WOOF: user defined news recommendation system

Ruisheng Shi(rshi7) Alexander Daniel Hadiwijaya(hadiwij2)

Fan Yang(fanyang5) Ruogu Zeng(rzeng4)
Lucas Qiu(shiqiu2) Ajay Nair(anair10)
Son Nguyen(nguyenb1) Sharu Jiang(sjiang19)
Victor Rodrigues(silvaro2) Paul Vijayakumar(pvijaya2)

ABSTRACT

Project WOOF! incorporates a personalized news recommendation system and increases users satisfaction rate. The objective of the system is to independently crawl news websites, filter out redundant news, classify the news into system generated and user specific categories and provide the user with a short summary of the news. The system also would incorporate collaborative filter techniques that can be captured using implicit user actions on the website. By crawling the 8 major news websites from March 13th to May 9th, WOOF has collected more the 12,000 cutting edges news from all aspect of the world. Our works are based on these news.

1. INTRODUCTION

The satisfaction rate for all the news and feeds we see everyday is incredibly low. Many news sites which we subscribe to present us news that would interest the general public may not interest us. A lot of the news we explore or search also tend to be repetitive and redundant. On the other hand, there are some information or news we are really interested in that are not seen by us. Our ultimate goal is to implement an effective user-focused news portal such that the satisfaction rate for the news presented to a user is increased to at least 50%.

Woof! News Retrieval and Personalized News Propagation Website (woofuiuc.com) pre-processes news streams from RSS feeds, extracts useful features from text and filters redundant news. WOOF! can efficiently summarize important ideas in news extracts for users review of content, cluster news articles in 66 clusters of roughly distributed documents, and learn from user's article selections and ratings to model and propagate personalized news to the user.

Compared to some similar news websites such as Stumbleupon.com, Google News, and Newsisfree.com, Woof! is real-time and relevant, allows direct search by keyword or category. Woof! provides personalized recommendations that meet user's need and interest, short summarization and near duplicate removal that saves user's time, and of course, wonderful user-friendly interface.

We are using the ACM alternative template. The sections are planned as following: section 1 will give a introduction of the problem; section 2 gives the designing of the WOOF; section 3 will focus on the previous existing works. Section 4 is about the Survey. Section 5 is about crawling method and our crawled data. Section 6 gives the report of our implementation. Section 7 gives the result evaluation. section 8 gives introduction about the future work. Section 9 contains the individual contribution
.

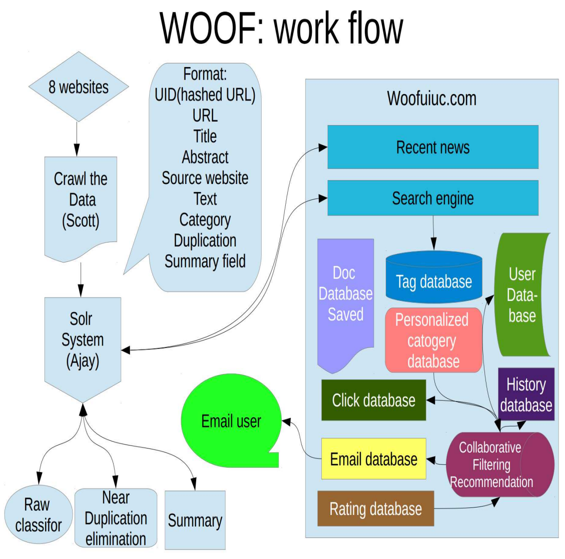


Figure 1: Workflow design for project WOOF

2. DESIGN OF THE WOOF

The diagram of the pipeline is shown in Figure 1. We firstly crawl real-time news from News sites RSS Feed (CNN, Reuters, BBC etc), then filter out near-duplicate news from the various sources, classify news into subcategories and generates summarizations for each news article per subcategory, finally data would be indexed in Solr and used by the User Component. In user component, we will pop the recent news every time the user visits the website. Then by creating a history database to record what user has viewed; rating database to record user's rating for each article and click database to record user's click through, we collecting and grouping users and news information and feed this information to a hybrid of a content-based and collaborative filtering recommendation system. The system will rank a list of article to each user and propagate it to them through the news portal or through our email delivery service.

3. RELATED WORK

In this section, we will list several websites provide similar service and give the PROs and CONs analysis for each.

Stumbleupon `http://www.stumbleupon.com`
PROS: Stumbleupon provides user an option to select topic they interested, then based on this topic, the system will recommend user related news.
CONS: Stumbleupon's system is linear and out of date. Also it is for amusement. We will focus on text. Focus on more 'important' and 'informative' information.

Google news `https://news.google.com/`
PROS: google news has millions of users, and they support customized preference. User can create label themselves
CONS: The preference is not really working well. After I select the only machine learning and data mining. The news are no significant different from the general news.

NewsIsFree `http://www.newsisfree.com/`
PROS: this website provides an amazing amount of accurate news. The system also keep track of the news user has viewed.
CONS: You have to search it instead. It is not a push system. Users are lazy. They don't have an automatic summary system

Newzpile `http://newzpile.com/`
PROS: they provides a crawling of lots of websites
CONS: they only keep a headline (title) of the news. Also, the UI is not user friendly.

Time `http://www.time.com/time/`
PROS: A bit more specific in categories, have some subcategories as well.
CONS: there are some unrelated post going to wrong category.

BrainPicking `http://www.brainpickings.org/`
PROS: all articles are selected by an expert reader. Choose the categories you are interested in, related articles will be pushed to your mailbox.
CONS: It is selected by human, limited by humanâĂŹs abilities.

4. SURVEY

4.1 Near Duplicate elimination

We looked at many different methods to be able to perform the features effectively. We tested some popular implementations with modifications tailored to our need. Near duplication removal can be done in many ways.But after a review of a number of papers we selected an approach utilizing named entities in the article text to identify and filter out the near-duplicate articles[11]. Named Entities, or Proper Nouns were the best representation of a unique document and we intended to utilize a vector-space model of these to determine a similiarity between two documents. The reason they are helpful in duplicate detection is that it gives a significant low similarity, close to 0 percent error rate amongst two unrelated news, similar to how stemming removes meaningless words.

After running some tests of our model for near-duplicate detection we ran some tests on some common methods to determine similarity such as the Jaccard Coefficient, Dot Product and Cosine Similiarity.

Utilizing the Named Entity model, we discovered that it is not very good at differentiating semantically sensitive news, or news that is talking about the same subject but in a different context or perspective. Therefore, we decided to apply either a Hashing or Shingling algorithm to identity the articles with the closest matching patterns of text.

Hashing, for this purpose, is essentially compute a fingerprint for each document by using locality-sensitive hashing. Then we look at the difference in their fingerprints to be able to tell how different/similar they are. The Hashing algorithms experiment with are the minHash and Charikar's hash algorithms.

Shingling, which is essentially an algorithm that computes similarity by comparing the set of k-consecutive terms of characters of documents.

In the end, we chose shingling to integrate with our Named Entity model because it gives the best overall performance while its computational cost is acceptable for our purpose.

4.2 News classification

Text classification, the assignment of free text documents to one or more predefined categories based on its content, can be considered similar to classification of discrete set-valued attributes. The training data is used to construct a classification model, which relates the underlying features of the new text document to one of the existing class labels. Then for a much larger set of given test instances for which the class is unknown, the training model is used to predict a class label for each instance in the set. Because of the high dimensional, low frequencies and sparse properties of the text documents, the general machine learning methods need to be special trailered or modified for text classification.

There are two different categories for text classifications. The first category is *Discriminative classifier*, including neural network classifiers and Support Vector Machine classifiers, will give a particular label explicitly to the given instance. The second category is *Generative classifier* or *Bayesian classifier*, will build a probabilistic classifier based on documents' underlying word feature in different class. Basing on the model, a probability will be assigned to the given instance. We can then rank this probability and assign multiple labels to the instance[8].

The general processes of text classification could be summarized as follow: we first read document, tokenize text, apply stemming, delete stopwords, represent the text in vector form and apply feature selection and transformation techniques to transfer the raw text documents into a representation suitable for the classification task. Then we use trained classification model to classified the free text documents.

4.2.1 Support Vector Machine

Support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep

Method	Accuracy	precision	recall
Nearest Neighbor	0.8725	0.6911	0.6716
Associate classification	0.8839	0.9160	0.5859
Instanse-based kNN	0.9113	0.7458	0.6514
Clustering-based kNN	0.9150	0.7405	0.8693
Support Vector machine	0.9306	0.9390	0.6639

Table 1: Accumulate experiment result from R3[1]

the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function K(x,y) selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vectors that occur in the data base. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation: $\sum_i \alpha_i \mathrm{K}(\mathrm{x}_i,\mathrm{x})$ = constant. Note that if K(x,y) becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding data base point x_i. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets which are not convex at all in the original space [1].

- Pros and Cons for SVM

  The advantages of support vector machines are:

  1. Effective in high dimensional spaces.
  2. Still effective in cases where number of dimensions is greater than the number of samples.
  3. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

  Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

  The disadvantages of support vector machines include:

  1. If the number of features is much greater than the number of samples, the method is likely to give poor performances.
  2. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation

- Evaluation for SVM

  Chen et al[1] in his paper proposed a two phase classification using associate classification and clustering-based kNN combining with experts' work experience to improve the accurate of the news classification. But it requires lots of human labor. Here we just briefly summarize his experiment result to check the performance of each method. It is clear that SVM outperforms kNN and associate classficiation.

[1]definition from Wiki

4.2.2 *Artificial Neural Network*

Artificial Neural Network are inspired by the biological neurons that form the basis of human learning. The analogy with human neurons that constitute its learning process has been inspirational in developing the artificial neural network. An artificial neuron combines its input signals for example by a weighted sum. The output is one numerical value, computed from a so called activation function, modeling approximately a "firing" of a coarsely modeled biological neuron.

One popular version of an artificial neural network for classification is a multilayer perceptron which is a feedforward artificial neural network. A multilayer perceptron model takes inputs and appropriately maps it to the one of the designated output layers. This model can be thought of to serve the purpose of classification where the inputs are may be the features of an unclassified document and the output layer can classify the document and assign the appropriate label to it. Between the output and the input layer can be one or more hidden layers that adjust the input features depending on the weight vector.

An artificial neuron combines its input signals for example by a weighted sum. The output is one numerical value, computed from a so called activation function, modeling approximately a "firing" of a coarsely modeled biological neuron.[9] It can be mathematically denoted as follows:

$$o = f(b + \sum_{i=1}^{n} w_i (a)_i)$$

o is the output
$(a)_i$ *are n inputs*
w_i *are summation weights*
f is the activation function
b is the bias term

The output value will be compared with the target. We can use the mean absolute error as the error function[6]. The error function can be represented as:

$$E_m = \frac{1}{2n} \sum_k \sqrt{(}T_k - O_k)^2$$

where n is the number of training patterns. O_k *and* T_k *are the output value and target value, respectively.*

Propagating this error backwards in the network and adjusting the weights W_i forms the core basis of the Back Propagation algorithm for training the data set. Remarks about multilayer perceptron using Artificial Neural Network:

1. An MLP operates on numerical vectors, so everything needs to be coded as ordered n-tuples of floating point numbers. This is particularly tricky in our context as our inputs are documents and we may need to express features in numbers.

2. The network layer can be trained using the back propagation algorithm. Back propagation algorithm simply adjusts the weight vectors of the layers to get the

desired output in the training set.

3. The algorithm will require deeper mathematical understanding but it seems to have more application in image classification as images are inherently described as numbers (pixels) in the computer.

Apache Mahout provides an inbuilt implementation of multilayer perceptron. However it only has a single computer implementation.

Verdict: May not be useful for our requirement. Steep learning curve.

4.2.3 Latent Semantic indexing

Latent Semantic indexing is an indexing and a retrieval method using a mathematical model to identify patterns. LSI basically creates a frequency profile of each document, and looks for documents with similar frequency profiles. If the remainder of the frequency profile is enough alike, it'll classify two documents as being fairly similar, even if one systematically substitutes some words. Conversely, if the frequency profiles are different, it will classify documents as different, even if they share frequent use of a few specific terms (e.g., "file" being related to a computer in some cases, and a thing that's used to cut and smooth metal in other cases). LSI is also typically used with relatively large groups of documents. The other documents can help in finding similarities as well – even if document A and B look substantially different, if document C uses quite a few terms from both A and B, it can help in finding that A and B are really fairly similar.

LSI is intended to show the degree to which items are similar or different and, primarily, find items that show a degree of similarity to an specified item. It can be coupled with k-nearest neighbor algorithm to classify documents. LSI can also be couple with the artificial neural network for developing a learning based classification model[6].

Remarks:

1. Its a popular model for indexing and retrieval.
2. An extension of latent semantic analysis.
3. It has been patented

Verdict: It might be a good option to try implementing LSI with kNN. We may want to try logistic regression for classification. Some implemented classification algorithms ready to be used can be found in Mahout

4.2.4 kNN

The k-nearest neighbors (kNN) rule is one of the oldest and simplest methods for pattern classification. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positiveinteger, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

The kNN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set. Its performance thus depends crucially on the distance metric used to identify nearest neighbors.

Comments:

1. we can combine kNN with some other methods, like tf-idf , Latent Semantic Indexing.
2. We can also combine it with SVM, there is an algorithm called large margin nearest neighbor (LMNN) classification. Its framework can be viewed as the logical counterpart to SVMs in which kNN classification replaces linear classification[13].

4.2.5 TF-IDF

tf-idf, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval andtext mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query, one common method is BM25.

- improved tf-idf

  There is also an improved method of tf-idf stated in the paper [17], This paper presents a new improved term frequency/inverse document frequency (TF-IDF) approach which uses confidence, support and characteristic words to enhance the recall and precision of text classification.

  The confidence degree of a feature word w_j is :

  $$conf(w_j, c_m) = \frac{N(w_j, c_m)}{N(w_j, all)} \tag{1}$$

  The support of a feature word w_j is :

  $$sup(w_j) = \frac{N(w_j, all)}{N(all)} \tag{2}$$

  The dominant measure is defined as:

  $$dom(w_j, c_m) = f(conf(w_j, c_m), sup(w_j)) =$$

  $$\begin{cases} 1, \\ ((conf(w_j, c_m) \geq threshold) \cap (sup(w_j) \geq threshold)) \\ 0, \\ ((conf(w_j, c_m) < threshold) \cup (sup(w_j) < threshold)) \end{cases} \tag{3}$$

  Feature words that meet the threshold ($dom(w_j, c_m) = 1$) are considered as characteristic words $cw(w_j, c_m)$. When the set of feature words of a document d_j contains characteristic word $cw(w_j, c_m)$, the text document d_j will be classified into the category cm. In other words, we can use characteristic word to determine whether a document is classified into a particular category or not.

  The improved tf-idf in [17] have better performance in precision and recall over the basic tf-idf. However, the tf-idf doesn't achieve the best performance in classification of documents, compared to some other method according to some conducted experiments.

- comparisons between tf-idf and other methods Comments: Paper[15] did the comparison among TFIDF, LSI (Latent Semantic Indexing) and multi-words for text classification, They used a Chinese and an English document collection to respectively evaluate the three methods in information retreival and text categorization. Experimental results have demonstrated that in text categorization, LSI (Latent Semantic Indexing) has better performance than other methods in both document collections. Also, LSI has produced the best performance in retrieving English documents. This outcome has shown that LSI has both favorable semantic and statistical quality and is different with the claim that LSI cannot produce discriminative power for indexing.

  Paper[7] did the comparison among Prototype (Rocchio's) algorithm, Nearest Neighbor, Naive Bayes, Compression Neural Network over standard Reuters dataset (Lewis, 1997), a preclassified collection of short articles. This is one of the standard test-beds used to test information retrieval algorithms (Dumais et al., 1998). The conclusion is:

  1. the one-class SVM gives the best overall performance. This is quite clear with respect to all the other algorithms except the compression NN algorithm which is comparable. Scholkopf's proposal has the usual advantages of SVM; in particular it is less computationally intensive than neural networks.
  2. Under the "macro" averaging, the NN and outlier-SVM were somewhat superior. This means that while the one-class SVM was more robust with regards to smaller categories, the NN and outlier-SVM showed good results by emphasizing success in the larger categories.
  3. On the other hand, the one-class SVM was very sensitive to the parameters and choice of kernel. The neural network method, in comparison, seemed relatively stable over these parameters. (In our experiments, the linear kernel was, however, fairly stable although its results were slightly worse than the neural network algorithm.) Thus, under current knowledge, i.e. if the understanding of the parameter choice is unclear, it would seem that the neural network method is the preferred one.

  Summary: We may use SVM+LSI to classify documents into very fine small categories, we can use kNN+LSI to classify news into row large categories .

4.2.6 *Naive Bayes Classifier*

In statistic and probability theory, Bayes' theorem expresses how a subjective degree of belief should rationally change to account for evidence. It means that, given proposition A and an evidence B, the probability A occurs given condition B can be described by formula:

$\mathrm{P(A|B)} = \frac{P(B|A)P(A)}{P(B)}$

In classification, Bayes' theorem is applied with independence assumptions. Naive Bayes classifiers considers each features contribute independently to a probability of an object belongs to a category.

The probabilistic model for Naive Bayes:

$\mathrm{P(C|F_1 \ldots F_n)} = \frac{P(F_1 \ldots F_n|C)P(C)}{P(F_1 \ldots F_n)}$

However, since we regard conditional independent of each features, assuming that each feature is conditionally independent of every other feature for given category C.

$\mathrm{P(C|F_1 \ldots F_n)} = \frac{P(C)}{P(F_1 \ldots F_n)} \prod_{i=1}^{n} \mathrm{P(F_i|C)}$

One of implementation model for Naive Bayes is multinomial naive Bayes for document classification. This model samples represent frequencies where certain events have been generated by multinomial. The problem with this model is that it required training data set to collect evidence before the test data. If a given category and feature never occur in training data, the frequency probability estimate will be zero. One way to solve this problem is by doing a smoothing, called Laplace smoothing to ensure that all probability estimates never be zero.

Verdict: It might not be useful for our project because we classify news based on the content and location.

4.3 User Feedback

We want to have a user-feedback learning system to tailor the news to the interests of the user. We first let user choose his topics of interest at sign up. Afterwards, user's clicks/taps are collected to learn his interests in certain topics. We decided to use vector space model because we can easily extend it to include other parameters like timestamps. Therefore, user profiles and news documents can all be represented as vectors with tags in them. Whenever a user looks at an article, we add the product of the news vector and a parameter to the user profile and do clustering on user's tags to adjust his top interests.

4.4 Automatic Summarization

In order to find a good summarization tool for our project to deliver concise news summary to our users, we studied several automatic summarization methods and compared their pros and cons. Guided by this survey on the current available methods, we aimed to find an open source auto-summarizer provided with good evaluations. Below is a brief summary about our survey on automatic summarization.

1. Automatic Extraction

   The most commonly used auto-summarization method is based on automatic extraction. The basic idea is to treat the whole document composed by sentences which are themselves composed by individual terms. Therefore, in order to extract the summary for the whole article, we can start by calculating the weight of each term, then the weight of each sentence. Rank the sentences and select those whose score are higher than the threshold value. Finally, output all the summary sentences according to their original order in the article to form the summary. Term weighting can be calculated using TF-IDF as discussed in section 4.2.5, and based where it appears in the original text (i.e. the term may appear in the tile), the weight can be determined according to multiple signals. The weight of a sentence is simply the sum of all the terms in that sentence[12].

2. Semantic Analysis Based Summarization

   Another main approach for auto-summarization is to extract the main points based on the semantic tree of the article. This abstraction based approach condense text more strongly than extraction-based approach in general[12]. However, it requires relatively sophisticated NLP analysis, which is still considered a difficult task today, so this method is not commonly used as compared to the extraction method.

3. Automatic Summarization Based on User-query

   The core feature of this method is the summary is query dependent. It gives user more flexibility to express the constraint on the summary. In general, users are expected to give a ratio on the portion of the summarized text or the maximum number of words in the summary, but this approach extends that to return a summary related to a specific topic defined by the user query. When calculating the term weight, this approach takes into consideration of the query words and normalizes its result by query length[3]. One popular auto-summarizer MEAD that we found on-line implemented this feature.

Based on the survey on those methods, we managed to find available candidate summerizars that implemented one or more of these features. Details will be presented in Implementation section 6.4.

5. CRAWLING

The basic crawler is provided by Minejie. The base code is provided from here[2]. The crawler will extract the html page file, one picture from the source you specified and the main text from the news page (Figure 2). However the original crawler only support 3 websites and contains some incompatibilities. The XPath tuning to parse enough amount of website pages is a tedious work and requires lots of time to apply to each individual websites. Because for different websites, they have different layouts and labels. Even for same website they may have dozens of page models. Take Guardian as an example, we have at least 14 XPaths and models for guardian in order to crawl enough amount of news from Guardian.

After 50 days crawling(time log available from April 20, before that we manually run twice per day), we have collected more than 12,000 news from the eight sources. The total size of news is 2G. The content of the txt file we crawled has <title>, <url>, <crawl time>, <publish time>, <abstract> and <text>. We use the URL as the unique ID to processing the articles in the following procedures. Furthermore, after the near duplicate removal, we will add a hashed value of the duplicate returning number. That means, similar news or same news will has the same value in the near duplicate field. <summary> Field contains the article summary text for the Summary, <category> field has the category information.

6. IMPLEMENTATION

6.1 Search Engine: Solr

Apache solr is a popular open source search engine from the apache lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search . Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Jetty. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language.

6.1.1 Implementation of Solr in Woof

Solr performs two major functions in our system

1. Indexing

   The data obtained from the crawler is fed to solr with the help of connector script written in python. The connector parses the crawled data converts the data into a solr compatible ready to be indexed XML format. The XML data created adheres to the indexing schema of Solr. The indexer schema species type of the data in the field, if the data should be indexed and if the value should be stored for retrieval. This is important as the all the fields may not be searched and hence need not be indexed. Thus facilitating optimum usage of memory. The indexer shell script uses this XML data to index the data into solr.

2. Searching

   Once the data has been indexed successfully search results would be returned on querying the solr via the REST api. A simple solr query may be generated by replacing white spaces with space and appending it to the query url given here[3]. Depending on the response expected by the webserver the output format may also be specified (XML by default).

6.1.2 Indexing and Searching query analysis flow

In order that our search queries produce high quality results we customized the solr configuration to accommodate two important features

1. Queries coming in to solr will be searched in the news title with a highest boost and then the news description is boosted. This is followed by equal boost search on the remaining fields. This is done as we expect a search queries that match the title more need to be higher ranking.

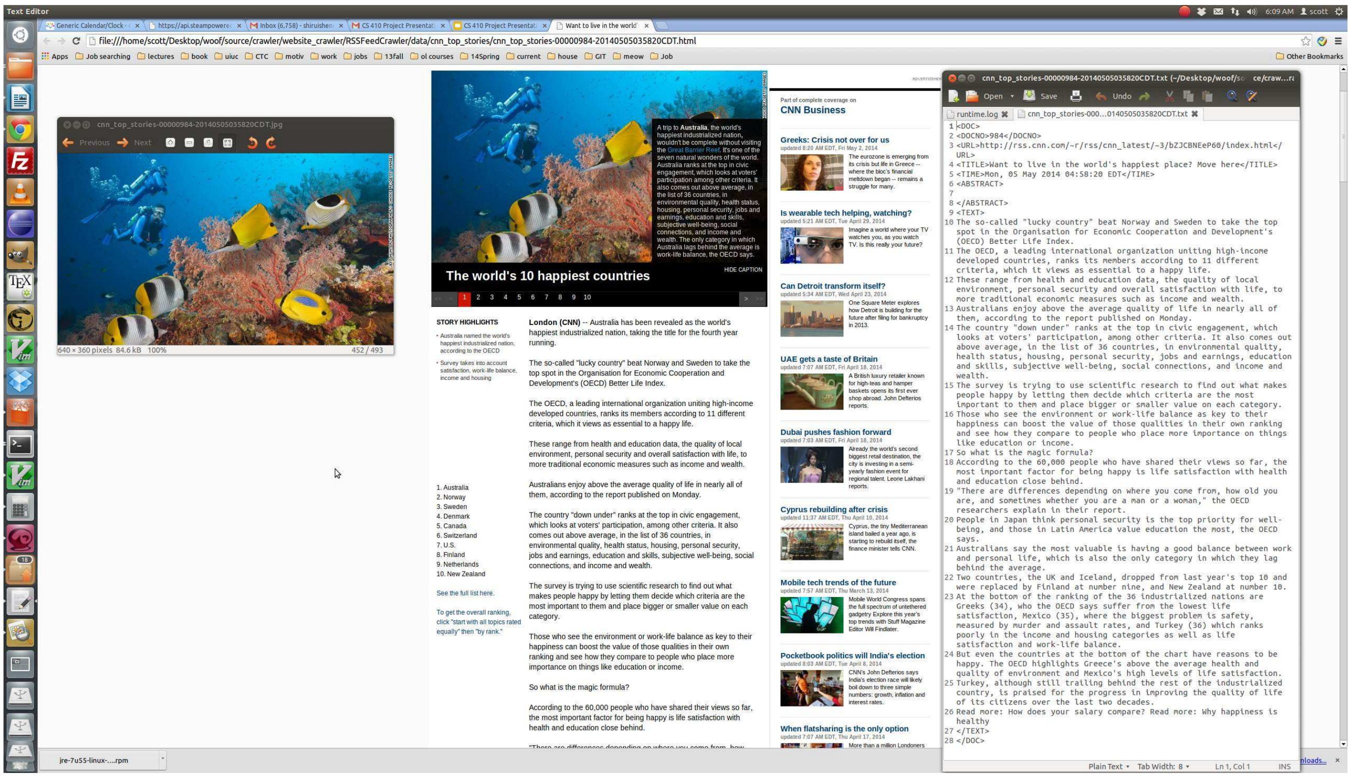
2. We also implemented various filters that help reformat the input query to return the best results to the user. The following filters were used before the query was searched in solr:

   - Stopword filter - This filter would remove stopwords from the query
   - Lowercase filter - This filter would make the query insensitive to cases
   - Porter stem filter - This filter uses the porter stemming algorithm to stem the query words.

It must be noted that these filters were also used while indexing the data.

[2] http://sourceforge.net/projects/rssfeedcrawler/files/RSSFeedCrawler/

[3] http://timan103.cs.illinois.edu:8983/solr/select/?q=malaysia+australia


Text Editor
file:///home/scott/Desktop/woof/source/crawler/website_crawler/RSSFeedCrawler/data/cnn_top_stories/cnn_top_stories-00000984-20140505035820CDT.html
cnn_top_stories-00000984-20140505035820CDT.jpg
Previous
Next
640 × 360 pixels 84.6 kB 100%
452 / 493
cnn_top_stories-00000984-20140505035820CDT.txt (~/Desktop/woof/so ce/craw...ra
Open
Save
Undo
runtime.log
cnn_top_stories-000...014050503582OCDT.txt
Plain Text
Tab Width: 8
Ln 1, Col 1
INS
jre-7u55-linux-....rpm

A trip to **Australia**, the world's happiest industrialized nation, wouldn't be complete without visiting the Great Barrier Reef. It's one of the seven natural wonders of the world. Australia ranks at the top in civic engagement, which looks at voters' participation among other criteria. It also comes out above average, in the list of 36 countries, in environmental quality, health status, housing, personal security, jobs and earnings, education and skills, subjective well-being, social connections, and income and wealth. The only category in which Australia lags behind the average is work-life balance, the OECD says.

HIDE CAPTION

The world's 10 happiest countries

STORY HIGHLIGHTS

- Australia named the world's happiest industrialized nation, according to the OECD
- Survey takes into account satisfaction, work-life balance, income and housing

1. Australia
2. Norway
3. Sweden
4. Denmark
5. Canada
6. Switzerland
7. U.S.
8. Finland
9. Netherlands
10. New Zealand

See the full list here.

To get the overall ranking, click "start with all topics rated equally" then "by rank."

London (CNN) -- Australia has been revealed as the world's happiest industrialized nation, taking the title for the fourth year running.

The so-called "lucky country" beat Norway and Sweden to take the top spot in the Organisation for Economic Cooperation and Development's (OECD) Better Life Index.

The OECD, a leading international organization uniting high-income developed countries, ranks its members according to 11 different criteria, which it views as essential to a happy life.

These range from health and education data, the quality of local environment, personal security and overall satisfaction with life, to more traditional economic measures such as income and wealth.

Australians enjoy above the average quality of life in nearly all of them, according to the report published on Monday.

The country "down under" ranks at the top in civic engagement, which looks at voters' participation, among other criteria. It also comes out above average, in the list of 36 countries, in environmental quality, health status, housing, personal security, jobs and earnings, education and skills, subjective well-being, social connections, and income and wealth.

The survey is trying to use scientific research to find out what makes people happy by letting them decide which criteria are the most important to them and place bigger or smaller value on each category.

Those who see the environment or work-life balance as key to their happiness can boost the value of those qualities in their own ranking and see how they compare to people who place more importance on things like education or income.

So what is the magic formula?

According to the 60,000 people who have shared their views so far, the most important factor for being happy is life satisfaction with health and education close behind.

ADVERTISEMENT

Part of complete coverage on
CNN Business

Greeks: Crisis not over for us
updated 8:20 AM EDT, Fri May 2, 2014
The eurozone is emerging from its crisis but life in Greece -- where the bloc's financial meltdown began -- remains a struggle for many.

Is wearable tech helping, watching?
updated 5:21 AM EDT, Tue April 29, 2014
Imagine a world where your TV watches you, as you watch TV. Is this really your future?

Can Detroit transform itself?
updated 5:34 AM EDT, Wed April 23, 2014
One Square Meter explores how Detroit is building for the future after filing for bankruptcy in 2013.

UAE gets a taste of Britain
updated 7:07 AM EDT, Fri April 18, 2014
A British luxury retailer known for high-teas and hamper baskets opens its first ever shop abroad. John Defterios reports.

Dubai pushes fashion forward
updated 7:03 AM EDT, Fri April 18, 2014
Already the world's second biggest retail destination, the city is investing in a semi-yearly fashion event for regional talent. Leone Lakhani reports.

Cyprus rebuilding after crisis
updated 11:37 AM EDT, Thu April 10, 2014
Cyprus, the tiny Mediterranean island bailed a year ago, is starting to rebuild itself, the finance minister tells CNN.

Mobile tech trends of the future
updated 7:57 AM EDT, Thu March 13, 2014
Mobile World Congress spans the full spectrum of untethered gadgetry Explore this year's top trends with Stuff Magazine Editor Will Findlater.

Pocketbook politics will India's election
updated 8:03 AM EDT, Tue April 8, 2014
CNN's John Defterios says India's election race will likely boil down to three simple numbers: growth, inflation and interest rates.

When flatsharing is the only option
updated 7:07 AM EDT, Thu April 17, 2014
More than a million Londoners

```
<DOC>
<DOCNO>984</DOCNO>
<URL>http://rss.cnn.com/~r/rss/cnn_latest/~3/bZJCBNEeP60/index.html</URL>
<TITLE>Want to live in the world's happiest place? Move here</TITLE>
<TIME>Mon, 05 May 2014 04:58:20 EDT</TIME>
<ABSTRACT>

</ABSTRACT>
<TEXT>
The so-called "lucky country" beat Norway and Sweden to take the top spot in the Organisation for Economic Cooperation and Development's (OECD) Better Life Index.
The OECD, a leading international organization uniting high-income developed countries, ranks its members according to 11 different criteria, which it views as essential to a happy life.
These range from health and education data, the quality of local environment, personal security and overall satisfaction with life, to more traditional economic measures such as income and wealth.
Australians enjoy above the average quality of life in nearly all of them, according to the report published on Monday.
The country "down under" ranks at the top in civic engagement, which looks at voters' participation, among other criteria. It also comes out above average, in the list of 36 countries, in environmental quality, health status, housing, personal security, jobs and earnings, education and skills, subjective well-being, social connections, and income and wealth.
The survey is trying to use scientific research to find out what makes people happy by letting them decide which criteria are the most important to them and place bigger or smaller value on each category.
Those who see the environment or work-life balance as key to their happiness can boost the value of those qualities in their own ranking and see how they compare to people who place more importance on things like education or income.
So what is the magic formula?
According to the 60,000 people who have shared their views so far, the most important factor for being happy is life satisfaction with health and education close behind.
"There are differences depending on where you come from, how old you are, and sometimes whether you are a man or a woman," the OECD researchers explain in their report.
People in Japan think personal security is the top priority for well-being, and those in Latin America value education the most, the OECD says.
Australians say the most valuable is having a good balance between work and personal life, which is also the only category in which they lag behind the average.
Two countries, the UK and Iceland, dropped from last year's top 10 and were replaced by Finland at number nine, and New Zealand at number 10.
At the bottom of the ranking of the 36 industrialized nations are Greeks (34), who the OECD says suffer from the lowest life satisfaction, Mexico (35), where the biggest problem is safety, measured by murder and assault rates, and Turkey (36) which ranks poorly in the income and housing categories as well as life satisfaction and work-life balance.
But even the countries at the bottom of the chart have reasons to be happy. The OECD highlights Greece's above the average health and quality of environment and Mexico's high levels of life satisfaction.
Turkey, although still trailing behind the rest of the industrialized country, is praised for the progress in improving the quality of life of its citizens over the last two decades.
Read more: How does your salary compare? Read more: Why happiness is healthy
</TEXT>
</DOC>
```

Figure 2: The data crawled from the eight source websites in three files, content.txt, local saved html and one picture

6.1.3 *Solr connectivity to other components of woof*

After the crawled data is indexed to Solr, 3 major components of woof try to access this new data and index back the results obtained after processing the data from solr. Summarizer Component: The summarizer component runs the query here[4] periodically on solr to fetch all documents that dont contain summary.

We built connector scripts that would facilitate indexing of the summarized data back to solr.

Filter and Classification component: Similar procedure was followed for the filter component and the classification component. Both would query solr periodically to obtain unfiltered and unclassified documents and index back the data using the connector scripts.

6.1.4 *Problem we have in Solr implementation*

The solr system is contructed under the domain of `http://timan103.cs.illinois.edu/` from our user defined port 8983.

But we didn't realize we can't access this port and domain from the woofuiuc.com until we have completed the solr indexing and have the data fully processed. Therefore, our system demo has been divided into two parts. The search engine from
`http://timan103.cs.illinois.edu:8983/solr/select/?q=`
and the website from `woofuiuc.com`

6.2 Near-duplication Removal

In order to implement the near-duplication removal method, we used three main ideas from our survey: Named Entities extraction, Jaccard Similarity and Shingling.

6.2.1 *Named Entity Extraction*

For the Named Entity extraction process we utilized the nltk python library extracting the tagged nouns from the and the chunking the text to also extract the named entities and specifying them through a isNamedEntity flag in the database schema.

6.2.2 *Jaccard Similarity*

For the Jaccard Similarity, let $\mathbb{N}$ be the set of news. For each news article d_i in $\mathbb{N}$, let E_i be the set of Named Entities extracted from d_i.
The Jaccard Similarity between two news articles d_i and d_j is defined by:

$$J_{ij} = \frac{|E_i \cap E_j|}{|E_i \cup E_j|} \tag{4}$$

Following equation 4, documents that present similarity J higher than a pre-defined threshold th are possible near-duplicate candidates, and they are passed to the Shingling method to confirm the near-duplication. This process is described in Algorithm 1.

6.3 Classification

We have more than 12,000 cutting edge news from all categories. Therefore we need enough training set to help us train the data.

[4]`http://timan103.cs.illinois.edu:8983/solr/select/?q=-summary:[*%20TO%20*]&rows=20`

Algorithm 1 Near-Duplicate Candidate Detection

```
$\mathbb{M}$ = matrix $|\mathbb{N}| \times |\mathbb{N}|$, initially filled with 0s
for $d_i \in \mathbb{N}$ do
    for $d_j \in \mathbb{N}$ do
        if $d_i = d_j$ then
            continue
        end if
        if $J_{ij} \geq th$ then
            $\mathbb{M}_{ij} = 1$
        end if
    end for
end for
return $\mathbb{M}$
```

6.3.1 Training set

- REUTERS-21578

  This dataset contains 21578 Reuters news documents from 1987. They were labeled manually by Reuters personnel. Labels belong to 5 different category classes, such as 'people', 'places' and 'topics'. The total number of categories is 672, but many of them occur only very rarely. Some documents belong to many different categories, others to only one, and some have no category. Over the past decade, there have been many efforts to clean the database up, and improve it for use in scientific research.

  CONS: Preprocessing of the dataset is tedious and challenging. Also, by manually check the news from this dataset for more than 100 news documents, we think this dataset is out of date and not fittable for our project

- 20 news Group

  This data set is a collection of 20,000 messages, collected from UseNet postings over a period of several months in 1993. The data are divided almost evenly among 20 different UseNet discussion groups. Many of the categories fall into overlapping topics; for example 5 of them are about companies discussion groups and 3 of them discuss religion. Other topics included in News Groups are: politics, sports, sciences and miscellanious.

  CONS: This dataset contains only a small part of the categories with high overlaps. The fine granularity of categories provides us a concept for our classification level. But we can't use this dataset for our classification

- Wikipedia

  Wikipedia is is a collaboratively edited, multilingual, free Internet encyclopedia. We have dumped the whole wiki pages(10GB) and partitioned it into 77,135 files. Wikification, commonly referred to as Disambiguation to Wikipedia (D2W), is the task of identifying concepts and entities in text and disambiguating them into the most specific corresponding Wikipedia pages. So for each categories, we extracting all children pages from that category. These pages can be treated as training set and train a classifier

  CONS: The accuracy of the classifer now will rely on not only the accuracy of the classification learners but also the accuracy of finding the right categories children pages(If we find only a small portion of the children pages, we'll have a bias understanding for the categories. The result is likely to inaccurate).

After several experiments and some back and forth tunings, we think non of the above data sets work for our scenarios. So at the end we adopt an unsupervised classification. The details of our approaching method will be explained later in this subsection.

6.3.2 Choice of the machine learning platform

Besides of the dataset, the choice of the machine learning platform is critical as well. We have tried both scikit and mahout. Mahout is based on the hadoop, which gives it better scalability. But the learning curve of mahout is higher. Considering the size of our data is relative small(2GB), we don't need to use Hadoop or Mapreduce which is targeted at peta-scale input data analysis.

6.3.3 Implementation of our classifier

First, we extract the text data from the 12000 news set. In particular, we have merged the title, abstract and text into json format(Figure 3). In particular, we have designed the parser to be able to enhance the title and abstract by changing of the function parameters. Because of the limitation of the time, we just set the default to be 1. Since we didn't use a cloud service like EC2, the amount of text news is relatively small and to minimize the learning curve for a large group, we didn't use any Database like mongoDB or Hbase. Instead, we read from the json file every time we need to process the file.

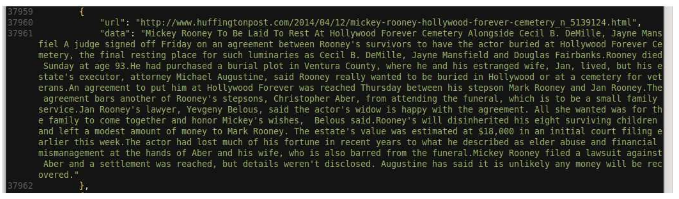
Then we preprocessed the news part in json file. The processing procedures are listed as below:

1. Replace every ',:().' with space
2. Use Porter stemmer to process the news
3. Use TF-IDF to vectorize the document with stop words applied

After preprocessing, we use MiniBatchKmeans and Kmeans algorithm to cluster the data. Assigning a label to each URl. Then we use the classifier to classify the output data. the classifer we tried includes RidgeClassifer, Perceptron, PassiveAggressiveClassifier, LinearSVC(l1 and l2 penalty), SGD, SGD with penalty set to 'elasticnet', nearestCentroid(aka Rocchio classifier) and Naive Bayes. For each classifer, we extracted the top 20 terms and use these top 20 terms as an indication of the potential categories these clusters refer to. The evaluation of the classification is showed in section 7.2

6.4 News Summary

In this project, we need to display the summary of several news articles crawled from various websites on our user's dashboard. It is crucial for us to give a concise and coherent summary to better capture their interests. We compared several popular summarizers and decided to use the Open Text Summarizer (OTS). In particular, we take into consideration of the efficiency of the summarizer, the algorithm that is implemented, and the features that the summarizer



```
        {
            "url": "http://www.huffingtonpost.com/2014/04/12/mickey-rooney-hollywood-forever-cemetery_n_5139124.html",
            "data": "Mickey Rooney To Be Laid To Rest At Hollywood Forever Cemetery Alongside Cecil B. DeMille, Jayne Mans
fiel A judge signed off Friday on an agreement between Rooney's survivors to have the actor buried at Hollywood Forever Ce
metery, the final resting place for such luminaries as Cecil B. DeMille, Jayne Mansfield and Douglas Fairbanks.Rooney died
 Sunday at age 93.He had purchased a burial plot in Ventura County, where he and his estranged wife, Jan, lived, but his e
state's executor, attorney Michael Augustine, said Rooney really wanted to be buried in Hollywood or at a cemetery for vet
erans.An agreement to put him at Hollywood Forever was reached Thursday between his stepson Mark Rooney and Jan Rooney.The
 agreement bars another of Rooney's stepsons, Christopher Aber, from attending the funeral, which is to be a small family
service.Jan Rooney's lawyer, Yevgeny Belous, said the actor's widow is happy with the agreement. All she wanted was for th
e family to come together and honor Mickey's wishes,  Belous said.Rooney's will disinherited his eight surviving children
and left a modest amount of money to Mark Rooney. The estate's value was estimated at $18,000 in an initial court filing e
arlier this week.The actor had lost much of his fortune in recent years to what he described as elder abuse and financial
mismanagement at the hands of Aber and his wife, who is also barred from the funeral.Mickey Rooney filed a lawsuit against
 Aber and a settlement was reached, but details weren't disclosed. Augustine has said it is unlikely any money will be rec
overed."
        },
```

Figure 3: URL and news in json format, in particular, the first few words contain the title and abstract of the news.

supports. The rest of this section will give more detailed explanation on each aspect.

Generally, there are two approaches to automatic summarization: extraction and abstraction. Extractive methods select a subset of existing words, phrases, or sentences in the original text to form the summary, whereas abstractive methods will first build an internal semantic representation and then try to use the natural language generation techniques to create a summary that is closer to what a human might generate. Since the abstract method may not use the same words as it appeared in the document, and the performance will be highly rely on the NLP technique, most automatic summarization research focus on improving the extraction method.

OTS uses extraction method. The main idea behind OTS is to TF-IDF weighting to give each sentence a score in the document. The summary will be composed by the sentences with highest scores[2]. The article is scanned once and a list of all the words and their frequencies in the text is stored in a list. After sorting this list by the occurrence of the words in the text we remove all the words that are common in the English language using a dictionary file. OTS allows it's user to customized this dictionary by re-setting the '- -key words' options. Each sentence will then be given a score based on TF-IDF weighting of the words in this sentence. In the end, a line that holds many important words but few frequency occurred words will be assigned a high score. To produce a 20% summary we print the top 20% sentences with the highest grade. OTS also implements stemming. This feature will group together all of the derivatives of a certain words, so that OTS can better figure out the subject of the article and select the important sentence.

OTS can be used as a command line tool, which can be easily plugged into our project by writing a shell script. The program can either print the summary as text or HTML. It also allows user to specify the ration of the summarized text to the original text.

We also considered MEAD, which is another popular auto-summarization tool that we found on-line. However, we think OTS is easier to be integrated into our system, it uses less overhead, and the performance is good enough to satisfy our requirements here. Details on the performance of different summeraziers will be presented in Evaluation 7.3.

6.5 Collaborative Filtering

Collaborative filtering tries to infer information about user preferences based on other user preferences. It is really important to process user feedback, and collaborative filtering plays a fundamental role in this part.

In order to feed our collaboration filtering algorithm, we have to define a matrix $|\mathbb{U}| \times |\mathbb{I}|$, where $\mathbb{U}$ is our set of users and $\mathbb{I}$ is a set of items. The set of items could be anything that is given a weight w by the users in an explicit or implicit way. For instance, $\mathbb{I}$ could represent the set of news articles, and w could then be the rating given by the users to the news articles. In that scenario, the output of the Collaborative Filtering would consist of a raked list of news that a user could like, based on the taste of similar users.

Another possible approach is to take $\mathbb{I}$ as the tags (refer to section 4.3) generated by the users. In this scenario, w could be defined as the number of times a user inputs a certain tag as a query. The output would be tags that are probably relevant to a user, based on the tags used by other users.

From the two situations above, we can notice that: first, the weight w can be defined as any function that maps any combination of the variables captured from the user's feedback over the set of items. Second, the similarities can be any function that captures the similarity between two users. It could be cosine-similarity, Jaccard Similarity, Pearson Correlation, or any other metric.

In our implementation, we considered $\mathbb{I}$ as the set of news articles, and w the ratings given by the users. We plan on expanding that to user tags in the future. We used Pearson Correlation as the similarity measure.

$$r_{ij} = \frac{\sum_{k=1}^{|\mathbb{I}'|}(U_{ik} - \overline{U_i})(U_{jk} - \overline{U_j})}{\sum_{k=1}^{|\mathbb{I}'|}(U_{ik} - \overline{U_i})^2 \sum_{k=1}^{|\mathbb{I}'|}(U_{jk} - \overline{U_j})^2} \tag{5}$$

In equation 5, U_{ik} is the value of w for each $k \in \mathbb{I}$ that is defined for user U_i. The same is valid for U_j. $\overline{U_i}$ and $\overline{U_j}$ are the means of the values of w for user U_i and U_j, respectively. $\mathbb{I}'$ is the set of items that is defined for both U_i and U_j. It is necessary to make such a constraint, otherwise we would have to make a regression of the missing values of w to calculate the correlation.
r_{ij} is the Pearson Correlation between users U_i and U_j, and its value ranges from -1 to 1.

Algorithm 2 Collaborative Filtering

```
input $\mathbb{W}$ = matrix $|\mathbb{U}| \times |\mathbb{I}|$, filled with weights $w$
input $U_i$ = user to whom recommend the items
$R$ = vector of size $|\mathbb{I}|$, initially filled with 0s
$S = 0$
for $U_j \in \mathbb{U}$ do
    if $U_j = U_i$ then
        continue
    end if
    for $k \in \mathbb{I}$ do
        if $k \in U_i$ then continue
        end if
        $R_k = R_k + r_{ij}(\mathbb{W}_{jk} - \overline{U_j})$
        $S = S + |r_{ij}|$
    end for
end for
$R_k = \frac{R_k + \overline{U_i}}{S}$ for $k \in R$
return sorted $R$ desc
```

Algorithm 2 describes the collaborative filtering system. In the end of the execution, R will contain the estimated weight values $\hat{w}$ for the inputted user U_i, over the k items. Since r_{ij} can assume negative values, the algorithm, as it is implemented, takes into consideration negative similarities. Most of the implementations that use Pearson's Correlation rule out negative similarity values.

6.6 Website

6.6.1 Language selection

We have chosen HTML/CSS/JavaScript to write our website's frontend, and PHP/Codeigniter as our back end. Code Igniter is a powerful PHP framework with a very small footprint. Because of the Timan domain can't be accessed from woofuiuc.com(we mentioned before), our website has been parted into two parts. The fully processed data and search engine could be found from:
`http://timan103.cs.illinois.edu:8983/solr/select/?q=`
and add the keywords you want to search.

6.6.2 Implementation

The website implementation consists of two parts: "Public", which accessible to every visitor, and "Registered User" which only accessible by registered user. Each part has several sections which provide similar functionality. We provide navigation bar to help users browse each sections of which each section is a news category. In each news section, we also provide subsection options. For example news "Technology" section have subsection in "Computer", "Mobile Phone", "Electronics", etc.

The "Home" section is the main page showing general news based on ranking. In this section, Top 10 news are displayed, regardless of category it belongs to. For each news section category, such as "Technology", "Sports", and "Politics" each of this section shows top rank news on its categories.

In every pages, visitor can choose to register from provided link to access more functionality in our website. User can register by providing an email and a password. User can also provide their news interest from categories options provided in registration page. Registered user can choose to login using their registered email account and password. Logged in user would not be displayed register link or login form on their page. Instead the page provide a logout button to terminate user session.

For registered user, the "Home" Page will display top rank news of ALL user's interest on the top. This is aimed to help user to get their interest news earlier without the need to browse each sections separately. Another functionality excluded for registered user is a search box. User can search news by typing keywords which will be directed to our system and display top rank news according to the queries. This is aimed to get more specific user's interest news.

Due to the limitation of the time and steep learning curves of the website construction, we didn't fully implement and test our website yet. We plan to finish this part in the near future.

7. EVALUATION

7.1 Near-duplicate Removal

For evaluating the near-duplication removal, a corpus containing 43 manually extracted news was created. The Named Entities were extracted and the near-duplications were tagged.

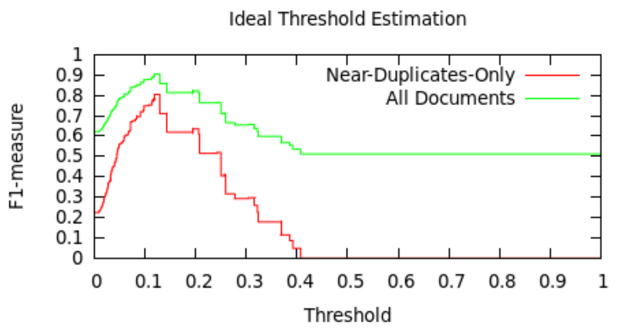


Figure 4: Evaluation of different threshold values for the Jaccard Similarity/Near-Duplicate Candidate Detection.

7.1.1 Jaccard Similarity

In order to define the best threshold for the Jaccard Similarity, we took the collection and compared the near-duplicate candidate detection with our manually tagged near-duplicate news articles. *Precision* and *Recall* were defined as follows: Let S_i be the set of documents that are similar to d_i, and S_i' be the set that our Near-Duplicate Candidate Detection method retrieved as the set of documents similar to d_i. Then:

$$Precision = \frac{|S_i \cup S_i'|}{|S_i'|}$$
$$Recall = \frac{|S_i \cup S_i'|}{|S_i|}$$

The **F1-measure** is the harmonic mean of *Precision* (P) and *Recall* (R):

$$\textbf{F1-measure} = \frac{2PR}{P+R}$$

Figure 4 illustrates the comparison of different values of threshold, and the F1-measure achieved by using that threshold.
The "Near-Duplicates-Only" line indicates the results for the documents that belong to a near-duplicate cluster only. The "All Documents" line indicates the results for all the documents in the collection. The importance of doing such an evaluation, is to get a notion of the average F1-measure for a document that contains a near-duplicate. Taking all documents into consideration gives a boost to the accuracy, because documents that don't result in false positives during the Near-Duplicate Candidate Detection always end up with F1-measure equal to 1, which is reflected for threshold values higher than 0.5 in Figure 4.

Based on this experiment, we conclude that the value for *th* that achieves the best result is 0.12, yielding **F1-measures** of 0.9048 for "All Documents" and 0.8050 for "Near-Duplicates-Only".

The accuracy we measured is between 50% - 70%.

7.1.2 Cosine Similarity

Cosine similarity is based on vector space model of representation of news documents. It computes the similarity by taking the dot product of two vectors(documents). The result for cosine similarity is similar to that of Jaccard, around 57%-72%.

7.1.3 Hashing

Based on a report by Jerry Zhao on minWise hashing implementation[18], we obtained a result of accuracy from between 64% - 100% with a recall rate between 95% - 33% respectively.

We also tested the Charikar's hash algorithm[5] which performs slightly better than the minWise hashing due to a bigger hash digits we used. Its accuracy ranges from 72% - 98% with a recall rate between 92%-58%.

7.1.4 Shingling

For Shingling, each article is converted to a set of shingles of length 3. Two articles are considered duplicates if the Jaccard similarity of the corresponding sets of shingles is larger than some threshold α. In order to find α, we conduct a training set labeled by human and pick the value of α that fits this training set the best (using **F1-measure**). This gives us the best accuracy around 82%-100% with higher recall rate 98%-88% respectively. The better accuracy and recall rate comes with a bigger computational cost or longer processing time.

7.2 Classification

7.2.1 Cross validation score for the clustering result

In this section we analysis the cross validation scores of the method we stated in section 6.3. First, we used the Kmeans and minibatch to classify the news without supervision. In order to test the accuracy, we used the 20 news group as an test sample. After cross validation of the 20 news group(containing 11314 documents), the average score is around 0.58(each time the result is different due to Kmeans random initial state). We believe our classifer will have at most 0.58 accuracy because in The 20 news group, every group is roughly evenly distributed. But this is not the case in our scenario(edge cutting real news). Because we don't have time to manually classify 12,000 news(and it is inaccuracy to do so by human). We can only use a small random sample of the result as evaluation of the accuracy in our implementation. The accuracy score is stated later in this subsection.

7.2.2 Methodology

Furthermore, because of randomness of the Kmeans and minibatch Kmeans, for each of the configurations we ran 3 times in order to achieve a better result. Since the number of the clusters is unknown as well, we iterated from 15 to 70. Therefore, we have $3 \times 2 \times (70 - 15) = 330$ times. For each time of running, we applied all classification methods(RidgeClassifer, Perceptron, PassiveAggressiveClassifier, LinearSVC(l1 and l2 penalty), SGD, SGD with penalty set to 'elasticnet', nearestCentroid(aka Rocchio classifier) and Naive Bayes). Then we saved all output for these 330 runs and corresponding classifiers(>120GB). We select one of the best from these 330 runs and use it as our classifer. The overall running time varies from 3 minutes to 7 minutes. In particular, we accumulate the classifier training time and displayed in Figure 6.

7.2.3 Searching for the right classifier

The next step is to find the best clustering result from these 330 result. In order to select the best clustering result from these 330 result. We accumulated the number of documents for each categories. Here, we have an assumption that the best classifiers should have roughly even distributed number of documents. Although this assumption is unlikely to be true, but comparing to some categories in most of classifiers has only one or two documents and a huge categories with more than three thousands of files. This assumption is reasonable in our opinions. By reading these 330 graphs, we found some common patterns here.

1. Most of the result has a max/average ratio below 10, but on the other hand, most of the max/mean ratio is above 25.

2. Most of the graph's largest category contains at least 2000 documents. The (max document count- median document count)/number of categories is larger than 65.

By manual checking the 330 graphs, we selected one of the graph where max/average = 6.63 and max/mean = 8.42. And (max - mean)/number of categories is 15.7. These 3 parameters are lowest among all 330 classifiers(Figure 5).

7.2.4 The accuracy evaluation for our classifier

In order to evaluate the accuracy of our result, we first use larger key terms sets and take these 66 key terms sets as 66 documents. Then each of the 12,000 news is treated as a query. By using the BM25, we have a ranked list of the categories for each news. We can evaluate our classification method by looking at the label's position in this ranked list(Figure 7).

7.2.5 Weakness of our classification methodology

Although the evaluation of the result showed that the document are classified into right categories with reasonable accuracy. But we think our classifier has fundemental weakness. They are:

1. The categories' labels are not human friendly labels like sports, entertainment etc. Instead it is a list of stemmed words.

2. Most of the news belong to multiple categories, instead our classifier blended them into one big categories. The side effect of this is that even the top 20 key terms in one categories conflict each other(some key terms lists clearly contains more than one category)

3. The classifier is very unfriendly to outlier news(standalone news). And it is somehow bias to popular news(Check clippers or MH370 in our news data set).

The final result could be found in github folder here[5] (it is a private folder, please send rshi7@illinois.edu an email to require the permission)

7.3 News Summary

Based on the paper[16], four current summarizers are evaluated, Essence, Copernicus Summarizer, Subject Search Summarizer(SSS), and Open Text Summarizer (OTS). Evaluation included the following procedures.

[5] `https://github.com/bravehearto0/woof/tree/master/source/classification/news_classifier`

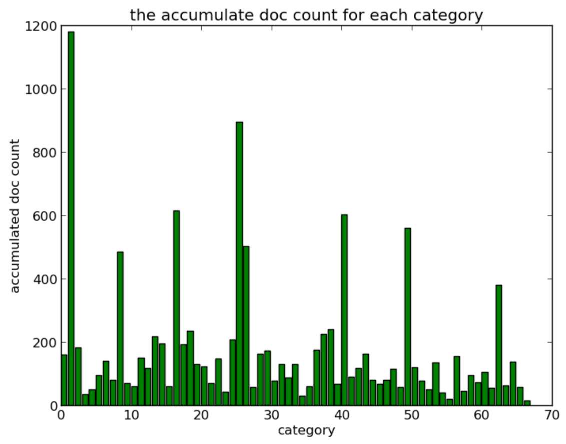


Figure 5: Accumulated count of each categories, for 66 categories

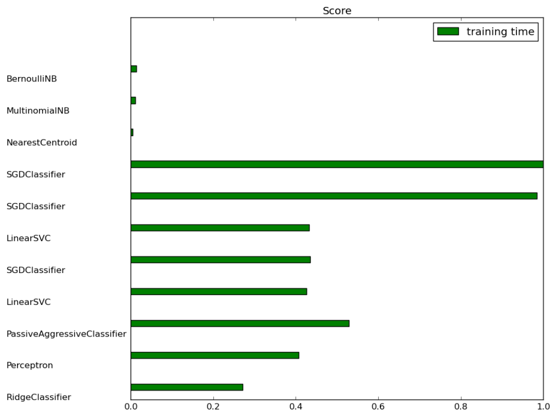


Figure 6: Evaluation of different training time for different classifier, The linearSVC are in l1, l2 penalty order, SGD is in l1, l2 and elastinet penaltoes orders.

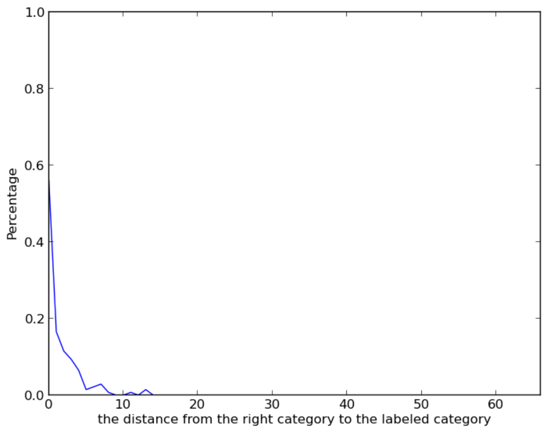


Figure 7: Evaluation for our classifier's accuracy

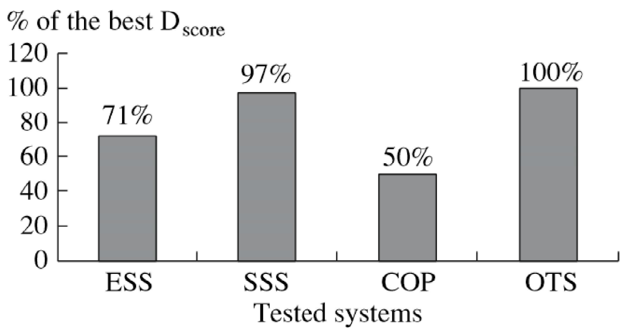


Figure 8: Quality of Summarization of tested systems in percents for the best result.

1. A comparative content analysis of the source text and of each of the 20 summaries by the formula

$$R = \frac{\sum_{k=1}^{31} (Q_i F_i)}{S_t} \tag{6}$$

where F_i is frequency of the i term from the vocabulary with all inflectional forms, Q_i is the functional weight, S_t is the total number of sentences in the given text, and R is the coefficient of the relevance of the given text. The calculation of the frequency of terms was done automatically with the help of a specially designed macros MS Word and the sum was calculated in MS Excel tables.

2. The value D score was obtained, that is, the sum R of the coefficients of all summaries generated by this system and representing its efficiency.

$$D = R_5 + R_{10} + R_{15} + R_{25} \tag{7}$$

Values D score of the tested systems were compared and the difference between them was calculated, which determined the difference in the quality of summarizing. The best result was taken as 100%. The value of D depended on two factors, including the number and frequency of the terms from the model vocabulary in summaries. Table 1 summarizes the coefficients of relevance (R) of summaries of different lengths and value D score of tested systems. Figure 8 shows the difference in the quality of summarizing in percents for the best result. Figure 9 shows the relevance of summaries of different lengths.

The best result was displayed by the Open Text Summarizer, which exceeded SSS by 3%, Essence by 29%, and Copernicus by 50%. The systems can obviously be divided into two groups, as OTS and SSS have shown far better results than Copernicus and Essence.

7.4 Collaborative & Content-based Filtering

7.4.1 Content-based Filtering

The Content-based Filtering utilizes a vector space model that makes use of the previously extracted nouns and named entities of every document. This component of our portal is intended to make use of the items that the user had clicked. It takes the sum of all the nouns of the articles that the

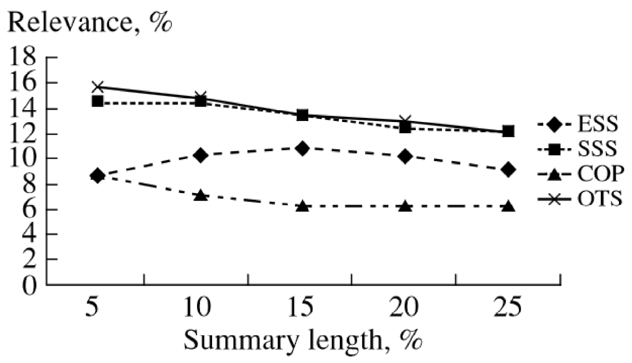


Figure 9: Relevance of summaries of different lengths

user has clicked and normalizes them by the total number of clicks. Using the terms in this generated user model, we then use the dot product of this user vector with all the latest documents and find the k most similar documents. For more advanced implementations, we use the user model to find the closest cluster centroid of documents using the 60+ clusters of documents we had obtained beforehand. We then randomly select k documents from that cluster. The purpose for this is that the news that is propagated to the user is similar to his interests but it does not overfit to his user model[10].

7.4.2 Collaborative Filtering

The Collaborative Filtering evaluation was made using a Book-Crossing Dataset[6] that contains $278,858$ users and $271,379$ items. We compared the absolute difference between actual weights w and the estimated values $\hat{w}$. The average absolute difference, in a scale from 0 to 10 was 0.9650 for the approach that rules out negative similarities, and 0.9484 for the approach that takes negative similarities into account. This implies in a difference of 0.2% in accuracy, which is statistically irrelevant.

8. FUTURE WORK

8.1 Website

The next step would be integrating the websites we have now, add more functions and test more.

8.2 Near-Duplicate Removal

For near-duplicate removal, the process of comparing all of the documents with all of the others is unnecessarily laborious. In that sense, some studies [4, 14] aim at reducing the cost of the first phase in the Near-Duplicates detection.

8.3 Open directory

One major problem of our classification is that we don't have a good training set. Open directory project uses a hierarchical ontology scheme for organizing site listings. Listings on a similar topic are grouped into categories which can then include smaller categories. We may use this dataset for our future classification

[6]`http://www2.informatik.uni-freiburg.de/~cziegler/BX/`

8.4 Hierarchy classification

Deploying Hierarchy classification may be another way to solve our problem. We can first sort the news based on its original categories. Then we deploy a multi-labels classifier to first classifier the news into raw categories and then apply a within category classifier to further divide the news into subcategories. For those mixed categories, we can automatically classify it into subcategories until the similar threshold is met for all news in the same category.

8.5 Name entity recognition

We can apply Name entity recognition in multiple place in our project, using this methodology in classification may achieve better result.

8.6 Integration with wiki

Although in our classification implementation analysis(section 6.3.1) we mentioned that using wikipedia may not boost our accuracy. But accuracy is not the only thing we care. Also there has to be some beneficial when you expand the query/doc/category with knowledge from wikipedia. This can be one direction for our future study as well.

9. INDIVIDUAL CONTRIBUTION

Ruisheng: Writed the design of the woof, both from back end and user's perspective, writed the survey in section 4.2.1, work on expansion of the crawler to support 8 sources and maintaining the daily running of the crawler(section 5), finished the classification part in implementation and evaluation(section 6.3, 7.2). Wrote a front-end of the website here [7]

Ajay: Designed the workflow of the project with Ruisheng, End to end handling of solr and scripts for the connecting components(section 6.1), Developed a naive bayes model in apache mahout for classifiying woof news documents using the 20 news groups data. However this was not used later as the 20 groups news classification was inadequate for our application and mahout worked better in a distributed setup with larger datasets. Surveyed on feasibility of Artificial Neural Networks and Latent semantic indexing for classification(section 4.2.2, 4.2.3).

Paul Jones Vijayakumar: Managed the survey and implementation of the near-duplicate filtering component, implemented the name entity and noun extraction component and the user-adaptive content based filtering component

Son: fixed some incompatibilities in crawler, implemented the shingling method(section 7.1.4) for duplicate detection and built a layer between the website and the database to encapsulate SQL requests.

Victor: worked on the Jaccard Similarity component of the Near-Duplicate detection (section 6.2.2), and also in the Collaborative Filtering (section 6.5). I implemented algorithms 1 and 2, and ran the experiments described in section 7.4 and in section 7.1.1.

Lucas Surveyed a list of databases for our backend. Researched, implemented and tested duplicate removal methods including minWise Hash[18] and Charikar's Hash[5] algorithms. Researched and worked with Paul on the design and implementation of user-feedback collecting and learning system.

[7]`http://woofuiuc.com/woof/index.html`

Fan: surveyed on the automatic summarization topic (section 4.4). Helped to find the OTS summarizer and integrated it into the Solr system with Ruogu. When deciding which summarization tool to use, I studied the main algorithm that OTS used and compared it other summarizer's approach (section 6.4). For website construction, I worked with Alex, Sharu, and Son for the initial design of our website, and wrote the registration page.

Ruogu: Select OST summarizer from several Open Source summarizers. Evaluated OTS and compared it with other similar summarizers(section 7.3). Summarized all articles for Solr platform. Set up database for our website.

Alex: worked on the MVC model of the website construction(section 6.6). cleaned the redundant front-end provided by Ruisheng, write the code of log in page. Fixed the bugs of the CSS display in woofuiuc.com. Survey the naive bayes classifier in classificationsection 4.2.6

Sharu: worked on survey the TF-IDF and kNN(section 4.2.4 4.2.5), design of the website UI and programmed the log in page of the website

10. REFERENCES

[1] Y.-L. Chen and T.-L. Yu. News classification based on experts' work knowledge. In *Proceedings of the 2nd International Conference on Networking and Information Technology IPCSIT*, volume 17, 2011.

[2] C. Cole. Open-text-summarizer. `https://github.com/neopunisher/Open-Text-Summarizer`.

[3] J.-l. FU and Q.-x. CHEN. Research on automatic summarization based on rules and statistics for chinese texts [j]. *Journal of Chinese Information Processing*, 5:001, 2006.

[4] C. Gong, Y. Huang, X. Cheng, and S. Bai. Detecting near-duplicates in large-scale short text databases. In *Proceedings of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'08, pages 877–883, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] V. Holub. Detection of near-duplicate documents. `http://d3s.mff.cuni.cz/~holub/sw/shash/`.

[6] C. H. Li and S. C. Park. Artificial neural network for document classification using latent semantic indexing. In *Information Technology Convergence*, 2007.

[7] L. M. Manevitz and M. Yousef. One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154, 2002.

[8] A. McCallum. Multi-label text classification with a mixture model trained by em. In *AAAIâĂŹ99 Workshop on Text Learning*, pages 1–7, 1999.

[9] N. Paavo. Classification and multilayer perceptron neural networks. University Lecture, 2010.

[10] X. Shen. *User-centered adaptive information retrieval*. ProQuest, 2007.

[11] E. Uyar. *Near-duplicate news detection using named entities*. PhD thesis, BILKENT UNIVERSITY, 2009.

[12] S. Wang, W. Li, F. Wang, and H. Deng. A survey on automatic summarization. In *Information Technology and Applications (IFITA), 2010 International Forum on*, volume 1, pages 193–196. IEEE, 2010.

[13] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 18:1473, 2006.

[14] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 131–140, New York, NY, USA, 2008. ACM.

[15] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

[16] V. A. Yatsko and T. N. Vishnyakov. A method for evaluating modern systems of automatic text summarization. *Automatic Documentation and Mathematical Linguistics*, 41.3:93–103, 2007.

[17] Z. Yun-tao, G. Ling, and W. Yong-cheng. An improved tf-idf approach for text classification. *Journal of Zhejiang University Science A*, 6(1):49–55, 2005.

[18] J. Zhao. An implementation of min-wise independent permutation family. `http://www1.icsi.berkeley.edu/~zhao/minwise/`.