3.1 Introduction

In this Chapter the architecture and implementation for two important focused crawling techniques is given. The first focused crawler makes use of Breadth First approach after applying Naive Bayes classification technique. The second focused crawler follows Best First approach after applying Naive Bayes classification technique. All the prerequisite for both the crawlers are given along with theirs structure and the way of use. The Chapter is concluded by showing the crawling results for both of the crawlers for retrieving the Web pages.

3.2 Naive Bayes

Naive Bayes Classifier is derived from Bayesian theorem. It is used for classifying the input patterns having large number of characteristics. To understand the concept of Naive Bayes Classification, consider the problem of classification of patterns representing feature of a mango, as shown in Fig. 3.1, in the Ripe (R) class or Not Ripe (NR) class. The aim is to classify a new mango pattern into the R or NR class i.e. to classify the pattern to a class to which it belongs by analyzing the already classified mango patterns.

The Fig. 3.1 shows that the number of patterns related to R are double than the number of patterns falling in NR class. Both of these patterns are being used by the classifier for training purposes.
It can also be deduced that the probability of a new pattern belonging to R class is double than the probability of its belongingness to NR class. This probability in Bayesian theorem is called as the prior probability. The prior probability is used to predict the outcome of an event before its happening.

The Prior Probability for belongingness to R is given by:

$$\frac{\text{Training patterns falling in R class}}{\text{Total number of patterns}}$$

(3.1)

The Prior Probability for belongingness to NR is given by:

$$\frac{\text{Training patterns falling in NR class}}{\text{Total number of patterns}}$$

(3.2)
As there are a total of 30 training patterns, out of which 20 belongs to R and 10 belongs to NR the prior probabilities for the classes can be given as:

\[
\text{Prior Probability for belongingness to } R = \frac{20}{30}
\]

(3.3)

\[
\text{Prior Probability for belongingness to } NR = \frac{10}{30}
\]

(3.4)

After calculating the prior probabilities, the class of an unseen pattern can be deduced. Let \( O \) denotes the pattern to be classified in Fig. 3.2.

![Mango Classification using Naïve Bayes Illustration (★ denoting the Ripe and △ denoting the Not Ripe class members and O denoting pattern to be classified)"

Fig. 3.2: Mango Classification using Naïve Bayes Illustration (★ denoting the Ripe and △ denoting the Not Ripe class members and O denoting pattern to be classified)
In Fig 3.2, all the patterns except the pattern to be classified are already clustered. The new pattern seems to belong with a greater degree to the class having closer clusters to the pattern than the class having far away clusters from the pattern. For measuring the relatedness, a circle is drawn across the pattern to be classified. Then the number of a particular class patterns coming inside the circle shows its belongingness to that class. Hence it can be derived from here that

\[
\text{Probability of O given } R = \frac{\text{Number of } R \text{ patterns enclosed in the circle}}{\text{Total number of patterns in } R}
\]  
\[
(3.5)
\]

\[
\text{Probability of O given NR} = \frac{\text{Number of NR patterns enclosed in the circle}}{\text{Total number of patterns in NR}}
\]

\[
(3.6)
\]

As shown by the Fig.3.2, 3 patterns belonging to R class and 4 patterns belonging to NR class appear in the circle. Hence

\[
\text{Probability of O given } R = \frac{3}{20}
\]  
\[
(3.7)
\]

\[
\text{Probability of O given NR} = \frac{4}{10}
\]  
\[
(3.8)
\]

According to Equation 3.3, O belongs to class R, but by Equation 3.8, O belongs to class NR. The Bayesian analysis makes use of a combination of both of these probabilities to deduce the class of a new pattern using Bayes rule [122].
Then the Posterior probability of O being R is:

\[ \text{Prior Probability of R} \times \text{Probability of O given R} \]

(3.9)

The Posterior probability of O being NR is:

\[ \text{Prior Probability of NR} \times \text{Probability of O given NR} \]

(3.10)

Posterior probability of O being R comes out to be \( \frac{1}{10} \) while the posterior probability of O being NR comes out to be \( \frac{4}{10} \) by the using above equations.

So the pattern O is classified to NR class. The Naive Bayes Classifier can be used to handle a large number of features.

If \( \{x_1, x_2, x_3, \ldots, x_d\} \) is the set of features constituting the pattern \( X \). Pattern \( X \) is to be classified in one of the \( n \) classes, \( \{C_1, C_2, C_3, \ldots, C_n\} \) then the posterior probability is given by the Bayes rule

\[ P(C_j|x_1, x_2, x_3, \ldots, x_d) \propto P(x_1, x_2, x_3, \ldots, x_d|C_j) \times P(C_j) \]

(3.11)

Where \( P(C_j|x_1, x_2, x_3, \ldots, x_d) \) is probability that the pattern \( X \) belongs to \( C_j \) class. As according to the Naive Bayes the conditional probability of \( X \) belonging to a class is given by

\[ P(X|C_j) \propto \prod_{k=1}^{d} P(x_k|C_j) \]

(3.12)
and the final posterior probability can be written as

\[ P(C_j|X) \propto P(C_j) \cdot \prod_{k=1}^{d} P(x_k|C_j) \]

(3.13)

The pattern \( X \) is classified to the class \( C_j \) having highest probability value.

---

**Algorithm 3.1: Naive Bayes Training**

**Input:** Collection \( D \) containing \( D_R \), set of domain related pages, and \( D_{NR} \), set of pages not related to the domain.

**Output:** Probability of each term’s belonging to the related or non-related class.

1. Let \( D \) denotes the collection of domain related and non domain related documents.
2. \( D_R \) and \( D_{NR} \) are collection of documents which are related to the domain and which are not related to the domain respectively, and both of these are subsets of \( D \), and let \( T_R \) and \( T_{NR} \) be the concatenation of documents in \( D_R \) and \( D_{NR} \) respectively.
3. Let \( V \) denotes the vocabulary of all words in \( D \).
4. Calculate probabilities
   \[ P(C_i) = \frac{|D_i|}{|D|} \text{ for } i = R \text{ and } NR \text{ (Related and Non Related).} \]
5. Set \( N_R = |T_R| \) and \( N_{NR} = |T_{NR}| \).
6. For each word \( W_j \in V \)
   a. Set \( n_{ij} \) to the number of occurrences of \( W_j \) in \( T_i \) for \( i = R \) and \( NR \).
   b. Calculate \( P(W_j|C_i) = \frac{n_{ij} + 1}{(n_i + |V|)} \text{ for } i = R \text{ and } NR. \)
3.3 Naive Bayes Training

Naive Bayes classifier considers that the existence (or non-existence) of a specific feature of a class is unrelated to the existence (or non-existence) of any other feature, given the class variable.

A mango can be called as ripe if it is pulpy and its color is yellow. Even if these features depend on each other or upon the existence of the other features, a Naive Bayes classifier considers all of these properties to independently contribute to the probability that this mango is ripe. Naive Bayes classifier can be trained in a supervised learning setting. The Naive Bayes is trained as according to the Algorithm 3.1. The training algorithm works upon the set of pages D containing DR, the set of domain related pages from ODP, and DNR, the set of pages not related to the domain from ODP. Step 2 of the algorithm concatenates the pages in DR to form TR, and the pages in DNR to form TNR. In Step 3 a vocabulary of all the documents in D is created and named as V i.e. the unique words from DR and DNR forms V. Step 4 calculates the prior probability of the Relevant and Irrelevant class. Step 6 calculates the probabilities for each word of the vocabulary to exist in the Relevant or Irrelevant class, which is then used to calculate the posterior probabilities.

3.4 Naive Bayes Classifier

The Naive Bayes classification algorithm works by deducing the class(relevant or irrelevant) of document X, by using the probability of a
random page belonging to relevant or irrelevant class and then multiplying this probability by sum of the probabilities of each word in X to relevant and irrelevant class respectively. The Naive Bayes classifier functions as according to Algorithm 3.2. The algorithm takes the document X as input. The number of words in X is determined and initialized to n in Step 1. The class of X is determined in Step 2 by multiplying the prior probabilities with the sum of the probabilities of each word to belong to the individual class.

Algorithm 3.2: Naive Bayes Classifier

**Input:** Text document X  
**Output:** Class of the document (whether related or not related to the domain)

1. Let n be the number of word occurrences in X.
2. Calculate class
   \[ \text{Class}_i = P(C_i) \times \prod_{j=1}^{n} P(a_j|C_i) \]
   \[ i \in \{R, NR\}, \text{where } a_j \text{ is the word occurring at } j^{\text{th}} \text{ position in } X. \]
3. If \( \text{Class}_R > \text{Class}_{NR} \) then return Related else return Not Related.

Step 3 derives the class by comparing the posterior probability of the two classes, and then classifying the X to the class having greater probability.

Based upon the Naive Bayes classifier, two crawlers one based upon the breadth first approach and other based upon the best first approach are developed.
3.5 Naive Bayes Breadth First Crawler

The architecture of a Naïve Bayes Breadth First Crawler is given in Fig. 3.3. The crawler first trains the Naïve Bayes using the set of seed pages coming from ODP as relevant pages and another set of pages from ODP as non relevant pages. These sets are used to train the Naïve Bayes classifier using Algorithm 3.1. The same set of seed pages is passed to the next module for pre-processing. After training, the Naïve Bayes is used to classify a page either in the relevant or irrelevant class. The relevant page is further explored for possible presence of related pages while non-relevant pages are discarded at that time only.

All the links contained within a relevant page are inserted at the end of the crawler queue. The whole process is repeated until the crawler queue is empty or the crawler limit is reached. The crawling process for breadth first approach is given as Algorithm 3.3. The algorithm works by initializing the set of seed pages or the set of relevant class pages from ODP in Step 1. Step 2 generates the set of URL which are irrelevant to the topic to be crawled from ODP.

The Naïve Bayes is trained using Step 3 using set of relevant pages and set of irrelevant pages. The $tf - idf$ scores for the collection of downloaded seed pages are generated in Step 4. Step 5 generates the VWV table which is going to be used for calculating the similarity score of the resultant pages.
Naive Bayes Classification Based Focused Crawlers

Fig. 3.3: Architecture for Naive Bayes breadth first crawler

All the seed URLs are inserted in the Crawler Queue by Step 6 of the algorithm. The crawling loop starts from Step 7 where the links from the downloaded page are inserted at the end of the Crawler queue, and also the similarity score for the downloaded contents is calculated and the page is inserted in the results database.

79
Algorithm 3.3: Naive Bayes Breadth First Crawler

Input: Domain topic and Open Directory Project (ODP) database.

Output: The set of pages retrieved by the crawler

1. Initialize the seed pages from the Open Directory Project (ODP) by passing the topic to be crawled to Algorithm 2.1 from Chapter 2.
2. Initialize the set of irrelevant pages from the Open Directory Project (ODP) by passing a list of topics, other than the topic to be crawled, to Algorithm 2.1 from Chapter 2.
3. Train the Naive Bayes, by using the output from Step 1 as the set of Relevant class pages and the output from Step 2 as the set of Irrelevant class pages, by using Algorithm 3.1.
4. Calculate tf – idf score for each term of the corpus of the pre-processed contents of seed pages using Equation 2.1 and Equation 2.2 of Chapter 2.
5. Generate VWV table using Algorithm 2.2 of Chapter 2.
6. Insert all the seed URLs into the Crawler Queue.
7. While Crawler Queue is not Empty or the crawler limit is not reached
   a. Pick the URL from the Crawler Queue.
   b. Fetch the URL from the Web.
   c. Deduce the class of the page by using Algorithm 3.2.
   d. If (page belongs to the Relevant class)
      i. Extract the links contained within document and insert them at the end of the Crawler Queue.
      ii. Calculate the page contents similarity score from VWV table.
      iii. Insert the page along with its similarity score into the result database.
3.6 Naive Bayes Best First Crawler

The architecture of a Naive Bayes Breadth First Crawler is given in Fig. 3.4. The crawler first trains the Naïve Bayes using the set of seed pages coming from ODP as relevant pages and another set of pages from ODP as irrelevant pages. These sets are used to train the Naive Bayes classifier using Algorithm 3.1. The same set of seed pages is passed to the next module for pre-processing. After training, Naïve Bayes is used to classify a page either in the relevant or irrelevant class. The seed URLs are inserted into the Crawler Queue in the order of their similarity score. The URL having the highest similarity score is fetched from the Web and its class is driven from the Naive Bayes.

The relevant page is further explored for possible presence of related pages while irrelevant page is discarded at that time only. A similarity score for the links contained within the relevant page is calculated by considering the parents similarity and the anchor text similarity score. All the links are inserted as according to their score’s position in the Crawler Queue. The crawling method for best first approach is given as Algorithm 3.4.

The algorithm works by initializing the set of seed pages or the set of relevant class pages from ODP in Step 1. Step 2 generates the set of URLs which are irrelevant to the topic to be crawled from ODP. The Naive Bayes is trained in Step 3 using set of relevant pages and set of irrelevant pages. The $tf-idf$ scores for the collection of downloaded seed pages are
generated in Step 4. Step 5 generates the VWV table which is to be used for calculating the similarity score of the resultant pages. The similarity score for the contents of the seed pages is calculated using Step 6. All the seed URLs are inserted into the Crawler Queue in the order of their similarity score, calculated in the previous step, using Step 7. The crawling loop starts from Step 8.
Algorithm 3.4: Naive Bayes Best First Crawler

Input: Domain topic and Open Directory Project (ODP) database.
Output: The set of pages retrieved by the crawler

1. Initialize the seed pages from the Open Directory Project (ODP) by passing the topic to be crawled to Algorithm 2.1 from Chapter 2.
2. Initialize the set of irrelevant pages from the Open Directory Project (ODP) by passing a list of topics, other than the topic to be crawled, to Algorithm 2.1 from Chapter 2.
3. Train the Naive Bayes by using the output from Step 1 as the set Relevant class pages, and the output from Step 2 as the set of Irrelevant class pages, by using Algorithm 3.1.
4. Calculate tf – idf score for each term of the corpus of the pre-processed contents of seed pages using Equation 2.1 and Equation 2.2 from Chapter 2.
5. Generate VWV table using Algorithm 2.2 from Chapter 2.
6. Calculate the similarity score for contents of the page corresponding to each seed URL.
7. Insert all the seed URLs into the Crawler Queue as according to their content’s similarity score from VWV table.
8. While Crawler Queue, which is a priority queue, is not Empty or the crawler limit is not reached
   a. Retrieve the topmost URL from the Crawler Queue, and fetch it from the Web.
   b. Deduce the class of the page by using Algorithm 3.2.
   c. If (The page belongs to the Relevant class)
      i. Insert the page itself into the result database with its VWV similarity score.
      ii. For each link present in the page calculate the similarity score as
          \[
          \frac{\text{parent.content.similarity}}{7} + \text{link.anchor.similarity}.
          \]
      iii. Insert the links into the Crawler Queue as according to their score’s location.
The topmost URL from the Crawler Queue is fetched and downloaded from the Web. The class of the downloaded page is predicted by using the Naïve Bayse. If the class of the page comes out to be relevant then the page is further explored for possible presence of other relevant pages from its links. The links are assigned a similarity score derived from VWV table. All the links are inserted in the Crawler Queue as according to the position of their scores. The crawling loop re-iterates until the Crawler Queue is not empty or the crawling limit is not reached.

3.7 Results

Both the crawlers are implemented in Java using a Windows 7 setup on an Intel(R) Core(TM) i5 Processor 3.20 GHz and 4.00 GB of RAM, and using MySQL as the backend database. An automated process for gathering seed pages is used. Open Directory Project (ODP) provides the categorical collection of URLs that are manually edited and not biased by any commercial user. From here one can find individual category links. One can find maximum number of URLs related to the concerned topic by just browsing the http://www.dmoz.org from root to the topic of interest.

For analysis and other purposes http://www.dmoz.org also provides the whole database free of cost, which can be used as according to the needs of the individual. This database was downloaded and its contents were extracted to MySQL using PERL (Practical Extraction and Report Language) scripts.
The URLs belonging to the categories which are pointing to "Animation", "Cricket", "Science" and "Computer" at the last level were retrieved. A total of 323, 199, 250 and 563 such URLs were found through a module written in Java for each of the "Animation", "Cricket", "Science" and "Computer" domain respectively. The set of these URLs acted as the initial seed pages for all the crawlers. The precision graphs for different similarity values for different topic domains are shown from Fig. 3.5 to Fig. 3.8.

It is evident to say from the graphs that the Naive Bayes Best First Crawler dominates the Naive Bayes Breadth First Crawler in terms of finding the quality pages. This dominance may be due to the fact that the possibility of reaching to the quality pages from among the best of quality pages is more than just any randomly selected quality pages.

Fig. 3.5: Precision graph for "Animation" domain
Fig. 3.6: Precision graph for "Computer" domain

Fig. 3.7: Precision graph for "Science" domain
3.8 Conclusion

In this chapter the architecture and implementation details for two crawlers based upon Naive Bayes classification are given. Algorithms for Naive Bayes training and classification for the contents using the trained Naive Bayes for Web pages are presented. The experimental results showing the precision values for the two crawlers were shown.