3.3 Top K Module

3.3.1 Preprocess and Build Train Data

In classification, different feature has different weight, thus, we need to focus on the key terms and remove the noise, such as escape characters.

Steps:

1. Remove \n, \t
2. Remove Punctuations.
3. Split Words.
4. Remove StopWords.
5. Lemmatize Words.

Implement based on nltk, which is a leading platform for building Python programs to work with human language data.

```
import nltk.data
from nltk.stem import WordNetLemmatizer
```

Figure 5: Import nltk in python

Next, we summarize the statistical information of terms for all cases to create term dictionary. In doing so, all terms are transferred to numeric values, that is easier for further processing.

We use one-hot encoding to represent law cases. In Figure 6, we assume that statistical dictionary of catchwords contains 3 words and statistical information of body sentences contain 5 words.

In one sample case, c1 and c3 exist in catchphrase and w1, w3 and w4 exist in body sentences.

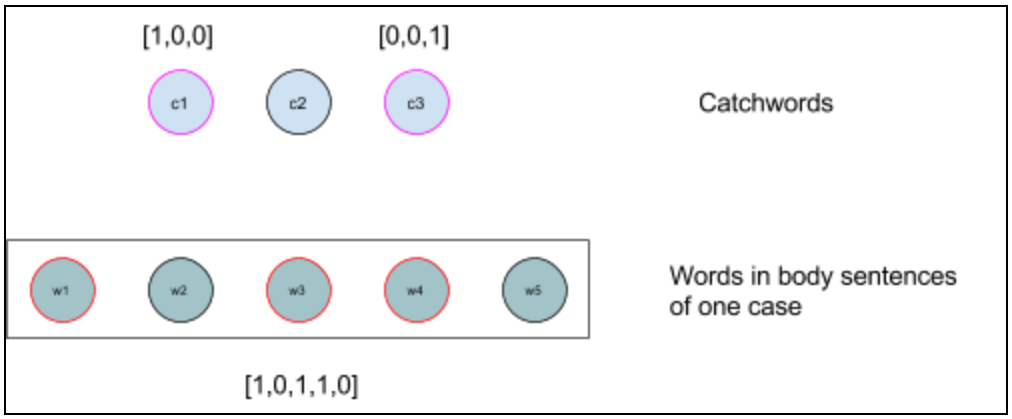

Figure 6: Vector representation of case

Thus, we can create two mapping relations according to f(x) → y as following:

[1, 0, 0] → [1,0,1,1,0]

[0, 0, 1] → [1,0,1,1,0]

We process all cases in this way and finally build the train data in train_data.npz.

3.3.2 Bayesian Training

We assume that, y is a variable with categorical distribution, its conjugate prior is dirichlet distribution.

x is a variable with bernoulli distribution, its conjugate prior is beta distribution.

y denotes catchword and x denotes term in body sentences.

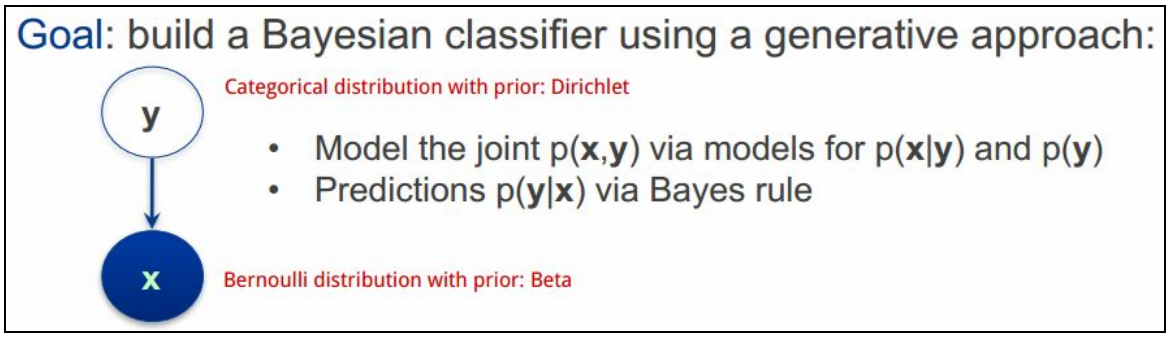

Figure 7: PGM for classifier

We calculate likelihood and get their posteriors.

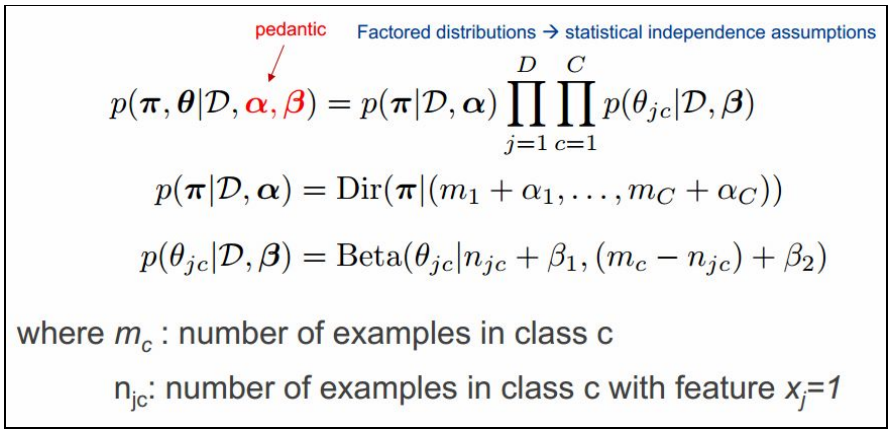

Figure 8: Prior distribution

We can calculate probability of prediction.

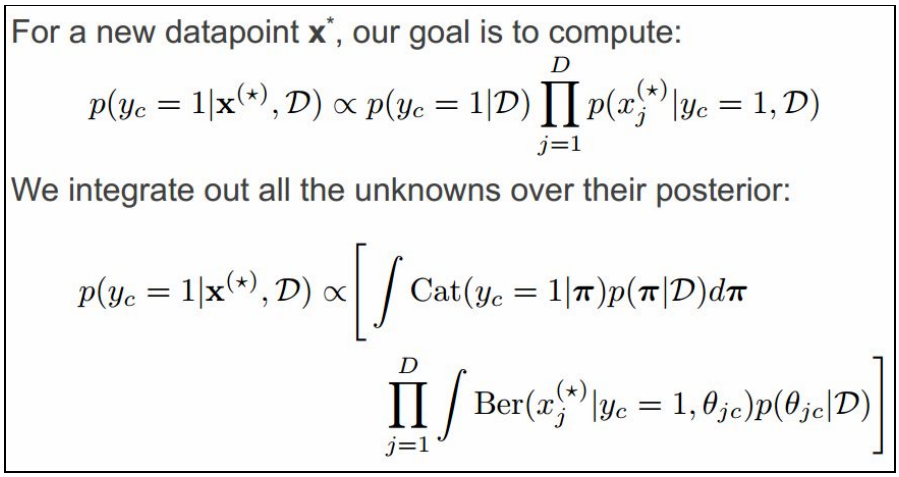

Figure 9: Prediction

Finally, we get an analytical solution which is expectation of posteriors distribution.

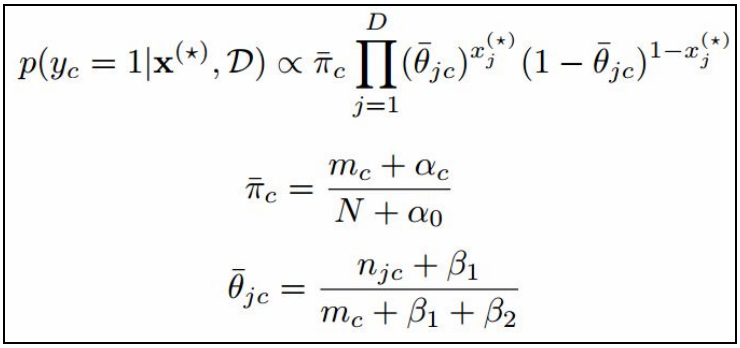

Figure 10: Posterior of parameter estimation

With more data, the parameter will converge to a more accurate result.

3.3.3 Bayesian Prediction

Here, theta is a matrix as following.

j denotes the index in catchword dictionary, and c denotes the index in body word dictionary.

According to the assumption in table 3-1, we can get the result of theta as following:

	c				
j	s11	s12	s13	s14	s15
	s21	s22	s23	s24	s25
	s31	s32	s33	s34	s35

Table 1: Theta demo

In this sample case, for its catchphrase which contains only two catchwords, c1 and c3, one sentence which contains w1, w3 and w4 will get a score s11+s13+s14+s31+33+34.

Next, we calculate all scores for each sentences and sort them. Finally bayesian model returns the score list.

3.3.4 Wordnet Synonyms System

This sub-module takes the un-preprocessed data of a case law document, specifically the extracted catchwords and sentences as input. The output is the score of all sentences in a list format.

This sub-module is suited for prediction on a single document, unlike the Bayesian prediction which trains over several documents.

First step:

The data is pre-processed using the same methods as the preprocessing stage for the Bayesian training, but only for the single relevant document requested by the user. Specifically, it uses the Summarizer class, which uses the Parser class, to handle the parsing of data through pre-processing for the catchwords and all body sentences of one document.

Second step:

Synonyms are taken for the catchwords and sentences in order to increase the probability of matching words. This is necessary due to the lemmatization stage of the pre-processing which previously reduced the word set. All synonyms are given an equal weighting to the original word due to to the above reason.

The wordnet.synsets() function from the nltk.corpus package for python is used to extract synonyms for each word. Wordnet also has a synonym similarity scoring feature, wordnet.wup_similarity(), but this was not implemented to eliminate the need to keep all synonyms for all words separately, and reduce the processing cost for speed purposes.

The scoring system for the Wordnet prediction sub-module is one point per word match between the catchphrase synonyms set and each sentence's synonyms set, with multiples of each matched word directly increasing the score in a linear 1:1 ratio.

The advantages of this method are an increased probability of matching for each sentence, hence reducing the probability of tied scores for all sentences.This is especially important for the smaller case documents which would otherwise need to separate their ranking for tied sentences by sentence order in the document.

The disadvantages include excessive matching of synonyms, especially as there are a variable number of synonyms from each word when using the Wordnet API. This may overly boost a sentence score beyond a more suitable sentence. However, as the words are lemmatized, this is likely not a large source of error in accuracy of sentence rankings.

Performance:

As the Wordnet.synsets() function is run for each word of the catchwords and all sentences, there is a small performance cost involved. This is minimised with the preprocessing stage which reduces the number of words to find synonyms for, and eliminating duplicate synonym searches by keeping a count for each word.

The preprocessing for the single document is the majority of the performance cost, however is negligible as the same pre-processing functions are run in the Bayesian prediction sub-module (hence kept in memory) that is run before this sub-module in the mixture model below.

3.3.5 Mixture Model

The mixture model takes a case law document url as an input, and outputs an ordered ranked list of all the sentences for use by the highlighter module.

It retrieves the catchwords and body sentences using the data cleaning module, and runs the Bayesian and Wordnet synonyms prediction sub-modules.

The normalised scores from each method are combined using a mixture model:

*Mixture Model = alpha * Bayesian Model + (1-alpha) * Wordnet Model*

Where alpha is set to 0.25 to lower the weighting on the comparatively less accurate Bayesian approach. The following graph shows the normalised scores for the Bayesian, Wordnet and Mixture models for the case law document found at http://www.austlii.edu.au/cgi-bin/viewdoc/au/cases/nsw/NSWCATCD/2016/63.html:

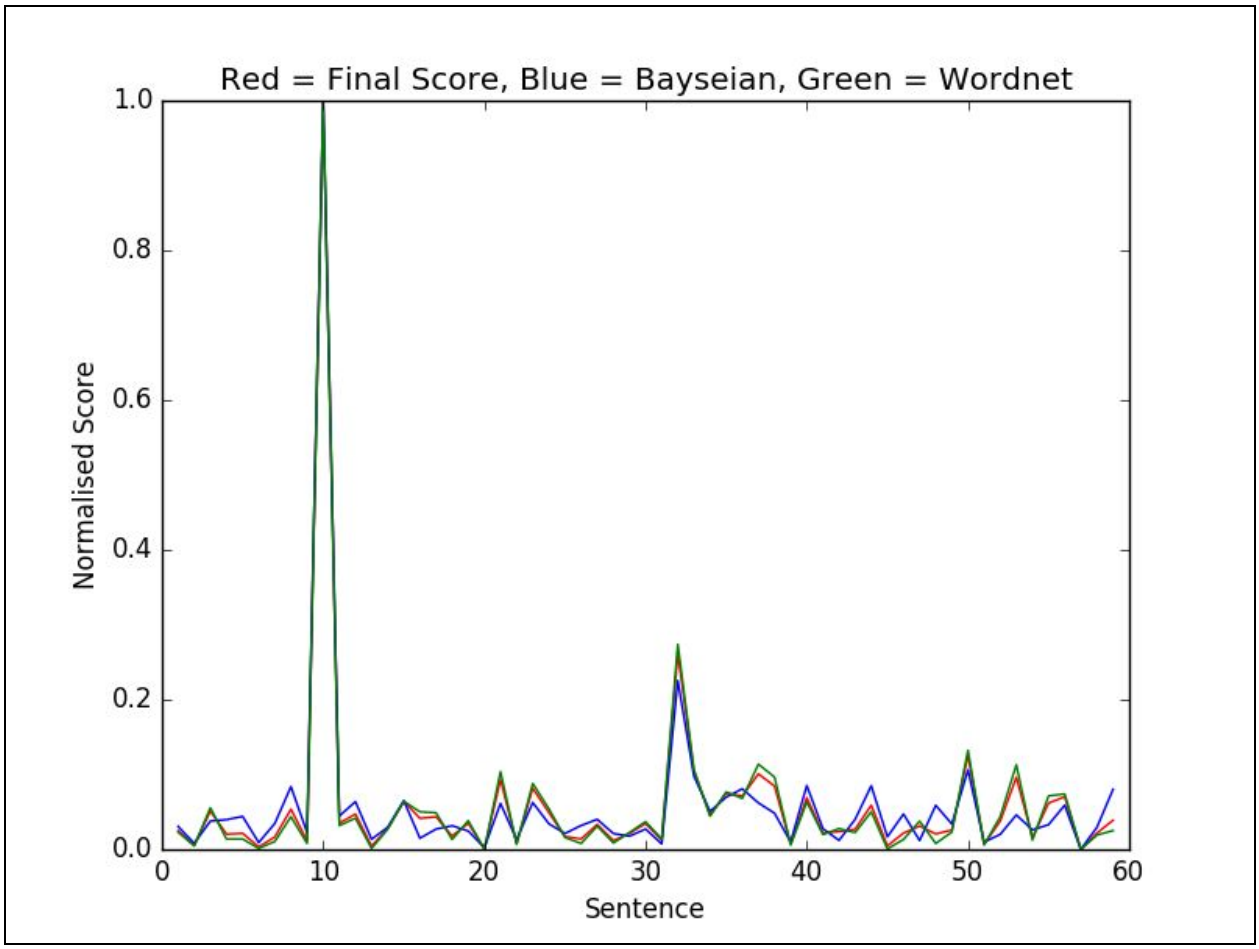

Figure 11: Mixture performance

This result is from a training data set of 10 case law documents, indicating the potential of the Bayesian predictor.

The combined score from the mixture model is used to generate a list ordered by the ranking of each sentence.