Matching Queries to Frequently Asked Questions: Search Functionality for the MRSA Web-Portal

Almer S. Tigelaar & Rieks op den Akker
{a.s.tigelaar, infrieks}@cs.utwente.nl
Database & Human Media Interaction Groups,
Computer Science Department,
University of Twente

Fenne Verhoeven
f.verhoeven@utwente.nl
Technical & Professional Communication Group,
Behavioural Sciences Department,
University of Twente

ABSTRACT

As part of the long-term EUREGIO MRSA-net project a system was developed which enables health care workers and the general public to quickly find answers to their questions regarding the MRSA pathogen. This paper focuses on *how* these questions can be answered using Information Retrieval (IR) and Natural Language Processing (NLP) techniques on a Frequently-Asked-Questions-style (FAQ) database.

Keywords

Information retrieval, question-answering, linguistic processing, performance evaluation.

1. INTRODUCTION

Methicillin-Resistant *Staphylococcus aureus* (MRSA) is a strain of pathogens that is resistant to common antibiotics and is therefore hard to combat. It forms a significant threat to people with a weakened immune system.

The MRSA web-portal was developed to provide information to health care workers and the general public regarding MRSA. It is actively used by several Dutch and German hospitals and also publically accessible[1]. This research has been conducted to support this portal.

The underlying databases consist of a set of reference questions that have been collected by a field investigation and answers to these questions written by expert microbiologists [22]. The web-interface enables users to browse through the questions categorically and has a search textfield in which users can enter a query. This research focuses on providing the underlying functionality for this textfield as a component named the MRSA-QA system.

The system utilises four domain-bound question-answer sets: Dutch/Professional, Dutch/Public, German/Professional, and German/Public. The two sets tailored for the professional domain each consist of about 160 pairs whereas the two sets geared towards the general public consist of approximately 220 pairs. The questions can be viewed as perfect indices to the answers resembling the set-up of a FAQ [5].

The domain of MRSA-QA is clearly demarcated and can be classified as a *Restricted Domain Question Answering* (RDQA) system [7]. The underlying database is explicitly structured. The answers consist of several fields: a title, short guideline, instruction video, long comments section, sources, examples and additional keywords. Semantic information is represented via links in the database implicitly such as categorisation, connections between questions and answers, and links between answers and other relevant answers. The system should be able to adjust to new data, since the database can change and grow over time.

The problem can be stated as: given a query in the form of a question or a set of keywords, the system has to display a list of appropriate answers ranked by relevance in descending order. Based on this problem statement the research question can be phrased as follows:

'How can the user be provided with the most appropriate answer(s) for his or her query within the restricted MRSA domain given the available structured MRSA corpus?'

This paper focuses on answering that question by looking at and evaluating a range of techniques using the contents of the MRSA corpus. While the results are corpus specific, the methods used are generic and likely to be useful for other RDQA systems.

2. DATA

Separate datasets exist for each domain (Professional and Public) and language (Dutch and German) combination. So, there are four in total. All of these are stored in databases exhibiting the same structure, shown in Figure 1.

Starting at the top: the categories table clusters related questions for example all questions that have something to do with treatment or discharge. Each question can belong to one or more categories and points to one answer in the database. Answers can be pointed to by one or more questions and each answer may point to other related answers thereby establishing semantic links.

The answers contain the most information that can be exploited for indexing and matching in the form of fields. The title field of the answer is usually a reformulation of a related question. The guideline and comment fields are the most relevant for matching, since they contain the most free-form text with content that actually answers a question. Sources and examples are less relevant fields. Both of them contain mostly links to external documents with more clari-

[1] `http://www.mrsa-net.nl/`, `http://www.mrsa-net.nl/de`

DIR 2009, 9^{th} Dutch-Belgian Information Retrieval Workshop,
February 2–3, 2009, University of Twente, Enschede, The Netherlands.

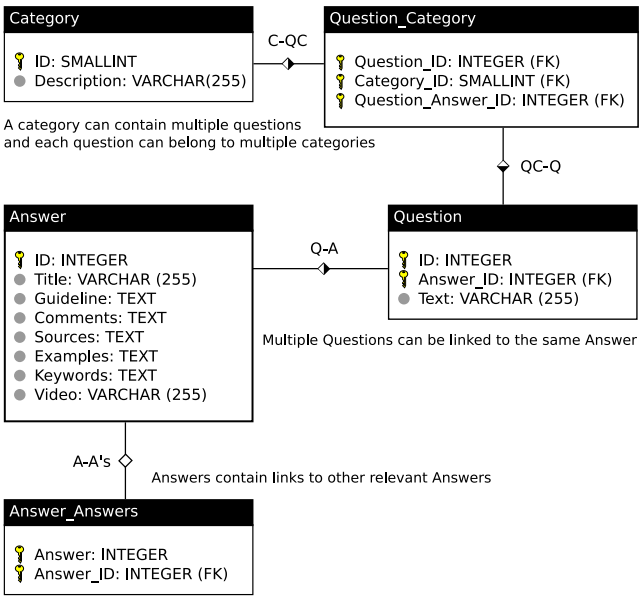

Figure 1: *Structure of an MRSA database (diamonds represent one-to-many (black/white) and many-to-many (white) relations).*

fication and are of use only when a relevant answer is being displayed. The keywords field contains a list of non-obvious keywords related to this answer and as such conveys additional semantics. The instruction video refers to non-text data. Hence, it will not be considered further.

Based on the data characteristics, we define a *document* within the MRSA-QA system as follows:

'A document consists of the title, guideline, comment and keywords of an answer plus the texts of all questions that refer to the answer.'

3. TECHNIQUES

Three categories of techniques can be distinguished: pre-processing techniques affecting the indexing and matching process, the ranking techniques and post-processing techniques which also affect the ranking process, but do not require any additional operations while indexing. The performance of six pre-processing techniques, two ranking techniques and three post-processing techniques was investigated. This section briefly explains each of the techniques.

3.1 Pre-Processing Techniques

McNamee & Mayfield and others report increases in matching performance using *character n-gramming*. In this technique a window of n characters is slid across a document. A snapshot is made of the characters that are visible in each position and used as index. There are two different approaches: n-gram each word in the text individually, called *within-word* character n-gramming, or also slide across the word boundaries in the text, called *between-word* character n-gramming. We chose for the latter approach, since it properly captures inter-word relations [8, 12].

The example presented in Table 1 provides some intuition for n-gramming. The value for n that was used is four. First the input document is padded with $n-1$ spaces on both sides (so all character sequences are properly captured). After this a window of length four is slid across the padded input text

Table 1: *Character n-gramming (spaces shown as underscores).*

Padded Input: _ _ _ _ t h e _ f o x _ _ _ _				
_ _ _ t	_ _ t h	_ t h e	t h e _	h e _ f
e _ f o	_ f o x	f o x _	o x _ _	x _ _ _

and all snapshots are recorded as shown in Table 1.

This process is applied to all documents in the corpus while indexing and on the query as well. The index thus becomes huge, since there are $k - n + 1$ n-grams for a document with k characters. Padding adds another constant $2 \cdot (n-1)$ to this. This is the primary argument against using n-grams, since such large indices adversely affect performance. For this method a value for n of five was used for both Dutch and German languages. We will show later why this is an optimal choice. Note that we use the n-grams *as replacement* of the original words they were derived from.

Word n-gramming bears much resemblance to character n-gramming. Here instead of regarding characters as the most atomic unit, words are used. While word n-gramming also leads to a larger indexing table with respect to bag-of-words indexing, the size increment is clearly not as big as with character n-grams. A value of n of two was determined to be the most optimal by experiment [8].

Decompounding is the process where a compound word is taken and broken into its individual components. A compound word consists of two or more words that can also be used individually. These are glued together in the compound to form a new or related meaning. Both Dutch and German are languages in which compounding is common. Two different approaches have been tested. One that breaks compounded words in the shortest possible units (Decompounding S) and one that breaks them into the longest possible units (Decompounding L). Wordlists were used that were available for both the Dutch and German language. These are wordlists that are installed by default on modern UNIX systems; respectively *wdutch* that follows the 1996 Dutch spelling and *wngerman* that follows the 1998 German new orthography spelling. The decompounding engine adds the component parts of decompounded words to the index for each document. The same is done at query-time [3, 6].

Words can also be reduced to their stem form. This procedure removes inflection from words, as in changing 'walking' to 'walk'. It also transforms derivatives to their root, as in stemming 'national' to 'nation'. For finding the stem of a word the *Porter stemmer* was chosen. Both the unstemmed word form and the stemmed variant are added to the index and the query [1, 8, 14].

Remember that the documents are constructed based on several fields. These are: title, guideline, comments, keywords and the texts of the referring questions. While normally these fields are simply combined into one document, field weighting uses a different approach and assigns a weight to each individual field. For example: when a keyword in a query is found in a document title it has a bigger influence on the final document rank than when the same keyword appears in the comment section of that document. We call this *field weighting*. Suitable weights were determined by experiment. Weight pre-multiplication was done during the indexing phase. This method does not require extra calculations during query execution and ranking time.

Table 2: *Categorical clustering.*

Rank	1	2	3	4	5	6	7	8	9	10
Document	a	b	c	d	e	f	g	h	j	k
Category	2	2	2	2	5	4	3	7	6	2

Finally, *word relation expansion* considers all words in a document and all words in a query and finds synonyms for each of them. Of course those synonyms also in turn have synonyms, but only one level of depth is used. All that is done is adding extra synonym words at the end of each document and at the end of a query as well. Those synonyms are extracted from EuroWordNet. Note that the German WordNet is only about one third the size of the Dutch WordNet. This somewhat impairs performance for German [10, 23].

3.2 Ranking Techniques

For the basic ranking process initially mostly tried and tested techniques were chosen like *term frequency / inverse document frequency* ($tf \cdot idf$) combined with the *vector-space model* (vsm). We later experimented with *Okapi Best Match 25* (BM25). We chose for these techniques since they are well understood and provide a solid foundation for a production system [16, 17, 18, 20].

3.3 Post-Processing Techniques

One of the interesting properties of the database is that all questions are categorised. Implicitly each answer can also belong to several categories, since an answer can be referred to by one or more questions. Thus, each document belongs to one or more categories. This characteristic can be exploited. The idea behind this approach is to look at the representation of the categories within the result list after executing a query. Imagine that there is a list with ten results with categories as displayed in Table 2. The first four documents and the tenth document belong to category two (the simplifying assumption that each document belongs to only one category is made).

Since category two is so strongly represented in the result set, document k that is now in position ten might actually better be in position five. This intuition is precisely what *categorical clustering* tries to capture. If a category is strongly represented in the result list, with respect to the entire dataset, the documents that also fall in that category are pushed up. To achieve this first the representation strength of each category is determined based on the entire database. The formula is similar to *idf*, but applied to category counts:

$$icf\,(c) = \frac{\#D}{\#\{d \mid (d,c) \in D\}} \tag{1}$$

where D is the set of all documents in the entire database and their category. Finally, recalculating the scores is done as follows:

$$score_2\,(d) = score\,(d) \cdot \sum_{c \in C_d} icf\,(c) \tag{2}$$

where $score\,(d)$ is the original strength of document d and C_d is the set of *all* categories document d belongs to.

Besides categories, the MRSA corpus also contains links between answers. Each document contains a list of references to other documents that might also be relevant. To exploit this, we take the same approach as with categorical clustering: first a query is executed as usual and then the result list is examined and re-ranked. The references to other documents in the top N documents in the result list are collected. After that all documents in the list are reweighted based on the frequency in which they appear in this collection of referred documents.

Table 3: *Referential clustering.*

Rank	*Weight*		*Reference*
1	10	→	6
2	8	→	6
3	6	→	6
4	4	→	7
5	2	→	8
6	1	→	9

Let us look at the example in Table 3 which shows a result list consisting of six items. The sixth item with weight one falls just outside of the top five. Notice that document one, two and three all carry a reference to document number six, this is called a referential cluster. Reweighting proceeds by simply multiplying the score of document six by three which yields $1 \cdot 3 = 3$. Hence, the document in position five with score two is pushed down in the result list and replaced by the document in position six. Hence, a document in the result list which is indicated to be relevant by the top of the list has been pushed up. Of course, only the top N documents should have influence on pushing up other documents. A value of N of 5 was determined by experiment. In addition to this the frequency of appearance in the referring document list is dampened by the log function which results in more conservative multiplication behaviour (this is not applied in the example).

To formalise we first calculcate the frequency of appearance of a document d within the references:

$$frequency\,(d) = \#\{r \mid (r,d) \in R_5\} \tag{3}$$

where R_5 is the partial rank list consisting of the first *five* ranks r and their reference(s) d. R_5 can actually contain more than five pairs in case of multiple references at the same rank.

The actual reweighting now proceeds as follows:

$$score_2\,(d) = score\,(d) \cdot f\,(frequency\,(d)) \tag{4}$$

where $score\,(d)$ is the original weight of document d and f is a monotonous function. We used the non-linear: $f\,(x) = \log_2\,(\max\,(x+1,2)))$. The formula ensures that only documents that are referred to *multiple times* can affect the ranking. Rescoring is performed over the entire ranked list.

Latent Semantic Indexing (LSI) is a dimensionality reduction method capable of finding semantic relations. Unfortunately it increases recall and tends to decrease precision which is the opposite of what is desirable for the MRSA-QA system. It is also of less use when the corpus is homogeneous and is computationally expensive. A cheaper method with similar capabilities is *blind relevance feedback*. Each document in a result set contains many more words than only the words in the query. This can be exploited by adding the top M most frequently occurring terms in the top N documents

in the ranked list to the original query and re-executing the search with this newly expanded query. So, each time a query is posed to the system two passes are made: one to determine the initial ranked result list and extract the top terms, and then another with the expanded query. For the number of documents N a value of at most[2] three was chosen for selecting the M top terms. The value of M was set to five. An extra constraint was added so only terms of at least length four are used. This is to avoid selecting smaller non content-words, such as articles [11, 12].

4. EVALUATION

4.1 Sets & Metrics

For evaluation purposes several sets of queries were created by hand. Every set has the following characteristics: For each query the most relevant document is indicated. For each document in each of the databases there is at most one query in a single set, but there may also be no query. Nevertheless, most sets contain queries for nearly all answers in the associated database. The difference between the evaluation sets is in the word usage in the queries: each set contains a different 'wording' which attempts to retrieve the same answer. An example:

EXAMPLE 1. *Ways to retrieve document number 64, in the evaluation set, concerning 'how many times someone can acquire MRSA' (English translations shown in upright font).*

hoeveel keer MRSA
how many times MRSA

risicofactoren infecties
risk factors infections

hoe is het mogelijk om vaker MRSA te krijgen?
how is it possible to acquire MRSA multiple times?

hoe groot is de kans dat infectie vaker optreedt?
how big is the chance that the infection re-occurs?

The first two queries shown are keyword queries, while the last two are question queries. For the final evaluations eight sets for each domain / language combination were used, four with keyword-based queries and four with question-based queries. Note that not all sets cover all documents in the database. They were weighted respective to their size for computations involving averages and deviations. All combined over 6000 queries were executed for every evaluation run. The assumption is made that the queries in the sets are somewhat representative of the queries posed by 'real' users of the system. It is difficult to give any guarantees regarding this, especially since the evaluation sets are based on the answers that are present in the databases (they are based on the answers that the system *should be able* to provide). Real end-users might come up with radically different phrasings. Also, closed domain systems are usually faced with longer queries on average than open domain systems. Usually the average query length is about five terms per query. This trait is also present in the evaluation query sets, which gives a positive indication of their representativity [13].

When a query is executed the system generates a list of documents that are sorted according to their presumed relevance. For measuring how well the system performs it is necessary to define what is considered to be a good list of returned documents. Note that for any single query that is executed three types of document may be returned: zero or one most relevant document, zero or more other documents that are partially relevant and zero or more irrelevant documents. Armed with these concepts the definition of what the system should do is: The most relevant document should be at position 1 in the list. All other n partially relevant documents should populate position 2 up to $n + 1$. In the evaluation sets the most relevant document is indicated explicitly and the partially relevant documents are those that the most relevant document refers to.

For expressing the position of the most relevant document the Mean Reciprocal Rank (MRR) was used. Mean Average Precision (MAP) and Mean R-Precision (MRP) are used for measuring the presence of other relevant documents. They measure precision and recall, but are in fact highly correlated as they both estimate the area under the recall-precision curve. Hence, they should show similar results during evaluation. A non-rank related performance measure was also used, namely the time it takes for a single query to execute. Significance testing was performed using one-tailed paired unequal variance t-tests. We recognise that these tests make assumptions about the shape of the data which might not necessarily hold, but we believe that the resulting alpha values can still be compared [2, 4, 13].

4.2 Baseline

From here on forward an implementation of a technique will be referred to as an engine. The theoretical techniques as explained in this paper provide the blueprints for these practical implementations.

For evaluating the effectiveness of the techniques described previously, we need some sort of baseline to compare against. Six candidate baseline engines were developed. All of these engines use a basic bag-of-words approach. The difference is in the weighting method that is used. Note that as part of standard pre-processing all documents are stripped of any mark-up tags they may have. Also, words are lowercased and stripped of diacritical marks. Words consisting of only one character, or that do not include alphanumeric characters, are ignored. If a query is entered the documents in which at least one of the words in the query appears, which is determined by looking at the index, become part of the resulting set of documents. After this the frequencies of the words are used to assign a score to each document and turn the set of selected documents into a ranked list.

Roughly there are two approaches that have been evaluated. The first works by treating both query and documents as vectors also known as the *vector-space model*. There are three such vector-based baseline engines: Vector, Vector Log and Vector Normalised Log. The idea behind all of them is the same. They differ only in the variants of the function that they apply. All of them performed quite poorly in comparison with the other methods that were tried [11, 18].

The other approach is conceptually simpler and works by summing the scores of each document on each individual query term. Three variants of this were also tried, namely Additive, Additive Log and a self-developed approach. Statistical tests were performed to determine which of the six engines performed best. Based on the MRR the Additive and self-developed variants scored best. While the latter is faster than the normal additive method it was not chosen since it was not tested outside the MRSA corpus.

[2]Depends on the size of the result list.

Two variants of the Additive approach were used, one employing the basic $tf \cdot idf$ formula (shown in equation 5 where d is a document and w_k is a query term) and the other (at a later stage) employing Okapi BM25 [15, 16].

$$score\,(d) = \sum_{k=1}^{n} tf\,(d, w_k) \cdot idf\,(w_k) \tag{5}$$

4.3 Individual techniques

First, we need to select techniques based on their performance increment over the baseline. We report the average performance and deviation over four *keyword* evaluation sets and four *question* evaluation sets. The numbers in the tables shown are for the Dutch *professional* database using $tf \cdot idf$ scoring. Colours (shades) and arrows are used to indicate performance increment △ (*green*) or decrement ▽ (*salmon*) over the baseline (*blue*). All numbers are rounded, so even if a performance number seems exactly the same as the baseline, it may still differ to the right of the last rounded digit. For the referential clustering engine the same data is used as for calculating the MAP and MRP scores which makes the increase in these values optimistic for this engine. Hence, those fields have been coloured ▲▼ (*pink*) in the various tables.

Table 4 shows that character n-gramming, decompounding and stemming positively affect the MRR. A similar pattern exists for the MAP and MRP shown in Table 5, but in contrast with the MRR the blind query expansion shows quite some improvement over the baseline here. Categorical clustering and word n-gramming decrease performance. Word relation expansion is fairly neutral and yields no convincing performance advantage. We believe this is due to the domain specific corpus and WordNet's generality [21].

Of course, besides the Dutch professional keyword sets there are also three other keyword evaluation sets. We describe performance of those sets in terms of differences with the tables:

- German/professional: Baseline keyword performance is quite a bit worse than for Dutch (MRR -0.06, MAP -0.05). The same holds for the question sets (MRR -0.08, MAP -0.05). The results show relatively similar performance patterns. However, German language appears to benefit more from both character n-gramming and stemming than Dutch, which is in line with findings of others [3, 8].
- Dutch/public: The overall performance is worse (MRR -0.10, MAP -0.01). Field weighting shows a slight performance increase which is the only exception with regard to the trends in performance on the Dutch professional dataset.
- German/public: Interestingly baseline performance on keywords is actually comparable to the Dutch professional dataset (MRR -0.01, MAP +0.03). Patterns are again similar, the exception being query expansion which performs a bit worse (MRR -0.02).

Performance on the Dutch question sets is shown in Table 6 and Table 7. Notice that the baseline performance on questions is much better than on keywords. This is probably caused in part by the inclusion of question-words (why, who, where, etcetera) in the question based queries.

Table 4: *Keywords: Reciprocal Rank & Timings (sec).*

	RR		*Time* $\times 10^{-3}$	
Engine	μ	σ	μ	σ
Baseline	0.51	0.40	7.1	5.5
Categorical clustering	0.44^{∇}	0.39^{Δ}	10.0^{∇}	8.4^{∇}
Character n-gramming	0.54^{Δ}	0.38^{Δ}	25.4^{∇}	11.3^{∇}
Decompounding large	0.52^{Δ}	0.39^{Δ}	9.6^{∇}	6.0^{∇}
Decompounding small	0.55^{Δ}	0.38^{Δ}	13.0^{∇}	7.0^{∇}
Field weighting	0.49^{∇}	0.40^{∇}	7.4^{∇}	6.8^{∇}
Query expansion	0.47^{∇}	0.39^{Δ}	20.1^{∇}	6.4^{∇}
Referential clustering	0.51^{∇}	0.39^{Δ}	7.2^{∇}	5.2^{Δ}
Stemming	0.52^{Δ}	0.39^{Δ}	7.9^{∇}	5.5^{∇}
Word n-gramming	0.32^{∇}	0.41^{∇}	5.6^{Δ}	5.0^{Δ}
Word relation expansion	0.48^{∇}	0.40^{Δ}	9.7^{∇}	6.9^{∇}

Table 5: *Keywords: Average Precision & R-Precision.*

	AP		*RP*	
Engine	μ	σ	μ	σ
Baseline	0.28	0.20	0.29	0.20
Categorical clustering	0.26^{∇}	0.20^{Δ}	0.27^{∇}	0.21^{∇}
Character n-gramming	0.32^{Δ}	0.19^{Δ}	0.30^{Δ}	0.19^{Δ}
Decompounding large	0.31^{Δ}	0.20^{∇}	0.31^{Δ}	0.20^{Δ}
Decompounding small	0.33^{Δ}	0.20^{Δ}	0.31^{Δ}	0.20^{Δ}
Field weighting	0.29^{∇}	0.21^{∇}	0.29^{Δ}	0.21^{∇}
Query expansion	0.32^{Δ}	0.21^{∇}	0.30^{Δ}	0.22^{∇}
Referential clustering	$0.33^{\blacktriangle}$	$0.23^{\blacktriangledown}$	$0.33^{\blacktriangle}$	$0.23^{\blacktriangledown}$
Stemming	0.29^{Δ}	0.20^{∇}	0.30^{Δ}	0.20^{Δ}
Word n-gramming	0.15^{∇}	0.20^{Δ}	0.16^{∇}	0.20^{Δ}
Word relation expansion	0.29^{Δ}	0.20^{∇}	0.28^{∇}	0.20^{Δ}

Positive increments are visible for character n-gramming, small decompounding and field weighting across all the metrics. Again, the performance differences on the other *question* evaluation datasets:

- German/professional: Similar performance as on keywords, baseline question performance is worse (MRR -0.06, MAP -0.05). Although the increase caused by stemming and character n-gramming is higher.
- Dutch/public: Shows a reduction in baseline performance (MRR -0.19, MAP -0.06). Relative to this character n-gramming yields a slightly higher improvement (MRR +0.05, MAP +0.04).
- German/public: Baseline performance is quite poor (MRR -0.22, MAP -0.09). The overall relative patterns are the same, but character n-gramming shows a slightly higher performance gain on par with Dutch public dataset.

The final choice is based on the significance of difference between MRR and MAP α values. Only methods that had at least one significant positive effect and no negative effect with respect to baseline on the professional dataset for both of these metrics and for both languages were selected. The final selection of five of the ten techniques is: character n-gramming, decompounding small, field weighting, referential clustering and stemming.

Table 6: *Questions: Reciprocal Rank & Timings (sec).*

	RR		*Time* $\times 10^{-3}$	
Engine	μ	σ	μ	σ
Baseline	0.60	0.39	17.2	7.9
Categorical clustering	0.47^{∇}	0.41^{∇}	23.4^{∇}	10.2^{∇}
Character n-gramming	0.63^{Δ}	0.37^{Δ}	61.4^{∇}	28.6^{∇}
Decompounding large	0.59^{∇}	0.39^{Δ}	30.0^{∇}	13.8^{∇}
Decompounding small	0.61^{Δ}	0.38^{Δ}	36.0^{∇}	16.2^{∇}
Field weighting	0.61^{Δ}	0.40^{∇}	41.1^{∇}	21.5^{∇}
Query expansion	0.55^{∇}	0.38^{Δ}	40.1^{∇}	13.3^{∇}
Referential clustering	0.58^{∇}	0.38^{Δ}	17.2^{Δ}	7.8^{Δ}
Stemming	0.60^{∇}	0.39^{Δ}	19.5^{∇}	9.0^{∇}
Word n-gramming	0.51^{∇}	0.41^{∇}	8.3^{Δ}	5.9^{Δ}
Word relation expansion	0.49^{∇}	0.40^{∇}	46.9^{∇}	28.0^{∇}

Table 7: *Questions: Average Precision & R-Precision.*

	AP		*RP*	
Engine	μ	σ	μ	σ
Baseline	0.34	0.20	0.33	0.20
Categorical clustering	0.28^{∇}	0.21^{∇}	0.26^{∇}	0.22^{∇}
Character n-gramming	0.36^{Δ}	0.20^{Δ}	0.34^{Δ}	0.19^{Δ}
Decompounding large	0.34^{∇}	0.20^{∇}	0.33^{∇}	0.21^{∇}
Decompounding small	0.35^{Δ}	0.20^{Δ}	0.33^{Δ}	0.20^{Δ}
Field weighting	0.35^{Δ}	0.20^{Δ}	0.33^{Δ}	0.21^{∇}
Query expansion	0.36^{Δ}	0.21^{∇}	0.33^{Δ}	0.21^{∇}
Referential clustering	$0.44^{\blacktriangle}$	$0.25^{\blacktriangledown}$	$0.41^{\blacktriangle}$	$0.24^{\blacktriangledown}$
Stemming	0.34^{∇}	0.20^{Δ}	0.33^{Δ}	0.20^{Δ}
Word n-gramming	0.25^{∇}	0.20^{Δ}	0.26^{∇}	0.19^{Δ}
Word relation expansion	0.29^{∇}	0.19^{Δ}	0.27^{∇}	0.20^{Δ}

4.4 Optimal value of n

A value of five was chosen for character n-gramming. This choice was initially based on the average length of the words in the professional Dutch and German datasets. Since character n-gramming is quite a fundamental difference with normal bag-of-words indexing, it is important to show that this is also the beste value for n. We tested several values of n for both the Dutch and German professional datasets. These tests were conducted on top of the additive $tf \cdot idf$ baseline that was chosen earlier.

Figure 2 shows how the value of n, ranging from two to eight, affects the MRR and MAP. For keyword queries 4-grams are slightly more optimal for Dutch with respect to 5-grams according to the MRR. Nevertheless, for German keyword queries this makes no difference. For question queries the picture is a bit different: the optimal value of n appears to be six for Dutch, while for German the performance tops-off after an n of five. A similar trend can be seen for the other measures. The differences however are not very large. Based on the trend-line the optimal value of n is four or five. This is the same finding as McNamee & Mayfield who show that these values give optimal performance for most European languages [8, 12].

An other measure is the retrieval time. Due to their length, this is significantly longer for question-based queries than for keyword-based ones. The lower the value of n, the larger the inverted index. Hence, higher values of n are preferred simply because of their retrieval speed advantage. But the speed difference between subsequent values of n becomes smaller as n increases: for 7 and 8-grams the difference is minimal. Choosing between 4 and 5-grams, based on speed

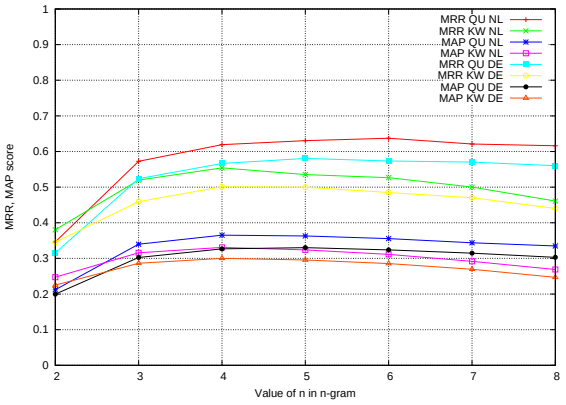

Figure 2: *MRR and MAP scores for n-grams on professional keyword (KW) and question (QU) evaluation sets for Dutch (NL) and German (DE).*

alone, 5-grams are preferred. McNamee & Mayfield report a tenfold speed penalty when using n-grams, but in our tests 5-grams are 'only' four times slower than the baseline approach. This confirms their own suspicions regarding the fact that the increase in processing time while using n-grams is an artefact of their implementation [12].

We eventually settled on a value for n of five, since it only slightly degrades performance on the Dutch MRR for keyword-based queries and has a mild positive effect for all question-based queries. 5-grams also have a speed performance advantage over 4-grams ($\sim$ 25% for question queries and $\sim$ 15% for keyword queries). It is safe to confirm McNamee & Mayfield's conclusion that higher values of n are beneficial for time-wise performance. It can be generally stated that: 'For n-grams and n+1-grams, the n+1-grams are preferred speed-wise when there exists no significant difference in ranking performance between the n-grams and n+1-grams'.

4.5 Combined techniques

Knowing the performance of individual techniques, we can combine them to improve performance. Abbreviations are used for the five previously selected techniques in this section: (C)haracter n-gramming, (D)ecompounding Small, (F)ield weighting, (R)eferential Clustering and (S)temming.

As can be seen in Table 8 and Table 9, combining character n-gramming with field weighting yields poor results for the Dutch MRR's. Hence, field weighting was dropped. The next combination that was tried was character n-gramming plus stemming, this does show improvement, especially for keyword based queries. After this small word decompounding was stacked on n-gramming and stemming. While this does not degrade retrieval performance overall, it does not increase it either. As a downside decompounding adds quite some overhead to the processing time (factor 1.5) and (not visible in the tables here) also adds significant processing time while indexing. Based on this and the lack of any significant performance increase, small decompounding was dropped. The last addition that was made is that of referential clustering. This does not really affect the MRR, but

Table 8: *Keywords: Combined performance on Dutch KW sets.*

	RR		*AP*		*Time* $\times 10^{-3}$	
Engine	μ	σ	μ	σ	μ	σ
Baseline	0.51	0.40	0.28	0.20	7.1	5.5
CF	0.47^{∇}	0.39^{Δ}	0.29^{Δ}	0.18^{Δ}	62.5^{∇}	28.6^{∇}
CS	0.55^{Δ}	0.38^{Δ}	0.34^{Δ}	0.19^{Δ}	33.1^{∇}	15.0^{∇}
CSD	0.56^{Δ}	0.38^{Δ}	0.34^{Δ}	0.19^{Δ}	55.2^{∇}	26.6^{∇}
CSR	0.55^{Δ}	0.38^{Δ}	0.40^{Δ}	0.23^{∇}	36.5^{∇}	16.2^{∇}

Table 9: *Questions: Combined performance on Dutch QU sets.*

	RR		*AP*		*Time* $\times 10^{-3}$	
Engine	μ	σ	μ	σ	μ	σ
Baseline	0.60	0.39	0.34	0.20	17.2	7.9
CF	0.55^{∇}	0.38^{Δ}	0.32^{∇}	0.18^{Δ}	162.6^{∇}	78.8^{∇}
CS	0.63^{Δ}	0.37^{Δ}	0.37^{Δ}	0.20^{Δ}	82.8^{∇}	40.5^{∇}
CSD	0.64^{Δ}	0.37^{Δ}	0.37^{Δ}	0.20^{Δ}	126.3^{∇}	64.0^{∇}
CSR	0.62^{Δ}	0.37^{Δ}	0.45^{Δ}	0.23^{∇}	85.3^{∇}	41.3^{∇}

it does increase the MAP. While this is to be expected, since the evaluation calculation uses the same data as is used for referential clustering, it is still an interesting addition. Since referential knowledge is available there is nothing against using it, especially because of the low processing overhead and the fact that it does not negatively affect the MRR.

The final engine consists of character n-gramming, stemming and referential clustering combined. We made some final optimizations to the implementation using acceleration tables and code improvements leading to a higher processing speed especially for question based queries ($\sim 2\times$ faster).

An important question: is the difference between the baseline engine and this new combined engine statistically significant? To answer this question we look only at the professional database and its evaluation results. Alpha values are shown in Table 10. With respect to the MRR we can conclude that the improvement of the CSR engine over the baseline is only significant for German keyword queries, even though the results on the other three sets are close to weak significance. However, the improvement to the MAP is significant. Even without the referential clustering all have $\alpha \leq 0.026$. Increasing the MRR is more difficult due to the fact that, even with different techniques applied, it has a high standard deviation.

Apart from $tf \cdot idf$ we further experimented with improving the performance of the final combined engine using Okapi BM25. For the public dataset using BM25 slightly worsened performance on keyword and question sets. However, on the professional dataset it did not adversely affect keyword performance and actually improved performance on questions: MRR +0.06 and MAP/MRP +0.04 with no change in the deviations. Hence, the decision was made to use the $tf \cdot idf$ based combined engine for querying the public databases and the BM25 variant for the professional ones.

Table 10: *CSR's significance of improvement over baseline.*

	Keyword Sets α		*Question Sets* α	
Language	*MRR*	*MAP*	*MRR*	*MAP*
Dutch	0.119	<0.001	0.251	0.001
German	0.039	0.001	0.112	<0.001

5. CONCLUSION

A working question-answering system for the MRSA domain has been developed based on research into Information Retrieval and Natural Language Processing techniques. Not all of the initially selected techniques work well on the dataset: only half of them show favourable performance increments. Appropriate techniques are highly dependent on both the size and content of the corpus. Curiously the relatively obscure character n-gram technique scored very well. This is presumably due to the fact that it properly captures local word sequence relations. The value chosen for n has been shown to be optimal. While some other techniques showed initial promise as well, combining them led to a decline in performance, which shows that techniques that improve performance individually may conflict with other such techniques.

The evaluation results show the need to base oneself not on only one (favourable) evaluation statistic, but on multiple ones to give a more accurate picture of the performance of the various techniques and the system as a whole. It is also important to look not only at the mean performance, but also at the stability using the *deviation*. The MRR has been shown to be quite hard to increase, which is largely due to the high deviation for this statistic.

Interestingly all the applied techniques and even the baseline yield better retrieval performance for question-based queries then keyword-based queries. This supports the conclusion that in general the retrieval task is easier to perform on question-based queries even without using techniques specifically geared towards these type of queries.

The initial research question was: 'How can the user be provided with the most appropriate answer(s) for his or her query within the restricted MRSA domain given the available structured MRSA corpus?'. This paper answers that question by providing a range of techniques that apply specifically to this corpus. The final selection of combined techniques is a blend between Information Retrieval (IR) (additive $tf \cdot idf$ and BM25), Natural Language Processing (NLP) (stemming, character n-grams) and novel usage of extra information present in the corpus (referential clustering). Several techniques might apply well to other corpora, but only decrease performance on this one. Also, there are approaches that increase the average precision, but decrease the reciprocal rank as side effect. This leads to the conclusion that the techniques to be chosen strongly depend both on the corpus, the query formulation that is used and the statistic for which one wants to optimise.

The evaluation results provide an interesting hint on the techniques that could be tried for IR use on other Dutch and German corpora. Especially character n-gramming shows much promise for broad application and is not commonly used for retrieval purposes. Also, when a corpus exhibits structural information it is highly recommended to investigate how this information can be used to increase retrieval performance.

Reasoning from the vantage point of the user is very important for IR systems that need to be used in practice, but this is unfortunately frequently neglected in favour of scientific performance measurements alone. While this paper has focused only on evaluating the system in a semi-automated fashion, real user investigations have also been carried out, leading to the current user-centered design of the MRSA web-portal [22].

6. FUTURE WORK

There are many other techniques that might be useful and could be tried on the MRSA corpus using the evaluation framework that was built for this research. We believe application of more sophisticated techniques to be a good direction for future research. This includes *explicit relevance feedback, ontology* and *word proximity* approaches, *s-grams*, and use of *syllables* instead of *n*-grams. *Part-of-speech tagging* could be used for example to filter out function words. An investigation into the EuroWordNet coverage of the medical domain and the word relations therein may lead to better results for word relation expansion. A functional enhancement that could be made is setting up the system as a dialog system that helps the user refine the posed queries. This could also be tied to a specific user profile for learning from and adapting the system to the user [6, 9, 11, 19].

Acknowledgements

We wish to thank Djoerd Hiemstra and Maarten Fokkinga for their support. Useful feedback and suggestions on prior incarnations of this paper were also provided by Danny Oude Bos, Desislava Dimitrova, Maarten Eykelhoff, Marco Gerards, Hendri Hondorp, Marco Pasch, Boris van Schooten, Rianne Tigelaar and Ruben Wassink.

This study was financially supported by the EUREGIO MRSA-net Twente/Münsterland project.

7. REFERENCES

[1] Ahlgren, P., and Kekäläinen, J. Swedish full text retrieval: Effectiveness of different combinations of indexing strategies with query terms. *Information Retrieval 9*, 6 (2006), 681–697.

[2] Aslam, J. A., Yilmaz, E., and Pavlu, V. A Geometric Interpretation of R-precision and Its Correlation with Average Precision. In *Proceedings of SIGIR* (Salvador, BR, August 2005).

[3] Braschler, M., and Ripplinger, B. How Effective is Stemming and Decompounding for German Text Retrieval? *Information Retrieval 7* (2004), 291–361.

[4] Buckley, C., and Voorhees, E. M. Evaluating Evaluation Measure Stability. In *Proceedings of SIGIR* (Athens, GR, July 2000).

[5] Burke, R. D., Hammond, K. J., Kulyukin, V., Lytinen, S. L., Tomuro, N., and Schoenberg, S. Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. Tech. Rep. TR-97-05, University of Chicago, 1997.

[6] Chen, A., and Gey, F. C. Multilingual Information Retrieval Using Machine Translation, Relevance Feedback and Decompounding. *Information Retrieval 7* (2004), 149–182.

[7] Hirschman, L., and Gaizauskas, R. Natural language question answering: the view from here. *Natural Language Engineering 7*, 4 (2001), 275–300.

[8] Hollink, V., Kamps, J., Monz, C., and de Rijke, M. Monolingual Document Retrieval for European Languages. *Information Retrieval 7* (2003), 33–52.

[9] Järvelin, A., Järvelin, A., and Järvelin, K. s-grams: Defining generalized n-grams for information retrieval. *Information Processing and Management 43* (2007), 1005–1019.

[10] Loerch, U., and Guesgen, H. Constructing an Intelligent Query Answering System. In *Proceedings of ANNES* (Dunedin, NZ, November 2001).

[11] Manning, C., and Schütze, H. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[12] McNamee, P., and Mayfield, J. Character N-Gram Tokenization for European Language Text Retrieval. *Information Retrieval 7* (2004), 73–97.

[13] Mishne, G., and de Rijke, M. Boosting Web Retrieval through Query Operations. In *Proceedings of ECIR* (Santiago de Compostela, ES, July 2005).

[14] Porter, M. F. An Algorithm for Suffix Stripping. *Program 14*, 3 (1980), 130–337.

[15] Robertson, S. E. Understanding Inverse Document Frequency: On theoretical arguments for IDF. *Journal of Documentation 60*, 5 (2004), 503–520.

[16] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. Okapi at TREC-3. In *Proceedings of TREC* (Gaithersburg, MA, USA, November 1994).

[17] Salton, G., and Buckley, C. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management 24*, 5 (1988), 513–523.

[18] Salton, G., Wong, A., and Yang, C. A Vector-Space Model for Automatic Indexing. *Communications of the ACM 18*, 11 (1975), 613–620.

[19] van Schooten, B., and op den Akker, R. Follow-up utterances in QA dialogue. *Traitement Automatique des Langues 46*, 3 (2007), 181–206.

[20] Spärck-Jones, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation 28*, 1 (1972), 11–21.

[21] Srinivasan, D. P. . G. R. . S. Passage Scoring for Question answering via Bayesian inference on lexical relations. In *Proceedings of TREC* (Gaithersburg, MA, USA, November 2003).

[22] Verhoeven, F., Hendrix, R. M., Daniels-Haardt, I., Friedrich, A. W., Steehouder, M. F., and van Gemert-Pijnen, J. E. The development of a web-based information tool for cross-border prevention and control of Methicillin Resistant Staphylococcus Aureus. *International Journal of Infection Control 4*, 1 (2008).

[23] Vossen, P. Introduction to eurowordnet. *Computers and Humanities 32* (1998), 73–89.