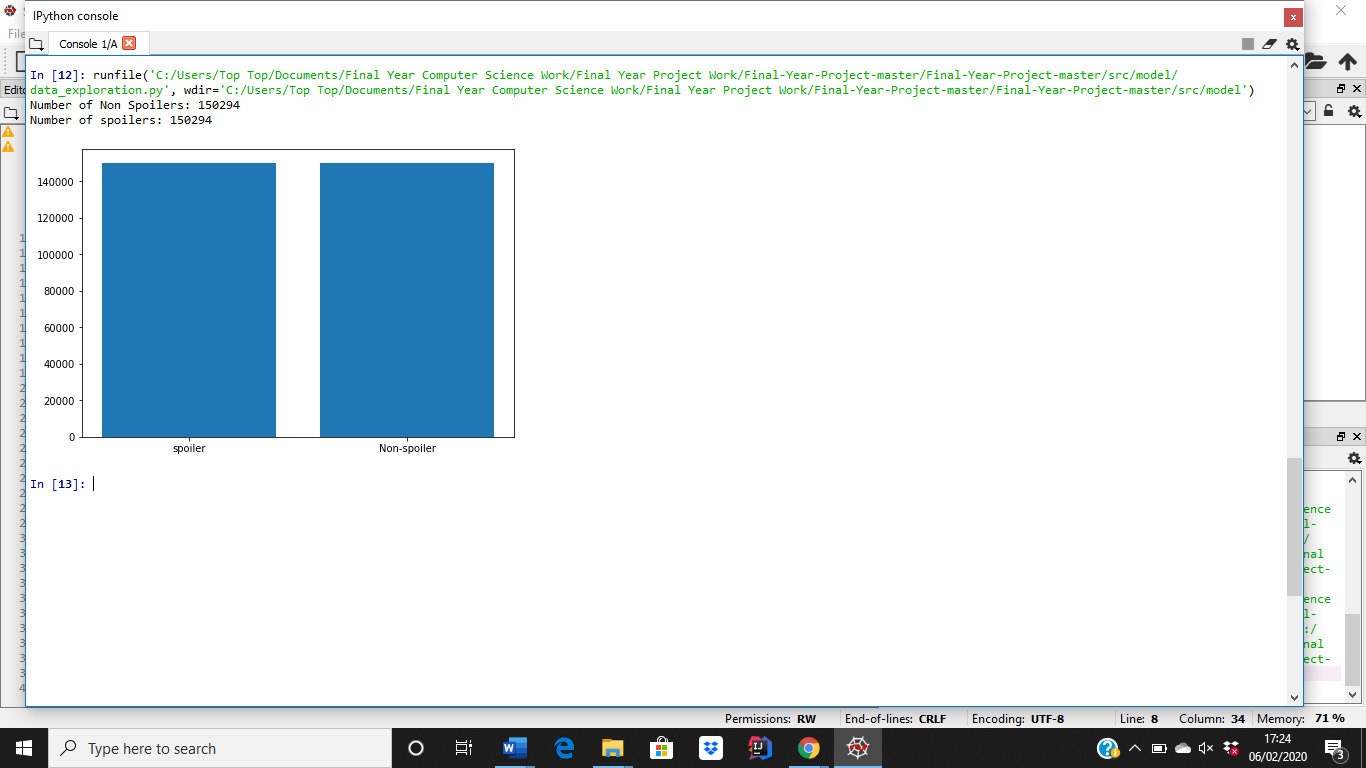
Most Machine Learning algorithms begin with text representation. How will the text from the data set be represented? Will it be represented numerically, Bag of Words, & Word embedding are common ways that the text can be represented. To work around this, I decided to create a small data structure for storing my training data. My data structure contains three items. Review text and “is\_ spoiler”. The review text just gives us the text in the paragraph. This will be key for training as our model can learn the key components that lead to the review text being a spoiler. The “is\_spoiler” section is self-explanatory as it shows us whether the entry is a spoiler or not. These two entries in my data structure are represented as text. I choose this method because I know Naïve Bayes is very good for text classification problems and the accuracy is very high.

Rather down doing traditional text representation techniques I decided to create my own data structure and represent the data as text.

**Data Exploration**

My data set is called: **IMDB\_reviews.json** .. The data set contains the review date, the movie ID, the user ID, rating and a review summary like the last data set. It also tells us who the film was directed by. However, the key attribute in this data set is that it actually tells us whether a spoiler is in the description or not. There is a field called “is\_spoiler”. When “is\_spoiler” is equal to true then we know that there is a spoiler present. When “is\_spoiler” = false, then we know that the description is not a spoiler. This is a simple but efficient way of classifying our data. The fact that the file is in json format makes things easier as there is already an explicit structure to the data. The accuracy of this data seems quite good due to the level of integrity of IMDB. IMDB has strict policies in relation to spoilers and to accurate data. A breach of IMBD policies results in your membership being blocked and posting privileges removed. For this task it is important to have data sets that have spoilers but the format of the movie spoiler is also important. Format refers to how the movie spoiler text is written. The spoilers I am focusing on are usually short pieces of text. Luckily this in this data set, all of the spoilers are no more than a couple of lines – matching the requirement.

When you have a data set it is very important to explore it to really understand the most efficient ways to use it. I previously mentioned that my data set contains two key fields “is\_spoiler = true” and “is\_spoiler\_false”. The first concern I had was figuring out whether my data is balanced or not. Is there a good ratio of spoilers to non- spoilers? This is important as it can have an affect on the performance of my classifier. When exploring I found that there were 422,989 non spoilers and there were 150,924 spoilers. There is roughly a 3:1 ratio of non-spoilers to spoilers. In terms of data this is not a terrible ratio although it could be better. I decided to balance out my data by under sampling. I concluded that I will match the number of spoilers and only consider 150,924 non spoilers. This will then fully balance my data and give me a 1:1 ratio of spoiler to non-spoiler. Balancing my data is important as it reduces the chances of underfitting and coming across false positives. Below are basic visualizations of my data before and after under sampling.

Pre Under Sampling Post Under Sampling

Another important aspect of my project is word frequency/ occurrence. I think it is important to see how many times certain words occur in my data set and to see if these words are more likely to produce spoilers or non spoilers.