Personalised content recommendation for Stack Overflow data

[Extended Abstract] *

Karthik Sheshadri
Department of Computer Science
NC State University
kshesha@ncsu.edu

Nandini Rangaswamy
Department of Electrical and Computer Engineering
NC State University
nrangas@ncsu.edu

ABSTRACT

We consider the problem of recommending personalized Stack Overflow content. We present two content based filtering solutions and one collaborative filtering approach using random forests. We comparatively evaluate these approaches on a simple classification task, to obtain relative performance metrics in terms of accuracy, processing time, and memory footprint. We consider two distinct feature choices, namely, tf-idf and information gain, and evaluate their relative performance for 1,2,3, and 4 gram keyword sets. Our results motivate the choice of an information gain based random forest classifier for our specific task.

Keywords

content based filtering, Stack Overflow, collaborative filtering, random forest, support vector machine, n gram, term frequency, inverse document frequency, information gain.

1. DATASET

We obtained our data from the Stack Exchange data dump[3]. The dump is hosted on archive.org, and contains data scraped from stackexchange.com over the period 2012-2015. The data is categorized by topic, and features 309 subjects ranging from religious areas such as Buddhism to computer programming subjects like Python and android coding.

We work off of a 10% subset of the data. We restricted our analysis to subjects relevant to Computer Science, namely Python,Android, Arduino, and Java.

2. ALGORITHMS USED

This section details the three approaches we implemented. We assume that there is a dataset D, split into training (T) and evaluation (E) data. Our task consists in picking the top k posts E_k of E, with k being an input parameter. The problem hence reduces to implementing three different methods of evaluating E_k.

For algorithms 1 and 3, we utilized bi-grams as our fundamental feature. This design choice is motivated by the optimal performance they exhibited on our classification task (see sec 3)

2.1 N-gram based similarity search

This algorithm essentially treats the recommendation problem as one of computing similarity exhibited between each test string and it's closest counterparts in the training data. We define $Sup(T)$ as the set of all n-grams that can be constructed from the elements of T, and similarly $Sup(E)$. Further, given two strings a and b, define the edit distance $d(a, b)$ to be the minimum-weight series of edit operations that transforms a into b.

We return the top k posts E_k such that
$E_k = min_k(d(Sup(E_k), Sup(T_i))), \forall i \in T$.

2.2 Random Forest

Random forests have recently demonstrated state of the art performance in tasks ranging from body tracking in computer vision to sentiment analysis and regression. The random forest[2] algorithm learns n decision trees, each trained in a greedy fashion on a randomly chosen subset of the data. Each split node of a decision tree chooses the n-gram that maximises utility among a randomly chosen subset of the n-gram set x.

As described in (see Sec 3), we conducted a binary classification experiment on Stack Exchange data to experimentally optimize the number of trees learnt by the forest. Figure 1 visualizes the results. Although the results appear bi-modal at first sight (with 2 trees achieving as high an accuracy as 6), we believe that this is an overfitting phenomenon and would dissipate with more trials. Therefore, we decided to utilize a 6 tree forest as suggested by the results.

2.3 Support Vector Machine

Support vector machines[2] (SVMs) have demonstrated consistent results in the area of text mining and have found uniform application in this field.

SVMs learn a separation between positive and negative examples in a high dimensional space by constructing an n-1

*A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX* at `www.acm.org/eaddress.htm`

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

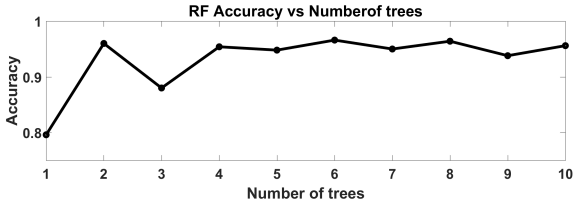

Figure 1: RF accuracy versus number of trees

dimensional hyperplane separating the two classes. The hyperplane is chosen so as to experimentally optimise the separation between the two classes. Since our decision problem is binary and we are not concerned in this case with learning a scaling function to reconcile multiple classifiers, we utilize a simple linear SVM.

Our algorithm was implemented in Matlab, which has a trainer for SVMs but restricts it's evaluation functionality to a simple binary class label. However, for the purposes of recommendation, we require a measure of posterior probability of class membership. We therefore define our own measure of posterior probability as distance of a particular test vector to the separating hyperplane. We evaluate this measure according to the following equation:

$$d_{FV} = \sum_{i=1}^{m} \alpha_i SV_i \cdot FV + b \tag{1}$$

where d_FV is the distance of feature vector FV to the separating hyperplane defined by the m support vectors SV. Alpha are the learned weights of the SVM, and b is the bias.

We implemented this functionality in Matlab. As can be seen from figure 4, our implementation is as efficient as Matlab's built in predict function.

3. DESIGN CHOICES

3.1 Classification problem

Given that our recommendation system lacks a well defined ground truth, it was necessary to arrive at a means of evaluating our algorithms without conducting a user survey every time a parameter was set. Therefore, we introduced a simple classification task to evaluate our algorithms.

For this purpose, we assumed a hypothetical user who is interested in a particular Stack Exchange topic, such as 'python'. This informs a simple binary classification task in which all python articles are positives and all other articles can be de facto treated as negatives.

Our algorithms can thence be evaluated in terms of accuracy, running time and memory footprint on this basis, as a preliminary means of evaluation without user involvement.

3.2 Feature Selection

3.2.1 N-grams

We define our feature space to be the corpus of all possible words in the English language. Therefore, our basic feature vector would have a dimensionality of around 10k. However, using individual words would fail to model any probabilistic relationships between individual dimensions.

The gold standard in literature to model probabilistic relationships is represented by the n-gram model[4], which considers every contiguous phrase of n words in a document to be an individual dimension in the feature vector. We adopt the n-gram model in our work and experimentally optimise the value of n. Figure. 2 depicts our experimental evaluation of performance as a function of n, for TF-IDF, and standard information gain. We note here that the variation of performance with n is consistent and independent of the measure of utility used. As recommended by figure. 2, we utilize bi-grams for our feature space.

3.2.2 Term Frequency - Inverse Document Frequency

The de-facto standard in recent years has been represented by Term Frequency-Inverse Document Frequency[7], which measures how "important" a word is to a document based on it's frequency of occurrence in the document relative to that frequency in the corpus as a whole. Each post is then described by it's tf-idf vector for every term in the corpus. This then can be used to feed in to a similarity based approach such as described in Sec 2.1 by sorting all terms according to their tf-idf, and reporting the top k of those.

While this is a logical approach that yields consistent results, vanilla tf-idf is an unsupervised approach that neither utilizes discriminative learning nor provides a method for evaluating the accuracy of it's results.

To overcome this shortcoming, we introduce a notion of evaluation based on learnt discriminative ability. The estimated tf-idf vectors should be able to discriminate relevant articles from random ones, and their accuracy on this classification task serves as an evaluation of their quality as a feature resresentation.

3.2.3 Information Gain

Our approach to post recommendation stems from the view of each post as a vector in an m space, in which every distinct n-gram represents a dimension. This motivates a binary classification problem in which a vector is either a relevant article, or it is not. It follows therefore, that an n-gram's utility in the binary classification problem so formulated can be used as a metric for evaluating how dominant a keyword it is.

To evaluate utility, we estimate each n-gram's information gain[5]. Accordingly, we define the utility of ngram x_i to be

$$IG(T, x_i) = H(T) - \sum_{i=1}^{m} \frac{|x \in T | x = x_i|}{|T|} H(x \in T | x = x_i) \tag{2}$$

where T denotes the set of training examples, and H the information entropy of T. We thus return the top k so estimated as our split nodes.

3.2.4 Evaluation with Random Forest

We comparatively evaluate the two notions of utility introduced and described above. We created three train/test splits of our data by randomly sampling 50% of examples each for our training and evaluation set. Then, we trained random forests using (i) TF-IDF utility (ii) information gain, and measured their accuracy on each test split using 1,2,3, and 4 gram based vector spaces.[1]

As can be seen from figure 3, information gain consistently yields better results than tf-idf. This is not surprising given that information gain is a dedicated discriminative

[1]Note that preprocessing steps such as stemming and stop word removal are carried out as described earlier

measure, as opposed to tf-idf which remains an unsupervised approach.

4. EXPERIMENTAL COMPARISONS

We evaluated our approaches based on three separate metrics: accuracy on our classification task, memory footprint, and running time.

4.1 Accuracy

We first deal with accuracy on our classification task as defined above. As can be inferred from Figure 3, all three approaches achieve an accuracy in excess of 95%, which to some degree makes it difficult to evaluate the algorithms on this basis. However, a few points are worthy of note.

Firstly, it is worthwhile to explain the consistent 100% performance of the n-gram approach. The N-gram approach, since it works on edit distance, is extremely unlikely to find low edit distance with articles not containing the relevant keywords. Therefore, from the standpoint of binary classification, the n-gram approach is perfect. However, this does not imply that the returned articles are in any sense the "best" or most relevant to a given user, simply that they contain the keywords that a user has specified to describe her interests.

Therefore, this approach does not truly achieve personalization, but instead guarantees that irrelevant content is filtered out. This approach is consistent with the classic content based filtering philosophy, which provides "good" but not "personal" content. The results from this approach actually motivate it's use as an **initial filter** to reduce the order of magnitude of possible articles, after which a SVM or RF based algorithm can be applied to further refine and personalize content recommendations.

Turning to the accuracy of SVM and Random Forest, we find a seemingly surprising trend of reduction in performance with the number of articles in the test set. To explain this paradox, consider the fact that the SVM and RF were trained on a **fixed** number of articles (500), and then tested on splits of size 1000 to 5000. With the increasing size of the test set, it becomes much more likely that the classifier encounters data which is unlike what it has seen before, hence resulting in degraded accuracy. Note however that the accuracy is still well above 94%, **demonstrating the ability of these ML based approaches to generalise with limited data**.

Overall, however, it is hard to recommend one approach over the others based purely on accuracy. We hence turn to our other metrics.

4.2 Running time

Although recommendation systems have traditionally been run offline, recent trends in movie and shopping recommendation have recognised the value of recommendations in real time, in a manner reactive to a user buying an item or ordering a movie. A relevant example is the Amazon "customers buying this also bought" recommendation system, which must compute effectively in real time. We therefore evaluate our algorithms from this lens, to help inform our choice of approach.

Figure 4 comparatively presents the running times of our three approaches. Perusing the figure, we note that two distinct regimes operate. The n-gram approach has an order of magnitude higher run time that the machine learning based approaches. This is inherently not a surprising result, since an article with n words has (very approximately) $\binom{n}{2}$ bigrams. Similarly, a user feeding in m keywords would generate $\binom{m}{2}$ bigrams. Therefore, $\binom{n}{2} \cdot \binom{m}{2}$ edit distances must be computed, and these must then be sorted. This algorithm therefore has an overall complexity $\in O(\binom{n}{2} \cdot \binom{m}{2}) log(\binom{n}{2} \cdot \binom{m}{2})$ which is $\in o(n^2)$.

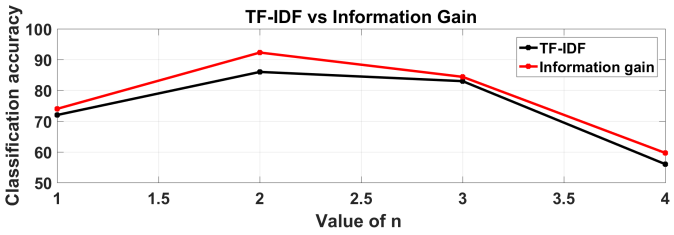

Figure 2: A comparison of classification accuracy for TF-IDF and Information Gain using 1,2,3 and 4 gram vector spaces.

In comparison to this growth level, the linear growths of the SVM and ML approaches seem insignificant and therefore cannot be discerned visually in Figure 4. In order to demonstrate their growth comparative to each other, we visualized these algorithms separately in figure 5. As can be seen, the majority of articles are processed in a timeframe less than or equal to 600 ms, less than a second. This therefore shows that both ML approaches can be used to carry out real time content recommendation in the collaborative filtering sense. While both approaches have a runtime satisfactory to this goal, the RF approach is marginally faster, and is consistently so over various test set sizes. Since their accuracies are very comparable, we argue that from this standpoint, the **RF approach beats the SVM and N-gram approaches**, and is therefore our recommendation from the standpoint of running time.

4.3 Memory Footprint

We now evaluate our algorithms from the lens of their memory footprint. Memory is in general considered subordinate to running time and accuracy as a means of evaluation, however we feel that this is a relevant metric for the following reasons: Most personal devices such as mobile phones and tablets have limited cache memory. Further, laptops running with limited RAM will expect background processes like recommendation systems to run without eating up a large percentage of the cache.

The memory footprint of our algorithms is visualized in Figure 6. Once again, we see a bimodal operation regime, with n-grams consuming much greater resources than the ML approaches. For the benefit of the reader, we pause to explain why this is the case.

Support Vector machines are compressive in the input training data set, since they only need to store a separating hyperplane and not the whole dataset at test time. The separating hyperplane is defined by an input parameter set of support vectors, and a few additional parameters like learnt weights (each a float), and an overall bias. Therefore the memory footprint of a single SVM is $\in O(k)$, i.e., **is constant with increasing data size**.

Similarly, a random forest learns a set of n trees, each of which is a set of split node decisions. This representation is

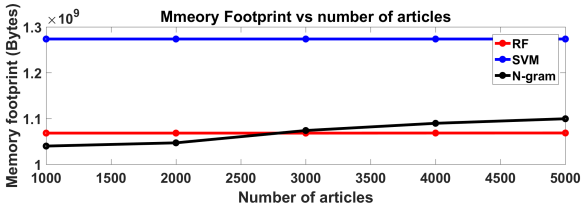

Figure 3: Memory vs. Number of articles

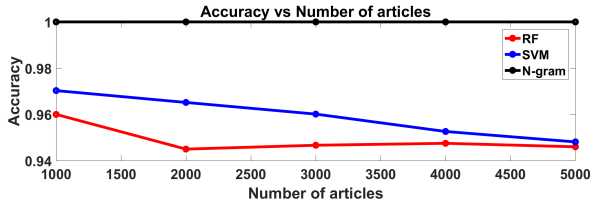

Figure 4: A comparative depiction of accuracy against the number of test articles.

also compressive, since only a very small subset of the training set needs to be stored in the forest. This representation is also $\in O(k)$ with respect to the size of the data set.

The n-gram approach is the only one that needs to store the entire training data set in cache at run time, since edit distance must be computed with every entry in the training set. Following a similar line of reasoning to the run time, we arrive at the conclusion that the memory footprint of this approach is also $\in O(\binom{n}{2} \cdot \binom{m}{2}) log(\binom{n}{2} \cdot \binom{m}{2})$ which is $\in o(n^2)$.

We argue that with such a running time and memory footprint, the n-gram based approach is infeasible for implementation in real world systems.

Once again, there is not to much of a significant difference between the SVM and RF approaches in terms of their memory footprints. Since both grow at a constant rate, we must choose based on the value of the constants. Since RF's have a marginally **smaller footprint**, we pick this approach.

4.4 Robustness to noise

Training data is likely to be noisy due to several factors causing performance degradation of even a stable algorithm. In order to evaluate the response of our three approaches to noise in the data, we added white Gaussian noise to our feature vectors. We utilized a signal to noise ratio of 30%. Figure 7 visualizes the response of our algorithms under conditions of noise.

There is a discernable difference in accuracy between N-gram and ML approach in response to noise.The 100% performance in N-gram approach can be attributed to it's use of edit distance. The presence of noise in the data can affect individual characters, leading to keywords appearing deformed. However, changing individual characters in a string leaves it's overall edit distance largely unchanged with respect to a list of external keywords. Therefore as long as the signal to noise ratio remains below any reasonable threshold, n-gram performance is largely unaffected.

Our RF algorithm demonstrates sensitivity to noise that is overall perhaps not too significant (since accuracy is still in the high 90's), but is still more so than either of the other two approaches. We see however that as the number of articles increases, the sensitivity to noise in our RF flattens out.

Turning to the SVM approach, we see that it's accuracy curve retains the linear pattern it exhibited before the noise was introduced. This is not particularly surprising, since perturbation of individual dimensions does not imply too much in a dimension space spanning several thousand n-grams. We must conclude **from this data** that the SVM approach is more robust to noise than the RF. **We acknowledge that many existing ML texts do not agree[6, 8]** with this conclusion. However, this is the pattern that we have observed from our data, and we must therefore report it. We acknowledge however that this pattern may be an artifact of the size of our data.

4.4.1 Kolmogorov-Smirnov statistical testing

Further, we conducted statistical tests to determine whether the effect of noise can be considered to significantly alter the results. We do not have normally distributed data, and therefore it would not be meaningful for us to implement a standard p-value test in terms of the standard deviation from the mean of our distribution. Instead, we utilize the standard Kolmogorov-Smirnov[1] non parametric test to estimate whether the results without and with noise correspond to the same distribution. Our Kolmogorov Smirnov p values for the SVM, RF, and n-gram algorithms are [0.994,0.989,1] respectively. These results clearly show that noise does not have a statistically significant impact on any of the algorithms, but an approach can still be chosen by maximising accuracy under conditions of noisy data.

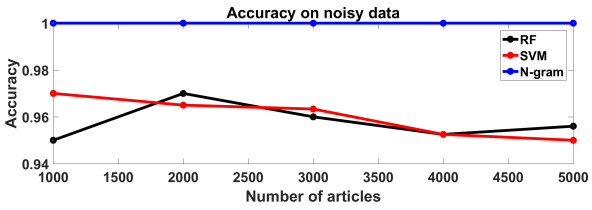

Figure 5: A comparison of accuracy under conditions of noisy data.

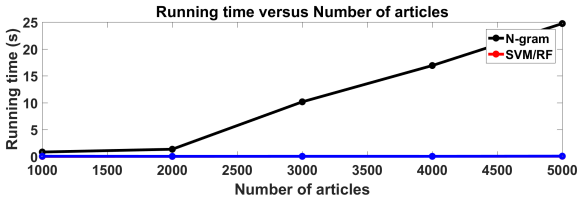

Figure 6: Running time vs. Number of articles

5. CONCLUSION

We presented one content based filtering (namely, N-gram based edit distance) and two collaborative filtering (namely, SVM and RF) approaches to the problem of recommending personalized Stack Overflow content. We evaluated these approaches from the four distinct perspectives of accuracy, running time, memory footprint, and robustness to noise. We noted that the N-gram approach has too high a run

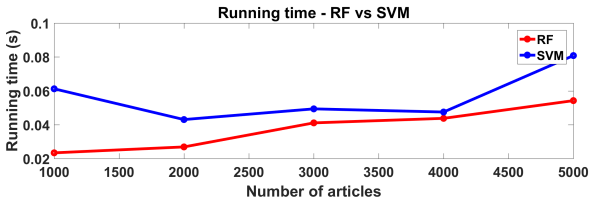

Figure 7: Running time vs. Number of articles

time and memory footprint to be a feasible solution, atleast for real time applications. We found that the SVM and RF approaches performed very similarly in most areas, however the RF approach was marginally faster but less robust to noise for our particular application. Given a virtual tie between the two algorithms, we are inclined to weigh RF's reputation for robustness in literature over our admittedly limited evaluation of it in this paper, and therefore recommend it's use as the system of choice.

6. REFERENCES

[1] https://en.wikipedia.org/wiki/kolmogorov

[2] https://en.wikipedia.org/wiki/supportvectormachine.

[3] http://stackoverflow.com/research/developer-survey-2015.

[4] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, Dec. 1992.

[5] E. Harris. Information gain versus gain ratio: A study of split method biases. In *ISAIM*, 2002.

[6] N. Japkowicz and M. Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, New York, NY, USA, 2011.

[7] J. Ramos. Using tf-idf to determine word relevance in document queries.

[8] L. B. Statistics and L. Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.