A Query Algebra for Quantum Information Retrieval

Annalina Caputo[1], Benjamin Piwowarski[2], Mounia Lalmas[3]

[1] University of Bari, Italy. acaputo@di.uniba.it
[2] University of Glasgow, UK. benjamin@bpiwowar.net
[3] Yahoo! Research, Spain. mounia@acm.org

Abstract The formalism of quantum physics is said to provide a sound basis for building a principled information retrieval framework. Such a framework is based on the notion of information need vector spaces, where events, such as document relevance, correspond to subspaces, and user information needs are represented as weighted sets of vectors. In this paper, we look at possible ways to build, using an algebra defined over weighted sets, sophisticated query representations and report on corresponding experiments on TREC.

1 Introduction

Quantum physics offers a formalism based on the mathematics of Hilbert spaces, to describe the behaviour of matter at (sub-)atomic scales. As van Rijsbergen claims [1], the "language" of quantum physics can be used to express the various geometrical, probabilistic and logical IR models within a unified framework. Driven by this motivation, research in quantum-inspired IR is investigating how the mathematical framework of quantum physics can address diverse challenges in IR. These efforts led to the recent proposal of a quantum IR (QIR) framework [2].

In this framework, the system view on the user information need (IN) is a set of weighted vectors, where each vector represents a possible atomic IN, i.e., an IN that cannot be further refined. Computing a query representation from the terms forming the query is more complex in the QIR framework, than in standard vector space models. In the latter, the query is represented as the sum of the vectors representing each query term; in the former, each term corresponds to a weighted set of vectors (the possible atomic INs corresponding to this term), and thus it is not straightforward to extend this summation technique. In addition, many different operators (beside an extension of the sum) can be used [2].

In the QIR framework, a query is processed in two steps. First, the different *aspects* of an IN are identified, where each aspect address a different requirement on the topic of a document (see Section 2.2). The aspects representations are computed using two possible operators, namely, the extension of the sum, which corresponds to a mixture of superpositions, and another operator, mixture. In [3], the best strategy was equalling each term to a different aspect. In [2], the two

operators led to an increase in performance depending on how some or all the terms forming a query were related to each other; more precisely, whether the terms together formed a "concept" (e.g. "social network") or whether they were (more) independent (e.g. "blue car").

Based on the findings in [2], and that it is possible to freely combine operators when representing a query (hence, to define an algebra with these operators), this paper explores how complex query representations can be built using an algebra based on quantum operators. In order to do so, we exploit query segmentation and computational linguistics techniques to identify relationships between terms in a query, and extract phrases. We propose a number of strategies, making use of the terms and phrases in the query, to build a structured query representation.

Structured queries have been used in the past in IR and when properly expressed, retrieval performance is often excellent [4]. The fact that structured queries, in general, are not widely used (e.g. [5]) stem from users difficulties to express them correctly, although this can be somehow tackled through (pseudo) relevance feedback. In this paper, we study operators from quantum physics as a means to express structured queries in the QIR framework.

The outline of this paper is as follows. We first introduce the QIR framework (Section 2). Section 3 contains the two paper contributions: We first define the query algebra (Section 3.1) and then define how to transform a query into the algebra (Section 3.2). These constructions are evaluated in an ad hoc retrieval scenario (Section 4). Finally, we conclude.

2 The quantum IR framework

2.1 Information Need Spaces

The basic assumption of the QIR framework [2] is that there exists a Hilbert space $\mathcal{H}$ of *information needs* (IN), called *information need space,* where each vector corresponds to what we call an *atomic IN*. The concept of an "atomic" IN[4] is central to our framework. It can be compared to the notion of "nugget" [6], used in summarisation and question-answering to assess the amount of relevant information a summary or an answer contains.

An *atomic* user IN is a unit vector φ (aka a *state*) in $\mathcal{H}$. States determine *statistically* any measure made on the user's IN, as for example the relevance of a document. Given a state space $\mathcal{H}$ and a state vector φ, a *probabilistic event* is represented as a subspace S of $\mathcal{H}$. A state vector φ induces a probability distribution on events (i.e., the subspaces). The probability of an event is given by the square of the length of the projection of φ onto the corresponding event subspace S, that is by computing the value $\left\|\widehat{S}\varphi\right\|^2$ where $\widehat{S}$ is the projector onto the subspace S.

In the QIR framework, we suppose that we have a system view on the user IN. We assume that the system usually does not know what is the *atomic* IN

[4] In this paper, we use "atomic" information need ("atomic IN") to distinguish it from information need ("IN") in its more usual sense in IR.

state of the user, but knows with a probability $\mathcal{V}(\varphi)$ that the user is in the state φ. It is convenient to view $\mathcal{V}$ as a function that associates a weight $\mathcal{V}(\varphi) \in [0,1]$ to each of its state vectors φ. We say that $\varphi \in \mathcal{V}$ if its weight is greater than zero – hence $\mathcal{V}$ can be thought of as a weighted set of INs. For notations, we use standard operations on function spaces: Adding two weighted sets $\mathcal{V}_1 + \mathcal{V}_2$, or scaling by a factor $\alpha\mathcal{V}$. We denote $\varphi \mapsto w$ the weighted set that associates w to φ and 0 to the other vectors.

As states are mutually exclusive (a user IN cannot be in two different states at the same time), we require that a weighted set has weights that sum up to 1. Given the weighted set $\mathcal{V}$ of possible state vectors, we then define the probability of an event S as

$$\Pr(S|\mathcal{V}) = \sum_{\varphi \in \mathcal{V}} \mathcal{V}(\varphi) \Pr(S|\varphi) = \sum_{\varphi \in \mathcal{V}} \mathcal{V}(\varphi) \left\| \widehat{S}\varphi \right\|^2 \tag{1}$$

Note that equation 1 reduces to $\left\| \widehat{S}\varphi \right\|^2$ if $\mathcal{V}$ consists of only one vector φ, i.e. when there is no uncertainty about the user's IN state. This probability can be computed efficiently when using low rank approximations (see [2] for more details).

2.2 Aspect Spaces

As discussed in [2], user INs may consist of several "aspects" that relevant documents should address. For example, in the TREC-8 topic 408, "What tropical storms (hurricanes and typhoons) have caused significant property damage and loss of life?", we can identify two (topical) IN aspects, namely tropical storms and significant damage/loss of life. Each IN aspect can be defined within an IN aspect space, where the state vectors are now called *atomic IN aspects*. Examples of atomic IN aspects are the vectors representing "hurricane" and "typhoon" for the first IN aspect (tropical storms). We use the terminology *atomic IN aspect,* since one atomic IN aspect addresses one aspect of the IN (tropical storms) in the same way that an atomic IN addresses an IN. In this paper, following [2] we only consider one aspect space, the *topical space* $\mathcal{T}$*,* and equal it to the standard term space where each term corresponds to a dimension.

In this topical space, we can represent the event S_d that a document is relevant to a given weighted set of topical IN aspects as a subspace. We give here a quick overview of the construction process, which is described in more details in [2]. The document representation is based on the assumption that a typical document answers various (atomic) IN aspects. It also assumes that each document can be split into (possibly overlapping and non-contiguous) fragments, where each fragment addresses one atomic IN aspect. This follows from research in focused retrieval, which states that answers to a query, and hence aspects of it, usually correspond to document fragments (sentences or paragraphs) and not full documents. In this work, following [2], a document is decomposed into a set of overlapping fragments of a fixed length of 10 terms.

Each fragment can be represented in the topical space $\mathcal{T}$ by a vector corresponding to the atomic IN this fragment is relevant to, which is approximated by a binary vector where components corresponding to terms in the fragment are set to 1. The subspace S_d is then defined as the subspace spanned by the set of vectors in $\mathcal{V}_d$, i.e. S_d is the smallest subspace such that a document is always a fully relevant answer to a pure IN aspect it contains, or more formally such that $\Pr(S_d|\varphi) = 1$ for any $\varphi \in \mathcal{V}_d$. The document will be relevant to an atomic IN with a probability that depends on the length of the projection of the atomic IN vector onto the subspace S_d.

2.3 Information Need and Aspect Spaces

The IN space is then defined as a tensor product of aspect spaces, i.e. of topical spaces $\mathcal{T}$. Tensor products are a generalisation of the product of probabilistic spaces for Hilbert spaces. We consider the simple case where the events in each space are independent from each other, i.e. where aspects are independent.

If we identify n aspects in a query, each represented by the weighted set of atomic aspects $\mathcal{V}_i$, then the IN space $\mathcal{H}$ is a tensor product of topical spaces $\bigotimes_{i=1}^{n} \mathcal{T}$. In this space $\mathcal{H}$ a document is represented by $S_d^{\otimes} = \bigotimes_{i=1}^{n} S_d$, where S_d is the subspace that represents the document in the topical space $\mathcal{T}$ (see previous section). The actual IN is a tensor product of weighted sets $\mathcal{V} = \bigotimes_{i=1}^{n} \mathcal{V}_i$. We describe several constructions of $\mathcal{V}$ based on a query q in Section 3.2 using the algebra defined in Section 3.1. Given a document representation $S_d^{\otimes}$ and a weighted set $\mathcal{V}$, we then compute the probability of relevance for the document d as:

$$\Pr\left(\bigotimes S_i \middle| \bigotimes \mathcal{V}_i\right) = \prod_i \Pr(S_i|\mathcal{V}_i) \tag{2}$$

Users might give different importance to certain aspects. This motivates the introduction of a weighting scheme for aspects that would enable us to express the fact that some aspects are more important than others. In our Hilbert space of aspects, we chose to introduce a new "*don't care*" dimension, defined by the state vector $\varphi_\top$. We then modify each document subspace S_d so that the subspace contains the "don't care" dimension. This implies that any document is relevant to the "don't care" aspect $\varphi_\top$, since $\Pr(S_d|\varphi_\top) = \left\|\widehat{S}_d\varphi_\top\right\|^2 = 1$; with respect to Equation 2, the probability of a document to be relevant to $\mathcal{V}_i$ is now $\Pr(S_i|\mathcal{V}_i) = \mathcal{V}_i(\varphi_\top) + (1 - \mathcal{V}_i(\varphi_\top))\Pr(S_i|\mathcal{V}_i \setminus \{\varphi_\top\})$, and hence $\mathcal{V}_i(\varphi_\top)$ defines how important the i^{th} IN aspect is within the product in Equation 2.

3 Representing queries

We formally define an algebra over weighted sets of atomic IN aspects (Section3.1). A query (sequence of terms) q is then transformed into a a structured representation $\mathcal{V}_q$ expressed in this algebra (Section 3.2). The algebra provides the mean

to express relationships between query terms, thus allowing for more complex query representations.

3.1 An algebra of weighted sets of atomic INs

We now define our algebra over weighted sets, i.e., its constants and operators. Constants are used to represent the set of atomic INs corresponding to a term or a phrase. This constant can be computed for each term/phrase at indexing time from the corpus (Section 3.1.1). We use two operators, each corresponding to one way to combine atomic INs: **mixture** (Section 3.1.2), where the possible atomic aspects are drawn from the weighted sets; and **mixture of superpositions** (Section 3.1.3), where the possible atomic aspects are (linear) combinations of the atomic IN aspects drawn from each weighted set. Other operators could be defined (e.g. projection), but we omit them due to the lack of space.

3.1.1 Constants The constants of the algebra correspond to weighted set of aspects associated with phrases. Let $p = t_1 \dots t_m$ be a phrase made of m terms, where $m \geq 1$. We assume that the documents fragments that possibly answer a query/phrase p are all the fragments containing this exact phrase. From Section 2.2 a fragment is mapped to an atomic aspect vector. Thus we can define over all the documents in the collection (1) the number of fragments N_p associated with the phrase p, and (2) the number of occurrences $n_p(\varphi)$ of the atomic IN aspect φ in the fragments containing the phrase p.

Each of the N_p fragments are associated with a vector, and as we have *a priori* no way to distinguish between these different vectors, we assume that each is equally likely to be a atomic IN aspect of a user that typed the phrase p as a query. Hence, we define a first weighted set $\mathcal{V}^\bullet(p)$ for a phrase p as $\mathcal{V}^\bullet(p) = \sum \varphi \mapsto n_p(\varphi)/N_p$. In practice, this representation means that the more vectors φ from $\mathcal{V}^\bullet(p)$ lie in the document subspace, the higher the relevance of the document to p.

A query can be composed with more than one phrase; thus to give more or less importance to a given phrase while composing the weighted sets, we use the fake state (Section 2.2) with a given weight. This weight will be useful to determine the contribution of a weighted set when combining it with other weighted sets in a structured query. Thus, given an aspect $\mathcal{V}_p$, we include the state $\varphi_\top$ as one of its possible atomic IN aspect states with a probability $f(p)$. We thus modify $\mathcal{V}^\bullet(p)$ as follows:

$$\mathcal{V}(p) = (1 - f(p))\,\mathcal{V}^\bullet(p) + (\varphi_\top \mapsto f(p)) \tag{3}$$

The second part of the sum assigns the weight $f(p)$ to the don't care atomic IN aspect, whereas the first part scales the previous weights so that the sum of all weights is 1 again. Equations 3 and 1 imply that the probability of a document to be relevant for the aspect defined by the phrase p is $\Pr(S_d|\mathcal{V}(p)) = f(p) + (1 - f(p))\Pr(S_d|\mathcal{V}^\bullet(p))$.

3.1.2 Mixture Mixture (by analogy with a probabilistic mixture) is the simplest operator combining n weighted sets of IN aspects $\mathcal{V}_1 \dots \mathcal{V}_n$. The underlying assumption in our context is that the atomic IN aspect can be an aspect coming from any of the $\mathcal{V}_i$. Unless there is a special reason to believe that the user atomic aspect is more likely to be from one than the other weighted sets, we can assume that each weighted set have an equal probability of being the one (that is, a probability of $\frac{1}{n}$). We define formally the *mixture* operator $\circledcirc_{i=1}^{n}\mathcal{V}_i$ of all the atomic IN aspect vectors associated with the weighted sets $\mathcal{V}_1 \dots \mathcal{V}_n$:

$$\circledcirc_{i=1}^{n}\mathcal{V}_i = \frac{1}{n}\sum_{i=1}^{n}\mathcal{V}_i \tag{4}$$

The mixture can be used when each IN aspect (weighted set) is an equally important alternative to describe the user IN aspect.

3.1.3 Mixture of superpositions In standard IR, a query is represented by a vector corresponding to a linear combination of the vectors associated with the query terms. In quantum physics, a normalised linear combination corresponds to the principle of superposition (normalised linear combination), where the description of a system state can be *superposed* to describe a new system state.

In our case, a system state is a user atomic IN aspect, and we use the superposition principle to build new atomic IN aspects from existing ones. For example, let φ_p, φ_{uk} and φ_{usa} be three pairwise orthogonal vectors in a three-dimensional IN space which, respectively, represent the atomic IN aspects "I want a pizza", "I want it to be delivered in Cambridge (UK)" and "I want it to be delivered in Cambridge (USA)". The atomic IN aspect vector "I want a pizza to be delivered in Cambridge (UK)" would be represented by a superposition $\varphi_{p/uk}$ of φ_p and φ_{uk}. We can similarly build the atomic IN aspect for Cambridge (USA). To represent the ambiguous query "pizza delivery in Cambridge" where we do not know whether Cambridge is in the USA or the UK, and assuming there is no other source of ambiguity, we would use a mixture of the two possible superposed atomic IN aspects. This brings us to another variant of combination of atomic INs, the mixture of superpositions, which forms our second algebra operator.

Each aspect vector weighted set resulting from the mixture of superpositions is constructed as follows. We first randomly select a vector from each set $\mathcal{V}_i$. In our previous example, the set $\mathcal{V}_1$ would be just one vector (φ_p), whereas $\mathcal{V}_2$ would contain two vectors (φ_{uk} and φ_{usa}). We then superpose the selected vectors (one for each term), and obtain a new vector φ. The above process is repeated for all the possible selections of vectors. The associated weighted set is thus formally defined using the *mixture of superpositions* operator:

$$\circledS_{i=1}^{n}\mathcal{V}_i = \sum_{\varphi_1\in\mathcal{V}_1} \dots \sum_{\varphi_n\in\mathcal{V}_n} \left(\frac{\sum \varphi_i}{\|\sum \varphi_i\|} \mapsto \prod_i \mathcal{V}_i\left(\varphi_i\right) \right) \tag{5}$$

In the weighted set, each atomic IN aspect is a linear combination $\sum_{i=1}^{n}\varphi_i$ of the IN aspect from each of the terms composing the query which is normalised

to ensure it is a unit vector. Each of these linear combinations is associated with a weight that is the product of the weights[5] $\prod_i \mathcal{V}_i(\varphi_i)$.

The mixture of superpositions differs from the mixture in the way we combine weighted sets. As an example, consider two terms t_1 and t_2 that always occur in a document, but half of the time in the same fragments, the other half in distinct fragments. A mixture $\mathcal{V}(t_1) \circledcirc \mathcal{V}(t_2)$ retrieves all the documents containing the terms, but a mixture of superpositions $\mathcal{V}(t_1) \textcircled{S} \mathcal{V}(t_2)$ gives a higher probability to the documents that contain both in the same fragments. The mixture of superpositions is better suited for terms whose meaning should be combined, i.e. forming phrases. Terms co-occurrence could, for instance, be used to determine whether two terms should be combined to form a phrase, e.g. "social network".

3.2 From queries to structured queries

In the previous section, we defined an algebra, i.e. its constants and operators, over weighted sets of IN aspects. In our QIR framework, a weighted set of IN aspects $\mathcal{V}$ is used to compute to which extent a document is relevant to the IN aspects in $\mathcal{V}$. With this algebra, we can express more sophisticated, aka structured, query representations, where the relationships between query terms are accounted for. In this section, we describe how a query q (a set of terms) is mapped to a structured query expressed as a weighted set $\mathcal{V}(q)$ in our algebra.

In [2], two main strategies were applied to construct the (non-structured) query representation: (1) the query had only one aspect and the weighted sets of each query term were combined using a mixture $\circledcirc$ or a mixture of superposition Ⓢ (2) each query term corresponded to an aspect or a tensor product (of n aspects). The latter strategy has been shown to perform best for a variety of TREC collections. Both strategies were, however, simple translations into our algebra.

Building a structured query involves two steps. First, we must indentify the different IN aspects, i.e. compute how many aspects the query has, and which query terms are linked to each. Second, we must characterise the relationships among terms within each aspect. We therefore make use of query analysers. The use of such tools is an active topic in IR, e.g. [7]. In this paper, the following tools are employed:

Chunkers Tools that splits a query into a series of chunks corresponding to the query "concepts". In this paper, we experimented with Two-stage query Segmentation *(TSQS)* [8], a machine learning-based chunker, and META (*Meta*), a Natural Language Processing tool [9];

Dependency parser A more powerful query analyser tool is MaltParser (*Malt*) [10], a dependency parser that builds a term graph based on grammatical relationships. Particular to MaltParser is that term dependence can be between non-adjacent terms. For example, in "the cat saw the black mouse", "saw" will be connected to "cat" (subject) and "mouse" (object), and "mouse" to "black" (modifier) and "the" (determinant).

[5] In practice, we used a modified formula for computational reasons (see [3]).

These tools were used to build a structured query representation $\mathcal{V}(q)$. A number of strategies have been experimented with; we associate with each an identifier (**in bold between parenthesis**), and a formula $\mathcal{V}(q)$ representing the query q, expressed within our algebra, i.e. phrase constants $\mathcal{V}(p)$, mixture $\circledcirc$, mixture of superpositions Ⓢ, and finally tensor product $\otimes$.

Two-term queries We wanted to compare all possible uses of operators. Thus to limit the number of possibilities, we restricted ourselves to queries composed of only two terms, t_1 and t_2[6]. We started with four such strategies (names prefixed by 2t) three using our operators: tensor (**2tT**) $\mathcal{V}(q) = \mathcal{V}(t_1) \otimes \mathcal{V}(t_2)$, mixture of superpositions (**2tS**) $\mathcal{V}(q) = \mathcal{V}(t_1) Ⓢ \mathcal{V}(t_2)$, mixture (**2tM**) $\mathcal{V}(q) = \mathcal{V}(t_1) \circledcirc \mathcal{V}(t_2)$ and one considering the two terms as forming a phrase (**2tP**) $\mathcal{V}(q) = \mathcal{V}(t_1 t_2)$. For the latter, we observed that in 30% of the cases, the phrase $t_1 t_2$ does not appear in the document collection, resulting in an empty weighted set. We hence performed two additional experiments on two-term queries where $\mathcal{V}(t_1 t_2)$ is combined with either a tensor product (**2tP-T**) $\mathcal{V}(q) = \mathcal{V}(t_1 t_2) \otimes \mathcal{V}(t_1) \otimes \mathcal{V}(t_2)$ or a mixture (**2tP-M**) $\mathcal{V}(q) = \mathcal{V}(t_1 t_2) \circledcirc \mathcal{V}(t_1) \circledcirc \mathcal{V}(t_2)$. In this way, $\mathcal{V}(t_1 t_2)$ has no effect if the phrase $t_1 t_2$ does not exist in the collection.

Reference For all experiments as a reference, we use the representation where each term corresponds to a different aspect, which was shown to be the most successful in [2]. The corresponding query representation is a tensor product of weighted sets, one for each query term t. We have two such references, one being specific to TSQS as the terms used by TSQS to construct the query representation is different to those used by all other strategies [7] (**TSQS-R, R**) $\mathcal{V}(q) = \bigotimes_{t \in q} \mathcal{V}(t)$.

Chunks We used both TSQS and META to decompose queries into chunks. These are sets of terms that can be considered as a whole, and usually define a concept. Hence, we decided to map each identified chunk using a mixture of superpositions, where each chunk is considered as an aspect of the topic. Thus each aspect is represented as a mixture of superpositions of the terms in the chunk. Denoting C the set of identified chunks, where each chunk is a set of terms, we have **(TSQS-S, Meta-S)** $\mathcal{V}(q) = \bigotimes_{c \in C} Ⓢ_{t \in c} \mathcal{V}(t)$.

Dependency Parsing We used MaltParser to extract term relationships in queries. In particular, we were interested by the "modifier" relationship, which "describes a noun phrase or restricts its meaning in some way"[8]. For example, in the TREC topic 357, "... Identify documents discussing international boundary

[6] This meant, 89 out of the 250 topics from the TREC Robust track (the detail of the test collection used in our experiments is given in Section 4).

[7] TSQS makes use of all query fields (title, description and narrative), in addiction to some external sources, to segment the original query into chunks. This results in a different bag-of-word and justify the computation of a separate *reference* for this strategy.

[8] From the Oxford dictionary.

disputes...", *international* and *boundary* are modifiers of *disputes.* We denote H_q the head terms of the query q, i.e. the terms that are not modifiers. In our example, *disputes* is a head term. For each head term t, we follow the "modified by" links of the dependency graph generated by MaltParser, and construct a set P_h of all the possible maximal sequences $t_1 \dots t_n h$, i.e. those where t_1 does not have any "modified by" link associated to it. In our example, the phrases "international disputes" and "boundary disputes" would be part of P_h.

Obviously (as also observed with 2tP) not all phrases in P_h may occur in the document collection. For such phrases, the corresponding weighted set will be empty. We therefore adopt two alternative strategies. In the first, an aspect corresponds to a mixture of all the possible subsequences of phrases constructed by following the "modified by" link. Formally **(PM)**:

$$\mathcal{V}(q) = \bigotimes_{t \in H_q} \bigotimes_{\substack{h \in H_q \\ t_1 \dots t_n h \in P_h}} (\mathcal{V}(t_1 t_2 ... t_n h) \circledcirc \mathcal{V}(t_2 ... t_n h) \circledcirc ... \circledcirc \mathcal{V}(h))$$

The second strategy is to consider that there are as many aspects as subsequences following the "modified by" link **(PT)**:

$$\mathcal{V}(q) = \bigotimes_{t \in H_q} \bigotimes_{\substack{h \in H_q \\ t_1 \dots t_n h \in P_h}} \left(\mathcal{V}(t_1 t_2 ... t_n h) \bigotimes \mathcal{V}(t_2 ... t_n h) \bigotimes ... \bigotimes \mathcal{V}(h)\right)$$

Following the same rationale as for 2tP, we decided to perform an additional experiment in which the sequences of phrases are combined with the tensor product of all the query terms. In that case, a term that is a modifier can have the same influence on the final query representation. For example, the query "A B" (where B is modified by A) becomes $\mathcal{V}_{AB} \otimes \mathcal{V}_A \otimes \mathcal{V}_B$. Thus, a query q is represented by **(PT2)**:

$$\mathcal{V}(q) = \bigotimes_{t \in q} \mathcal{V}(t) \quad \otimes \bigotimes_{\substack{h \in H_q \\ t_1 \dots t_n h \in P_h}} \mathcal{V}(t_1 t_2 ... t_n h) \bigotimes \mathcal{V}(t_2 ... t_n h) \bigotimes ... \bigotimes \mathcal{V}(t_n h)$$

Next, we experiment all these structured query constructions. While not exhaustive, they provide a good starting point to study sophisticated query representations in our QIR framework.

4 Experiments

4.1 Experimental Setup

We used the TREC ROBUST 2004 collection (topics 301-450, 601-700). All topics were automatically converted into a query using their "title" part, with the exception of the TSQS strategy (see Footnote 7). We experimented with the various strategies defined in Section 3.2. To define the weight $f(p)$ of a phrase p

(a sequence of terms), following [2], we used a normalised inverse document frequency. To create the document subspace and compute each term representation, we define each fragment as a window of 10 words, to which we apply stemming and remove stop-words before using a binary weighting scheme for the vector representing the topical aspect covered by the fragment. We then follow [2] for the computation of the document and query representation.

4.2 Results

Our performance results are reported using Mean Average Precision (MAP). Results for our three sets of experiments, *two-term queries*, *chunks*, and *dependency parsing*, are shown in Tables 1a, 1b, and 1c, respectively. We also show, for comparison, results using our Reference (TSQS-R when comparing to TSQS-S, and R when comparing with all other strategies), and the classic baseline BM25.

Overall, none of the results (except for 2tP-T) have shown an improvement over BM25. However, as already indicated in [3], although using mixture/mixture of superpositions to construct query representations do not increase overall performance, they increase the performance of specific types of topics. We return to this throughout this section.

Generally, all strategies follow similar trends across all three sets of experiments. The tensor product (2tT, TSQS-T, R) outperforms the mixture of superpositions (2tS, TSQS-S, Meta-S). But again, when looking on a per topic basis, in the *chunks* set of experiments for instance, we saw that, for 25% of the queries, the mixture of superpositions gives the best result[9].

We now discuss results with respect to the use of phrases in constructing query representations using our algebra. For two-term queries, the strategy using the tensor product (2tT) led to better performance than that using phrases made of the simple concatenation of the two terms (2tP). Furthermore, even when using a query analyser tool (MaltParser) to indentify phrases (here restricted to head and modifiers), the tensor product of *all* query terms (R) led to better performance than the tensor product of weighted sets, one for each aspect, where an aspect corresponds to a mixture of all phrases (the set P_h) associated with the head term h (PM); this is even the case when the tensor product is used to compute the weighted set for each query aspect, where each aspect corresponds to a phrase in P_h (PT).

Only two strategies making use of phrases (2tP-T, PT2) led to higher performance compared to the tensor product only strategy (no phrases) was used (2tT, R). Both 2tP-T and PT2 combine phrases with the tensor product of all query terms. This would indicate that using the tensor product of the query terms is important in representing structured queries.

We performed an in-depth manual analysis of the topics used in our two-term queries experiments. Tensor product works well when the query expresses distinct aspects that should be contained in the same document (e.g. "poppy cultivation," "L-tryptophan deaths"). Using the mixture of superpositions increases

[9] For space constraint, we do not report performance results per topic.

effectiveness for queries containing terms forming a concept (e.g., "space station", "computer viruses", "radioactive waste"). Finally, although using phrases (2tP, 2tP-M, 2tP-T) may allow capturing the narrower meaning coming from the modifier (e.g. "home schooling", "marine vegetation"), only the combined use of the tensor product on both the phrase and the two query terms (2tP-T) results in effective performance. It is not fully clear whether this comes from and only the fact that many phrases, as such, do not appear in document fragments, or whether their representation is not optimal (i.e., the right weighted set is built due to approximations issues).

Overall our results indicate that retrieval performance depends strongly on how the relationships between query terms are represented within our algebra (i.e. which operators are used and how). In future work, we will look at automatic means to assign construction strategies to topics, depending on their type. We however performed two upper-bound experiments, where we chose the best performing strategy for each topic, and calculate MAP over these best performing strategies. The first experiment was conducted over the two-term queries strategies (2tM, 2tS, 2tT, 2tP, 2tP-M and 2tP-T), and the resulting MAP value is 0.2801, which corresponds to an increase of +13% over BM25 (0.2552). The second experiment considered R, TSQS-S, TSQS-T, Meta-S, PM, PT, and PT2, leading to a MAP value of 0.2726, which corresponds to an increase of +12.59 over BM25 (0.2421).

Table 1: Mean Average Precision (MAP) values for the TREC Ad Hoc Robust Track 2004 collection.

Two-term queries	
Run	MAP
BM25	0.2552
2tM	0.1843
2tS	0.1596
2tT	0.2408
2tP	0.1629
2tP-M	0.0971
2tP-T	0.2612
upper-bound	0.2801

(a)

Chunks	
Run	MAP
BM25	0.2421
R	0.2262
TSQS-S	0.1557
TSQS-R	0.1659
Meta-S	0.1780
upper-bound	0.2726

(b)

Dependency parsing	
Run	MAP
BM25	0.2421
R	0.2262
PM	0.1916
PT	0.1993
PT2	0.2301
upper-bound	0.2726

(c)

To summarise, using phrases *and* the tensor product operator seems to be the most effective strategy (2tP-T, PT2). However, the best results are achieved with short queries (2tP-T outperforms BM25 whereas PT2 although better than R is outperformed by BM25). In future work, we intent to investigate whether this comes from the tools used to analyse the queries (e.g. we saw that Malt-

Parser failed to identify a large number of head-modifier relationships), or their representations using our algebra. As shown in our upper-bound experiments, there is much room for improvement with respect to the latter.

5 Conclusion

This paper describes a query algebra defined within a quantum-inspired approach to IR[2], where user information needs are represented as weighted sets of vectors. The algebra, defined over weighted sets, is used to construct more sophisticated, aka structured, query representations, where relationships between terms are accounted for through the consideration of phrases. We proposed several strategies to transform a query (set of terms) into a structured query (set of terms and phrases), using the operators from this algebra. Some of the strategies make use of query analyser tools to extract proper phrases.

Our experimental results indicate that each specific operator improves performance for a specific type of queries (tensor for multi-aspect queries, mixture of superpositions for concepts, and phrase operator for queries where a narrower meaning is given by a modifier). Future work includes looking at automatic means to assign operators (and strategies) to queries, in particular for representing phrases. Other query analyser tools will be employed, as properly extracting phrases is important to build effective structured queries using our algebra.

Acknowledgements This research was supported by an Engineering and Physical Sciences Research Council grant (Grant Number EP/F015984/2). This paper was written when Mounia Lalmas was a Microsoft Research/RAEng Research Professor and based on the work done by Annalina Caputo as an intern, both at the University of Glasgow.

References

1. van Rijsbergen, C.J.: The Geometry of Information Retrieval. Cambridge University Press, New York, NY, USA (2004)
2. Piwowarski, B., Frommholz, I., Lalmas, M., van Rijsbergen, K.: What can quantum theory bring to IR? In: CIKM, ACM (2010)
3. Piwowarski, B., Frommholz, I., Lalmas, M., van Rijsbergen, K.: Exploring a multidimensional representation of documents and queries. In: RIAO. (2010)
4. Metzler, D., Croft, W.B.: Combining the language model and inference network approaches to retrieval. IPM **40**(5) (2004)
5. Spink, A., Wolfram, D., Jansen, M.B.J., Saracevic, T.: Searching the web: The public and their queries. JASIST **52**(3) (2001)
6. Clarke, C.L., Kolla, M., Cormack, G.V., Vechtomova, O., Ashkan, A., Büttcher, S., MacKinnon, I.: Novelty and diversity in information retrieval evaluation. In: Proceedings of the 31st ACM SIGIR, Singapore, ACM (July 2008)
7. Croft, B., Bendersky, M., Li, H., Xu, G., eds.: SIGIR 2010 Workshop on Query Representation and Understanding. (2010)
8. Bendersky, M., Croft, W.B., Smith, D.A.: Two-stage query segmentation for information retrieval. In: Proc. of the 32nd ACM SIGIR, ACM (2009)
9. Basile, P., de Gemmis, M., Gentile, A., Iaquinta, L., Lops, P., Semeraro, G.: META-MultilanguagE Text Analyzer. In: Proc. of LangTech. (2008)
10. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E.: MaltParser: A language-independent system for data-driven dependency parsing. Natural Language Engineering **13**(02) (2007)