The University of Amsterdam at WePS2

Krisztian Balog Jiyin He Katja Hofmann Valentin Jijkoun
Christof Monz Manos Tsagkias Wouter Weerkamp Maarten de Rijke

ISLA, University of Amsterdam
Science Park 107, 1098 XG Amsterdam
{k.balog,j.he,k.hofmann,v.jijkoun,c.monz,e.tsagkias,w.weerkamp,m.derijke}@uva.nl

ABSTRACT

In this paper we describe our participation in the Second Web People Search workshop (WePS2) and detail our approaches. For the clustering task, our focus was on replicating the lessons learned at WEPS1 on the data set made available as part of WEPS2 and on experimenting with a voting-based combination of clustering methods. We found that clustering methods display the same overall behavior on the WEPS1 and WESP2 data sets and that a hierarchical clustering approach delivers the best performance, even outperforming voting-based combinations.

For attribute extraction, we explore approaches using pattern matching with manually and automatically constructed patterns. Manual patterns were constructed using expert knowledge and following analysis of sample data. Automatic pattern construction extracts textual and syntactic context around training samples and selects patterns which are expected to perform well based on leave-one-out evaluation. Experimental results show that manually constructed patterns are very effective for obtaining high recall. For automatically extracted patterns performance varied widely depending on the attribute type. Larger amounts of training data may help improve these approaches in the future.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.4 [**Information Systems Applications**]: H.4.2 Types of Systems; H.4.m Miscellaneous

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Information extraction, Attribute extraction, Clustering, Person name disambiguation

1. INTRODUCTION

The second edition of the Web People Search (WePS2) workshop featured two tasks: clustering [2] and attribute extraction [10].

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

The goal of the clustering task was to group web pages, returned by a web search engine using a person's name as a query, so that pages referring to the same individual are assigned to the same cluster. Previous work conducted using the WePS1 data set [1] showed that this task can effectively be treated as a document clustering problem, and standard document clustering methods deliver excellent performance [3, 4]. The introduction of the WePS1 data set was a very significant step, as it provided the first common platform for evaluating techniques for person name disambiguation in a web setting. Yet, issues were raised concerning the reliability of the results obtained using this resource, due to the large difference between the average ambiguity in the training and test subsets, and because of the standard clustering evaluation measures that may not be appropriate for this specific scenario [1]. One of our major aims for this year, therefore, was to validate whether our findings generalize to the WePS2 collection, and whether looking at other evaluation metrics leads to the same conclusions drawn before. A secondary aim was to experiment with a voting method in order to combine the advantages of multiple individual clustering methods. We found that clustering methods display the same behavior across collections and metrics. As to our second goal, our combination technique was able to improve over all individual methods in terms of F-scores, but not for the single precision—recall and purity—inverse purity measures. Overall best performance was delivered by the hierarchical clustering approach.

For the attribute extraction task we experimented with two fundamentally different families of methods, manual and automatic, for constructing patterns. We found that our manual approach is very effective in terms of recall, as it achieved the highest overall recall among all participating systems [10]. As to the automatic methods, we found that performance strongly depends on the attribute classes; for some attribute types no generalizable patterns were extracted, while, for others, the automatically extracted patterns outperform the manually designed ones.

The paper is organized as follows. First, we describe our document preprocessing in Section 2. Next, we discuss our work on the clustering task (Section 3) and on the attribute extraction task (Section 4) in two largely independent sections. We conclude our findings and put forward suggestions for future work in Section 5.

2. PREPROCESSING

Preprocessing included the removal of HTML tags, sentence splitting, and named-entity tagging.

For HTML tag removal, we used a perl program based on the module `HTML::TreeBuilder`. This program detected the encoding of pages, converted text to ASCII format, and returned all text that would be rendered by a web browser while preserving paragraph boundaries for paragraph and block level elements.

The cleaned web pages were then processed by a sentence splitter using the module `Lingua::EN::Sentence`. The output is one text file per webpage, with one sentence per line.[1]

For the clustering task a separate index was created for each person name, using the Lemur toolkit.[2] We removed a standard list of stopwords, but did not apply stemming. Note that the clustering run `UvA_1` employed a different kind of pre-processing. Details are discussed in Section 3.1.1.

For attribute extraction we used a named entity tagged version of the documents. Named entity detection was performed using the Stanford NER software [5].

3. CLUSTERING

3.1 Approach

We experimented with five different clustering methods that correspond to our submitted runs `UvA_1`–`UvA_5`, that we detail in Sections 3.1.1–3.1.4 below.

3.1.1 Hierarchical Agglomerative Clustering

We experimented with two hierarchical agglomerative clustering implementations. The first implementation was entirely implemented by us to give us more control about the internal aspects, while the second implementation used the Lemur toolkit. First we describe our own implementation (run `UvA_1`) followed by the system using Lemur (run `UvA_2`).

Our clustering approach follows a general hierarchical clustering approach [8] with a few specific additions for this task. In general, hierarchical clustering uses a greedy clustering procedure where clusters with the highest similarity score are merged into the same cluster. The procedure continues iteratively until no further clusters can be merged. There are two stopping criteria: (a) fixed number of iterations or (b) minimum similarity threshold. In our experiments we only used stopping criterion (b). Following Balog et al. [4] the threshold was set to 0.1.

At the heart of hierarchical clustering lies the definition of similarity between clusters, which can be individual documents (singleton clusters) or sets of documents. Here, the general similarity between two clusters c_1 and c_2 is defined as the cosine similarity:

$$sim(c_i, c_2) = \sum_{t \in c_1 \cap c_2} \frac{w(t, c_1) \cdot w(t, c_2)}{\sum_{t' \in c_1} \sqrt{w(t, c_1)^2} \cdot \sum_{t' \in c_2} \sqrt{w(t, c_2)^2}}$$

where the weight of a term t in cluster c is defined as

$$w(t, c) = tf(t, c) \cdot \log\left(\frac{N}{n_t}\right)$$

and $tf(t, i)$ is the frequency of term t in cluster c, N is the number of documents in the collection and n_t is the number of documents in which t occurs. This is the standard TF.IDF definition used in information retrieval.

[1]Sentence information is used by some of the attribute extraction runs.

[2]`http://www.lemurproject.org`

In order to improve on the standard cosine similarity we experimented with a number of variations, in particular, instead of using the actual term frequency, we normalized it with respect to the average frequency of a term in this document:

$$w(t, c) = \frac{1 + \log(\text{freq}_{t,c})}{1 + \log(\text{avg}_{t' \in c}\text{freq}_{t',c})} \cdot \log\left(\frac{N}{n_t}\right)$$

What constitutes the collection in clustering is somewhat unclear. Is it the number of documents that were retrieved for a particular name or is it the number of all documents in the test? Here we experimented with both options, but the latter led to slightly better performance. This is probably due to the fact that a somewhat larger collection gives more reliable counts.

We were also interested to find out to what extent preprocessing the input has an impact on clustering quality. For converting the input HTML files into text files we opted for a simple but yet effective approach by using `w3m`, a UNIX command line web browser and piped the output into a text file. The text files were then tokenized by a simple tokenizer just separating punctuation symbols.

Instead of indexing the actual words, they were stemmed using Porter's stemmer [9]. We found that stemming improved our results considerably.

When clustering two smaller clusters together, there are a number of strategies:

minimum: This strategy defines the similarity between two clusters as the similarity between the two closest documents from each cluster.

maximum: This strategy defines the similarity between two clusters as the similarity between the two farthest documents from each cluster.

centroid: This strategy defines the similarity between two clusters as the similarity between the two centroids each cluster.

average: This strategy defines the similarity between two clusters as the similarity between all documents in both clusters.

We experimented with all strategies, and selected those for submission that performed best with the given implementation.

The final run submitted for this clustering approach (`UvA_1`) used stemming, simple `w3m` HTML clean-up, normalized document frequency and the **minimum** strategy. Further than that no task-specific preprocessing or feature extraction was applied. While the approach is rather simple, it shows that even simple and robust strategies can lead to very competitive performance in this task.

The second system (run `UvA_2`) which uses Lemur, employs the **centroid** strategy, and there are a number of differences with respect to preprocessing and morphological normalization. Firstly, preprocessing is done as described in Section 2. Secondly, no stemming is applied and the actual surface forms of the words are used for indexing. Finally, TF.IDF scores are computed in the standard way and are based only on documents of the specific person.

Throughout the remainder of the section, we use the standard preprocessing as described in Section 2. The following subsections correspond to runs `UvA_3` to `UvA_5`.

3.1.2 Single Pass Clustering

Single Pass Clustering [6] assigns pages to clusters using the following algorithm: The first document is taken and assigned to the first cluster. Then each subsequent document is compared against each cluster with a similarity measure. A document is assigned to the most likely cluster, as long as the similarity score is higher than a threshold γ (set to 0.1 following [4]); otherwise, the document is assigned to a new cluster.

For estimating the similarity between a document and a cluster ($\text{sim}(D, C)$) we employ the Naive Bayes similarity measure. It uses the log odds ratio to decide whether the document is more likely to be generated from that cluster or not. Assuming that terms in a document are sampled independently and identically, the odds ratio is calculated as follows:

$$O(D, C) = \frac{p(D|\theta_C)}{p(D|\theta_{\bar{C}})} = \frac{\prod_{t \in D} p(t|\theta_C)^{n(t,D)}}{\prod_{t \in D} p(t|\theta_{\bar{C}})^{n(t,D)}},$$

where $n(t, D)$ is the number of times term t appears in document D, θ_C is the cluster language model, and $\theta_{\overline{C}}$ is the language model that represents "not being in the cluster." The cluster language model is estimated by performing a linear interpolation between the empirical probability of a term occurring in the cluster $p(t|C)$ and the background model $p(t)$. The "not in the cluster" language model is approximated by using the background model $p(t)$. For a more elaborate discussion of the method, the reader is referred to Balog et al. [4].

3.1.3 Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis (PLSA) [7] can be used to cluster documents based on the semantic decomposition of the term document matrix into a lower dimensional latent space. Formally, PLSA can be defined as:

$$p(t, d) = p(d) \sum_z p(t|z)p(z|d), \quad (1)$$

where $p(t, d)$ is the probability of term t and document d co-occurring, $p(t|z)$ is the probability of a term given a latent topic z and $p(z|d)$ is the probability of a latent topic in a document. The prior probability of the document, $p(d)$, is assumed to be uniform. This decomposition can be obtained automatically using the EM algorithm [7]. Once estimated, we make the simplifying assumption that each latent topic represents one of the different senses of the person name. The document d is assigned to one of the person-topics z if (i) $p(z|d)$ is the maximum argument, and (ii) the odds of the document given z, i.e., $O(z, d)$, is greater than a threshold γ (set to 1.0 based on [4]), where

$$O(z, d) = \frac{p(z|d)}{p(\bar{z}|d)} = \frac{p(z|d)}{\sum_{z', z' \neq z} p(z'|d)}.$$

Note that the decomposition takes the number of latent topics, z, as a parameter. This value, however, corresponds to the number of person senses, which is unknown. We use an EM algorithm to select z that maximizes the log-likelihood of the decomposition; see [4] for details.

3.1.4 Combination of Methods

Based on the observation that different clustering methods result in different clustering structures, we decide to submit a combined run (`UvA_5`), which takes the initial results generated by different clustering methods as input and using a voting scheme to generate the final results. The voting procedure is defined as follows.

For a given pair of documents (d_i, d_j) and M clustering algorithms, we count the number of clustering algorithm that assign the same cluster for the two documents, i.e., the two document belong to a same cluster. If the votes reaches or exceeds a given a given threshold θ, the two documents are assigned to a same cluster. Formally, it is defined as follows:

$$\begin{cases} c(d_i) = c(d_j), & \text{if } \#n\left(c_m(d_i) = c_m(d_j)\right) \leq \theta, m = 1, ..., M \\ c(d_i) \neq c(d_j), & \text{otherwise.} \end{cases}$$

where $\#n\left(c_m(d_i) = c_m(d_j)\right)$ is the number of clustering algorithms agree that d_i and d_j belong to the same cluster.

When applying the combination method, we follow the procedure:

- collect the instances on which all m algorithms agree that they should belong to a same cluster to form a set of *seed clusters*; if none of the instances pairs has an agreement from all algorithms, $m = m - 1$
- start from the seeds, get the instances on which $m - 1$ algorithms agree that they should belong one of the seed clusters, add them in; if they do not have any relation with the existing seed clusters, form a new cluster; repeat until $m \leq \theta$.
- the rest instances form clusters with singularities.

In order to facilitate the voting decision, we take the initial results from 3 runs as input for combination and set the threshold θ to 2 based on the results obtained from the training set.

3.2 Results

Table 1 shows the results achieved by our different clustering methods. We find that runs `UvA_2` to `UvA_4` display the same behavior as on the WePS1 collection [4]. Our overall best performance was delivered by the hierarchical clustering approach. Most of the difference between `UvA_1` and `UvA_2` can be attributed to the differences in pre-processing, and the usage of a stemmer.

As to the combination of methods, run `UvA_5`, we see improvements over the individual methods in terms of F-scores (both for F-0.2 and F-0.5), but not for the single precision-recall and purity-inverse purity measures.

			F-0.2		F-0.5			
Run	BP	BR	BP-BR	P-IP	BP-BR	P-IP	IP	P
`UvA_1`	0.85	**0.80**	**0.80**	**0.87**	**0.81**	**0.87**	**0.87**	0.89
`UvA_2`	**0.92**	0.51	0.54	0.66	0.61	0.73	0.63	0.93
`UvA_3`	**0.92**	0.40	0.43	0.51	0.49	0.58	0.48	**0.94**
`UvA_4`	0.52	0.75	0.57	0.70	0.50	0.63	0.81	0.62
`UvA_5`	0.86	0.60	0.63	0.74	0.67	0.78	0.71	0.92

Table 1: Results for the clustering task. Best scores for each measure are in boldface. The metrics are: BCubed precision (BP) and recall (BR), purity (P), and inverse purity (IP).

4. ATTRIBUTE EXTRACTION

4.1 Approach

Our aim for the attribute extraction task was to experiment with two fundamentally different families of methods: manual and automatic pattern construction. Both are applied on top of a named entity tagged version of the documents.

4.1.1 Manual pattern construction

Under the first group of approaches, a separate extraction strategy was developed for each attribute type. We submitted two runs, using a baseline (`UvA_1`) and an advanced (`UvA_2`) version of these patterns, as follows:

- **Affiliation**. As affiliations of a person, we simply extracted all organizations (as identified by a named entity tagger) from the sentences containing the person's name.
- **Award.** The baseline for awards is a regular expression in the form of: "the ... AWARDTYPE for|the ...", where ... denotes alphanumeric sequences with their first letter capitalised. AWARDTYPE is a closed set consisting of *award*, *prize*, *competition*, and *medal*. Specifically for medals, we tried to capture the medal's rank by adding the subpattern: "with ... star(s)". The advanced system extended the baseline twofold. Firstly, new members were added to the AWARDTYPE: *fellowship*, *hall of fame*, *honour*, *distinction*, *grant*, and *scholarship*. Secondly, we introduced the pattern "... -of-the- PERIOD", for capturing time-dependent awards (i.e., "The Author of the Year"), where PERIOD is a closed set of *month*, *day*, *year*, *week*.
- **Birth place**. Both runs use the output of the named entity tagger as basis: For run `UvA_1` we return all mentions of LOCATION in the pages. For run `UvA_2` we follow up on the advanced identification of date of birth and return only those locations that are preceded by a possible date of birth: "DATE OF BIRTH{1..150} LOCATION".
- **Date of birth**. For both our runs we use the same regular expressions to select dates from pages. E.g., expressions can look like "02/03/1945", "3 Feb 1948", and "February 3, 1948". In run `UvA_1` we return all dates identified on the pages. Run `UvA_2` filters dates: Only dates that are contained in either "PERSON {1..50} (DATE)" or "PERSON {1..50} born {1..20} DATE" are returned.
- **Degree**. We construct a list of degrees from Wikipedia and filtered out the degrees that also refer to a common word (e.g., AS). The filtered list contains 89 degrees and 110 abbreviated versions. In run `UvA_1` we mark all entries in the list as degree, while in run `UvA_2` we take the same approach but remove orphan degrees (i.e., if no major or school is present, the degree is removed as well).
- **Email address**. Both runs `UvA_1` and `UvA_2` use the same regular expression to select email addresses. The selected addresses are filtered on the criterion whether or not they contain the first or last name of the person.
- **Major**. Run `UvA_1` uses the pattern "degree in.." to detect majors: When this pattern is followed by either capitalized words (e.g., "degree in Applied Mathematics") or a sequence of words that could indicate a school (e.g., "degree in mathematics from yyy") we mark the word(s) as major. In the `UvA_2` run we use the degrees extracted in the advanced degree setting in addition to the "degree in"-pattern (so also "DEGREE in ... "), and follow the same patterns as in the previous run to identify majors.
- **Mentor**. Mentor is considered any pattern starting with the words *influence*, *with*, *by*, or *for* followed by at least two capitalised words. The advanced system is the same as the baseline.
- **Nationality**. We construct a list of 201 nationalities from Wikipedia. In run `UvA_1` we mark each of the entries in this list as nationality of that person. Run `UvA_2` follows the same path, but requires the first or last name of the person to be mentioned at most 30 characters before the nationality (i.e. "PERSON {1..30} NATIONALITY")
- **Occupation**. We compiled two gazetteers of words and phrases describing occupations: one using Wordnet and another using English Wikipedia. From Wordnet, we extracted all (recursive) hyponyms of synsets *professional*, *entertainer*, *author*, *leader*, *worker*, *creator*, *engineer*, *preserver* and *expert*. This resulted in a list of 3,295 terms (2,271 words and 1,024 multi-word expressions). From Wikipedia, we started with two categories *Occupations* and *Sports positions* and recursively extracted all sub-categories whose titles end with *occupations, ranks, positions* or *professions*. Furthermore, for these categories, we extracted (non-recursively) all sub-categories with titles containing plural nouns or a string *Lists of ...*. Finally, we extracted all pages that are annotated with one of the extracted categories. This resulted in a list of 11,375 page and category titles (7,261 words and 4,114 multiword expressions).

  To extract possible occupations of a person in the test collection, we simply identified all occurrences of terms from our gazetteer in the documents for this person. The run `UVA_1` used the gazetteer created using Wordnet. The run `UVA_2` used both Wordnet- and Wikipedia-based gazetteers.
- **Other name**. We use regular expressions to select capitalized words immediately before or after the first and last name of the person, or in between the first and last name (e.g. "OTHERNAME PERSON" or "PERSON OTHERNAME"). In run `UvA_1` we return all the identified occurrences. In run `UvA_2` we use a list of 2,626 common (first) names, constructed from the web and select only the occurrences that also appear in this list.
- **Phone** and **Fax**. For both runs `UvA_1` and `UvA_2` we return the same set of candidates: we use regular expressions to match phone and fax numbers (e.g., "111 111-111" or "+11 (111) 111-11") and all numbers matching the patterns are used both as telephone and fax number.

- **Relatives**. A dictionary of terms describing family relations (father, mother, grandson, etc.) was constructed manually. Run UvA_1 returns the first person name occurrence after a relation term. Run UvA_2 uses a refined dictionary of relations and returns all names within a fixed window size around relation terms (specifically, this window was defined as 5 terms before and 20 terms after the relation term, based on the training data).

- **School**. Run UvA_1 extracts possible schools that follow simple patterns like "University of ...", "... College", and "School of ...". In run UvA_2 we look for occurrences of degrees or majors and we follow more sophisticated patterns to recognize schools. Examples of patterns used here are "DEGREE from the SCHOOL", "MAJOR at SCHOOL", and "degree from SCHOOL".

- **Web site** We use the same results for runs UvA_1 and UvA_2: Regular expressions are used to select possible web sites, after which only the web sites containing the last name of the person are returned.

4.1.2 Automatic pattern construction

In runs UvA_3 to UvA_5 we apply an automatic approach for "learning" patterns from training examples. We experiment with two different ways to construct patterns and with using named entity tags as an additional source of information. We use a two-step approach: (1) pattern extraction and (2) pattern selection. Example patterns that were obtained following this approach are shown in Table 2.

In the first step, we extract candidate patterns using the supplied sample data. For each name in the test set, and for each attribute, we extract from the text of the supplied web pages each sentence that contains the person's last name and an attribute value. We then construct patterns for 1 up to 5 tokens that precede and follow the attribute value in that sentence. Next, we generalize patterns by replacing elements of person names by placeholders, and, when using NE-tags, by additionally replacing named entities by their tag.

In the second step, we select patterns following a leave-one-out approach. The patterns extracted from all but one person name are applied to extract attributes from the web pages belonging to the held-out person name. Thus, we evaluate the extracted attribute values and keep the patterns where F-score exceeds a threshold value. This is repeated until each person name has been held-out once. The resulting set of patterns is applied to extract attributes for unseen names.

4.2 Results

Table 3 gives an overview of the results of our 5 submitted attribute extraction runs. Best performance is achieved by UvA_2, which uses the improved version of manually constructed patterns. Both runs using manual patterns (UvA_1, UvA_2) achieve similarly high recall, while UvA_2 makes substantial gains in precision.

The runs using automatically generated patterns (UvA_3, UvA_4, UvA_5) perform substantially lower than the runs using manually constructed patterns. However, UvA_5 achieves relatively high precision.

Attribute name	Example patterns
Affiliation	`Member of :ORG:\s*(\w+)\s*`
Award	`\s*(\w+)\s*for his services to`
Birthplace	`born in :LOC:\s*(\w+)\s*`
Date of birth	`born in :LOC:[\w\.\-]+? in\s*(\d+)\s*`
Degree	`graduated from the :ORG:[\w\.\-]+? with a\s*(\w+)\s*`
Major	`a bachelor 's degree in\s*(\w+)\s*`
Nationality	`one of :LOC:\s*(\w+)\s*'s most`
Occupation	`, as a\s*(\w+ \w+)\s*`
Other name	`\s*(\w+)\s*\(`
Relatives	`and his wife , :PER:\s*(\w+)\s*,`
School	`graduated from :ORG:\s*(\w+)\s*`

Table 2: Example patterns acquired by the automatic pattern extraction approach.

In comparison with other WePS2 participants, our runs using manually constructed patterns achieve the highest recall scores [10].

Run	Prec.	Recall	F-measure
UvA_1	2.717	27.317	4.942
UvA_2	**4.417**	**27.415**	**7.608**
UvA_3	0.667	0.151	0.246
UvA_4	0.181	0.027	0.046
UvA_5	3.349	2.844	3.076

Table 3: Results for the attribute extraction task. Best scores for each measure are in boldface.

4.3 Analysis

Experimental results showed that our manual patterns are very effective, in terms of recall. Further, the refined versions considerably improve precision, without losing any of the recall. For 7 out of the 16 attributes (*occupation*, *award*, *major*, *mentor*, *relatives*, *phone*, and *fax*) our UvA_2 run achieved the highest recall among all participating systems [10].

As to the automatic methods, we found that performance strongly depends on the attribute type. For some attribute types, for example *phone* and *website*, no generalizable patterns were extracted. For other attributes, for example *award* and *degree*, patterns were extracted, but did not generalize to the test set. Finally, run UvA_5 extracted patterns that performed better than manually designed patterns for the attributes *birthplace* and *other name*, and high precision was achieved for *nationality*.

Analysis of automatically extracted patterns for attributes where high performance was achieved could help improve manually created patterns. For the attribute birthplace, we find that 14 patterns were extracted, all of which make use of the named-entity type "location." Most of these are combined with the clue word "born", and many contain dates as well. For *nationality*, one of the clues used is again named entities of type "location." Additional clues are references to occupations, such as *"Goethe is one of Germany's most famous writers."* For *other name* the extracted patterns make use of syntactic clues such as commas (indicating appositions), and opening or closing parentheses. From these examples we can conclude that named entities, co-occurring attribute types, and punctuation can form important cues

		UvA_1			UvA_2			UvA_3			UvA_4			UvA_5		
Attribute	#no	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Affiliation	3,105	10.5	**20.3**	13.9	**10.7**	20.2	**14.0**	—	—	—	—	—	—	2.9	0.5	0.9
Award	264	**2.6**	**21.2**	**4.7**	**2.6**	16.7	4.5	—	—	—	—	—	—	—	—	—
Birth place	301	0.2	**32.1**	0.5	15.8	1.0	1.9	0.1	0.3	0.2	—	—	—	**18.6**	8.0	**11.2**
Date of birth	370	1.3	**39.6**	2.5	**33.5**	15.4	**21.2**	5.3	1.4	2.2	—	—	—	9.7	11.7	10.6
Degree	335	9.6	**23.3**	**13.6**	**14.2**	10.4	12.0	—	—	—	—	—	—	—	—	—
Email	209	**24.3**	**54.5**	**33.6**	**24.3**	**54.5**	**33.6**	—	—	—	—	—	—	—	—	—
Fax	65	**3.9**	**70.8**	**7.4**	**3.9**	**70.8**	**7.4**	—	—	—	—	—	—	—	—	—
Major	173	**18.0**	10.4	13.2	13.6	**16.8**	**15.0**	—	—	—	—	—	—	—	—	—
Mentor	343	**0.3**	**16.9**	**0.5**	**0.3**	**16.9**	**0.5**	—	—	—	—	—	—	—	—	—
Nationality	250	3.7	**66.8**	6.9	27.2	27.2	**27.2**	—	—	—	—	—	—	**39.5**	6.8	11.6
Occupation	3,292	**6.6**	34.8	**11.1**	5.1	**38.3**	9.0	2.7	0.3	0.5	0.6	0.0	0.1	4.5	5.2	4.8
Other name	797	15.0	0.8	1.4	**62.5**	0.6	1.3	0.2	0.3	0.2	0.2	0.3	0.2	1.1	**6.1**	**1.8**
Phone	219	**13.8**	**74.4**	**23.3**	**13.8**	**74.4**	**23.3**	—	—	—	—	—	—	—	—	—
Relatives	914	**7.0**	30.3	**11.4**	4.1	**55.0**	7.7	—	—	—	—	—	—	—	—	—
School	494	2.6	8.3	4.0	**45.6**	**8.3**	**14.1**	—	—	—	—	—	—	—	—	—
Web site	154	**25.4**	**22.7**	**24.0**	**25.4**	**22.7**	**24.0**	—	—	—	—	—	—	—	—	—
Overall	–	2.717	27.317	4.942	**4.417**	**27.415**	**7.608**	0.667	0.151	0.246	0.181	0.027	0.046	3.349	2.844	3.076

Table 4: Breakdown of results per attribute type. #no denotes the total number of times that attribute occurs in the test set. For each run precision (P), recall (R), and F-measure (F) are reported. Best scores for each attribute and measure are in boldface. Scores of zero are denoted by '—' to improve readability.

for the attribute extraction task.

In cases where no suitable patterns were found, the reasons could be (1) the limited amount of sample data available for the particular attribute, (2) tokenization and the specific way in which patterns are extracted and matched, and (3) errors of the NE-tagger for runs where named entities were used. Further research is required to quantify the influence of each of these on extraction performance.

We conclude that it is possible to automatically learn patterns for attribute extraction. However, performance of the approach used strongly varies with the attribute type. Interesting directions for future work are the use of more training data (possibly using bootstrapping approaches), improved matching of training samples and patterns, and improved pattern selection. Also, extracted patterns could be inspected manually to further improve performance of manual extraction.

5. CONCLUSIONS

We described our participation in the Second Web People Search workshop (WePS2) and detail our approaches. For the clustering task, our focus was on replicating the lessons learned at WEPS1 on the data set made available as part of WEPS2 and on experimenting with a voting-based combination of clustering methods. We found that clustering methods display the same overall behavior on the WEPS1 and WESP2 data sets and that the hierarchical clustering approach delivers the best performance, even outperforming voting-based combinations. For attribute extraction, we explored approaches using pattern matching with manually and automatically constructed patterns. Manual patterns were constructed using expert knowledge and following analysis of sample data. Automatic pattern construction extracted textual and syntactic context around training samples and selects patterns which are expected to perform well based on leave-one-out evaluation. Experimental results showed that manually constructed patterns are very effective for obtaining high recall. For automatically extracted patterns performance varied widely depending on the attribute type.

As to future work, we are keen to set up a detailed analysis of the differences between the clustering methods used and to consider alternative combination methods. For the attribute extraction task we are particularly interested in finding out wether larger amounts of training data help improve our approaches.

Acknowledgments

This research was supported by the the DuOMAn project carried out within the STEVIN programme which is funded by the Dutch and Flemish Governments (http://www.stevin-tst.org) under project number STE-09-12, and by the Netherlands Organisation for Scientific Research (NWO) under project numbers 017.001.190, 640.001.501, 640.002.501, 612.-066.512, 612.061.814, 612.061.815, and 640.004.802.

References

[1] J. Artiles, J. Gonzalo, and S. Sekine. The SemEval-2007 WePS evaluation: Establishing a benchmark for the web people search task. In *SemEval, ACL*, 2007.

[2] J. Artiles, J. Gonzalo, and S. Sekine. WePS 2 evaluation campaign: overview of the web people search clustering task. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, April 2009.

[3] K. Balog, L. Azzopardi, and M. de Rijke. Personal name resolution of web people search. In *WWW2008 Workshop: NLP Challenges in the Information Explosion Era (NLPIX 2008)*, April 2008.

[4] K. Balog, L. A. Azzopardi, and M. de Rijke. Resolving person names in web people search. In R. Baeza-Yates and I. King, editors, *Weaving Services, Location, and People on the WWW*. Springer, 2009.

[5] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370, 2005.

[6] D. R. Hill. A vector clustering technique. In Samuelson, editor, *Mechanised Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam, 1968.

[7] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999. URL `citeseer.ist.psu.edu/hofmann99probabilistic.html`.

[8] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7:217–240, 1971.

[9] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[10] S. Sekine and J. Artiles. WePS 2 evaluation campaign: overview of the web people search attribute extraction task. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, April 2009.