XMedia: Web People Search by Clustering with Machinely Learned Similarity Measures

Lorenza Romano
FBK-irst
via Sommarive 18
Povo Trento, Italy
romano@fbk.eu

Krisztian Buza
ISMLL
University of Hildesheim
Marienburger Platz 22
Hildesheim, Germany
buza@ismll.de

Claudio Giuliano
FBK-irst
via Sommarive 18
Povo Trento, Italy
giuliano@fbk.eu

Lars Schmidt-Thieme
ISMLL
University of Hildesheim
Marienburger Platz 22
Hildesheim, Germany
schmidt-thieme@ismll.de

ABSTRACT

In this paper we present an approach to person name disambiguation that clusters documents on the basis of textual features using cosine similarity and a machinely learned meta similarity measure. The approach achieves an F-measure of B-Cubed Precision and Recall of 0.74[1] on the Clustering Subtask for WePS-2. Such task consists of clustering a set of documents that mention an ambiguous person name according to the actual entities referred to that name.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Measurement, Performance, Experimentation

Keywords

Web People Search, WEPS, clustering, Entity Disambiguation, SVM

1. INTRODUCTION

Finding information about people on the World Wide Web is one of the daily activities of Internet users. Because of the high ambiguity of person names and the increasing amount of information on the web, the results of person name searching are often a mix of pages about different people sharing the same name. So the user is forced to browse the documents and identify those referring to the person she/he is actually looking for. An ideal search engine should present results in as many clusters as different people.

[1]Such result has been obtained after the official evaluation, the official result is 0.72 obtained with the 3rd run submitted xmedia_3 (see Table 1).

WWW2009, April 20-24, 2009, Madrid, Spain.

The SemEval 2007 Web People Search task [1] and the Clustering Subtask for WePS-2 [2] try to formally evaluate systems on this task. Systems receive as input a list of ranked web search results obtained using a possibly ambiguous person name as a query, and the expected output is a clustering of the web pages, where each cluster is assumed to contain all (and only those) pages that refer to the same individual.

Our assumption on this task is that most of the discriminating information to disambiguate people name mentions is in the immediate surroundings of the mentions themselves. As an example consider the 5th ranked web page for "Alvin Cooper" in the training data (Figure 1). The page contains a lot of non relevant information: on the left an index referring to other people, and in the lower part a list of events *"that affected the communities where Alvin Wheeler lived"* that does not contain any mentions of "Alvin Cooper".

The most discriminating pieces of information seem to be the ones on which we focus (Figure 2) having a first look at the web page. And they are the surroundings of the "Alvin Cooper" mentions, a photo and two paragraphs, one beside the photo *"Alvin Cooper Wheeler was born on 11/10/1928 in Lanett, AL and died on 6/18/2005 in Hiram, GA. He was 77 years old."* and the caption of the photo *"Alvin Cooper Wheeler 11/10/1928 - 6/18/2005 (Shown at age 71)"*.

Even if all the objects in the surroundings of the mentions seem good to discriminate people (e.g. images), in our participation to the task we focus on the textual part and we try to detect the geometry and extension of these mention surroundings.

As an early work in name disambiguation [3] we cluster documents using vector representations based only on lexical or bag-of-words content. As clustering algorithms we applied the Quality Threshold (QT) clustering algorithm [7] and a slightly different version of it which exploits the web page ranking. In our clustering algorithms, we applied both cosine similarity and a machinely learned meta similarity measure. In particular, we used support vector machines to learn a meta similarity measure over a set of standard similarity measures.

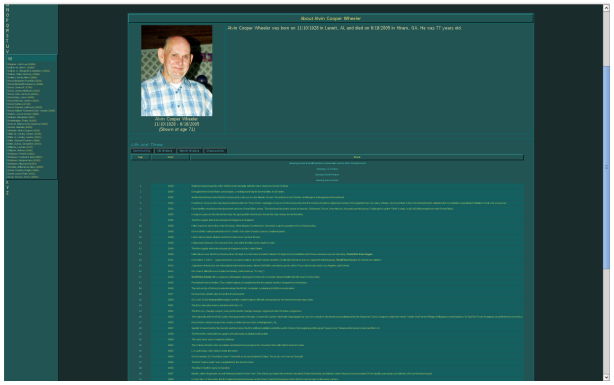

Figure 1: 5th ranked Alvin Cooper web page

Figure 2: Alvin Cooper focus

The paper is organised as follows. Section 2 provides a description of dataset and feature construction. Section 3 gives a description of the clustering algorithms applied. Section 4 presents results, and finally Section 5 draws conclusions.

2. DATASET CONSTRUCTION

We focused on the textual part so we first needed to extract the text from the web pages. According to our idea of surroundings, we decided to extract text from the HTML rendering using the Java Swing component `JTextPane` that allowed us to extract text substantially preserving its layout on the web page.

We obtained a text file for each web page, and we processed it by the OpenNLP sentence detector[2]. At this point for each web page we constructed a pair of files, one containing the sentences with at least one mention of the person name (*sents*) and one with the paragraphs (*pars*) containing at least one mention. We tokenized them using Java `breakIterator`.

Finally we constructed a feature vector for each *sents* file considering the set of unigrams occurring in sentences. To weight the features (unigrams) that represent each document we used the *self information*[3] calculated on the Web1T corpus [5]. If we were not able to construct a vector using sentences (e.g. because they contain only the mention of the person name and nothing else), we extended the region to be considered extracting features from the paragraph containing the mention. In this way however we totally miss documents not containing name mentions, for this reason on the test data we were not able to cluster about 130 pages.

By means of pair-wise comparisons of these document vector representations we computed a symmetric similarity matrix for the whole set of web pages of each person name, and we used this matrix as the input of the clustering algorithms.

On the training set we made some experiments adding title and URL information to the *sents* files without any relevant increase in performance. In order to detect sentences containing person name mentions we used an almost exact match. I.e., we look for case insensitive matching of "name surname" or "surname name" expressions, any generalization of such patterns to include full or abbreviated middle names has shown a decrease in performance.

3. CLUSTERING ALGORITHMS

Essentially, the number of different people entities that are present in a web people search cannot be known in advance, in general varying from very common names that have high ambiguity to famous people which seem to monopolize most of the documents in the web search results.

Cluster analysis requires specifying either the number of clusters a priori, or some kind of termination criterion to stop clustering. Due to the nature of the problem we decided for a threshold-based termination criterion for clustering. In particular, we applied Quality Threshold (QT) clustering algorithm [7] and a slightly different version of it, which we call QT^*. In both cases (QT and QT^*) we only had to set a threshold parameter for cluster's diameter. We estimated such threshold on the annotated training data (i.e. the data distributed for training/developing systems).

Differently from the original QT algorithm, our variant QT^* considers the set of documents relative to a specific person name according to the search engine ranking order $D_1, D_2, \ldots D_n$. It starts with the first (more prominent) document D_1 and determines which other following documents are similar to it, that is have a distance not exceeding the threshold. Documents similar to D_1 are put in the same cluster C_1. In general at step i the algorithm compares D_i with the following documents $D_{i+1}, \ldots D_n$ and add them to the C_i cluster if it exists, or create a C_i cluster containing D_i and add them to it.

Algorithm 1 QT^* algorithm

```
for i = 1 to n − 1 do
  for j = i + 1 to n do
    if D_j similar to D_i then
      if ∃C_i then
        C_i = C_i ∪ {D_j}
      else
        C_i = {D_i} ∪ {D_j}
      end if
    end if
  end for
end for
```

[2]http://opennlp.sourceforge.net

[3]In information theory [10] *self information* is a measure of the information content associated with the outcome of a random variable. It is expressed in a unit of information, for example bits.

Note, that QT^* does not necessarily produce disjoint clusters, in this sort of sense this is similar to fuzzy clustering algorithms, like FCM [4]. In fact QT^* doesn't use the transitive property to merge clusters, and even if a document D_j is similar to D_i and D_i was previously found similar to a preceding document such as D_1, it doesn't consider D_j referring to the same individual as D_1 if D_j was previously not found to be similar to D_1. In a certain sense we assume D_1 to be more representative of its cluster (i.e. of the person D_1 refers to) than D_i because D_1 is ranked better.

3.1 Machinely learned similarity measure

Clustering algorithms often use some kind of a similarity measure. One can run clustering algorithms like k-Medoids or QT for example with Euclidean or Cosine similarity. In fact, if one "invents" a new similarity measure, or one designs a similarity measure, which fits the current task, one can just use her/his own similarity measure in number of classic clustering algorithms.

What similarity measure fits the current task best is often not trivial. The design of a new similarity measure, or just the choice of an appropriate one from the high number of existing similarity measures can be time-consuming and needs massive domain experience. Alternatively one can learn a meta similarity measure machinely like in [6, 8]. This process can be scaled to large databases as described in [9].

In this section we describe how we learned a meta similarity measure machinely for the WePS-2 Clustering Subtask. Our meta similarity measure was basically a combination of some standard similarity measures using support vector machines.

As described before, the preprocessing of the data resulted in a set of weighted unigrams for each text. Based on these weighted unigrams, one can calculate similarities for each **pairs** of texts referring to the same name. We calculated the following similarity measures:

- **Cosine Similarity**
- **Euclidean Distance**
- **Adapted (weighted) Jaccard Coefficient**
  The weighted sum of the common unigrams of the both texts (i.e. $\Sigma_i w_{1,i} + \Sigma_i w_{2,i}$, where $w_{1,i}$ and $w_{2,i}$ denote the weights of **common** unigrams in the first and second text respectively) divided by the weighted sum of unigrams occuring in at least one of the texts (i.e. $\Sigma_j w_{1,j} + \Sigma_k w_{2,k}$, where $w_{1,j}$ and $w_{2,k}$ denote the weights of unigrams in the first and second text respectively).
- **L1 (Manhattan) Distance**
- **Weighted Sum of common unigrams.**
  $\Sigma_i w_{1,i} + \Sigma_i w_{2,i}$, where $w_{1,i}$ and $w_{2,i}$ denote the weights of common unigrams in the first and second text respectively.
- **Adapted Jaro Similarity**
  $\Sigma_i w_{1,i} / \Sigma_j w_{1,j} + \Sigma_k w_{2,k} / \Sigma_m w_{2,m}$, where $w_{1,i}$ and $w_{2,k}$ denote the weights of **common** unigrams in the first and second text respectively, $w_{1,j}$ and $w_{2,m}$ denote the weights of unigrams in the first and second text respectively

For each pair of texts referring to the same name we calculate the 6 similarity measures above. This results in a vector of length 6 for each pair of texts. For the texts belonging to the training corpus we know, if two texts are about the same person or not. Thus the vectors resulting from the training corpus are labeled training instances: the labels are 1 if the two texts in the pair are about the same person, 0 else. In the case of the test corpus, we calculate the vectors of the 6 similarity measures in the same way, but we do not know if two texts are about the same person or not. Thus in the case of the test corpus we have unlabeled test instances.

As machinely learned model we use the SVMreg and SMOreg support vector machine implementations from Weka[13, 11, 12]. They perform support vector regression, thus they are more suitable as similarity measures than a binary classification model: this way the test instances (as described above, each test instance represents a pair of documents) will be associated with a real number, which indicates how likely the two documents of the pair refer to the same person. A classification model, however, would only return a binary decision value, i.e. if the two documents are about the same person or not.

We use the support vector machines with polynomial kernel. To speed-up the process of learning the similarity measure, we discard all training instances where the weighted jaccard coefficient was less than 0.01 or greater than 0.99 (we assumed that in these cases the two texts of the pair are respectively about different or same persons). This can be regarded as a form of blocking [8, 9].

We used machinely learned similarity measure both in QT [7] clustering algorithm and in our variant QT^*.

4. RESULTS AND DISCUSSION

4.1 Experimental Settings

Due to time limitations, when submitting our results for the competition, we used the default WEKA-hyperparameters of support vector machines (exponent $e = 1$ and complexity constant $c = 1$) when machinely learning the meta similarity measure.

The diameter threshold for the clustering algorithm was learned on the training dataset using a hold-out subset of the training dataset. For cosine similarity we found that diameter thresholds 0.11 between 0.15 work best, for the machinely learned meta similarity measure we had the best results with diameter threshold of 0.20. The distribution of test data is significantly different from the training data, thus these learned threshold parameters are not optimal for the test data. Even though, to be fair, in section 4.2 we report our results on the test set using these settings. We refer to section 4.3 for a short discussion about handling this difference.

After the submission deadline for the competition, we tried to learn the hyperparameters (exponent, complexity constant) of the support vectors on a hold-out subset of the training data. We report our experiences in section 4.3.

4.2 Results

Our results are summarised in Table 1. According to the competition, we use the F-Measure resulting of B-Cubed Precision and Recall as quality measure. In the table with *xmedia_n* we denoted the runs submitted for the official evaluation.

Sim. measure	Clust. alg.	Diam. thres.	BEP	BER	F	run
cosine	QT	0.11	0.94	0.44	0.55	
cosine	QT	0.15	0.96	0.41	0.53	
cosine	QT^*	0.11	0.82	0.66	0.72	xmedia_3
cosine	QT^*	0.13	0.87	0.61	0.70	xmedia_4
cosine	QT^*	0.15	0.90	0.56	0.67	xmedia_5
SMO	QT	0.18	0.88	0.51	0.60	xmedia_1
SMO	QT	0.20	0.89	0.50	0.60	xmedia_2
SMO	QT^*	0.20	0.75	0.72	0.73	
SVM	QT	0.20	0.89	0.50	0.61	
SVM	QT^*	0.20	0.77	0.74	0.74	

Table 1: Experimental Results

4.3 Outlook

As mentioned in 4.1, we tried to learn the hyperparameters (complexity constant, exponent) of the support vector machines (SVMs) using a hold-out subset of the training dataset. Surprisingly, this did not lead to the expected improvement. This is probably due to the fact, what has been remarked by many participants of the competition, that the distribution of the test data significantly differs from the distribution of the training data. As a result of this difference, the diameter threshold parameter learned on the training set, was far from the optimal when applying clustering with cosine similarity on the test set.

Machine learning – clustering can also be regarded as unsupervised or semi-supervised machine learning (e.g. parameters of clustering algorithm can be learned) – assumes, that the training and test data are more or less from the same distribution. Of course, in practise, these distributions are often only approximately equal. However, in the other extreme case, if training and test distributions would differ totally, one could not learn anything machinely.

Consequently, when studying the task of person name disambiguation from the machine learning point of view in the future, one could consider to experiment with other splits of test and training data (i.e. merging the training and test data of the competition and splitting them differently, so that training and test splits have similar distributions).

Alternatively the WEPS challenge can be regarded as an interesting transfer learning problem, when the model learned on the training data has to be adapted in order to be applied successfully on the test data having a different distribution.

As another direction of future work, we consider machinely learning the count of clusters a priori and trying other clustering algorithms which exploit this information, like k-Medoids, or hierarchical clustering.

As for the textual features, we used a combination of mention surroundings with different extensions (*sents* and *pars*), in the future we would try to learn automatically the size of the regions to be considered in term of some spatial distance (e.g. characters, lines) from the mention. We have also to deal with web pages without complete name mentions.

5. CONCLUSIONS

In this paper we presented an approach to person name disambiguation that clusters documents on the basis of textual features using cosine similarity and a machinely learned meta similarity measure. As clustering algorithm we used an own variant of QT, called QT^*.

6. ACKNOWLEDGMENTS

This work has been funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

7. REFERENCES

[1] J. Artiles, J. Gonzalo, and S. Sekine. The SemEval-2007 WePS Evaluation: Establishing a benchmark for the Web People Search Task. In *Proceedings of the Fourth International Workshop on Semanti c Evaluations (SemEval-2007)*, pages 64–69, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[2] J. Artiles, J. Gonzalo, and S. Sekine. WePS 2 Evaluation Campaign: overview of the Web People Search Clustering Task. In *Proceedings of the 2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, April 2009.

[3] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the Vector Space Model. In *Proceedings of the 17th international conference on Computational linguistics*, pages 79–85, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[4] J. C. Bezdek, R. Ehrlich, and W. Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[5] T. Brants and A. Franz. Web 1T 5-gram corpus version 1.1. Technical report, Google Research, 2006.

[6] P. Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In Y. Li, B. Liu, and S. Sarawagi, editors, *KDD*, pages 151–159. ACM, 2008.

[7] L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring Expression Data: Identification and Analysis of Coexpressed Genes. *Genome Research*, 9(11):1106–1115, 1999.

[8] S. Rendle and L. Schmidt-Thieme. Object identification with constraints. In *ICDM*, pages 1026–1031. IEEE Computer Society, 2006.

[9] S. Rendle and L. Schmidt-Thieme. Scaling Record Linkage to Non-uniform Distributed Class Sizes. In T. Washio, E. Suzuki, K. M. Ting, and A. Inokuchi, editors, *PAKDD*, volume 5012 of *Lecture Notes in Computer Science*, pages 308–319. Springer, 2008.

[10] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[11] S. Shevade, S. Keerthi, C. Bhattacharyya, and K. Murthy. Improvements to the SMO Algorithm for SVM Regression. In *IEEE Transactions on Neural Networks*, 1999.

[12] A. Smola and B. Schoelkopf. A tutorial on support vector regression. Technical report, 1998. NeuroCOLT2 Technical Report NC2-TR-1998-030.

[13] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, 2005.