Clustering Web People Search Results using Fuzzy Ants

E. Lefever[*,a,b], T. Fayruzov[b], V. Hoste[a,b], M. De Cock[b,c]

[a] *LT3 Language and Translation Technology Team, University College Ghent, Groot-Brittanniëlaan 45, 9000 Gent, Belgium*
[b] *Department of Applied Mathematics and Computer Science Ghent University, Krijgslaan 281 (S9), 9000 Gent, Belgium*
[c] *Institute of Technology, University of Washington Tacoma, WA–98402, USA*

Abstract

Person name queries often bring up web pages that correspond to individuals sharing the same name. The Web People Search (WePS) task consists of organizing search results for ambiguous person name queries into meaningful clusters, with each cluster referring to one individual. This paper presents a fuzzy ant based clustering approach for this multi-document person name disambiguation problem. The main advantage of fuzzy ant based clustering, a technique inspired by the behavior of ants clustering dead nestmates into piles, is that no specification of the number of output clusters is required. This makes the algorithm very well suited for the Web Person Disambiguation task, where we do not know in advance how many individuals each person name refers to. We compare our results with state-of-the-art partitional and hierarchical clustering approaches (k-means and Agnes) and demonstrate favorable results. This is particularly interesting as the latter involve manual setting of a similarity threshold, or estimating the number of clusters in advance, while the fuzzy ant based clustering algorithm does not.

Key words: Web People Search, Web Person Disambiguation, document clustering, fuzzy ant based clustering

1. Introduction

Searching for people on the web is a very popular online activity that is receiving increasing support from specialized search engines. Besides the major search engine Yahoo! offering a people search service[1], there are a fair number

[*] Corresponding author
Email addresses: els.lefever@hogent.be (E. Lefever), timur.fayruzov@UGent.be (T. Fayruzov), veronique.hoste@hogent.be (V. Hoste), martine.decock@UGent.be (M. De Cock)

[1] http://people.yahoo.com/

of smaller search engines, such as pipl[2] and spock[3], whose core business is Web People Search. One significant problem faced by all of these "web people finders" is person name ambiguity due to the fact that one person name can refer to different individuals. Given the high ambiguity of person names (the most common name in the UK "David Jones" e.g. refers to 157,630 different individuals [4]) and the rapidly increasing amount of information on the web, the person name ambiguity problem becomes increasingly important in many natural language processing (NLP) fields, such as information extraction, cross-document summarization and question-answering. In the context of this paper, Web People Search (WePS) refers to the problem of organizing all search results for a person name query into meaningful clusters that group together all web pages corresponding to an individual.

At first sight the Web People Search or "Web Person Disambiguation" problem can be linked to the better known Word Sense Disambiguation (WSD) problem, as both try to disambiguate instances of usage based on context information. The main difference between both natural language processing applications is the sense inventory involved, which is rather fixed for common words (the target of WSD), but not known in advance for person names, where we do not know in advance how many unique persons are involved. In this way, the WePS problem has more in common with Word Sense Discrimination, where similar examples of word senses are grouped together. As a consequence, it is much harder to develop a classification algorithm for person name disambiguation (since the classes are not known in advance), which motivates the proposed clustering approach.

As mentioned above, a particularly challenging aspect of the WePS problem is that it is usually not known beforehand how many clusters to expect in the search results for a person name query. Some names occur rarely, while others are shared by a large group of people. Furthermore, celebrity names tend to monopolize search results. For instance, while there may be many people out there with a not very unusual name like "Michael Jordan", the first 100 results returned by a search engine might be for a large part about the American basketball player and about the computer science professor with the same name at UC Berkeley. This makes it hard to predict how many different individuals will be covered in the first search results, which is problematic since most clustering algorithms require an estimate of the number of clusters that needs to be found in the data.

In this paper we propose the use of a fuzzy ant based clustering algorithm that does not require specification of the number of clusters. Ant based clustering algorithms are inspired by the clustering of dead nestmates, as observed to be done by several ant species under laboratory conditions. Without direct communication about where to gather the corpses, ants manage to cluster all corpses into one or two piles. The conceptual simplicity of this phenomenon,

[2] http://www.pipl.com/
[3] http://www.spock.com/

together with the lack of centralized control and the lack of a need for a priori information, is the main motivation for using clustering algorithms inspired by this behavior. Real ants are, because of their very limited brain capacity, often assumed to reason only by means of rules of thumb. The fuzzy ant based clustering method proposed in [37] is inspired by the idea that the behavior of ants in picking up and dropping items can be expressed flexibly by fuzzy IF–THEN rules. As in other ant based clustering algorithms, no initial partitioning of the data is needed, nor need the number of clusters be known in advance. The machinery of approximate reasoning from fuzzy set theory endows the ants with some intelligence. As a result, on each time step the ants are capable of deciding for themselves whether to pick up or drop an item or a heap, and a clustering automatically emerges from this process.

The fuzzy ant based clustering algorithm has been successfully applied to group search results for ambiguous queries such as *rem*, *travelling to Java*, and *salsa* [37]. Snippets[4] returned by the search engine are turned into bags of words with a binary weighting scheme and subsequently clustered. While this method is useful for detecting documents that discuss a similar topic, the WePS problem calls for a different approach. Indeed, different web pages about the same individual might contain a significantly different kind of content, such as a professional web page versus an account page on a social networking site. Furthermore, the presence of a biographical fact or a telephone number by itself might give an important clue to the identity of the individual at hand. While such information is rather rare, it is very reliable: we can be almost sure that identical phone numbers in different documents refer to the same person. In our approach we therefore represent every web document as a rich feature vector, containing biographical facts, named entities[5], URL and email addresses, IP location, as well as distinctive keywords from the title and the snippet, and a weighted bag of words extracted from the full text of the document.

Existing approaches to Web Person Disambiguation use either a classification or a more classical (partitional or hierarchical) clustering approach. The main drawback of these approaches is that the number of classes/clusters must be defined in advance. We contribute to solving the WePS task by using a fuzzy ant based clustering algorithm and by comparing its performance with a partitional and an agglomerative hierarchical clustering approach (k-means and Agnes). This use of a fuzzy ant based algorithm overcomes the main drawback of existing approaches that require the number of clusters to be defined in advance.

This paper is organized as follows: in Section 2 we give an overview of related research. In Section 3 we provide details about the construction of the feature vectors and the keyword matrices, while the flow of the clustering algorithm is explained in Section 4. Section 5 describes experimental results on the WePS

[4] A snippet is a small sample of content shown to users on the search results page.

[5] Named entities are names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. The NLP task of Named Entity Recognition (NER), which is a subtask of information extraction, aims to locate and classify words into these predefined categories.

data sets from the SemEval competitions. In Section 6, we summarize our main findings and suggest future work.

2. Related research

The Web Person Disambiguation task has already been approached from different angles. The approaches differ from each other concerning the data that is used, being the features that are extracted from the document as well as external data that can be used, and the type of algorithm that is applied, viz. classification versus clustering, different clustering techniques, etc. One direction leads to unsupervised clustering based on a rich feature space. Mann and Yarowsky [26], for example, extract features containing biographical facts, such as birth and death place or date, that are combined with associated names such as family and employment relationships, and nationality. To extract these patterns, they further develop the method of Ravichandran and Hovy [35] that bootstraps information extraction patterns from a set of examples (e.g. for birth date an example query such as "Mozart, 1756" returns a set of examples that can be used for generating patterns, which are then used to extract the biographical information from the data). They finally combine learned patterns and hand-coded rules. In this way each instance of the ambiguous name is represented by a feature vector and clustering is done by grouping the most similar feature vectors, using bottom-up centroid agglomerative clustering based on the cosine similarity distance.

Pedersen [32] presents an unsupervised approach that uses statistically significant bigrams in the documents to be clustered ("significant" meaning that the log-likelihood ratio between the two words is greater than a given threshold). A matrix based on these bigrams is built with the rows representing the first word, and the columns representing the second word in the bigram; each cell contains the log-likelihood ratio associated with the bigram. Because of its large size and sparsity, singular value decomposition is applied to reduce the dimensionality. In this way, the words that make up the columns are reduced from a word level feature space into a concept level semantic space that is supposed to capture indirect relationships between words. Afterwards, instances (documents) that have similar context vectors are placed into the same cluster by means of a hybrid clustering approach called repeated bisections clustering. This clustering algorithm combines a hierarchical divisive approach with partitioning: it starts with all instances in a single cluster and at each iteration a cluster is bisected using the standard k-means method with $k = 2$.

Other researchers use additional web resources for better measuring document similarities. Vu et al. [30] use web directories such as Dmoz's www.dmoz.org, Google's directory.google.com, and Yahoo!'s dir.yahoo.com as an additional knowledge base. These collections of web documents are categorized according to different topics and can be used to enrich the extractable information in web documents. In this way, the authors determine the topic of the web document and link other documents containing similar contexts. Bollegala et al. [10] try to disambiguate personal names by using automatically extracted

keyphrases. The proposed unsupervised algorithm extracts key phrases from the web that uniquely identify a person. These phrases are then added to the person name query to narrow down the web search. Clustering is done with a group average agglomerative clustering algorithm, which is a hybrid of single-link and complete-link clustering. Clustering stops when the cluster quality drops below a preset threshold that is experimentally obtained.

A problem faced by most clustering approaches is how to accommodate nominal features such as string features. Such features pose a problem to the similarity measure computation because typically such measures accept only numerical input. Several approaches were proposed to overcome this problem. Lee and Brouwer [22] propose a method to integrate a similarity measure between ordinal features in the fuzzy c-means algorithm. However, ordinal features assume an underlying ordering relation such as *child,youth,adult*; many nominal features do not fit into this category. Another approach aimed at dealing with nominal features is explained in Guha et al. [18], who propose a clustering algorithm called ROCK that groups similar sets of categorical features, i.e. features that take values from a certain set. However, this approach is not applicable when every feature uses its own category, as it is not possible to infer the relations among the values of one category from the feature vector.

The Web Person Disambiguation task can also be considered as a multi-document coreference problem. Bagga and Baldwin [5] opt for personal name disambiguation using document context profiles or vectors. They first do coreference resolution within each document in order to form coreference chains. Next they take the surrounding context of these reference chains for creating summaries about each entity and convert these summaries into a bag of words. Documents get clustered using the standard vector space model; each vector contains the words that occur in the window around the ambiguous name and the similarity is measured using the cosine measure. Clustering continues until a predefined threshold is reached.

Web Person Disambiguation can also be treated as a classification task. Fleischman and Hovy [17], for example, introduce a maximum entropy model in which a binary classifier determines whether two concept-instance pairs (the concept is represented by a complex noun phrase such as "president of the united states", the instance by a name) refer to the same individual. The classification is based on a trained probability threshold. The features they use for the classification are related to the names themselves (common/uncommon names, famous names), web features (e.g. number of hits returned for a query using the instance name and the heads of each concept in the match pair), overlap features (similarity between the sentential contexts of the concept-instance pairs, expressed in word overlap score) and semantic features (semantic relatedness of the concepts using the WordNet ontology).

In previous work we presented a hybrid approach that combined the results of both classification and clustering [23]. First, supervised classification based on feature vectors containing binary and symbolic disambiguation information on pairs of documents is done by means of the eager RIPPER rule learner [12] which induces a set of easily understandable if-then classification rules for the

minority class, and a default rule for the remaining class. Second, different clustering approaches are applied to the weighted keyword matrices. In a final step, the seed clusters obtained by the classification algorithm are enhanced by the results from the clustering algorithms.

Our present work, that further elaborates the approach presented in [24], is mostly related to the research of Mann and Yarowsky, in the sense that we also do unsupervised clustering on a rich feature space. It is different from the approaches mentioned above because of the clustering algorithm we use and because of the integration of very different features. In particular, we use biographical facts, named entity overlap, IP location information, URL and email overlap, and weighted keywords, combined into one feature vector per document. As an alternative to existing approaches that typically use a top-down or bottom-up agglomerative clustering algorithm, we have experimented with a fuzzy ant based algorithm that is able to cluster without using any kind of a priori information, such as the required number of clusters.

3. Feature vector construction

For the construction of the feature vectors we extract information from the content of the web pages themselves, the snippets and the titles. Our choice of features is inspired by previous research (see Section 2) where certain features have already shown to be very well suited for the task, as well as new features (e.g. location features) that are likely to be informative to distinguish a specific person.

All data is preprocessed by means of a memory-based shallow parser (MBSP) [13]. The following preprocessing steps are taken. Tokenization (i.e. splitting punctuation from adjoining words) is performed by the MBSP using a rule based system with regular expressions. Part-of-Speech (PoS) tagging and text chunking is performed by the memory based tagger MBT [14], which was trained on the Wall Street Journal corpus in the Penn Treebank [28], the Brown corpus [21] and the Air Travel Information System (ATIS) corpus [19]. During text chunking syntactically related words are combined into non-overlapping phrases.

On the basis of the preprocessed data we construct a rich feature vector that combines biographical facts and distinctive characteristics for a given person, a list of weighted keywords and metadata information about the web page.

3.1. Extracted features

We extract the following features from the preprocessed web pages:

- **Biographical facts**, such as date of birth and death. We have training data available in which these kind of biographical facts have been manually labeled (see Section 5 for more details on the training data). This training data serves as the basis for the development of a regular-expression based procedure similar to the approach used by Ravichandran and Hovy [35].

- **Information on named entities** is extracted on the basis of the shallow syntactic information provided by the memory-based shallow parser and additional gazetteer information (in-house constructed lists of male and female person names and lists of locations). Three named entity features are extracted:
  1. The first feature aims at the extraction of further interesting name information, e.g other surnames or family names of the person in focus, leading to the extraction of for example *Ann Hill Carter Lee* and *Jo Ann Hill* for the person *Ann Hill*.
  2. The second named entity feature contains all geographic locations named in the particular document.
  3. The last feature extracts all other named entity information, such as other person names, company names, etc.

  Another location feature we extract is the **IP location**, based on the simple assumption that if two documents are hosted in the same city, they most probably refer to the same person (but not vice versa). For converting IP-addresses to city locations, we use the MaxMind GeoIP(tm) open source database[6].

- As URLs and email addresses can be used to identify a person, we also extract **URL, email and domain addresses** from the web pages. In order to do so, we combine pattern matching rules and markup information such as the HTML <href> tag. The URL of the document itself is added to the set of URL links. Some filtering on the list is performed concerning the length of extracted URLs (to exclude garbage) and the content (to exclude non-distinctive URL addresses such as index.html).

Table 1 gives an overview of all extracted features. In an early version of the algorithm, we also extracted telephone and fax numbers. However, as the WePS2 data set (see Section 5) contains very little information of this kind, we decided to omit it in the final version as presented in this paper. For similar reasons, we rejected an early idea of mapping locations to their geographic coordinates (which can be done using e.g. the GeoNames geographic database[7]) and computing the distance between them.

Example 1. Below are some examples of features extracted from web search results for the name "Alexander Macomb". All web pages listed here, which are referred to as "Doc" followed by a 3 digit identifier, are from the SemEval data sets. We refer to Section 5 for more details about these data sets. The list below is not exhaustive.

[6] http://www.maxmind.com/app/geolitecity
[7] http://www.geonames.org/about.html

Biographical Facts	Date of birth Date of death
Named Entities	Person name Locations Other named entities
URL and Email	URL addresses Email addresses Domain names
Location	IP addresses

Table 1: Overview of features extracted from the web documents

PERSON NAME: Alexander Macomb

1. Biographical facts
   (a) Date of birth
      Doc 004: was born in 1717
      Doc 009: 4 February 1888
      Doc 017: (1783
      Doc 019: (1782
      Doc 064: (1748
   (b) Date of death
      Doc 004: death in 1796
      Doc 009: DEATH : 3 March 1970
      Doc 010: death in 1841
      Doc 014: death in 1841
      Doc 019: died on June 25 , 1842
2. Named entities
   (a) Name
      Doc 004: General Alexander Macomb
      Doc 007: General Alexander Macomb
      Doc 075: Alexander Macomb Miller
      Doc 016: Colonel Alexander Macomb
   (b) Location
      Doc 004: Belle Isle, Cherokee, Chicago, Detroit, Grosse Ile, ...
      Doc 076: Belle Isle, Detroit, Grosse Isle, Macomb County, ...
3. URL and Email
   (a) URL page
      Doc 000: http://www.answers.com/main/business.jsp
      Doc 000: http://librarians.answers.com
      Doc 000: http://www.answers.com/topic/macomb-illinois

Doc 000: http://www.answers.com/topic/alexander-macomb-j

(b) Email

Doc 004: mlloyd@sms-va.com

Doc 005: mlloyd@sms-va.com

Doc 014: sales@galleryofhistory.com

Doc 008: booklists@historiclakes.org

Doc 054: editor@wreckhunter.net

(c) Domain

Doc 000: librarians.answers.com

Doc 000: teachers.answers.com

Doc 006: teachers.answers.com

4. IP location

Doc 03: www.multied.com

resolved to Saddle Brook

3.2. Keyword features

Besides the features mentioned above, we also use distinctive keywords from both the entire content of the web pages as well as from the snippets and titles of the documents. The keyword selection is based on a preprocessed text file which is cleaned by removal of markup and other HTML information, tokenized, and PoS tagged (see Section 5.1). Only the content words, including nouns, adjectives, and verbs, are kept for further processing. The person name itself is removed from the snippets and titles because it is not useful for the disambiguation process.

To determine the relevance of the words extracted from the content of the web pages, we compute the TF-IDF score (term frequency – inverse document frequency) for each term-document pair, i.e. the relative frequency of the term in a document compared to the relative frequency of the term over all documents [36]. Words with high TF-IDF scores imply a strong relationship with the document they appear in. The relevance scores are stored in a document-term matrix DT. The rows of this matrix correspond to the documents, while the columns correspond to all keywords of all documents. An entry in the document-term matrix denotes the weight of a particular word in a particular document.

As the keywords extracted from snippets and titles are likely to be highly informative, we associate with every document a set of snippet words and title words. This snippet and title keyword feature is treated in a similar way as e.g. the named entity or biographical facts features discussed above, in the sense that any snippet or title keyword overlap is considered to be a strong indication that the documents are related to the same person.

4. Clustering

4.1. Document similarity measure

At this point, we performed document analysis and extracted a set of discriminatory features that can be useful for person disambiguation. However, the clustering algorithm can not be readily applied to these features vectors that are of mixed nature; some features are numeric while others are nominal. To apply a clustering algorithm we require a way of comparing two feature vectors by developing a custom similarity measure.

To solve this problem, we construct a comparison vector for each pair of documents (d_1, d_2):

$$C(d_1, d_2) = (C_1(d_1, d_2), C_2(d_1, d_2), \ldots, C_n(d_1, d_2))$$

which is used by the clustering algorithm to compare documents d_1 and d_2. Each component $C_i(d_1, d_2)$ of the comparison vector corresponds to a feature i.

If A and B are sets of values for this feature for d_1 and d_2 respectively, then

$$C_i(d_1, d_2) = \begin{cases} 1, & \text{if } A \cap B \neq \emptyset, \\ 0, & \text{otherwise} \end{cases}$$

Example 2. From the information for Alexander Macomb in Example 1 we obtain for instance

$$C_{DateOfBirth}(d_{19}, d_{64}) = 0$$

but

$$C_{Location}(d_4, d_{76}) = 1$$

because both documents d_4 and d_{76} contain "Belle Isle" and "Detroit".

Note that this approach is very tolerant in the sense that the smallest overlap in the sets of values for a particular feature already results in a perfect match of the two documents for that feature; we opted for this approach because all features discussed in Section 3 (except for the weighted keywords, see below) are strong indicators that two documents refer to the same person. A less tolerant approach would be to use the Jaccard measure as a gradual similarity measure between the sets A and B, reflecting that the more values A and B have in common, the better the match is between d_1 and d_2 on the feature at hand, and a perfect match would then only be achieved when A equals B. In a more tolerant approach, on the other hand, one could take into account gradual similarities of the values of the features themselves, acknowledging that two sets of values A and B overlap if they contain at least one element that is similar, even if not exactly the same. In practice the similarity between feature values such as location names ("Belle Isle" and "Belle Ile") could then be computed using the edit distance.

The weighted bags of keywords in our approach are compared using the cosine similarity measure

$$C_{kws}(d_1, d_2) = \sum_{t=1}^{m} DT_{d_1,t} \cdot DT_{d_2,t}$$

where $DT_{d_1,t}$ and $DT_{d_2,t}$ are values from the document-term matrix DT (cfr. Section 3.2), i.e. the TF-IDF scores of term t in documents d_1 and d_2 respectively, and m is the number of columns in DT.

The second step is the aggregation of the components of the comparison vector into one value in $[0, 1]$ which will be used by the clustering algorithm as the similarity measure between the two documents. We aggregate the comparison vector by means of a weighted average. In other words, let $w_1, w_2, \ldots, w_n$ be nonnegative numbers that sum up to 1, then

$$S(d_1, d_2) = \sum_{i=1}^{n} w_i \cdot C_i(d_1, d_2)$$

This is a number between 0 and 1 that represents the overall similarity between d_1 and d_2. An important question that arises, is what the value of the weights should be, because these weights reflect the importance of the various components for person name disambiguation.

As a baseline system, one can give all components of the comparison vector equal weight, i.e. $w_1 = w_2 = \ldots = w_n = \frac{1}{n}$.

In order to give higher weights to components that contribute more to a correct clustering decision, we refine this by using the gain ratio of the different components as their weights [33]. Note that every document comparison vector $C(d_1, d_2)$ needs to be classified in one of two classes: it belongs to class c_{same} if documents d_1 and d_2 are about the same person, while it belongs to class c_{diff} if d_1 and d_2 concern different persons. Every component $C_i(d_1, d_2)$ gives some clue about whether $C(d_1, d_2)$ belongs to c_{same} or c_{diff}. However, some components contain more information than others. The information gain of a given component $C_i(d_1, d_2)$ is the reduction in uncertainty about which class $C(d_1, d_2)$ belongs to, when we know the value of $C_i(d_1, d_2)$. Gain ratio is a normalized version of information gain that reduces the bias of information gain towards selecting components with a large number of values. We have experimentally optimized the gain ratio weights output in order to determine the weight settings that have been used for all clustering experiments (see Section 5).

4.2. Fuzzy ant based clustering

The third step, after feature extraction and document similarity computation, is clustering. A particular challenge of the WePS problem is that it is not known in advance how many people in the data set share the same name. Clustering algorithms mimicking the behavior of ants are interesting candidates for tackling this problem. In this section we describe a fuzzy ant based algorithm, that does not depend on a priori knowledge of the number of clusters,

while in Section 4.3 and Section 4.4 we describe a hierarchical and a partitional clustering algorithm, that require the expected number k of clusters as an input.

Ant based clustering algorithms are usually inspired by the clustering of dead nestmates, as observed in the behavior of several ant species under laboratory conditions. Without negotiating about where to gather the corpses, ants manage to cluster all corpses into one or two piles. The conceptual simplicity of this phenomenon together with the lack of centralized control and a priori information are the main motivations for designing clustering algorithms inspired by this behavior (see e.g. [15], [25], [31]).

Monmarché's algorithm [31] involves a pass in which artificial ants can only pick up one item and a separate pass during which ants can only pick up an entire "heap" of items. In [37] a fuzzy ant based clustering algorithm was introduced where the artificial ants are endowed with intelligence in the form of IF–THEN rules that allow them to do approximate reasoning. As a result, at any time the ants can decide for themselves whether to pick up a single item or an entire heap, which makes a separation of the clustering into different passes superfluous. The corresponding algorithm is not only more elegant but it also outperforms Monmarché's algorithm on benchmark data sets as shown in [37].

In the algorithm, a certain stimulus and a response threshold value are associated with each task an ant can perform. The response threshold value is fixed, but the stimulus can change and represents the need for the ant to perform the task. The probability that an ant starts performing a task with stimulus s and response threshold value θ is defined as

$$T_n(s;\theta) = \frac{s^n}{s^n + \theta^n}$$

where parameter n is a positive integer, $s \in [0,1]$ and $\theta \in]0,1]$.

An ant with a load has only one possible task, which is to drop its load. Let s_{drop} be the stimulus associated with this task and θ_{drop} the threshold value. The probability of dropping the load is then given by

$$P_{drop} = T_{n_i}(s_{drop};\theta_{drop})$$

where $i \in \{1,2\}$ and parameters n_1 and n_2 are positive integers. When the ant is only carrying one item, n_1 is used, otherwise n_2 is used. An ant without a load can perform two tasks when encountering a heap: picking up one item or picking up all items. Let s_{one} and s_{all} be the respective stimuli, and θ_{one} and θ_{all} the respective response threshold values. The probabilities for picking up one item and picking up all the items of a heap are given by

$$P_{pickup\_one} = \frac{s_{one}}{s_{one} + s_{all}} \cdot T_{m_1}(s_{one};\theta_{one})$$

$$P_{pickup\_all} = \frac{s_{all}}{s_{one} + s_{all}} \cdot T_{m_2}(s_{all};\theta_{all})$$

where parameters m_1 and m_2 are positive integers.

As described below, the values of the stimuli s_{drop}, s_{one}, and s_{all} are calculated by evaluating fuzzy IF-THEN rules, using the rule bases provided in [37]. The values we used for parameters n_1, n_2, m_1, and m_2 are discussed in Section 5.

A major asset of humans is their flexibility in dealing with imprecise, granular information, that is, their ability to abstract from superfluous details and to concentrate instead on more abstract concepts (represented by words from natural language). One way to allow a machine to mimic such behavior is to construct an explicit interface between the abstract symbolic level, i.e. linguistic terms like *high*, *old*, etc. and an underlying, numerical representation that allows for efficient processing; this strategy lies at the heart of fuzzy set theory, which, since its introduction in the 1960s [41], has rapidly acquired an immense popularity as a formalism for the representation of vague, linguistic information, and which in [37] is exploited as a convenient vehicle for constructing commonsense rules that guide the behavior of artificial ants in a clustering algorithm.

A fuzzy set A in a universe U is a mapping from U to the unit interval $[0, 1]$. For any u in U, the number $A(u)$ is called the membership degree of u to A; it expresses to what extent the element u exhibits the property A. A is often also referred to as a membership function. A fuzzy rule base is of the form

$$\begin{array}{c} \text{IF } X \text{ is } A_1 \text{ and } Y \text{ is } B_1 \text{ THEN } Z \text{ is } C_1 \\ \text{IF } X \text{ is } A_2 \text{ and } Y \text{ is } B_2 \text{ THEN } Z \text{ is } C_2 \\ \cdots \\ \text{IF } X \text{ is } A_n \text{ and } Y \text{ is } B_n \text{ THEN } Z \text{ is } C_n \end{array}$$

where X, Y, and Z are variables taking values in the respective universes U, V, and W, and where for i in $\{1, \ldots, n\}$, A_i (resp. B_i and C_i) is a fuzzy set in U (resp. V and W). The aim is then to deduce a suitable conclusion about Z for every specific input of X and Y. This can, of course, be generalized to an arbitrary number of variables in the antecedent and the consequent.

The stimulus for a loaded ant to drop its load L on an already existing heap H is based on the average similarity A among the items in H and the average similarity B between the center of H and items of L [37]. If B is smaller than A, the stimulus for dropping the load should be low, because this would destroy, or at least decrease, the homogeneity of heap H; if B is greater than A, the stimulus should be high. Because heaps should be able to grow, we should also allow the load to be dropped when A is approximately equal to B. An ant will perceive the values of A and B to be very high (VH), high (H), medium (M), low (L), or very low (VL). The stimulus will be perceived as very very high (VVH), very high (VH), high (H), rather high (RH), medium (M), rather low (RL), low (L), very low (VL), or very very low (VVL). These linguistic terms can be represented by fuzzy sets in $[0, 1]$ (see [37] for their exact shape). The rules for the stimulus for dropping the load L onto an existing heap H are presented in Table 2.

The stimuli for an unloaded ant to pick up one item or a whole heap are computed in a similar way. An unloaded ant should pick up the most dissimilar

	VH	H	M	L	VL
VH	RH	H	VH	VVH	VVH
H	L	RH	H	VH	VVH
M	VVL	L	RH	H	VH
L	VVL	VVL	L	RH	H
VL	VVL	VVL	VVL	L	RH

Table 2: Fuzzy rules to infer the stimulus for dropping a load L on an existing heap H. The columns correspond to the average similarity A of H, while the rows correspond to the average similarity B between the center of H and the items of L. For example, if A is high, i.e. the heap is homogeneous, and B is low, i.e. the load is quite different from the heap, then the stimulus to drop the load on the heap is very very low.

item from a heap if the similarity between this item and the center of the heap is far less than the average similarity of the heap. This means that by taking the item away, the heap will become more homogeneous. An unloaded ant should only pick up an entire heap, if the heap is already homogeneous. Thus, the stimulus for an unloaded ant to pick up a single item from a heap and the stimulus to pick up all items from that heap are based on the average similarity of the heap and the minimal similarity among two items in the heap. We refer to [37] for the respective fuzzy rule bases to compute these stimuli.

During execution, the algorithm maintains a list of all heaps. Initially there is a heap, consisting of a single element, for every item in the data set. Picking up an entire heap H corresponds to removing a heap from the list. At each iteration an ant acts as follows (k is a small integer constant, set to 5 in our experiments):

- If the ant is unloaded, a heap H from the list is chosen at random.
  - If H consists of a single item, this item is always picked up.
  - If H consists of two items, a and b, both items are picked up with probability $S(a,b)^k$ and one of the two items is picked up with probability $(1 - S(a,b))^k$, with $S(a,b)$ the similarity of items a and b.
  - If H consists of more than two items, a single element is picked up with probability P_{pickup_one}, while all elements are picked up with probability P_{pickup_all}.
- If the ant is loaded, a new heap containing the load L is added to the list of heaps with a fixed probability (set to 0.001 in our experiments). Otherwise, a heap H from the list is chosen at random.
  - If H consists of a single item a and L consists of a single item b, L is dropped onto H with probability $S(a,b)^k$.
  - If H consists of a single item and L consists of more than one item, the ant does nothing. The main reason for separating this special case is efficiency. Because the average similarity $avg(H)$ will always

be 1 in this case, the only situation where it would be desirable to merge H and L is when all the items in L are approximately equal to the single element in H. But in this unlikely case, L and H would be merged at a later iteration of the algorithm.

– If H consists of more than one item, L is dropped onto H with probability P_{drop}.

4.3. Agglomerative hierarchical clustering

The output of a hierarchical clustering algorithm is a tree where the root corresponds to a cluster containing all documents, and the leaves correspond to clusters containing only one document. An agglomerative clustering algorithm builds this tree from the leaves to the top, in each step merging the two clusters with the largest similarity. Cutting the tree at a given height gives a clustering at a selected number k of clusters. Alternatively, one can define a cutting threshold, i.e. the clustering stops when the similarity between clusters is less than a given threshold.

Techniques to automatically determine the best value for k often rely on the use of a validity index such as the Dunn index [16] or the silhouette statistic to estimate the quality of a particular clustering of the data set. In [11], the silhouette index is mentioned as the index of choice, but at the same time this study warns that, depending on the model of the data, validity indices do not always correlate well to the actual error rate of the clustering algorithm. For an overview of the well known and more recent fuzzy clustering validity indices, we refer to [40] and [42].

In Section 5 we compare the performance of the fuzzy ant based clustering algorithm with an implementation of *Agnes* (Agglomerative Nesting) that is fully described in [20]. Agnes is run with single linkage (or single-link), meaning that in each step the algorithm merges the two clusters with the smallest minimum pairwise distance (which comes down to the nearest neighbor method). In order to determine the number of output clusters, we have decided to cut the tree at different similarity thresholds between the document pairs, with intervals of 0.1 (e.g. for threshold 0.2 all document pairs with similarities above 0.2 are clustered together).

4.4. Partitional clustering

Partitional clustering aims to directly decompose the data set into a set of disjoint clusters. Typically, the clustering algorithm tries to minimize some dissimilarity measure in the samples within each cluster, and to maximize the dissimilarity of different clusters.

A commonly used partitional clustering method is the k-means clustering algorithm. The k-means clustering method aims to partition n data items into a predefined number of k clusters, in which each data item belongs to the cluster whose center (or centroid) is nearest. The algorithm starts off by randomly generating k clusters and determining the cluster centers. In a next step each point is assigned to the nearest cluster center and the new cluster centers are

recomputed. The two final steps are repeated until the convergence criterion is met.

In Section 5 we compare the performance of the fuzzy ant based clustering algorithm with k-means clustering. To run k-means clustering we need to provide a desired number of clusters as an input. Obviously, there is no universal number of clusters that can fit to any type of data, thus this requirement poses a serious restriction on the algorithm's applicability. To partially overcome this problem we decided to use not an absolute number of clusters but a proportion of the number of documents per person. In particular, we look at results where k is 20, 40 and 60% of the document set size.

5. Evaluation

In this section we compare the results of the fuzzy ant based algorithm on web people data with both an agglomerative hierarchical clustering approach and a k-means clustering approach, since these two methods are most frequently used in the related research. One of the main weaknesses of the hierarchical and k-means clustering algorithms is that they require a predefined number of output clusters or a similarity treshold as an input. This is an impractical restriction for the problem at hand since the number of clusters strongly depends on the name, a phenomenon that is clearly reflected in the striking difference between the training data described below (an average of 11 clusters per name) and the test data (an average of 46 clusters per name). The similarity threshold, in turn, depends on the clusters density, i.e. on the similarity of the objects within one cluster. For one name it could be very high (e.g. one person is a politician and another is a sportsman), while for another relatively low (e.g. both persona are actors). In this case, no universal threshold can be set to fulfill both clustering cases. The fuzzy ant based algorithm does not depend either on the number of clusters, nor on the cluster density, which makes it a very suitable candidate for solving this task.

5.1. Data sets

Given the high relevance of the Web People Search problem for various NLP domains, a first WePS task was organized in the framework of the SemEval 2007 Fourth International Workshop on Semantic Evaluations[8], an international competition on semantic evaluation which was organized in conjunction with the Annual Meeting of the Association for Computational Linguistics (ACL-2007). In this SemEval framework, the task organizers [2] provided the participants with training data and test data[9]. The training set covers different degrees of ambiguity, including very common names, uncommon names, and celebrity names, which tend to monopolize search results.

[8] http://www.senseval.org/

[9] Available at http://nlp.uned.es/weps

The names were selected from the US Census corpus (32 names), from Wikipedia (7 names) and from the Program Committee listing of the ECDL-2006 conference (10 names). The Wikipedia and ECDL sets contain documents corresponding to the first 100 results for a person name query to the Yahoo! search engine, whereas the US Census sets contain a varying number of search engine results (from 2 to 405 documents) per person name. These documents were manually clustered. Documents that could not be clustered properly were put in a "discarded" section.

The test data were constructed in a similar way (30 sets of 100 web pages). Unfortunately, there was a general increase in ambiguity relative to the training set. The global ambiguity average (number of different entities per person name) is 10.76 for the training data, whereas for the test data it is 45.93 [2]. The largely different distributions in the training and test sets makes the task very challenging for a machine learning approach (e.g. for training the distance threshold for clustering).

The first WePS competition has been followed by a second WePS workshop [3], that was organized in conjunction with the International World Wide Web Conference (WWW2009). The test set for the second competition again contains 30 data sets (each corresponding to a person name) and the names were selected from Wikipedia, the Program Committee listing of the ACL-2008 conference and from the US Census corpus. The second test set has on average 18.64 entities per person name.

5.2. Evaluation metrics

Clustering quality can be evaluated against internal criteria (how similar are documents within a cluster and how dissimilar are documents from different clusters) or external criteria (by comparing the output clusters against a set of reference clusters) [27]. The first approach is often the method of choice when no reference clusters (also called categories) are available. In our evaluation however, we focus on the second criterion and compare our clustering output against the gold standard that has been made available within the WePS competition framework.

Different evaluation metrics have been proposed for measuring clustering quality when reference clusters or categories are available: purity and inverse purity (and their combined F-measure) [2], mutual information [39], clusters and class entropy [6], rand statistics [34], Jaccard coefficient [7], etc. Amigó et al. [1] tested different clustering metrics against a number of formal constraints, being:

- cluster homogeneity: clusters should only contain items belonging to one category
- cluster completeness: items belonging to the same category should be grouped into the same cluster
- "rag bag": it is preferable to have clean clusters plus a "rag bag" containing miscellaneous

- cluster size versus cluster quality: a small error in a large cluster is preferable to a large number of small errors in small clusters

Although there is no single metric satisfying all constraints, the BCubed metric combines most features of the existing metrics and scores best compared to the output of a human quality assessment test [1]. This motivates our choice for using purity and inverse purity, since these are the official evaluation metrics of the WePS competition, and the BCubed measure, since that has been found to be a reliable clustering metric. We briefly recall their definitions below.

5.2.1. Purity and inverse purity

The purity and inverse purity metrics assume a one to one mapping between the predicted clusters and the manually labeled clusters in the gold standard data set (henceforth "categories") and rely on the precision and recall concepts from the field of information retrieval. Purity focuses on the proportion of the most frequent class in each cluster, while inverse purity considers the cluster with highest recall for each category. Perfect scores for purity are obtained by assigning all items to separate clusters, while putting all items together in one single cluster results in maximum scores for inverse purity. In our evaluation, we have implemented purity, inverse purity and their harmonic mean F according to the definition in [1].

Given a set of manually labeled categories (gold standard) $L = \{L_1, L_2, \ldots, L_m\}$, a set of clusters $C = \{C_1, C_2, \ldots, C_k\}$ to be evaluated, and the number of clustered items n, purity is obtained by taking the weighted average of the maximal precision values:

$$Purity = \sum_{i=1}^{k} \left(\frac{|C_i|}{n} \right) \max_{j=1}^{m} \text{Precision}(C_i, L_j)$$

where precision is defined as

$$Precision(C_i, L_j) = \frac{|C_i \cap L_j|}{|C_i|}$$

Inverse purity looks for the cluster with maximum recall for each category:

$$InversePurity = \sum_{i=1}^{m} \left(\frac{|L_i|}{n} \right) \max_{j=1}^{k} Recall(C_j, L_i)$$

where $Recall(C_j, L_i) = Precision(L_i, C_j)$.

To calculate their harmonic mean, the F-measure of [38] is used:

$$Purity\_F = \sum_{i=1}^{m} \left(\frac{|L_i|}{n} \right) \max_{j=1}^{k} \{F(C_j, L_i)\}$$

where the maximum is taken over all clusters. $F(C_j, L_i)$ is defined as:

$$F(C_j, L_i) = \frac{2 \cdot Recall(C_j, L_i) \cdot Precision(C_j, L_i)}{Recall(C_j, L_i) + Precision(C_j, L_i)}$$

5.2.2. BCubed

The BCubed metric [5] was originally designed for scoring coreference algorithms. Precision and recall are computed for each entity in the document, and are combined to calculate final precision and recall for the entire output. If we extend the BCubed algorithm for clustering evaluation, we can state that two items are correctly linked (clustered together) if they share a category and appear in the same cluster. The precision score then expresses how many items in the same cluster belong to its category, while the recall represents how many items from its category appear in the cluster. The main advantage compared to the purity measure is that the calculated precision of the items depends on the reference item and not on the predominant category in the cluster.

Amigó et al. [1] describe the BCubed measure using a function *correctness* that maps a pair of items to 0 or 1. In particular, $correctness(e, e') = 1$ when items e and e' share a category if and only if they appear in the same cluster. In other words, with $L(e)$ the category of an item e, and $C(e)$ the cluster to which item e is assigned:

$$correctness(e, e') = \begin{cases} 1, & \text{if } L(e) = L(e') \text{ and } C(e) = C(e') \\ 0, & \text{otherwise} \end{cases}$$

The BCubed precision of an item is then defined as *the proportion of correctly related items in its cluster* (including the item itself), and the overall BCubed precision as *the averaged precision over all items*:

$$PrecisionBCubed = Avg_e Avg_{e' \in C(e)} correctness(e, e')$$

The BCubed recall of an item is defined as *the proportion of correctly related items in its category*, and the overall BCubed recall as *the averaged recall over all items*:

$$RecallBCubed = Avg_e Avg_{e' \in L(e)} correctness(e, e')$$

Finally, the harmonic mean $BCubed\_F$ (with parameter $\alpha = 0.5$) of BCubed precision and recall is calculated as follows:

$$BCubed\_F = \frac{1}{\alpha \frac{1}{PrecisionBCubed} + (1 - \alpha) \frac{1}{RecallBCubed}}$$

5.3. Evaluation results

In order to evaluate the quality of the fuzzy ant based clustering algorithm for Web People Search, we have compared it, using all evaluation metrics, with a state-of-the-art hierarchical clustering approach (Agnes) as well as with a well-known partitional clustering approach (k-means). The evaluation metrics used

are BCubed precision, recall and F-score, purity, inverse purity and purity–inverse purity F-score. We have fine-tuned all algorithms parameters on the SemEval training data, and evaluated their performance on the WePS test sets (referred to as WePS1 and WePS2). Table 3 lists the weights for all features we used (common for all algorithms).

Biographical Facts	Date of birth	0.10
	Date of death	0.10
	Year of birth	0.10
Named Entities	Person name	0.02
	Locations	0.07
	Other named entities	0.04
URL and Email	URL addresses	0.10
	Email addresses	0.10
	Domain names	0.10
Location	IP addresses	0.07
Keywords	Content of web page	0.08
	Title and snippet	0.12

Table 3: Overview of all extracted features and the corresponding feature weights

We first present the performance results obtained with Agnes and k-means using three different clustering thresholds. Next, we compare these results with the fuzzy ant based clustering.

5.3.1. Hierarchical clustering

Figure 1 shows the results for Agnes with three different clustering thresholds (threshold 0.1, threshold 0.2, and threshold 0.3) applied to the training data. Figure 2 shows the results for Agnes applied to the first WePS test set, whereas Figure 3 shows the results on the second WePS test set. Note the trade-off between BCubed precision and BCubed recall in all figures: when the precision goes up, the recall decreases, and vice versa. A similar tendency can be observed for the purity and inverse purity on the test data. The F-scores (BCubed_F and Purity_F) are in this sense the most informative, as they provide a summary of the performance in terms of both precision and recall.

The Agnes results show the large impact of the clustering threshold. The first threshold (only documents with similarity above 0.1 get clustered) clearly outperforms the other two thresholds when applied to the training data. Indeed, both BCubed_F and Purity_F decrease as the threshold increases. When applied to the first test set however, this trend is reversed for BCubed_F, indicating the first threshold as the worst choice, while for Purity_F all thresholds lead to relatively similar results. The scores on the second test set show again much better performance for the first threshold compared to the other two.

The remarkable difference in performance among the various data sets is probably due to the different ambiguity level for the training and test data sets.

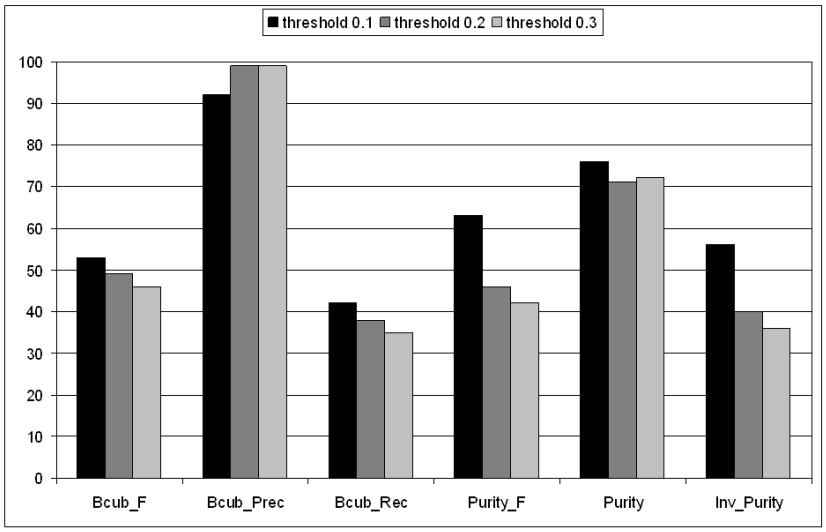

Figure 1: Results for Agnes using three different clustering thresholds on training data

In the training data there are on average 10.67 entities per person name, while the first test set has on average 45.93 and the second test set 18.64 entities per person name. The large impact of the threshold setting is important, as the threshold has to be predefined for each hierarchical clustering task, and for Web People Search we do not know the ambiguity, or in other words, the number of desired output clusters in advance.

5.3.2. Partitional clustering

Figure 4 represents the results for k-means on the training data with the number of clusters k equal to 20, 40 and 60% of the document set size. It can be seen in Figure 4 that the trade-off between BCubed recall and BCubed precision for k-means becomes more exposed with the growth of the number of clusters, while a smaller number of clusters provides more balanced scores. The results suggest that setting the number of clusters as 20% of the document set size is an optimal choice in terms of BCubed_F and Purity_F.

However, the disadvantage of explicit specification of the number of clusters becomes evident when we apply the algorithm to the test sets. In Figure 5 it is clear that 20% of the data set size is not the best choice here since it provides the worst results both in terms of BCubed_F and Purity_F. This effect is due to the drastic drop in the precision for the smaller number of clusters. The results for the second test set provided in Figure 6 show the same trend as for the training data, however the values for all metrics are significantly smaller.

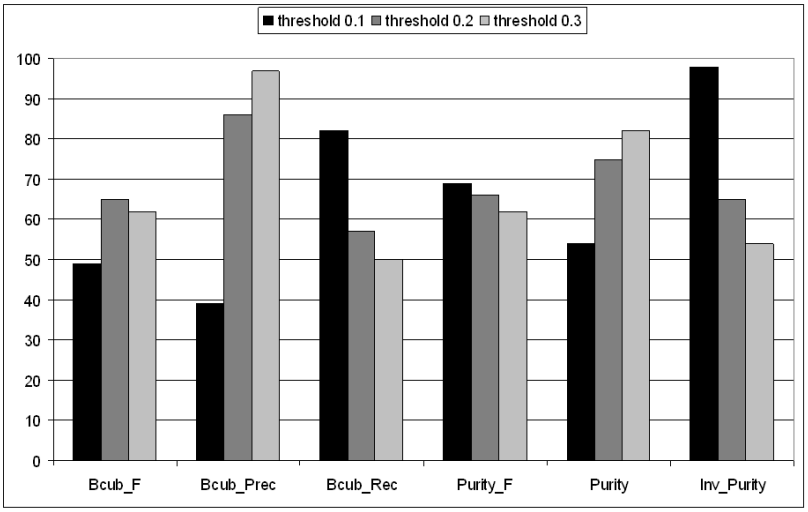

Figure 2: Results for Agnes using three different clustering thresholds on WePS1 test data

It is interesting to observe that the performance of the k-means algorithm on the training data set is in line with that on the second test set, but diverges from that on the first test set. Notice how we observed a similar phenomenon for the hierarchical clustering approach.

5.3.3. Fuzzy ant based clustering

Table 4 lists the parameter values we used in all experiments involving the fuzzy ant based clustering algorithm.

n_1	parameter for probability of dropping one item	1
m_1	parameter for probability of picking up one item	1
n_2	parameter for probability of dropping an entire heap	5
m_2	parameter for probability of picking up a heap	5

Table 4: Parameter settings for fuzzy ant based clustering

Figure 7 shows the results for our fuzzy ant based clustering approach when applied to the training data, whereas figure 8 and figure 9 show the results for our fuzzy ant based clustering implementation when applied to both test data sets. All three figures display results after 800 000 ant runs in which an individual ant can decide to do a drop or a pick up. In addition, the figures include the results obtained with Agnes and with k-means with the parameter

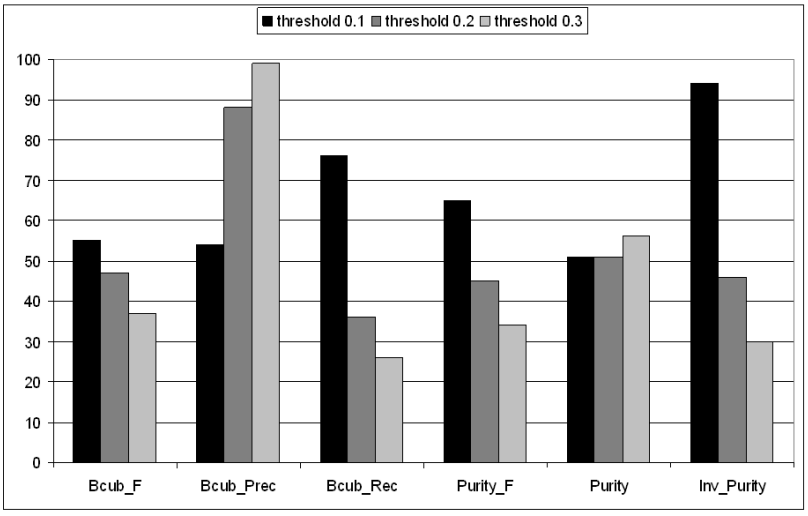

Figure 3: Results for Agnes using three different clustering thresholds on WePS2 test data

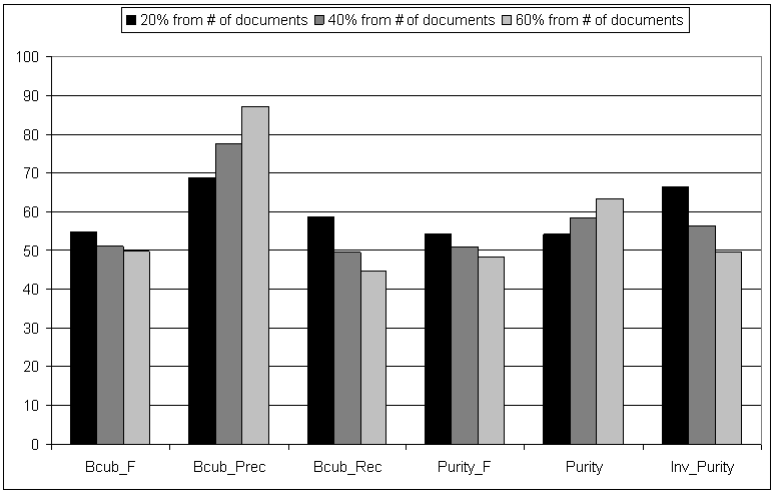

Figure 4: Results for k-means using three different numbers of clusters on training data

settings that came out as the best ones on the training data, i.e. a threshold of 0.1 for Agnes, and a number of clusters equal to 20% of the document set size

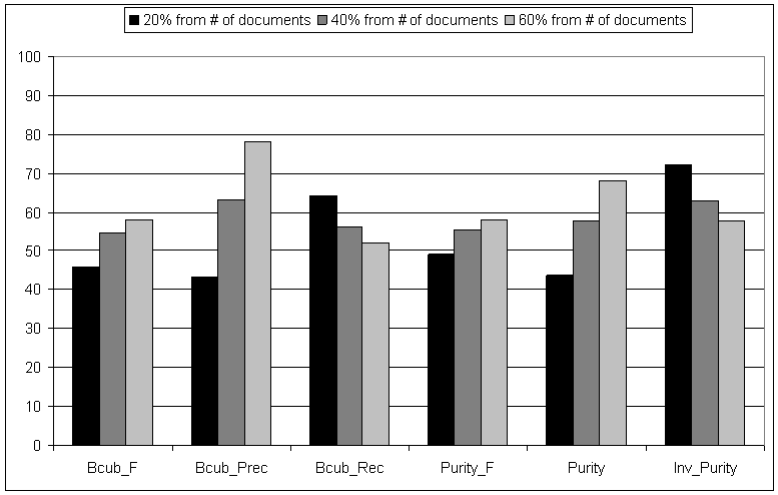

Figure 5: Results for k-means using three different numbers of clusters on WePS1 test data

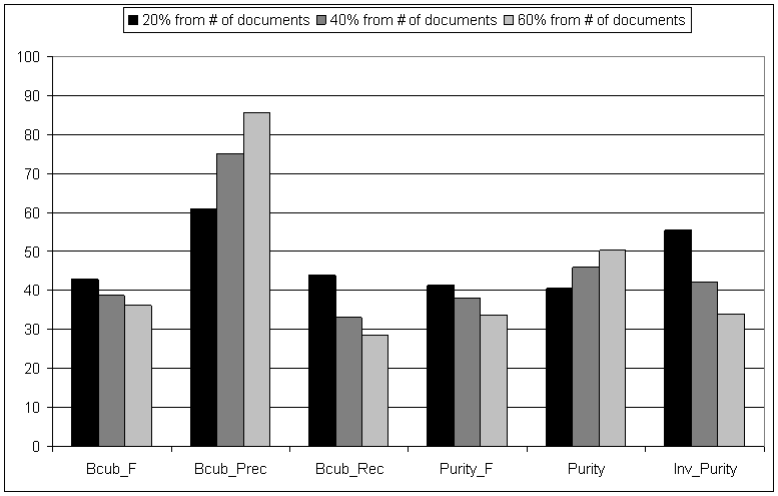

Figure 6: Results for k-means using three different numbers of clusters on WePS2 test data

for k-means.

When applied to the training data, all three algorithms (fuzzy ants, Agnes, k-means) show a comparable best performance in terms of BCubed_F, with a slightly worse performance of k-means in terms of Purity_F. The results for the first test set show a completely different picture. The fuzzy ant based algorithm clearly outperforms both other approaches in terms of both BCubed_F and Purity_F. On the second test set, Agnes is the winner, with the fuzzy ant based

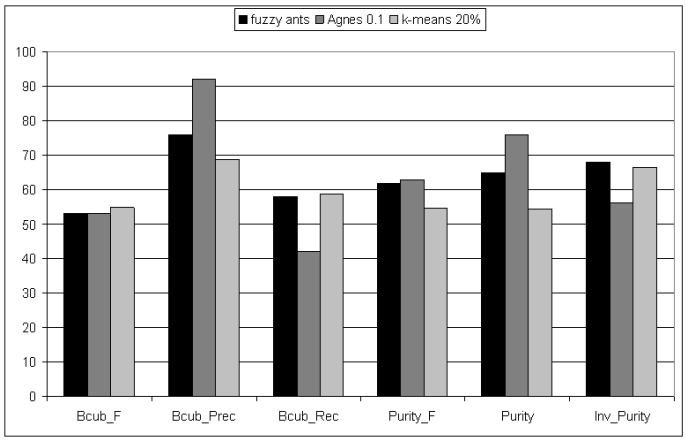

Figure 7: Results for fuzzy ant based clustering compared with Agnes and k-means on training data

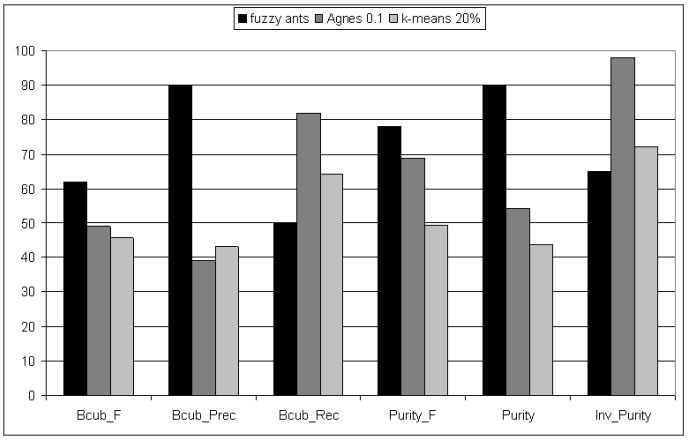

Figure 8: Results for fuzzy ant based clustering compared with Agnes and k-means on WePS1 test data

algorithm being a close runner-up. Note how the fuzzy ant based algorithm outperforms k-means clustering for both test sets.

The differences in results over the various data sets are due to the highly imbalanced composition of the training data set (the number of documents per person name varies from 2 to over 400 documents, which makes it hard to train a learning algorithm) and the different level of ambiguity between the data sets.

To further differentiate the described clustering algorithms we performed a statistical significance testing for every pair of methods. The null hypothesis in this case is that the two algorithms being tested perform equally. To perform

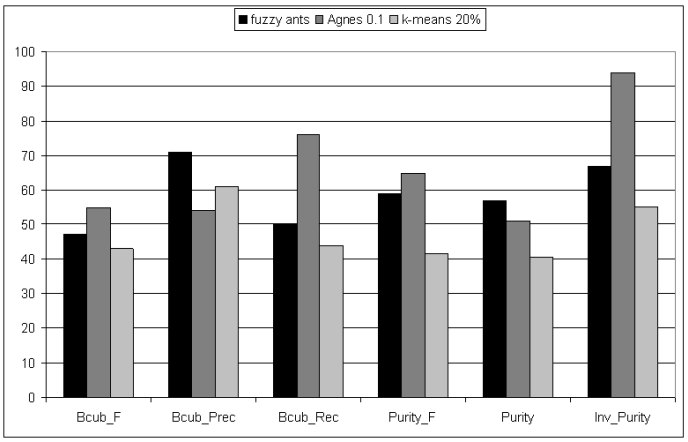

Figure 9: Results for fuzzy ant based clustering compared with Agnes and k-means on WePS2 test data

	WePS1 test data	WePS2 test data
Agnes vs. Fuzzy Ants	27.03	15.56
Agnes vs. k-means	1.05	0
Fuzzy Ants vs. k-means	3.75	4.21

Table 5: Results of McNemar's test for clustering algorithms

the test we consider every pair of documents and observe whether this pair was correctly assigned to the same cluster by both algorithms, i.e. we have paired observations. On the other hand we are not able to draw a definite conclusion about the data distribution, thus we choose to perform a non-parametric test. Since we can have only two possible outcomes — two documents are assigned to the same cluster correctly or not correctly — we choose a McNemar's test [29] to define whether the algorithms provide statistically significant results. The significance was calculated for every person name, and we consider the minimal significance results to evaluate the difference between the algorithms. These significance numbers are provided in Table 5. It can be seen that all algorithms are very different in terms of statistical significance except Agnes and k-means. which produce an identical cluster for one of the person names in the second WePS test set (hence the 0 in the table); however the results for other names are still significantly different.

6. Conclusions

We have proposed a new approach to Web People Search, which is the problem of organizing search results for ambiguous person name queries into meaningful clusters, each cluster referring to one single individual. In particular we:

- represent every search result (i.e. document or web page) as a feature vector containing relevant information extracted from the search result
- calculate the similarity of two search results as the weighted average of the similarity of the individual features, using gain ratio to determine the weights
- apply a fuzzy ant based algorithm for clustering the search results according to similarity

The fuzzy ant based clustering approach has as its main advantage that the number of output clusters does not have to be known or estimated a priori, which makes it very well suited for the Web People Search task. We have applied our algorithm to the WePS data sets that were used for the WePS-2007 and WePS-2009 competitions, and compared its performance to that of agglomerative hierarchical clustering (Agnes) and k-means clustering.

Our main finding is that the fuzzy ant based algorithm is more robust than the two state-of-the-art algorithms. The choice of a specific clustering threshold or a heuristic to determine the number of clusters before the start of the clustering process, has a very large impact on the hierarchical and k-means clustering performance, while the fuzzy ant based algorithm does not require any a priori information. This advantage of the fuzzy ant based algorithm is very clearly noticeable on the WePS1 test data set, which has characteristics that are quite different from those of the training set. Indeed, the global ambiguity average (number of different entities per person name) is 10.76 for the training data, whereas for the WePS1 data set it is 45.93. Information about the expected number of clusters or the similarity threshold learned on the training data, and consumed by Agnes and k-means, is not the best fit for the WePS1 data set. As a result, these algorithms are outperformed by the fuzzy ant based clustering algorithm. On the WePS2 test data set, which adheres closer to the original training data, Agnes performs the best, but the fuzzy ant based algorithm is still a close runner-up, making it the best choice overall.

In future research we would like to apply a genetic algorithm that combines feature selection with parameter optimization for finding the best settings for this WePS task. The different variables that have to be combined and optimized are the feature weight settings, the probability parameter settings, the fuzzy sets, and the optimal number of ants runs. In addition we would also like to investigate the use of adding a postprocessing step after the ants runs in order to merge additional clusters based on the cluster similarities.

References

[1] Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F., 2008. A comparison of extrinsic clustering evaluation metrics based on formal constraints. Information Retrieval Doi:10.1007/s10791-008-9066-8.

[2] Artiles, J., Gonzalo, J., Sikine, 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.

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

[4] Ash, R., 2006. The top 10 of Everything. Hamlyn, Palazzo Bath UK.

[5] Bagga, A., Baldwin, B., 1998. Entity-based cross-document co-referencing using the vector space model. In: Proceedings of the 17th international conference on Computational linguistics. pp. 75–85.

[6] Bakus, J., Hussin, M., Kamel, M., 2002. A SOM-based document clustering using phrases. In: ICONIP'02 Proceedings of the 9th International Conference on Neural Information Processing. Computational Intelligence for the E-Age. pp. 2212–2216.

[7] Ben-Hur, A., Elisseeff, A., Guyon, I., 2002. A Stability Based Method for Discovering Structure in Clustered Data. In: Pacific Symposium on Biocomputing. pp. 6–17.

[8] Berger, A., Caruana, R., Cohn, D., Freitag, D., Mittal, V., 2000. Bridging the lexical chasm: Statistical approaches to answer finding. In: Proc. Int. Conf. Reasearch and Development in Information Retrieval. pp. 192–199.

[9] Bezdek, J.C., Hathaway, R.J., Huband J.M., 2007. Visual Assessment of Clustering Tendency for Rectangular Dissimilarity Matrices. IEEE T. Fuzzy Systems 15 (5), pp. 890–903.

[10] Bollegala, D., Matsuo, Y., Ishizuka, M., 2006. Disambiguating personal names on the web using automatically extracted key phrases. In: Proc. ECAI 2006. Trento, Italy, pp. 553–557.

[11] Brun, M., Sima, C., Hua, J., Lowey, J., Carroll, B., Suh, E., Dougherty, E., 2007. Model-based evaluation of clustering validation measures. In: Pattern Recognition. Vol. 1. pp. 807–824.

[12] Cohen, W.W., 1995. Fast effective rule induction. In: Prieditis, A., Russell, S. (Eds.), Proceedings of the 12th International Conference on Machine Learning. Morgan Kaufmann, Tahoe City, CA, pp. 115–123.

[13] Daelemans, W., van den Bosch, A., 2005. Memory-Based Language Processing. Cambridge University Press.

[14] Daelemans, W., Zavrel, J., Berck, P., Gillis, S., 1996. Mbt: A memory-based part of speech tagger generator. In: Proceedings of the 4th ACL/SIGDAT Workshop on Very Large Corpora. pp. 14–27.

[15] Deneubourg, J., Goss, S., Franks, N., Sendova-Franks, A., Detrain and, C., Chrétien, L., 1990. The dynamics of collective sorting robot-like ants and ant-like robots. In: From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour. pp. 356–363.

[16] Dunn, J.C., 1973. A Fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters. In: Journal of Cybernetics. Vol. 3(3). pp. 32–57.

[17] Fleischman, M., Hovy, E., 2004. Multi-document person name resolution. In: Proceedings of 42nd Annual Meeting of the Association for Computational Linguistics (ACL), Reference Resolution Workshop.

[18] Guha S., Rastogi R., Shim K., 2000. ROCK: A robust clustering algorithm for categorical attributes. In: Information Systems. Vol. 25(5). pp. 345–366.

[19] Hemphill, C., Godfrey, J., Doddington, G., 1990. The atis spoken language system pilot corpus. In: Proceedings of the DARPA Speech and Natural Language Workshop. pp. 96–101.

[20] Kaufman, L. and Rousseeuw, P.J., 1990. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley, New York, NY.

[21] Kucera, H., Francis, W., 1967. Computational analysis of present-day English. Brown University Press, RI.

[22] Lee M. and Brouwer R. K., 2007. Fuzzy Clustering and Mapping of Ordinal Values to Numerical. In: Proceedings of the IEEE Symposium on Foundations of Computational Intelligence. pp. 538–543.

[23] Lefever, E., Fayruzov, T., Hoste, V., 2007. A combined classification and clustering approach for web people disambiguation. In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 105–108.

[24] Lefever, E., Fayruzov, T., Hoste, V., De Cock, M., 2009. Fuzzy Ants Clustering for Web People Search. In: 2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference.

[25] Lumer, E., Faieta, B., 1994. Diversity and adaptation in populations of clustering ants. In: From Animals to Animats: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour. pp. 501–208.

[26] Mann, G., Yarowsky, D., 2003. Unsupervised personal name disambiguation. In: Proceedings of CoNLL-2003. Edmonton, Canada, pp. 33–40.

[27] Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval, online version available at: http://nlp.stanford.edu/IR-book/html/htmledition/mybook.html. Cambridge University Press.

[28] Marcus, M.P., Santorini, B., Marcinkiewicz, M. A., 1993. Building a large annotated corpus of english: The penn treebank. Computational Linguistics 19 (2), pp. 313–330.
http://citeseer.nj.nec.com/marcus93building.html

[29] McNemar, Q., 1947. Note on the sampling error of the difference between correlated proportions or percentages. In: Psychometrika. Vol. 12(2). pp. 153–157.

[30] Ming Vu, Q., Takasu, A., Adachi, J., 2007. Improving the performance of personal name disambiguation using web directories. Information Processing & Management Doi:10.1016/j.ipm.2007.11.001.

[31] Monmarché, N., 2000. Algorithmes de fourmis artificielles: Applications à la classification et à l'optimisation. Ph.D. thesis, Université François Rabelais, Tours, France.

[32] Pedersen, T., Purandare, A., Kulkarni, A., 2006. Name discrimination by clustering similar contexts. In: Proceedings of the World Wide Web Conference (WWW).

[33] Quinlan, J., 1993. C4.5: Programs for machine learning. Morgan Kaufmann, San Mateo, CA.

[34] Rand, W.M., 1971. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66, pp. 846–850.

[35] Ravichandran, D., Hovy, E., 2001. Learning surface text patterns for a question answering system. In: Proceedings of the 40th annual meeting on association for computational linguistics (ACL). Morristown, NJ, USA, pp. 41–47.

[36] Salton, G. and Buckley, C., 1998. Term-weighting approaches in automatic text retrieval. Information Processing & Management 24 (5), pp. 513–523.

[37] Schockaert, S., De Cock, M., Cornelis, C., Kerre, E., 2007. Clustering web search results using fuzzy ants. In: International Journal of Intelligent Systems. Vol. 22. pp. 455–474.

[38] Van Rijsbergen, C.J., 1974. Foundation of evaluation. Journal of Documentation 30 (4), pp. 365–373.

[39] Xu, W., Liu, X., Gong, Y., 2003. Document clustering based on non-negative matrix factorization. In: SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval. Toronto, Canada, pp. 267–273.

[40] Yue, S., Wang, J.-S., Wu, T., Wang, H., 2010. A new separation measure for improving the effectiveness of validity indices. In: Information Sciences. Vol. 118(5). pp. 748–764.

[41] Zadeh, L.A., 1965. Fuzzy sets. In: Information and Control. Vol. 8, pp. 338-353.

[42] Zhang, Y., Wang, W., Zhang, X., Li, Y., 2008. A cluster validity index for fuzzy clustering. In: Information Sciences. Vol. 178. pp. 1205–1218.