Table 2: Extracted terms from the AlterEgo website and the Wikipedia page about Mallorca

AlterEgo	Mallorca
add-ons	majorca
addons	island
Nicolaas	palma
Matthijs	islands
CSTIT	spanish
Nicolaas Matthijs	balearic
Cambridge	mallorca
Language Processing	cathedral
Google	Palma de Mallorca
keyword extraction	port

re-extraction using the Viterbi algorithm. The outcome of this algorithm run on two sample web pages can be seen in Table 3.

Noun Phrases

Noun phrases were extracted by taking the text from each web page and splitting it into sentences using a sentence splitter from the OpenNLP Tools[5]. The OpenNLP tokenization script was then run on each sentences. The tokenized sentences were then tagged using the Clark & Curran Statistical Language Parser[6] [3], which assigns a constituent tree to the sentence, part of speech tags to each word. Noun phrases were then extracted from this constituent tree.

3.1.3 Term List Filtering

To reduce the number of noisy terms in our user representation, we also tried filtering terms using two different approaches.

The first is dictionary filtering using the lexical database WordNet 3.0[7], retaining only words marked as nouns. However, WordNet is far from complete and hence tends to remove many nouns words, in particular most proper nouns.

The alternative way we tried was removing uncommon words using the Google N-Gram corpus[8]. This approach is general successfully remove non-sensical word, while allowing less common nouns such as proper nouns. However, as part of speech tags are not a part of the Google N-Gram corpus, it could not filter out stop words and determiners.

3.1.4 Term Weighting

After the list of terms has been accumulated and potentially filtered, we computed weights for each term using three methods.

TF Weighting

The most straightforward implementation we consider is Term Frequency (TF) weighting. We define a frequency vector $\vec{F}$ that contains the frequency counts of a given term t_i for all of the input data sources, as shown in equation (1). For example, f_{title} is the number of times a given term t_i occurs in all of the titles in the user's browsing history. We calculate a term weight based on the dot product of these frequencies with a weight vector $\vec{\alpha}$:

$$\vec{F}_{t_i} = \begin{bmatrix} f_{title_{t_i}} \\ f_{mdesc_{t_i}} \\ f_{textt_i} \\ f_{mkeyw_{t_i}} \\ f_{terms_{t_i}} \\ f_{nphrases_{t_i}} \end{bmatrix} \quad (1)$$

$$w_{TF}(t_i) = \vec{F}_{t_i} \cdot \vec{\alpha} \quad (2)$$

For simplicity, in our experiments we limit ourselves to three possible values for each weight α_i: 0, ignoring the particular data field, 1, including the particular data field, and $\frac{1}{N_i}$, where N_i is the total number of terms in field i. This gives more weight to terms in shorter fields (such as the meta keywords or title fields). We call the latter *relative weighting*.

TF-IDF Weighting

The second possibility for term weights we consider is TF-IDF (or Term Frequency, Inverse Document Frequency) weighting. Here, words appearing in many documents are down-weighted by the inverse document frequency of the term:

$$w_{TFIDF}(t_i) = \frac{1}{log(DF_{t_i})} \times w_{TF}(t_i) \quad (3)$$

To obtain IDF estimates for each term, we use the inverse document frequency of the term on all web pages using the Google N-Gram corpus.

Personalized BM25 Weighting

The final weight method we consider is taken from Teevan et al [31]. They propose a personal term weight similar to how BM25 weights terms. Specifically, they propose a weight

$$w_{BM25}(t_i) = log\frac{(r_{t_i}+0.5)(N-n_{t_i}+0.5)}{(n_{t_i}+0.5)(R-r_{t_i}+0.5)}, \quad (4)$$

where N represents the number of documents on the web (estimated from the Google N-Gram corpus, 220,680,773), n_{t_i} is the number of documents in the corpus that contain the term t_i (estimated using the Google N-Gram corpus), R is the number of documents in the user's browsing history and r_{t_i} is the number of documents in the browsing history that contains this term within the selected input data source.

While this method allows us to compare our results against the approach proposed by Teevan et al., note that we do not have access to users' full Desktop index, and are limited to their browsing history, making our implementation potentially less effective.

3.2 Re-ranking Strategies

Like previous work, we use the user profile to re-rank the top results returned by a search engine to bring up results that are more relevant to the user. This allows us to take advantage of the data search engines use to obtain their initial ranking to first obtain a small set of results that can then be personalized. In particular, [31] noted that chances are high that even for an ambiguous query the search engine will be quite successful in returning pages for the different meanings of the query. We opt to retrieve and re-rank the

[5]http://opennlp.sourceforge.net/
[6]http://svn.ask.it.usyd.edu.au/trac/candc/wiki
[7]http://wordnet.princeton.edu/
[8]http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html