WIT: Web People Search Disambiguation using Random Walks

José Iria, Lei Xia, Ziqi Zhang
The University of Sheffield
211 Portobello Street
Sheffield S1 4DP, United Kingdom
{j.iria, l.xia, z.zhang}@sheffield.ac.uk

Abstract

In this paper, we describe our work on a random walks-based approach to disambiguating people in web search results, and the implementation of a system that supports such approach, which we used to participate at Semeval'07 Web People Search task.

1 Introduction

Finding information about people on the Web using a search engine is far from being a quick and easy process. There is very often a many-to-many mapping of person names to the actual persons, that is, several persons may share the same name, and several names may refer to the same person. In fact, person names are highly ambiguous: (Guha and Garg, 2004) reports that only 90.000 thousand different names are shared by 100 million people according to the U.S. Census Bureau. This creates the need to disambiguate the several referents typically found in the web pages returned by a query for a given person name.

The Semeval'07 Web People Search challenge (Artiles et al., 2007) formally evaluated systems on this task. In this paper, we describe our work on a random walks-based approach to disambiguating people in web search results, heavily influenced by (Minkov et al., 2006). This particular model was chosen due to its elegance in seamlessly combining lexico-syntactic features local to a given webpage with topological features derived from its place in the network formed by the hyperlinked web pages returned by the query, to arrive at one single measure of similarity between any two pages.

2 Proposed Method

In a nutshell, our approach 1) uses a graph to model the web pages returned by the search engine query, 2) discards irrelevant web pages using a few simple hand-crafted heuristics, 3) computes a similarity matrix for web pages using random walks over the graph, and 4) finally clusters the web pages given the similarity matrix. The next subsections detail these steps.

2.1 Web People Search Graph

We build a directed weighted typed graph from the corpus. The graph is a 5-tuple $G = (V, E, t, l, w)$, where V is the set of nodes, $E : V \times V$ is the ordered set of edges, $t : V \rightarrow T$ is the *node type function* ($T = \{t_1, \ldots, t_{|T|}\}$ is a set of types), $l : E \rightarrow L$ is the *edge label function* ($L = \{l_1, \ldots, l_{|L|}\}$ is a set of labels), and $w : L \rightarrow \mathbb{R}$ is the *label weight function*. We structure our problem domain with the types and labels presented in Figure 1.

In order to transform the text into a graph that conforms to the model shown, we take the output of standard NLP tools and input it as nodes and edges into the graph, indexing nodes by string value to ensure that identical contents for any given node type are merged into a single node in the graph. To process the corpus, we run a standard NLP pipeline seperately over the metadata, title and body of the HTML pages, but not before having transformed its contents as much into plain text as possible, by removing HTML tags, javascript code, etc. The pipeline used is composed of tokenization, removal of stop words and infrequent words, and stemming with Porter's algorithm. The resulting graph at this

Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 480–483,
Prague, June 2007.

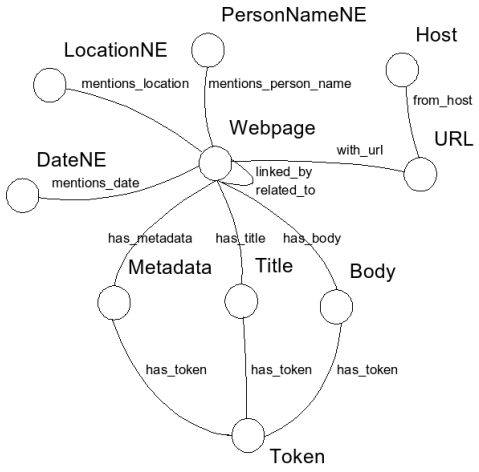

Figure 1: The data representation model adopted

stage consists of the nodes of type *Token*, *Webpage*, *Metadata*, *Title* and *Body*, properly interconnected. We then run a named entity recognizer to associate NE tags to the respective documents, via the constituent words of the NE. The information about the original *URL* of page is given by the corpus, while *Host* is trivially obtained from it. We finalise the graph by inserting an edge of type *linked_by* between any web page linked by another in the corpus, and an edge of type *related_to* between any web page related to another in the corpus, as given by Google's *related:* operator.

For the named entity recognition task, we have compared GATE and OpenNLP toolkits. Although both toolkits show comparable results, OpenNLP demonstrated faster performance. Moreover, some documents in the corpus consisted of very extensive lists of names (e.g. phonebook records) which slowed the NER to a halt in practice. To compensate for this, we applied a chunking window at the beginning and end of each body content and around each occurrence of the person name being considered (and its variants determined heuristically). The window size used was 3000 characters in length, and an overlap between windows results in a merged window.

2.2 Discarding using heuristics

To discard irrelevant documents within the corpus, we manually devised two heuristics rules for classification by observing the training data at hand. The heuristics are 1) whether the page has content at all, 2) whether the page contains at least one appearance of mentioned person name with its variants. This simple classification showed high precision and low recall on the training data. We also tried a SVM-based classifier trained on a typical bag-of-words feature vector space obtained from the training data, but found the such classifier not to be sufficiently reliable.

2.3 Random Walks Model

We aim to determine the similarity between any two nodes of type *Webpage* in the graph. In our work, similarity between two nodes in the graph is obtained by employing a random walks model. A random walk, sometimes called a "drunkard's walk," is a formalization of the intuitive idea of taking successive steps in a graph, each in a random direction (Lovász, 2004). Intuitively, the "harder" it is for a drunkard to arrive at a given webpage starting from another, the less similar the two pages are.

Our model defines weights for each edge type, which, informally, determine the relevance of each feature type to establish a similarity between any two pages. Let $L_{t_d} = \{l(x,y) : (x,y) \in E \wedge T(x) = t_d\}$ be the set of possible labels for edges leaving nodes of type t_d. We require that the weights form a probability distribution over L_{t_d}, i.e.

$$\sum_{l \in L_{t_d}} w(l) = 1 \tag{1}$$

We build an adjacency matrix of locally appropriate similarity between nodes as

$$W_{ij} = \begin{cases} \sum_{l_k \in L} \frac{w(l_k)}{|(i,\cdot) \in E : l(i,\cdot) = l_k|}, & (i,j) \in E \\ 0, & otherwise \end{cases} \tag{2}$$

where W_{ij} is the ith-line and jth-column entry of W, indexed by V. Equation 2 distributes uniformly the weight of edges of the same type leaving a given node. We could choose to distribute them otherwise, e.g. we could distribute the weights according to some string similarity function or language model (Erkan, 2006), depending on the label.

We associate the state of a Markov chain to every node of the graph, that is, to each node i we associate the one-step probability $P^{(0)}(j|i)$ of a random walker traversing to an adjacent node j. These

probabilities are expressed by the row stochastic matrix $D^{-1}W$, where D is the diagonal degree matrix given by $D_{ii} = \sum_k W_{ik}$. The "reinforced" similarity between two nodes in the graph is given by the t-step transition probability $P^{(t)}(j|i)$, which can be simply computed by a matrix power, i.e., $P^{(t)}(j|i) = [(D^{-1}W)^t]_{ij}$.

Note that t should not be very large in our case. The probability distribution of an infinite random walk over the nodes, called the stationary distribution of the graph, is uninteresting to us for clustering purposes since it gives an information related to the global structure of the graph. It is often used as a measure to rank the structural importance of the nodes in a graph (Page et al., 1998). For clustering, we are more interested in the local similarities inside a cluster of nodes that separate them from the rest of the graph. Also, in practice, using $t > 2$ leads to high computational cost requirements, as the matrix becomes more dense as t grows.

Equation 2 introduces the need to learn the function w. In other words, we need to tune the model to use the most relevant features for this particular task. Tuning is performed on the training set by comparing the standard purity and inverse purity measures of the clusters against the gold standard, and using a simulated annealing optimization method as described in (Nie et al., 2005).

2.4 Commute Time Distance

The algorithm takes as input a symmetric similarity matrix S, which we derive from the random walk model of the previous section as follows. We compute the Euclidean Commute Time (ECT) distance (Saerens et al., 2004) of any two nodes of type *Webpage* in the graph. The ECT distance is (also) based on a random walk model, and presents the interesting property of decreasing when the number of paths connecting two nodes increases or when the length of any path decreases, which makes it well-suited for clustering tasks. Another nice property of ECT is that it is non-parametric, so no tuning is required here. ECT has connections with principal component analysis and spectral theory (Saerens et al., 2004).

In particular, we are interested in the *average commute time* quantity, $n(i, j)$, which is defined as the average number of steps a random walker, starting in state i, will take before entering a given state j for the first time, and go back to i. That is, $n(i, j) = m(j|i) + m(i|j)$, where the quantity $m(j|i)$, called the *average first-passage time*, is defined as the average number of steps a random walker, starting in state i, will take to enter state j for the first time. We compute the average first-passage time iteratively by means of the following recurrence:

$$\begin{cases} m(i|j) = 1 + \sum_{k=1,k\neq i}^{|V|} P^{(t)}(k|j)m(i|k), & j \neq i \\ m(i|i) = 0 \end{cases} \tag{3}$$

where $P^{(t)}(\cdot|\cdot)$ is the t-step transition probability of the random walk model over G presented in the previous section.

Informally, we may regard the random walk model presented in the previous section as a "refined" document similarity measure, replacing, e.g., the typical TF-IDF measure with a measure that works in a similar way but over all features represented in the graph, whereas we can regard the ECT measure presented in this section as a "booster" to a basic clustering techniques (cf. next section), achieved by means of coupling clustering with a random walk-based distance which has been shown to be competitive with state-of-the-art algorithms such as spectral clustering (Luh Yen et al., 2007).

2.5 Clustering

Clustering aims at partitioning n given data points into k clusters, such that points within a cluster are more similar to each other than ones taken from different clusters. An important feature of the clustering algorithm that we require for the problem at hand is its ability to determine the number k of natural clusters, since any number of referents may be present in the web search results. However, most clustering algorithms require this number to be an input, which means that they may break up or combine natural clusters, or even create clusters when no natural ones exist in the data.

We use a form of group-average agglomerative clustering as described in (Fleischman and Hovy, 2004), shown in Table 1, which works fast for this problem. A difficult problem (with any clustering approach) has to do with the number of initial clusters or, alternatively, with setting a threshold for when to stop clustering. This threshold could po-

Input: symmetric similarity matrix S, threshold θ
Output: a set of clusters C
1. $(i, j) \leftarrow$ find min score in S
2. if $S_{ij} > \theta$ then exit
3. place i and j in the same cluster in C (merging existing clusters of i and j if needed)
4. (average pairs of edges connecting to nodes i,j from any node k)
4a. $S_{ik} \leftarrow (S_{ik} + S_{jk})/2, k \neq i, j$
4b. $S_{ki} \leftarrow (S_{ki} + S_{kj})/2, k \neq i, j$
5. remove j-th column and j-th line from S (effectively merging nodes i,j into a single node)
6. goto 1
7. return clusters C

Table 1: The simple group-average agglomerative clustering algorithm used

tentially also be optimized using the training data; however, we have opted for unsupervised heuristics to do that, e.g. the well-known Calinski&Harabasz stopping rule (Calinski&Harabasz, 1974).

3 Results Obtained

The results obtained by the system are presented in the following table. The evaluation measures used were f-measure, purity and inverse purity - for a detailed description refer to the task description (Artiles et al., 2007).

aver_f05	aver_f02	aver_pur	aver_inv_pur
0,49	0,66	0,36	0,93

The results are below average for this Semeval task, and should not be regarded as representative of the approach adopted, since the authors have had limited time available to ensure a pristine implementation of the whole approach.

References

Artiles, J., Gonzalo, J., & Sekine, S. (2007). The SemEval-2007 WePS Evaluation: Establishing a benchmark for the Web People Search Task. *In Proceedings of Semeval 2007, Association for Computational Linguistics*.

Calinski and Harabasz (1974). A Dendrite Method for Cluster Analysis *Communications in Statistics, 3(1), 1974, 1-27*.

Erkan, G. (2006). Language model-based document clustering using random walks. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* (pp. 479–486). Association for Computational Linguistics.

Fleischman, M. B., & Hovy, E. (2004). Multi-document person name resolution. *Proceedings of the ACL 2004*. Association for Computational Linguistics.

Guha, R. V., & Garg, A. (2003). Disambiguating People in Search. *TAP: Building the Semantic Web.*. ACM Press.

Luh Yen, Francois Fouss, C. D., Francq, P., & Saerens, M. (2007). Graph nodes clustering based on the commute-time kernel. *To appear in the proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*. Lecture Notes in Computer Science (LNCS).

Minkov, E., Cohen, W. W., & Ng, A. Y. (2006). Contextual search and name disambiguation in email using graphs. *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 27–34). ACM Press.

Nie, Z., Zhang, Y., Wen, J. R., & Ma, W. Y. (2005). Object-level ranking: Bringing order to web objects. *Proceedings of WWW'05*.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). *The pagerank citation ranking: Bringing order to the web* (Technical Report). Stanford Digital Library Technologies Project.

Saerens, M., Fouss, F., Yen, L., & Dupont, P. (2004). The principal components analysis of a graph, and its relationships to spectral clustering. *Proceedings of the 15th European Conference on Machine Learning*.

László Lovász (1993). Random Walks on Graphs: A Survey. *Combinatorics, Paul Erdos is Eighty (Volume 2), Keszthely (Hungary), 1993, p 1-46.*.