# Querying and Ranking News Items in the Hermes Framework

Arnout Verheij
308057av@student.eur.nl

Allard Kleijn
303118ak@student.eur.nl

Flavius Frasincar
frasincar@ese.eur.nl

Damir Vandic
vandic@ese.eur.nl

Frederik Hogenboom
fhogenboom@ese.eur.nl

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

## ABSTRACT

Hermes is a Web-based framework designed to build personalized news services using Semantic Web technologies. It makes use of ontologies for knowledge representation, natural language processing techniques for semantic text analysis, and semantic query languages for specifying the desired information. This paper presents the Hermes Graphical Query Language (HGQL). HGQL makes it possible to create structured queries in Hermes. Structured queries use disjunctive, conjunctive, negation, and pattern operators. In addition, this paper presents a ranking algorithm based on the queries made using HGQL.

## 1. INTRODUCTION

The Web is a great source of information, which is easy to access. However, the user confronts himself with information overflow. This calls for a way to filter this information such that the results represent the user's wishes. Many techniques are available for information searching of which the most common probably is keyword matching. Basically, this means the user enters words that are important to his query and the system selects information that matches these words in order of relevance. However, this immediately creates a problem: how will a computer system cope with the multiple meanings of the keywords that are used? Let us take the word 'apple' for example: it could be the fruit, the company behind Macintosh computers or it could be a person's last name. At the current moment search engines do not make use of word sense disambiguation techniques to find the correct meaning of polysemous words.

Another often used format for retrieving news is Really Simple Syndication (RSS). In RSS information is grouped in 'feeds'. A user can subscribe to a certain feed, which has a certain subject. In this way information on a certain topic can be retrieved, for example: health or financial.


*SAC'12* March 25-29, 2012, Riva del Garda, Italy.


The strengths of the aforementioned techniques are employed in the Hermes framework [4, 16], used for news personalization, along with Semantic Web technologies to define the meaning of domain concepts. This overcomes the problem with keywords, which have no well-defined semantics. In addition, a graphical query language would allow users, with little understanding of query languages, to create fairly complex structured queries in an intuitive way. Structured queries use AND, OR, and NOT operators, and can also be pattern-based. Pattern-based queries follow the form of a sentence: they have a subject, a predicate, and an object.

The graphical query language in turn calls for a ranking algorithm to provide the user with a list of news items sorted by relevance. In this paper we present the graphical query language and evaluate the corresponding ranking algorithm.

Several weighting schemes for concept importance have been proposed in the literature. Most of the schemes can cope with AND and/or OR operators for queries, but few solutions have been devised to use these operators together with the NOT operator. Here we propose to enhance the popular extended Boolean model [13] with negation.

The contributions of this paper are threefold: it proposes a graphical query language for searching news that goes beyond current keyword-based approaches, it devises a ranking algorithm for sorting news relevant for a user query, and it evaluates several weighing schemes for the proposed algorithm. The structure of the paper is organized as follows. Section 2 describes related work on graphical query languages and relevance sorting algorithms. Section 3 proposes the Hermes Graphical Query Language (HGQL). Section 4 devises a ranking algorithm for HGQL. Section 5 discusses the implementation of HGQL and its ranking algorithm. Section 6 evaluates the ranking algorithm with different weighting schemes. Last, Section 7 presents our conclusions and suggestions for future work.

## 2. RELATED WORK

### 2.1 Graphical Query Languages

Often graphical representations are considered easier to comprehend by humans than textual representations, assuming that the representations are not too complex. Additionally textual representations often require the user to be familiar with a certain syntax, raising the learning threshold. In order to cope with these issues we introduce a graphical

query language which can be used in the Hermes framework [4, 16]. This section provides an overview of the research on the topic of graphical query languages.

In [5] a technique for building graphical queries for RDF is introduced. RDF and OWL are used in Hermes as well. RDF is built up from triples, which are composed of a subject, predicate, and object. The language discussed in [5] uses rounded rectangles with a predicate and optionally an object to visualize queries. Multiple of these rounded rectangles can be interconnected, creating conjunctive queries (logical AND). Object nesting is also supported providing the possibility to link different resources to each other. This means it is possible to create a query, for which the object is defined by another query. A disadvantage of this language is that no functionality is given for disjunctive queries.

RDF-GL [7] is a graphical query language for RDF and it is based on the query language SPARQL. It uses boxes, circles, and arrows in different colors to represent different elements. RDF-GL covers the SELECT statement from SPARQL. However, it is not a very intuitive language as the different lines, shapes and colors can confuse the user.

Another graphical query language is GLOO [3], in which the user can specify concepts, individuals, relations, and logical operators (AND/OR). Ovals, squares, and arrows are the building blocks for GLOO. GLOO is a powerful graphical query language, allowing the user to make relatively complex queries in an intuitive way. However, there is no method discussed for the implementation of filters, such as those in RDF-GL, nor it has support for negation.

## 2.2 Relevance Sorting Algorithms

### 2.2.1 Models

Many models for information retrieval and relevance sorting have been developed throughout the years. The most basic one is the Boolean model, developed by George Boole [2]. The Boolean model uses the structured query operators AND representing the logical product, OR representing the logical sum, and NOT representing the logical difference. This model does not provide any ranking mechanism for text querying.

One of the earliest models for ranked retrieval is the Vector Space model [14]. The Vector Space model is based on measuring the similarity between a query and a document. The simplest of similarity measures is the vector inner product [9] which just calculates the sum of the products of components of the query vector and the document vector. The document vector is $d = (d_1, d_2, ..., d_m)$ of which each component $d_k$ ($1 \leq \mathrm{k} \leq \mathrm{m}$) is associated with an index term; for the query there is a similar vector $q = (q_1, q_2, ..., q_m)$ of which the components are associated with the same terms. Terms are usually words, keywords, or longer phrases. If the words are chosen to be terms, the dimensionality of the vector is the number of words in the vocabulary. The vectors can handle binary values to indicate if a term occurs or not in the document or query, and positive integers to indicate the importance of a term in the document or query. The Vector Space model maps every term to a different dimension. The query and document are considered vectors in the high dimensional Euclidean space determined by the terms. The similarity measure is the cosine of the angle between the query vector and the document vector (if terms are normalized to the unit length the cosine becomes the scalar product). This model needs an additional term weighting model, because it does not describe what the values of the vector components should be. Also, the model is only able to cope with conjunctive queries.

The Vector Space model can be extended to the p-norm extended Boolean model so it is able to support disjunctive queries as well. Let us consider a query of 2 terms and the vectors are normalized to unit length. Point (1,1) represents the situation that both query terms are present with weight 1. Point (0,0) represents the situation that both query terms are not present. Documents in point (1,1) have the highest relevance for conjunctive queries while documents in point (0,0) have the lowest relevance if the query is disjunctive. Therefore, the documents should be ranked in order of increasing distance from the point (1,1) for conjunctive queries and in order of decreasing distance from point (0,0) for disjunctive queries. This reasoning gives the definition of the following scores [14]:

$$\begin{aligned} score\,(d, a \text{ OR } b) &= \sqrt{\frac{(d_a - 0)^2 + (d_b - 0)^2}{2}} \quad . \\ score\,(d, a \text{ AND } b) &= 1 - \sqrt{\frac{(1 - d_a)^2 + (1 - d_b)^2}{2}} \quad . \end{aligned} \tag{1}$$

Generalizing these formulas and extending them with term weights introduces a p-norm which provides a certain softness to Boolean operators [14].

$$\begin{aligned} score\left(d, q \text{ OR}_{(p)}\right) &= \left(\frac{\sum_{k=1}^{m} (q_k)^p (d_k)^p}{\sum_{k=1}^{m} (q_k)^p}\right)^{1/p} \quad . \\ score\left(d, q \text{ AND}_{(p)}\right) &= 1 - \left(\frac{\sum_{k=1}^{m} (q_k)^p (1 - d_k)^p}{\sum_{k=1}^{m} (q_k)^p}\right)^{1/p} \quad . \end{aligned} \tag{2}$$

Salton et al. propose a recursive algorithm based on the two formulas above in order to cope with queries which contain both conjunctive and disjunctive operators [13]. The use of '-1' for query terms with the NOT-operator is mentioned in literature [8] in order to cope with negation queries. Terms which do not appear in the document are represented as '-1' in the document vector. This approach has nevertheless no theoretical nor empirical support so far. The vector space model and the p-norm extended Boolean model both work with all the terms appearing in all documents and in the query. In Hermes concepts are used instead of terms, so with this adjustment these models can be reused for Hermes.

Another solution aiming at incorporating negation in vector spaces is based on linear algebra [18]. A negated concept is represented as its orthogonal complement under the scalar product. Compared to this approach, our research aims to build on the simplicity of the extended Boolean model proposed by Salton [13] by enhancing it with negation.

Fuzzy set theory uses degrees of membership to a certain set. Fuzzy set models in information retrieval rank the documents based on this degree of membership of the document to the terms in the query. Paice's model [11] extends the basic fuzzy models by adding different computations of membership for disjunctive and conjunctive queries, but it does not support negation queries. Like the Vector Space model, this model needs an additional term weighting algorithm.

### 2.2.2 Term Weighting

Most of the discussed models need an additional term weighting algorithm. Many term weighting algorithms have been developed in the past 25 years [6]. We will briefly discuss some of the most successful ones.

The weighting algorithms treat the query like a document. Both the document and the query terms are assigned a weight, and then the similarity of the weights of the query and the document are calculated using one of the models described in the previous section.

Term Frequency (TF) weighting simply counts how often a term occurs in the document and query. The more often a term occurs in a document, the more relevant that term is considered to be to the particular document. Inversed Document Frequency (IDF) weighting is the inverse of the count of the number of documents a term occurs in. A term which occurs in a low number of documents is considered to be specific and therefore documents with this term should get a high weight. TF and IDF weighting combined results in TF.IDF weighting [15]. Most modern weighting algorithms are based on this concept. TF.IDF weighting simply multiplies the weights obtained by TF and IDF weighting and normalizes the outcome using cosine normalization.

In 1988 it was discovered that it is better to map the document and query weights differently [12]. Many variations of TF.IDF have been proposed for this purpose. They are named by 2 three letter combinations, of which the first combination represents the document term weight and the second combination the query term weight. The first letter of such a combination indicates the TF component, the second letter the IDF component, and the third letter indicates the normalization. An example of this is the tfc.nfc algorithm, which uses a normalized TF factor for the query weights. The original TF.IDF algorithm as previously described is called tfc.tfc.

An important discovery is that weights that are logarithmic in TF outperform weighting algorithms which are linear in TF [1]. An example of this is the lxc.ltc formula, where the 'l' stands for weights with a logarithmic TF or IDF component. The normalization part of these formulas is left out here because it follows the usual cosine normalization. The lxc.ltc formula is shown in Eq. 3.

$$\begin{aligned} d_k &= 1 + \log TF_k \\ q_k &= (1 + \log TF_k) * \log \frac{N+1}{df_k} \quad . \end{aligned} \tag{3}$$

A more recent algorithm which is said to be outperforming the cosine normalization is the Lnu.ltu algorithm [17]. This algorithm uses a combination of the document length and the average document length for normalization.

$$\begin{aligned} d_k &= L * u \\ q_k &= l * t * u \\ l &= 1 + \log TF \\ L &= \frac{1 + \log TF}{1 + \log TF_{avg}} \\ t &= \log \frac{N+1}{df} \\ u &= \frac{1}{(1-s) + s \ \frac{uw}{uw_{avg}}} \quad . \end{aligned} \tag{4}$$

$TF_{avg}$ = Average TF of all terms in document
$uw$ = Number of unique words in document
$uw_{avg}$ = Average number of unique words in a document (taken over all documents)
$s$ = Slope factor dependent on the number of unique terms in the document and which is determined experimentally

The best value for the slope s is experimentally determined to be 0.25 when using pivoted unique normalization [10], as in this formula.

# 3. HERMES GRAPHICAL QUERY LANGUAGE

This section discusses the theoretical framework for the Hermes Graphical Query Language (HGQL). Any query in HGQL is a directional graph, with differently colored rounded rectangles (nodes). A set of elements provide the building blocks for a query in HGQL.

Concepts represent classes or individuals from the knowledge base. They can be people, countries, companies, etc. Concepts are labeled with representative lexical representations and their label is purple.

The relations allow the user to employ HGQL's triple-based pattern querying. A triple-based pattern query is constructed with the structure: *concept* → *relation* → *concept*; which returns all news items matching that relation. The possible instances of a relation are extracted from the knowledge base. Relations are represented by verb phrases. Instances of the relation category are denoted as green nodes.

Edges are used to model the triple-based pattern query structure. There are two groups of edges: logical connectivity edge and pattern connectivity edges. Logical connectivity edges always point from the operator to its children and are denoted as gray arrows. Pattern connectivity edges follow the structure: *concept* → *relation* → *concept* and are denoted by black arrows. Figure 1 shows a combined query consisting of a pattern-based query and a concept-only query. The query shown in this figure represents the textual query '(EBAY Buys UNKNOWN) OR (GOOG AND YHOO)'.

Wild cards are partially or entirely unknown concepts or relations to the user. Partially unknown concepts are based on a text match; the concept or relation for which a lexical representation best matches a given text is considered relevant. Entirely unknown concepts can be used if the user is interested in a certain relationship, but wants to leave out the subject or object. For example, the user is interested in every event in which Google buys something. The query "Google buys Unknown" can be constructed. Unknown relations can be constructed as well, allowing the user to retrieve information about all interactions between certain concepts. For example, "United States Unknown France". Wild card colors depend on what they map: concepts or relations (use the samelxc.ltc colour as these).

Operators are used to model logical operations. In the HGQL there are two different types of operators: intra-query and inter-query operators. The first group are operators used within one query, while the second group is used to connect different queries together. Although these are logically the same they provide the user with a better conceptual distinction (they are convenience operators used

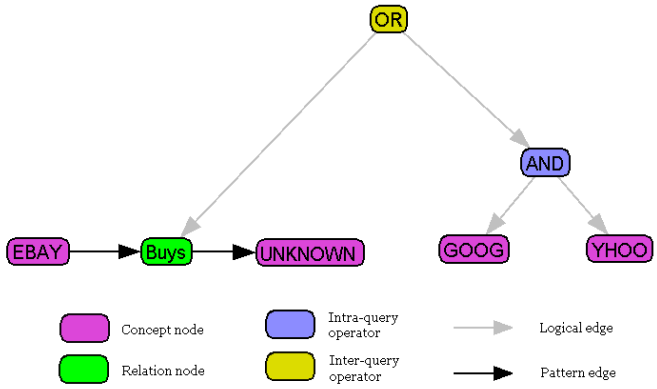


**Figure 1: A combined query consisting of a pattern-based query and a concept-only query with legenda.**

to split the query into subqueries). Inter-query operators point to the root of each query. Inter-query operators are denoted as yellow nodes and intra-query operators as blue nodes.

A set of rules is defined to provide the syntax for HGQL. Firstly, nodes or groups of nodes that 'float' are not allowed. This means that every node in the graph is either directly or indirectly connected to another node in the graph. Secondly, every query should contain at least one node of type concept, with an additional rule for triple-based pattern queries: every pattern query has to have at least one subject, one predicate, and one object. Thirdly, it is not allowed to connect a node to a node of the same type, with an exception for operator nodes. An operator connected to another operator creates a nested query. Finally, a set of connection conditions is specified, which denotes for every node type the allowed number of incoming and outgoing edges.

# 4. HERMES RANKED RESULTS

Hermes is a framework aimed at personalizing news. In order to show the news items desired by a user, Hermes has to rank all news items based on the queries created by this user. Hermes Ranked Results is an extension to the Hermes framework which sorts the news on relevance based on queries created in HGQL, the querying language of Hermes.

It is impossible to define different formulas for each possible query, so we need to normalize the query form. An example of a normal form is the negation normal form (NNF). A query is in negation normal form if negation is only applied to single concepts or patterns, and if only disjunctive, conjunctive, and negation operators are used. Two possible extensions of NNF are the disjunctive normal form (DNF) and the conjunctive normal form (CNF). A query is in DNF if this query is a disjunction of clauses, where a clause is a conjunction of possibly negated concepts or patterns. All rules for NNF also apply to DNF. A query is in CNF if it is a conjunction of disjunctive clauses, where a clause consists of concepts or patterns which are possibly negated. Basically it is the same as DNF, except the disjunction and conjunction operators are swapped. Each query can be converted to NNF, DNF, or CNF.

For Hermes, DNF is used as normal form, interpreting a query as a disjunction of its conjunctive subqueries. The queries made in HGQL are converted to DNF using double negation elimination, De Morgan's laws, and the distributive law. In order to be able to also rank the news for queries which, besides concepts, also contain patterns, patterns are treated like concepts. To reach this goal, we first need to convert pattern queries to a form which is compatible with the ranking algorithm. In HGQL, one can create a pattern based query with logical operators within a single pattern. The ranking algorithm can only process queries which consist of simple patterns with operators between them. Such a simple pattern has only one subject, one predicate, and one object, and represent a so-called complex concept. A complex pattern, a pattern that uses logical operators in a pattern, is reduced to simple patterns (complex concepts) connected by logical operators by outer moving the logical operators.

Chained queries are not explicitly part of Hermes HGQL. This type of query is simulated by two complete patterns connected by a conjunctive operator, where the first pattern's object is the same as the second pattern's subject.

## 4.1 Ranking Algorithm

Now that we discussed the conversion of any possible query created in HGQL to DNF, we discuss the actual ranking algorithm which uses these queries to sort the news based on their relevance to the user.

### *4.1.1 Document and Query Representation*

To be able to use a relevance sorting algorithm we need to represent query and documents as weight vectors. We discussed some term weighting algorithms for this purpose in Section 2.2.2. In this study we calculate term weights with four different algorithms to check which one returns the best results.

The first method we use is a simple binary weight. We call this method the extended Boolean method, because it uses Boolean weights. The document weight is 0 if the concept does not occur and 1 if it does occur at least one time in the document. The same goes for the query weights. Please note that this method is an extension of the original extended Boolean model by Salton et al. [13], which does not support negation operators.

As second method, the basic TF.IDF weighting algorithm with cosine normalization is used. This method is called

tfc.tfc with the two three letter combinations naming system. The third method used is lxc.ltc. This is a TF.IDF variation with logarithmic TF and IDF weights, as depicted in Eq. 3. The purpose of using the second and third method is to check if the third method (lxc.ltc) also outperforms the second method (tfc.tfc) in a concept space. The fourth and last method used is the Lnu.ltu algorithm as described in Eq. 4 which is said to be outperforming the previous algorithms using cosine normalization. This algorithm uses a combination of the document length and the average document length for normalization. As slope we use a value of 0.25, since it is proven that this value provides the best results.

### 4.1.2 Disjunctive & Conjunctive Queries

Hermes uses a combination of the two formulas of the p-norm extended Boolean model with p = 2. One of these formulas addresses conjunctive queries, whereas the other copes with disjunctive queries. The input queries for this model are in DNF. This means that there are basically two parts, a disjunctive query and a collection of conjunctive subqueries. At first, all the weights of the subqueries are calculated by the formula which addresses conjunctive queries. After this, the formula which addresses disjunctive queries calculates the total relevance of the document using the obtained weights of the subqueries. The query weights in the formula which calculates the total score of the document are 1 because all subqueries are present one time. The results are shown in Eq. 5.

$$\begin{aligned} score(w_i \text{ OR}_{(p)}) &= \left(\frac{\sum_{k=1}^{n}(w_i)^p}{n}\right)^{1/p} . \\ w_i\left(d, q \text{ AND}_{(p)}\right) &= 1 - \left(\frac{\sum_{k=1}^{m}(q_k)^p(1-d_k)^p}{\sum_{k=1}^{m}(q_k)^p}\right)^{1/p} . \end{aligned} \tag{5}$$

d = Document's index representation vector
q = Query's index representation vector
m = Total number of terms in documents and query
n = Total number of conjunctive clauses
$w_i$ = Weight of clause $i$

### 4.1.3 Negation

The previously stated relevance scoring does not take in consideration the negation operator. Therefore we make an adjustment to the query and document weights to cope with this operator. The query weights for concepts which are not part of the query are 0, and the query weights for concepts which occur in the query and are not negated are calculated with the four different methods described above. The weights of query concepts which are negated are computed as formerly mentioned, however now they are multiplied with -1. Document weights remain the same for concepts which occur in the document, while concepts which do not occur in the document are given weight -1 instead of 0.

With the original vector inner product, this approach is correct with respect to the intended query semantics. The query and document weights are simply pairwise multiplied in this approach. Therefore, documents get a negative relevance when the query concept is negated while the concept does occur in the document, or when the query concept is not negated but the concept does not occur in the document. The relevance will be positive when a non-negated concept of the query occurs in the document, or a negated query concept does not occur in the document. Concepts which do not occur in the query will not affect the relevance score as the assigned 0 weight to these concepts will be multiplied with the document weight, resulting in a relevance score of 0.

The ranking model described above was made for a Boolean weighting model using only weights between 0 and 1, where 0 represented absence and any value higher than 0 presence of a term. As we changed the term weighting, we need to change the ranking algorithm itself correspondingly. In some cases, the original ranking algorithm gives the same results for negated queries as for non-negated queries. If the document weight is 1, the query weight is multiplied with 0 according to Eq. 5. This results in 0, independent from the value of the query weight. The equation also provides incorrect relevance scores for document weights of -1. In this case, the query weight will be multiplied with a value higher than 1, which may result in values below 0. Eq. 5 is only normalizing to query weights. Therefore, the outcome of the formula could be below 0, which is not within the desired positive weight range.

In order to be able to calculate the relevance score of news items based on structured queries consisting of disjunctive, conjunctive, and negation operators, we need to adjust the equations. We need to change Eq. 1, where the OR part is based on the distance to the worst case (0,0). With negation we have a new worst case, namely $(-q_a, -q_b)$ for the specific case of two query terms a and b. In general, the worst case is $-q_k$. The AND part of this formula is based on best case (1,1). The new best case for negation queries in case of two query terms a and b is $(q_a, q_b)$, so in general, the best case is $q_k$. The old formulas adjusted with this knowledge to make them compatible with negation queries result in Eq. 6. Because the only case when $d_a > q_a$ is when $d_a = 1$ and $q_a = -1$, the scaling factor is $(2 * q_a)^2$. As the scaling factor is squared, this value is always bigger than $(d_a + q_a)^2$.

$$\begin{aligned} score\left(d, Q_{or}\right) &= \sqrt{\frac{(d_a+q_a)^2+(d_b+q_b)^2}{(2*q_a)^2+(2*q_b)^2}} . \\ score\left(d, Q_{and}\right) &= 1 - \sqrt{\frac{(q_a-d_a)^2+(q_b-d_b)^2}{(2*q_a)^2+(2*q_b)^2}} . \end{aligned} \tag{6}$$

$Q_{or}$ is either $a \vee b$, $a \vee \neg b$, $\neg a \vee b$, or $\neg a \vee \neg b$, and $Q_{and}$ is either $a \wedge b$, $a \wedge \neg b$, $\neg a \wedge b$, or $\neg a \wedge \neg b$. From these formulas we can deduce the generalization as in Eq. 2 for negation queries. This results in Eq. 7.

$$\begin{aligned} score\left(d, q \text{ OR}_{(p)}\right) &= \left(\frac{\sum_{k=1}^{m}(q_k)^p(d_k+q_k)^p}{\sum_{k=1}^{m}(2*q_k)^p}\right)^{1/p} . \\ score\left(d, q \text{ AND}_{(p)}\right) &= 1 - \left(\frac{\sum_{k=1}^{m}(q_k)^p(q_k-d_k)^p}{\sum_{k=1}^{m}(2*q_k)^p}\right)^{1/p} . \end{aligned} \tag{7}$$

Based on these formulas, we can instantiate our formula for combined disjunctive and conjunctive queries as in Eq. 5 compatible with negation queries. Because the range of the weights from the AND part of the formula is now from 0

to 1, we do not have to change the OR part of the formula which calculates the disjunction of the subquery weights. The obtained model is shown in Eq. 8.

$$\begin{aligned} score(w_i \text{ OR}_{(p)}) &= \left(\frac{\sum_{k=1}^{n}(w_i)^p}{n}\right)^{1/p} . \\ w_i\left(d, q \text{ AND}_{(p)}\right) &= 1 - \left(\frac{\sum_{k=1}^{m}(q_k)^p(q_k - d_k)^p}{\sum_{k=1}^{m}(2 * q_k)^p}\right)^{1/p} . \end{aligned} \tag{8}$$

$d$ = Document's index representation vector
$q$ = Query's index representation vector
$m$ = Total number of terms in documents and query
$n$ = Total number of conjunctive clauses
$w_i$ = Weight of clause $i$

# 5. IMPLEMENTATION

In this section we discuss the implementation of the HGQL and the ranking algorithm based on this query language as described in this paper. These are implemented in such a way that the user can use our implementation to access all features of the Hermes framework. The implementation is called the Hermes News Portal (HNP), and can be integrated in the standard implementation of Hermes. In HNP the user searches for news by selecting the concepts (s)he is interested in using the Original Graph tab. The user can choose either single concepts or concepts which are related to a certain concept. These concepts can be stored in the Search Graph from the Original Graph. When the user selected all the concepts of his interest, he can query the news based on these using the Search Graph tab. He can also select time constraints for the resulting news items, and start the querying procedure. The HNP automatically switches to the Results tab, presenting a ranking of news items along with a relevance score between 0 and 100 percent.

In the HGQL tab, the user is able to build queries using the Boolean operators AND, OR, and NOT. HGQL also offers the ability to create pattern queries, consisting of a subject, a predicate, and an object. Once the query is finished, the user can search the news by means of it by clicking the 'Execute Query' button. The system switches to the Results tab, showing only the news items which exactly match the query. If the user tries to retrieve news with an invalid query, an error message will pop up, generated by the query validator.

The ranked list of news items based on the relevance algorithm of HGQL can be obtained by checking the 'Ranked' box in the HGQL tab. The user also has to select one of the radio buttons in order to choose a weighting algorithm. After the 'Execute query' button is pressed, the system will rank all news items in the database based on the given query. The HNP automatically switches to the Results tab showing the ranked list of news items, as shown in Figure 2.

HNP uses OWL as ontology language and is written in Java. OWL is supported by W3C and offers some useful additional features we need, e.g., the ability to describe disjoint classes, cardinality, and symmetry. The classified news items, the knowledge base, the update rules, the user profile, and queries in HGQL are stored in OWL ontologies. SPARQL is used to query these ontologies because it is supported by W3C as the query language for RDF languages including OWL. To represent the ontologies in a visual graph Prefuse is employed. Java has many libraries to manipulate, reason with, query and visualize ontologies, like GATE for the Natural Language Processing steps, ARQ to execute SPARQL, SPARQL/update to update ontologies, and OWL2Prefuse for the visualization of the knowledge base. The conversion of queries to DNF is done by the Orbital library.

# 6. EVALUATION

In this section we evaluate the proposed ranking algorithms using the implementation in Hermes. Two different measures are used in order to evaluate the ranking algorithm. The first measure is the precision for the first ten documents in our results list. For $n$ different queries we calculate how much of the news items in the top ten of the ranked list are considered relevant. This is referred to as the 'Mean Precision @ 10' (MP@10).

The second measure that is used to evaluate the ranking algorithm is the Mean Average Precision. The MAP provides a single-figure measure of quality across recall levels. The Average Precision (AP) is the average of the precision values obtained for the set of top $k$ documents in the search results before each relevant document is retrieved for a certain query. We take the mean of this value from $n$ different queries.

We evaluate the ranking algorithm based on a list retrieved for the same queries from various test users. We consulted 5 test users who have a Computer Science background and who are familiar with Semantic Web technologies and Information Retrieval techniques. The list obtained from these users is regarded as the ultimate goal to compare the ranking algorithm with. The users were told to select all news items which they considered relevant to the queries. For this obtained 'golden standard' we used only the results where the majority of annotators agreed. The test set used is a database of 927 news items about various subjects. The methods used for document and query weight calculation are the extended Boolean Model (Rank ($e\mathbb{B}$)), the traditional TF.IDF weights (Rank (tfc.tfc)), the TF.IDF model with logarithmic weights (Rank (lxc.ltc)) as given in Eq. 3, and the TF.IDF model with a combination of the document length and the average document length for length normalization (Rank (Lnu.ltu)), as given in Eq. 4. The calculations for the different possibilities of concept presence in documents and queries are shown in Table 1.

The results of the comparison to the user list are displayed in Table 2. We can note from this table that the extended Boolean Model performs best on both measures. Rank(lxc.ltc) performs a bit less good than the extended Boolean Model. Rank(tfc.tfc) and Rank(Lnu.ltu) have the lowest scores, with the latter performing slightly better than the first.

Figure 3 shows the 11-point precision/recall graph averaged over the 10 given queries. As can be noted from this graph, the extended Boolean model performs best across all recall levels. From the term weighting methods, Rank(lxc.ltc) performs best for most recall levels, while Rank(Lnu.ltu) performs best on high recall levels. Rank(tfc.tfc) performs poor on all recall levels as expected.

We have used the different TF.IDF weighting schemes for our ranking algorithm. The previous results [12] showing that it is better to map the document and query vectors

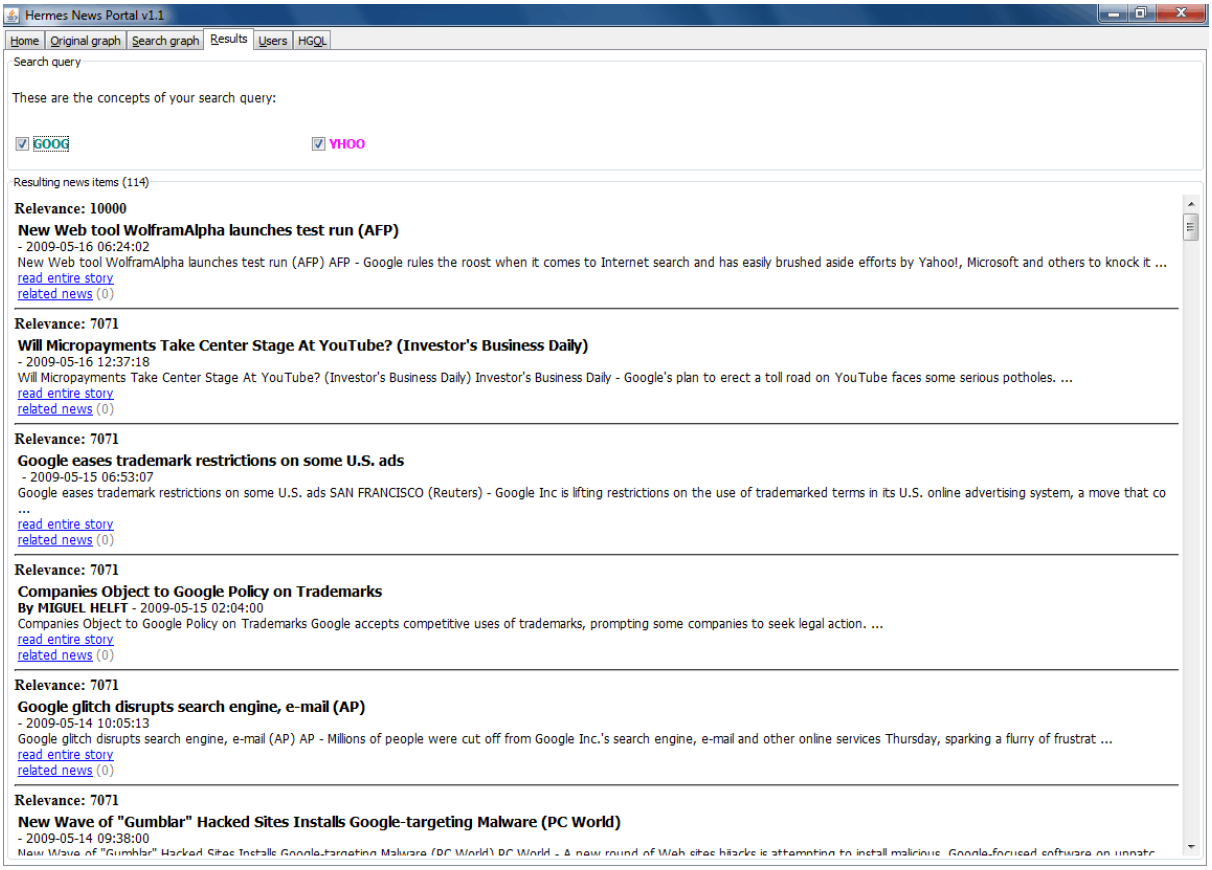


**Figure 2: Ranked query results in HNP.**

differently in the vector space have been confirmed here, because both Rank(lxc.ltc) and Rank(Lnu.ltu) perform better than the basic TF.IDF model Rank(tfc.tfc) where the query and document vectors are mapped equally. However, the previous results about algorithms which use a combination of the document length and the average document length for normalization performing better than algorithms which use cosine normalization [17] could not be confirmed here, as can be noted from the fact that Rank(Lnu.ltu) does not perform better than Rank(lxc.ltc).

## 7. CONCLUSION

The Hermes Graphical Query Language (HGQL) is a query language developed in the context of news personalization by Hermes. Hermes is a Web-based news personalization service using Semantic Web technologies. HGQL makes it possible to build structured queries without any knowledge of SPARQL using building blocks: concepts, relations, operators, and wildcards as nodes, and edges to connect the different nodes. Structured queries are queries which consist of only concepts and logical operators, triple-based patterns, or a combination of these two. HGQL can either return a list of news items which completely match the query (the Boolean Model), or a ranked list of news items based on their relevance to the query.

The ranking algorithm of HGQL is based on a combination of the different formulas for disjunctive and conjunctive queries of the p-norm extended model. Negation operators are addressed by using negative weights for query and document terms. Four different weighting algorithms have been tested in this study, namely the extended Boolean model Rank($e\mathbb{B}$), which uses document weights of 1, 0 and -1, and three different TF.IDF weighting algorithms, Rank(tfc.tfc), Rank(lxc.ltc), and Rank(Lnu.ltu). The main contribution of this paper is the ability to address negation operators

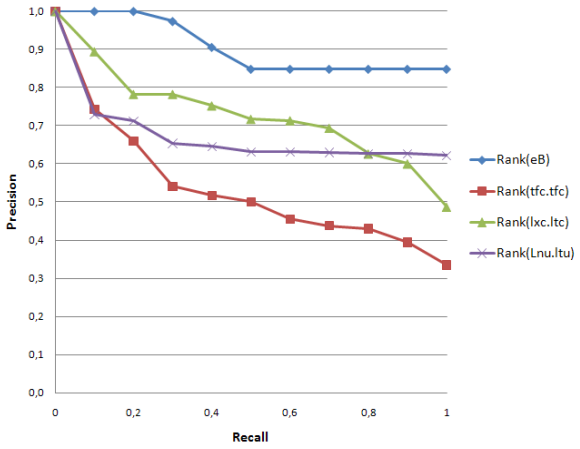


**Figure 3: The 11-point precision/recall graph with the user list as benchmark.**

**Table 1: Document weight calculation.**

| | Document Weight | | Query Weight | | |
|---|---|---|---|---|---|
| | Concept present | Concept absent | Concept present | Concept absent | Concept negated |
| Rank($e\mathbb{B}$) | 1 | -1 | 1 | 0 | -1 |
| Rank(tfc.tfc) | tfc.tfc | -1 | tfc.tfc | 0 | -1 * tfc.tfc |
| Rank(lxc.ltc) | lxc.ltc | -1 | lxc.ltc | 0 | -1 * lxc.ltc |
| Rank(Lnu.ltu) | Lnu.ltu | -1 | Lnu.ltu | 0 | -1 * Lnu.ltu |

**Table 2: Evaluation results using user list benchmark.**

| Measure | Rank($e\mathbb{B}$) | Rank(tfc.tfc) | Rank(lxc.ltc) | Rank(Lnu.ltu) |
|---|---|---|---|---|
| MP@10 | 0.850 | 0.470 | 0.730 | 0.480 |
| MAP | 0.874 | 0.467 | 0.694 | 0.572 |

within a query which consists of conjunctive and disjunctive operators using a ranking algorithm.

Our study showed that the extended Boolean model is performing best with a Mean Precision at 10 (MP@10) of 0.85 and a Mean Average Precision (MAP) of 0.874. The lxc.ltc algorithm provides the second best results with an MP@10 of 0.73 and a MAP of 0.694. The Lnu.ltu algorithm and the tfc.tfc algorithm perform relatively poor with an MP@10 of respectively 0.48 and 0.47 and a MAP of 0.572 and 0.467, respectively.

In the future we would like to extend the ranking algorithm by employing user-defined weights for query concepts and evaluate its performance with respect to the user-specified interests. Another research direction that we would like to pursue is perform a user-based evaluation with respect to the ease of use of the query language.

## Acknowledgment

The authors are partially sponsored by the NWO Physical Sciences Free Competition project 612.001.009: Financial Events Recognition in News for Algorithmic Trading (FERNAT) and the FES COMMIT Infiniti project Information Retrieval for Information Services.

## 8. REFERENCES

[1] C. Buckley, J. Allan, and G. Salton. Automatic Routing and Retrieval Using Smart: TREC-2. *Information Processing & Management*, 31(3):315–326, 1995.

[2] G. G. Chowdhury. *Introduction to Modern Information Retrieval*. John Wiley and Sons, 1998.

[3] A. Fadhil and V. Haarslev. GLOO: A graphical query language for OWL ontologies. In *International Workshop on OWL: Experiences and Directions (OWLED 2006)*, volume 216. CEUR-WS.org, 2006.

[4] F. Frasincar, J. Borsje, and L. Levering. A Semantic Web-Based Approach for Building Personalized News Services. *International Journal of E-Business Research*, 5(3):35–53, 2009.

[5] A. Harth, S. R. Kruk, and S. Decker. Graphical representation of RDF queries. In *15th international conference on World Wide Web (WWW 2006)*, pages 859–860. ACM, 2006.

[6] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Technology, 2001.

[7] F. Hogenboom, V. Milea, F. Frasincar, and U. Kaymak. RDF-GL: A SPARQL-based graphical query language for RDF. In *Semantic Web Information Management: A Model-Based Perspective*, pages 225–246. Springer, 2010.

[8] J. Hynek. Document Classification in a Digital Library. Technical report, University of West Bohemia in Pilsen, 2002.

[9] H. P. Luhn. A statistical approach to the mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[10] F. Oroumchian, A. Aleahmad, P. Hakimian, and F. Mahdikhani. N-gram and local context analysis for Persian text retrieval. In *9th International Symposium on Signal Processing and its Applications (ISSPA 2007)*, pages 1–4. IEEE Computer Society, 2007.

[11] C. P. Paice. Soft evaluation of boolean search queries in information retrieval systems. *Information Technology: Research and Development*, 1(3):33–42, 1984.

[12] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24(5):513–523, 1988.

[13] G. Salton, E. A. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[14] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[15] G. Salton and C. S. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 4(29):351–372, 1973.

[16] K. Schouten, P. Ruijgrok, J. Borsje, F. Frasincar, L. Levering, and F. Hogenboom. A Semantic Web-Based Approach for Personalizing News. In *The 25th Symposium on Applied Computing (SAC 2010)*, pages 854–861. ACM, 2010.

[17] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. *Information Processing and Management*, 32(5):619–633, 1996.

[18] D. Widdows. 41st annual meeting on association for computational linguistics (ACL 2003). In *Orthogonal Negation in Vector Spaces for Modelling Word-Meanining and Document Retrieval*, pages 136–143. ACL, 2003.