Batch Document Filtering Using Nearest Neighbor Algorithm

Ali Mustafa Qamar, Eric Gaussier, Nathalie Denos
Laboratoire d'Informatique de Grenoble (LIG)
ali-mustafa.qamar@imag.fr
eric.gaussier@imag.fr
nathalie.denos@imag.fr

Abstract

This paper describes the participation of LIG lab, in the batch filtering task for the INFILE (INformation FILtering Evaluation) campaign of CLEF 2009. As opposed to the online task, where the server provides the documents one by one, all of the documents are provided beforehand in the batch task, which explains the fact that feedback is not possible in the batch task. We propose in this paper a batch algorithm to learn category specific thresholds in a multiclass environment where a document can belong to more than one class. The algorithm uses k-nearest neighbor algorithm for filtering the 100,000 documents into 50 topics. The experiments were run on the English corpus. Our experiments gave us a precision of 0.256 while the recall was 0.295. We had participated in the online task in INFILE 2008 where we had used an online algorithm using the feedbacks from the server. In comparison with INFILE 2008, the recall is significantly better in 2009, 0.295 vs 0.260. However the precision in 2008 were 0.306. Furthermore, the anticipation in 2009 was 0.43 as compared with 0.307 in 2008.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software

General Terms

Measurement, Performance, Experimentation, Algorithms

Keywords

Information Filtering, Batch algorithms, k-Nearest neighbor algorithm

1 Introduction

The goal of the INFILE (INformation FILtering Evaluation) [2] campaign is to filter 100,000 documents into 50 topics (plus a category 'other'). Out of 50 topics, 30 are related to general news and events (e.g. national and international affairs, sports, politics etc.), whereas the rest concern scientific and technical subjects. A document can belong to zero, one or more topics, each topic being described by a set of sentences. In comparison with INFILE 2009, where there was only an online task, an additional batch filtering task was added in 2009. As opposed to the online task, where the server provides the documents one by one to the user, all of the documents are provided beforehand in the batch task. This explains the fact that feedback is not possible in

the batch task. We had participated in the online task in 2008 [3], and restricted ourselves to the batch one in 2009.

The k-nearest neighbor (kNN) algorithm is a largely investigated supervised learning algorithm due to its simplicity and performance. It aims at finding the k nearest neighbors of an example x (based either on similarity or distance) and then finding the most represented class in the nearest neighbors in order to classify x. Similarity has deemed to be more appropriate as compared to distance, specially while dealing with texts. In such case, the cosine measure is used instead of euclidean or mahanalobis distances.

In this paper, we develop a batch algorithm to learn category-specific thresholds in a multiclass environment. Our algorithm uses kNN algorithm along with cosine similarity, in order to filter the documents into various topics.

The rest of the paper is organized as follows: Section 2 describes the batch algorithm developed for the INFILE campaign, experiments and results are discussed in Section 3 while we conclude in Section 4.

2 Batch Algorithm for the INFILE Campaign

In order to filter the documents into various topics, we use a similarity measure between new documents and topics, along with a set of thresholds on this similarity that evolve over time. The similarity between a new document d, to be filtered, and a topic t_i can given as:

$$\text{sim}(t_i, d) = \alpha * \underbrace{cos(t_i, d)}_{s_1(t_i,d)} + (1-\alpha) \underbrace{\max_{(d' \neq d, d' \in t_i)} cos(d, d')}_{s_2(t_i,d)} \quad (1)$$

where $\alpha \in [0,1]$. The similarity given in equation 1 is based on two similarities: one based on a direct similarity between the new document and the topic (given by $s_1(t_i, d)$), and another one between the new document and the set of documents already assigned to the topic ($s_2(t_i, d)$. One might think that only the first similarity would suffice. However, this is not the case since the topics and the documents do not share the same kind of structure and content. The second similarity helps us to find documents which are closer to documents which had already been assigned to a topic. α is used to control the importance of the two similarities. In the beginning, when no documents are assigned to any topic, only the similarity, $s_1(t_i, d)$, is taken into account. This similarity is used to find a certain number of nearest neighbors for each of the document (we used 10 nearest neighbors) which eventually helps us to use the second similarity. A threshold was used for each of the 50 topics. We now describe the batch algorithm to filter the documents. As already mentioned, the feedback is not possible in this case since the complete set of documents is transferred to the user in one go.

Construction of initial set:

for each topic i ($i \in$ {101,102,...,150}

  find 10 nearest neighbors based on $s_1 = cos(t_i, d)$

  for each nearest neighbor found

    $t_i \Leftarrow NN$

Assignment of remaining documents to topics:

$\alpha = 0.7$

for each topic i

  $\theta_i = \min_{d \in t_i} \text{sim}(t_i, d))$

for each document d

  for each topic i

    if ($\text{sim}(t_i, d) \geq \theta_i$)

      $t_i \Leftarrow d$

$\theta_i = \min(\theta_i, \min_{d \in t_i} \text{sim}(t_i, d))$

Yang et al. [5] have described a similar method, whereby they learn category-specific thresholds based on a validation set. An example is assigned to a particular category only if its similarity with the category surpasses a certain learned threshold. In contrary, we do not have a validation set to learn thresholds, however, we create a simulated one, by finding nearest neighbors for each of the 50 categories.

3 Comparison with Online Campaign 08

We present here, a detailed comparison between the batch algorithm of 2009 and the online algorithm developed for the online campaign, 08. For time constraints, we were not able to participate in the online task this year. The online algorithm is presented below:

Construction of initial set: $\alpha = 0.7$, $\theta^1 = 0.42$
for each new document d
  for each topic i ($i \in$ {101,102,...,150}
    if ($l_i < 10$)
      if ($s_1(t_i, d) > \theta^1$)
        Ask for feedback (if possible) and $t_i \Leftarrow d$ if feedback positive

Assignment of remaining documents to topics:
for each new document d
  for each topic i ($i \in$ {101,102,...,150}
    if ($\text{sim}(t_i, d) > \theta_i^2$)
      $t_i \Leftarrow d$
    $\theta_i^2 = \min_{d \in t_i} \text{sim}(t_i, d)$

---

where l_i represents the number of documents assigned to a topic i. For each topic, two thresholds are used: the first one (θ^1) allows filtering the documents in the early stages of the process (when only a few documents have been assigned to the topic). The value chosen for this threshold was 0.42. The second threshold (θ^2), however, works with the global similarity, after a certain number of documents have been assigned to the topic. In addition, the algorithm makes use of the feedbacks (50 in total), so that only relevant documents are placed in the different topics.

The main difference between the two algorithms (batch and online) lies in the manner in which we construct the initial set of documents pertinent to the topics. In batch algorithm, we just rely on finding the 10 nearest neighbors for each topic, with the assumption that the nearest neighbors for a topic, would in general, belong to the topic under consideration. However, for the online algorithm, we use feedbacks if the similarity between a topic t_i and a document d, is greater than a certain threshold (θ^1). The rest of the algorithms are almost the same.

4 Experiments

We have run our algorithm on INFILE English corpus. For all of the documents, stemming was performed using Porter's algorithm. This was followed by the removal of stop-words, XML tags skipping and the building of a document vector (which associates each term with its frequency) using Rainbow [4]. A single run was submitted during the INFILE campaign where α was chosen to be 0.7. Initially, 10 nearest neighbors were found for each of the document based on the similarity s_1 (between a document and the topic). These documents were subsequently used to find s_2. The experiment was divided into 4 sub-parts, each sub-part being run in parallel to increase the

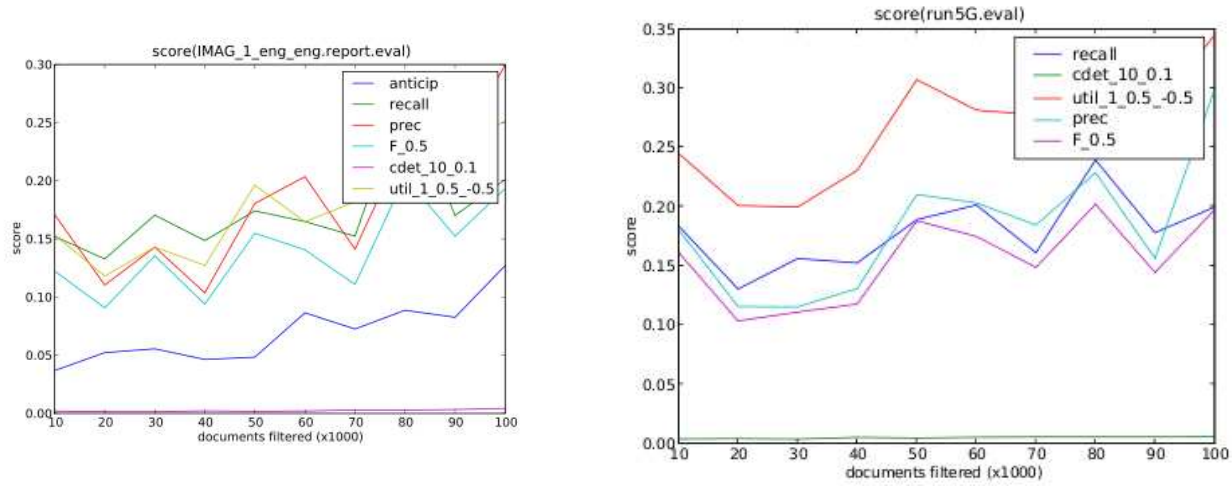

Figure 1: Score Evolution for Run 1 and 2

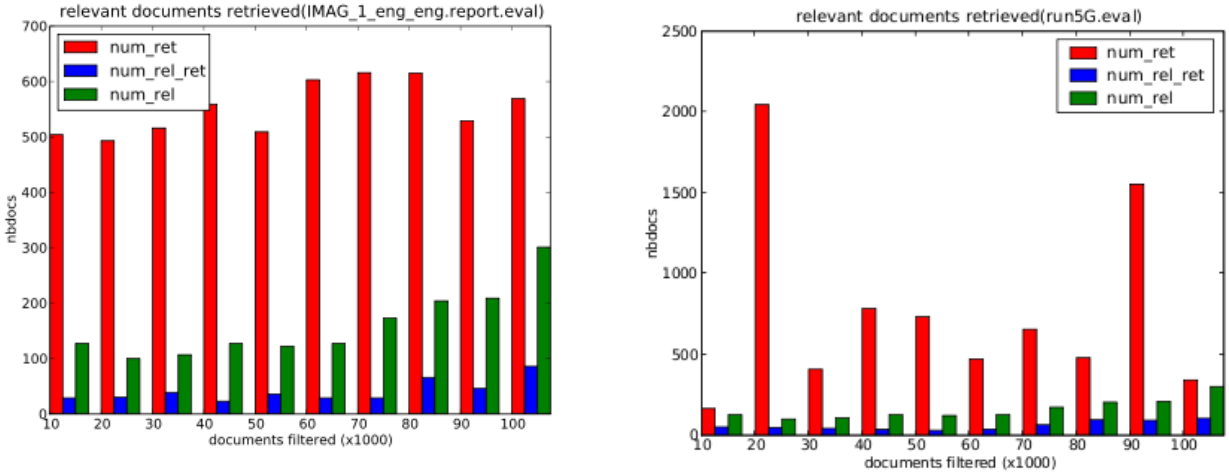

Figure 2: Number of documents retrieved for Run 1 and 2

efficiency. However, this setting meant that the thresholds for the 50 topics were different for the different sub-parts.

There are 1597 documents relevant to one or more topics in the INFILE data. The results for the different runs were evaluated based on different measures, namely, precision, recall, F-measure, linear utility, anticipation (added in 2009) and detection cost (see [1] and [2]). Utility is based on two parameters: importance given to a relevant document retrieved and the cost of a non-relevant document retrieved. Anticipation measure is designed to give more importance to systems that can find the first document in a given profile. Figure 1 give us an insight on the number of relevant documents retrieved during Run 1 and 2. We do not see a significant change for Run 1, in terms of the number of documents retrieved during the entire process. However, Run 2 returns much more documents between 10,000-20,000 and 80,000-90,000 documents. The evolution of these measures, computed at different times in the process, after each 10,000 documents, are given in the Figure 2. The curve, at the bottom represents the detection cost. Similarly, for Run 1, the curve just above the one meant for detection cost, describes the anticipation. For Run 1, all of the measures randomly vary but increase significantly as compared to the initial values (for example, 0.04 in

Table 1: Detail about the different runs

	Name	Campaign	Algorithm	Doc. ret	Doc. ret - relevant
Run 1	IMAG_1	Batch 09	Batch (w/o feedback)	5513	413
Run 2	run5G	Online 08	Online (with feedback)	7638	601

Table 2: Run Scores

	Precision	Recall	F-measure	Linear Utility	Detection Cost	Anticipation
Run 1	0.256	0.295	0.206	0.205	0.002	0.430
Run 2	0.306	0.260	0.209	0.351	0.007	0.307

the beginning vs 0.125 at the end for anticipation, 0.12 to 0.19 for the F-measure etc.) during the course of the filtering process. For Run 2, all of the measures, except utility and precision (0.18 vs 0.30), randomly vary but remain the same at the end.

Table 1 describes the different runs along with the number of documents retrieved and the number of relevant documents found. The average score (the values are first computed for each of the 50 profiles and then averaged) is given in Table 2. We can note that Run 1 has the best recall (0.295) as compared with the second run. The F-measure for the two runs is roughly the same. However, Run 2 surpasses Run 1 in terms of average precision. The overall detection cost is very low in these two runs, with Run 1, being more economical. This is a strong point for these runs. The linear utility of Run 2 is greater than that of Run 1. On contrary, the anticipation for Run 1 (0.430) is significantly better than that of Run 2 (0.307).

We can easily conclude from these results, that for the online algorithm, the initial set of documents can be constructed in the same manner as that for the batch algorithm. In this case, we can find 10 nearest neighbors for each topic instead of relying on feedbacks from the server.

5 Conclusion

We have presented, in this paper, a simple extension of the kNN algorithm using thresholds to define a batch filtering algorithm. The results obtained can be deemed encouraging as the F-measure equals approximately 20%, for a collection of 100,000 documents and 50 topics, out of which only 1597 documents are relevant. In comparison with online results of 2008, we have a much better recall (almost 30% against 26% in 2008) along with a lower detection cost (0.002 vs 0.007) and a much better anticipation (0.430 vs 0.307). Considering the evolution of different measures, we had observed that the values for all of the measures increase, with the increase in the number of documents filtered. Furthermore, the comparison between batch and online algorithms indicate that, while building the initial set of documents, one can use the nearest neighbors for each topic, rather than relying on feedbacks from the server.

References

[1] Romaric Besancon, Stéphane Chaudiron, Djamel Mostefa, Olivier Hamon, Ismail Timimi, and Khalid Choukri. Overview of clef 2008 infile pilot track. In *Working Notes of the Cross Language Evaluation Forum (CLEF 2008), Aarhus, Denmark*, 17–19 September 2008.

[2] Romaric Besancon, Stéphane Chaudiron, Djamel Mostefa, Ismail Timimi, and Khalid Choukri. The infile project: a crosslingual filtering systems evaluation campaign. In ELRA, editor, *Proceedings of LREC'08*, Morocco, May 2008.

[3] Vincent Bodinier, Ali Mustafa Qamar, and Eric Gaussier. Working notes for the infile campaign : Online document filtering using 1 nearest neighbor. In *Working Notes of the Cross Language Evaluation Forum (CLEF 2008), Aarhus, Denmark*, 17–19 September 2008.

[4] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996.

[5] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *SIGIR '99*, pages 42–49, USA, 1999. ACM Press.