Data Models and Languages for Agent-Based Textual Information Dissemination*

M. Koubarakis, C. Tryfonopoulos, P. Raftopoulou, and T. Koutris

Dept. of Electronic and Computer Engineering
Technical University of Crete
73100 Chania, Crete, Greece
manolis@ced.tuc.gr, {trifon,rautop,koutris}@mhl.tuc.gr
www.ced.tuc.gr/~manolis

Abstract. We define formally the data models $\mathcal{WP}$, $\mathcal{AWP}$ and $\mathcal{AWPS}$ especially designed for the dissemination of textual information by distributed agent systems using communication languages such as KQML and FIPA-ACL. We also define the problems of satisfaction and filtering and point out that these problems are fundamental for the deployment of our models in distributed agent architectures appropriate for information dissemination. One such architecture currently under development in project DIET is sketched in some detail in this paper. Finally, we present algorithms for the problems of satisfaction and filtering, prove the correctness of these algorithms, and calculate their computational complexity.

1 Introduction

The selective dissemination of information to interested users is a problem arising frequently in today's information society. This problem has recently received the attention of various research communities including researchers from agent systems [16, 22, 12, 28, 29], databases [18, 2, 26, 14], digital libraries [15], distributed computing [6, 4] and others.

We envision an information dissemination scenario in the context of a *distributed peer-to-peer (P2P) agent architecture* like the one shown in Figure 1. Users utilize their *end-agents* to post *profiles* or *documents* (expressed in some appropriate language) to some *middle-agents.* End-agents play a dual role: they can be information producers and information consumers at the same time. The P2P network of middle-agents is the "glue" that makes sure that published documents arrive at interested subscribers. To achieve this, middle-agents forward posted profiles to other middle-agents using an appropriate P2P protocol. In this way, matching of a profile with a document can take place at a middle-agent that is as close as possible to the origin of the incoming document. Profile

* This work was carried out as part of the DIET (Decentralised Information Ecosystems Technologies) project (IST-1999-10088), within the Universal Information Ecosystems initiative of the Information Society Technology Programme of the European Union.

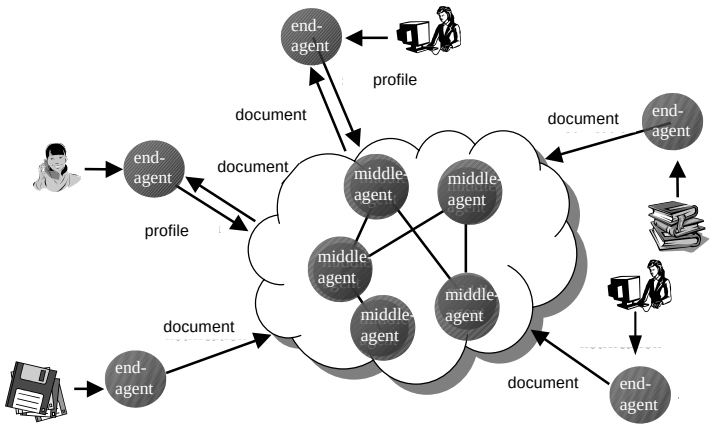

Fig. 1. A distributed P2P agent architecture for information dissemination

forwarding can be done in a sophisticated way to minimize network traffic e.g., no profiles that are less general than one that has already been processed are actually forwarded.

In their capacity as information producers, end-agents can also post *advertisements* that describe in a "concise" way the documents that will be produced by them. These advertisements can also be forwarded in the P2P network of middle-agents to *block* the forwarding of *irrelevant* profiles towards a source. Advertisement forwarding can also be done in a sophisticated way using ideas similar to the ones for profile forwarding.[1]

Our work in this paper concentrates on models and languages for expressing documents and queries/profiles in *textual information* dissemination systems that follow the general architecture of Figure 1.[2] We are motivated by a desire to develop useful agent systems in a *principled* and *formal* way, and make the

[1] Most of the concepts of the architecture sketched above are explicit (or sometimes implicit) in the KQML literature and subsequent multi-agent systems based on it [16, 22, 12, 28, 29]. Unfortunately the emphasis in most of these systems is on a single central middle-agent, making the issues that would arise in a distributed setting difficult to appreciate. In our opinion, the best presentation of these concepts available in the literature can be found in [6] where the distributed event dissemination system SIENA is presented. SIENA does not use terminology from the area of agent systems but the connection is obvious.

[2] We use the terms *query* and *profile* interchangeably. In an information dissemination setting, a profile is simply a long-standing query. We do not consider advertisements, but it should be clear from our presentation that appropriate subsets of the query languages that we will present could be used for expressing advertisements as well.

following technical contributions. We define formally the models $\mathcal{WP}$, $\mathcal{AWP}$ and $\mathcal{AWPS}$, and their corresponding languages for textual information dissemination in distributed agent systems. Data model $\mathcal{WP}$ is based on free text and its query language is based on the *boolean model with proximity operators*. The concepts of $\mathcal{WP}$ extend the traditional concept of proximity in IR [3, 8, 9] in a significant way and utilize it in a content language targeted at information dissemination applications. Data model $\mathcal{AWP}$ is based on *attributes* or *fields* with finite-length strings as values. Its query language is an extension of the query language of data model $\mathcal{WP}$. Our work on $\mathcal{AWP}$ complements recent proposals for querying textual information in distributed event-based systems [6, 4] by using linguistically motivated concepts such as *word* and not arbitrary strings. This makes $\mathcal{AWP}$ potentially very useful in some applications (e.g., alert systems for digital libraries or other commercial systems where similar models are supported already for retrieval). Finally, the model $\mathcal{AWPS}$ extends $\mathcal{AWP}$ by introducing a "similarity" operator in the style of modern IR, based on the vector space model [3]. The novelty of our work in this area is the move to query languages much more expressive than the one used in the information dissemination system SIFT [33] where documents and queries are represented by free text. The similarity concept of $\mathcal{AWPS}$ is an extension of the similarity concept pioneered by the system WHIRL [11] and recently also used in the XML query language ELIXIR [10]. We note that both WHIRL and ELIXIR target information retrieval and integration applications, and pay no attention to information dissemination and the concepts/functionality needed in such applications. The models $\mathcal{WP}$ and $\mathcal{AWP}$ are also discussed in [20, 21] but no connection to agent systems and architectures is made. The first presentation of model $\mathcal{AWPS}$ is the one given in this paper.

In the second part of our paper, we built on the formal foundations of the first part and study the computational complexity of the problems of matching and filtering in the three models we have defined. These results are original and are currently leading to an implementation of a prototype information dissemination system in the context of project DIET [24, 30, 19].

The rest of the paper is organised as follows. Section 2 presents data model $\mathcal{WP}$ based on free text and its sophisticated query language. Then Sections 3 and 4 build on this foundation and develops the same machinery for data models $\mathcal{AWP}$ and $\mathcal{AWPS}$. Section 5 presents our complexity results for the problems of satisfaction and filtering. Finally, Section 6 gives our conclusions and discusses future work. The proofs of the results of Section 5 are omitted. They can be found (together with a very detailed discussion of related work) in the long version of this paper which is available at:
`http://www.intelligence.tuc.gr/~manolis/publications.html`.

2 Text Values and Word Patterns

In this section we present the data model $\mathcal{WP}$ and its query language. $\mathcal{WP}$ assumes that textual information is in the form of *free text* and can be queried

by *word patterns* (hence the acronym for the model). The basic concepts of $\mathcal{WP}$ are subsequently used in Section 3 to define the data model $\mathcal{AWP}$ and its query language.

We assume the existence of a finite *alphabet* $\mathbf{\Sigma}$. A *word* is a finite non-empty sequence of letters from $\mathbf{\Sigma}$. We also assume the existence of a (finite or infinite) set of words called the *vocabulary* and denoted by $\mathcal{V}$.

Definition 1. *A* text value *s of length n over vocabulary $\mathcal{V}$ is a total function $s : \{1, 2, \ldots, n\} \rightarrow \mathcal{V}$.*

In other words, a text value s is a finite sequence of words from the assumed vocabulary and $s(i)$ gives the i-th element of s. Text values can be used to represent finite-length strings consisting of words separated by blanks. The length of a text value s (i.e., its number of words) will be denoted by $|s|$.

We now give the definition of word-pattern. The definition is given recursively in three stages.

Definition 2. *Let $\mathcal{V}$ be a vocabulary. A* proximity-free word pattern *over vocabulary $\mathcal{V}$ is an expression generated by the grammar*

$$WP \rightarrow \mathbf{w} \mid \neg WP \mid WP \wedge WP \mid WP \vee WP \mid (WP)$$

where terminal $\mathbf{w}$ *represents a word of $\mathcal{V}$. A proximity-free word pattern will be called* positive *if it does not contain the negation operator.*

Example 1. The following are proximity-free word patterns that can appear in queries of a user of a news dissemination system interested in holidays:

$$Athens \wedge hotel \wedge \neg Hilton, \quad holiday \wedge (beach \vee mountains)$$

Word patterns made of words and the Boolean operators $\wedge, \vee$ and $\neg$ should be understood as in traditional IR systems and modern search engines. These systems typically have a version of negation in the form of binary operator $AND\text{-}NOT$ which is essentially set difference thus safe (in the database sense of the term [1]). For example, a search engine query wp_1 $AND\text{-}NOT$ wp_2 will return the set of documents that satisfy wp_1 *minus* these that satisfy wp_2. In our information dissemination setting, there is no problem considering an "unsafe" version of negation since word patterns are checked for satisfaction against a single incoming document. Note that the previous work of [9] has *not* considered negation in its word pattern language (but has considered negation in the query language which supports attributes; see Section 3).

We now introduce a new class of word patterns that allows us to capture the concepts of *order* and *distance* between words in a text document. We will assume the existence of a set of *(distance) intervals* $\mathcal{I}$ defined as follows:

$$\mathcal{I} = \{[l, u] : \ l, u \in \mathbb{N}, l \geq 0 \text{ and } l \leq u\} \cup \{[l, \infty) : \ l \in \mathbb{N} \text{ and } l \geq 0\}$$

The symbols $\in$ and $\subseteq$ will be used to denote membership and inclusion in an interval as usual.

The following definition uses intervals to impose lower and upper bounds on distances between word patterns.

Definition 3. *Let $\mathcal{V}$ be a vocabulary. A* proximity word pattern *over vocabulary $\mathcal{V}$ is an expression $wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{n-1}} wp_n$ where $wp_1, wp_2, \ldots, wp_n$ are positive proximity-free word patterns over $\mathcal{V}$ and $i_1, i_2, \ldots, i_{n-1}$ are intervals from the set $\mathcal{I}$. The symbols $\prec_i$ where $i \in \mathcal{I}$ are called* proximity operators. *The number of proximity-free word patterns in a proximity word pattern (i.e., n above) is called its* size.

Example 2. The following are proximity word patterns:

$$Holiday \prec_{[0,0]} Inn, \quad Mini \prec_{[0,0]} Palace \prec_{[0,0]} Hotel,$$
$$luxurious \prec_{[0,3]} (hotel \vee apartment), \quad hotel \prec_{[0,\infty)} view$$
$$holiday \prec_{[0,10]} beach \prec_{[0,10]} (clean \wedge sandy)$$

The proximity word pattern $wp_1 \prec_{[l,u]} wp_2$ stands for "word pattern wp_1 is *before* wp_2 and is separated by wp_2 by *at least* l and *at most* u *words*". In the above example $luxurious \prec_{[0,3]} hotel$ denotes that the word "hotel" appears after word "luxurious" and at a distance of at least 0 and at most 3 words. The word pattern $Holiday \prec_{[0,0]} Inn$ denotes that the word "Holiday" appears exactly before word "Inn" so this is a way to encode the string "Holiday Inn". We can also have arbitrarily long sequences of proximity operators with similar meaning (see the examples above). Note that proximity-free subformulas in proximity word-patterns can be more complex than just simple words (but negation is *not* allowed; this restriction will be explained below). This makes proximity-word patterns a very expressive notation.

Definition 4. *Let $\mathcal{V}$ be a vocabulary. A* word pattern *over vocabulary $\mathcal{V}$ is an expression generated by the grammar*

$$WP \rightarrow PFWP \mid PWP \mid WP \wedge WP \mid WP \vee WP \mid (WP)$$

where non-terminals $PFWP$ and PWP represent proximity-free and proximity word patterns respectively. A word pattern will be called positive *if its proximity-free subformulas are positive.*

Example 3. The following are word patterns of the most general kind we allow:

$$holiday \wedge (luxurious \prec_{[0,0]} hotel) \wedge \neg Hilton,$$
$$holiday \wedge (hotel \prec_{[0,10]} (cheap \wedge clean)),$$
$$Vienna \wedge ((Dolce \prec_{[0,0]} Vita \prec_{[0,0]} Hotel) \vee (Mini \prec_{[0,0]} Palace \prec_{[0,0]} Hotel))$$

We have here completed the definition of the concept of word pattern. We now turn to defining its semantics. First, we define what it means for a text value to satisfy a proximity-free word pattern.

Definition 5. *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$ and wp a proximity-free word pattern over $\mathcal{V}$. The concept of s* satisfying *wp (denoted by $s \models wp$) is defined as follows:*

1. *If wp is a word of $\mathcal{V}$ then $s \models wp$ iff there exists $p \in \{1, \ldots, |s|\}$ and $s(p) = wp$.*

2. *If wp is of the form $\neg wp_1$ then $s \models wp$ iff $s \not\models wp_1$.*
3. *If wp is of the form $wp_1 \wedge wp_2$ then $s \models wp$ iff $s \models wp_1$ and $s \models wp_2$.*
4. *If wp is of the form $wp_1 \vee wp_2$ then $s \models wp$ iff $s \models wp_1$ or $s \models wp_2$.*
5. *If wp is of the form (wp_1) then $s \models wp$ iff $s \models wp_1$.*

The above definition mirrors the definition of satisfaction for Boolean logic [25]. This will allow us to draw on a lot of related results in the rest of this paper.

Example 4. Let s be the following text value:

During our holiday in Milos we stayed in a luxurious hotel by the beach

Then $s \models holiday \wedge Milos$.

The following definition captures the notion of a set of positions in a text value containing only words that contribute to the satisfaction of a proximity-free word pattern. This notion is then used to define satisfaction of proximity word patterns.

Definition 6. *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$, wp a proximity-free word pattern over $\mathcal{V}$, and P a subset of $\{1, \ldots, |s|\}$. The concept of s* satisfying *wp with set of positions P (denoted by $s \models_P wp$) is defined as follows:*

1. *If wp is a word of $\mathcal{V}$ then $s \models_P wp$ iff there exists $x \in \{1, \ldots, |s|\}$ such that $P = \{x\}$ and $s(x) = wp$.*
2. *If wp is of the form $wp_1 \wedge wp_2$ then $s \models_P wp$ iff there exist sets of positions $P_1, P_2 \subseteq \{1, \ldots, |s|\}$ such that $s \models_{P_1} wp_1$, $s \models_{P_2} wp_2$ and $P = P_1 \cup P_2$.*
3. *If wp is of the form $wp_1 \vee wp_2$ then $s \models_P wp$ iff $s \models_P wp_1$ or $s \models_P wp_2$.*
4. *If wp is of the form (wp_1) then $s \models_P wp$ iff $s \models_P wp_1$.*

Now we define what it means for a text value to satisfy a proximity word pattern.

Definition 7. *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$ and wp a proximity word pattern over $\mathcal{V}$ of the form $wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{n-1}} wp_n$. Then $s \models wp$ iff there exist sets $P_1, P_2, \ldots, P_n \subseteq \{1, \ldots, |s|\}$ such that $s \models_{P_j} wp_j$ and $min(P_j) - max(P_{j-1}) - 1 \in i_{j-1}$ for all $j = 2, \ldots, n$ (the operators max and min have the obvious meaning).*

Example 5. The text value of Example 4 satisfies the following word patterns:

$$luxurious \prec_{[0,0]} hotel \prec_{[0,5]} beach$$
$$luxurious \prec_{[0,0]} (hotel \vee apartment) \prec_{[0,5]} beach,$$
$$(holiday \wedge Milos) \prec_{[0,10]} luxurious \prec_{[0,0]} hotel$$

The sets of positions required by the definition are for the first and second word pattern $\{10\}$, $\{11\}$ and $\{14\}$, and for the third one $\{3, 5\}$, $\{10\}$ and $\{11\}$.

If the structure of wp falls under the four cases of our most general definition (Definition 4), satisfaction is similarly defined in a recursive way as in Definition 5 (for Cases 1, 3 and 4) and Definition 7 (for Case 2).

Example 6. The text value of Example 4 satisfies word pattern

$$holiday \wedge (luxurious \prec_{[0,0]} hotel \prec_{[0,5]} beach).$$

3 An Attribute-Based Data Model and Query Language

Now that we have studied the data model $\mathcal{WP}$ in great detail, we are ready to define our second data model and query language. Data model $\mathcal{AWP}$ is based on *attributes* or *fields* with finite-length strings as values (in the acronym $\mathcal{AWP}$, the letter $\mathcal{A}$ stands for "attribute"). Strings will be understood as sequences of words as formalised by the model $\mathcal{WP}$ presented earlier. Attributes can be used to encode textual information such as author, title, date, body of text and so on. $\mathcal{AWP}$ is restrictive since it offers a rather flat view of a text document, but it has wide applicability as we will show below.

We start our formal development by defining the concepts of document schema and document. Throughout the rest of this paper we assume the existence of a countably infinite set of attributes $\mathbf{U}$ called the *attribute universe*.

Definition 8. *A* document schema $\mathcal{D}$ *is a pair* $(\mathcal{A}, \mathcal{V})$ *where* $\mathcal{A}$ *is a subset of the attribute universe* $\mathbf{U}$ *and* $\mathcal{V}$ *is a vocabulary.*

Example 7. An example of a document schema for a news dissemination application is

$$\mathcal{D} = (\{SENDER,\ EMAIL,\ BODY\}, \mathcal{E}).$$

Definition 9. *Let* $\mathcal{D}$ *be a document schema. A* document d *over schema* $(\mathcal{A}, \mathcal{V})$ *is a set of attribute-value pairs* (A, s) *where* $A \in \mathcal{A}$*,* s *is a text value over* $\mathcal{V}$*, and there is at most one pair* (A, s) *for each attribute* $A \in \mathcal{A}$*.*

Example 8. The following is a document over the schema of Example 7:

$$\{\ (SENDER, \text{“}John\ Brown\text{”}), (EMAIL, \text{“}jbrown@yahoo.com\text{”}),$$
$$(BODY, \text{“}During\ our\ holiday\ in\ Milos\ we\ stayed\ in\ a\ luxurious\ hotel\ by\ the\ beach\text{”})\ \}$$

The syntax of our query language is given by the following recursive definition.

Definition 10. *Let* $\mathcal{D} = (\mathcal{A}, \mathcal{V})$ *be a document schema. A* query *over* $\mathcal{D}$ *is a formula in any of the following forms:*

1. $A \sqsupset wp$ *where* $A \in \mathcal{A}$ *and* wp *is a* positive *word pattern over* $\mathcal{V}$*. The formula* $A \sqsupset wp$ *can be read as "A contains word pattern wp".*
2. $A = s$ *where* $A \in \mathcal{A}$ *and* s *is a text value over* $\mathcal{V}$*.*
3. $\neg\phi$ *where* ϕ *is a query containing no proximity word patterns.*
4. $\phi_1 \vee \phi_2$ *where* ϕ_1 *and* ϕ_2 *are queries.*
5. $\phi_1 \wedge \phi_2$ *where* ϕ_1 *and* ϕ_2 *are queries.*

Example 9. The following are queries over the schema of Example 7:

$$SENDER \sqsupset (John \prec_{[0,2]} Smith),$$
$$\neg SENDER = \text{“}John\ Smith\text{”} \wedge (BODY \sqsupset (Milos \wedge (hotel \prec_{[0,5]} beach)))$$

Let us now define the semantics of the above query language in our dissemination setting. We start by defining when a document satisfies a query.

Definition 11. *Let $\mathcal{D}$ be a document schema, d a document over $\mathcal{D}$ and ϕ a query over $\mathcal{D}$. The concept of document d* satisfying *query ϕ (denoted by $d \models \phi$) is defined as follows:*

1. *If ϕ is of the form $A \sqsupset wp$ then $d \models \phi$ iff there exists a pair $(A, s) \in d$ and $s \models wp$.*
2. *If ϕ is of the form $A = s$ then $d \models \phi$ iff there exists a pair $(A, s) \in d$.*
3. *If ϕ is of the form $\neg\phi_1$ then $d \models \phi$ iff $d \not\models \phi_1$.*
4. *If ϕ is of the form $\phi_1 \wedge \phi_2$ then $d \models \phi$ iff $d \models \phi_1$ and $d \models \phi_2$.*
5. *If ϕ is of the form $\phi_1 \vee \phi_2$ then $d \models \phi$ iff $d \models \phi_1$ or $d \models \phi_2$.*

Example 10. The first query of Example 9 is not satisfied by the document of Example 8 while the second one is satisfied.

4 Extending $\mathcal{AWP}$ with Similarity

Let us now define our third data model $\mathcal{AWPS}$ and its query language. $\mathcal{AWPS}$ extends $\mathcal{AWP}$ with the concept of *similarity* between two text values (the letter $\mathcal{S}$ stands for similarity). The idea here is to have a "soft" alternative to the "hard" operator $\sqsupset$. This operator is very useful for queries such as "I am interested in documents sent by John Brown" which can be written in $\mathcal{AWP}$ as

$$SENDER \sqsupset (John \prec_{[0,0]} Brown)$$

but it might not be very useful for queries "I am interested in documents about the use of ideas from agent research in the area of information dissemination".

The desired functionality can be achieved by resorting to an important tool of modern IR: the *weight* of a word as defined in the Vector Space Model (VSM) [3, 23, 31]. In VSM, documents (text values in our terminology) are conceptually represented as vectors. If our vocabulary consists of n distinct words then a text value s is represented as an n-dimensional vector of the form $(\omega_1, \ldots, \omega_n)$ where ω_i is the weight of the i-th word (the weight assigned to a non-existent word is 0). With a good weighting scheme, the VSM representation of a document can be a surprisingly good model of its semantic content in the sense that "similar" documents have very close semantic content. This has been demonstrated by many successful IR systems recently (see for example, WHIRL [11]). [3]

In VSM, the weight of a word is computed using the heuristic of assigning higher *weights* to words that are frequent in a document and *infrequent* in the collection of documents available. This heuristic is made concrete using the concepts of word frequency and the inverse document frequency defined below.

[3] Note that in the VSM model and systems adopting it (e.g., WHIRL [11]) word *stems*, produced by some stemming algorithm [27], are forming the vocabulary instead of words. Additionally, *stopwords* (e.g., "the") are eliminated from the vocabulary. These important details have no consequence for the theoretical results of this paper, but it should be understood that our current implementation of the ideas of this section utilizes these standard techniques.

Definition 12. *Let w_i be a word in document d_j of a collection C. The* term frequency *of w_i in d_j (denoted by tf_{ij}) is equal to the number of occurrences of word w_i in d_j. The* document frequency *of word w_i in the collection C (denoted by df_i) is equal to the number of documents in C that contain w_i. The* inverse document frequency *of w_i is then given by $idf_i = \frac{1}{df_i}$. Finally, the number $tf_{ij} \cdot idf_i$ will be called the* weight *of word w_i in document d_j and will be denoted by ω_{ij}.*

At this point we should stress that the concept of inverse document frequency assumes that there is a *collection* of documents which is used in the calculation. In our dissemination scenario we assume that for each attribute A there is a collection of text values C_A that is used for calculating the *idf* values to be used in similarity computations involving attribute A (the details are given below). C_A can be a collection of recently processed text values as suggested in [33].

We are now ready to define the main new concept in $\mathcal{AWPS}$, the similarity of two text values. The similarity of two text values s_q and s_d is defined as the cosine of the angle formed by their corresponding vectors:[4]

$$sim(s_q, s_d) = \frac{s_q \cdot s_d}{\|s_q\| \cdot \|s_d\|} = \frac{\sum_{i=1}^{N} w_{q_i} \cdot w_{d_i}}{\sqrt{\sum_{i=1}^{N} w_{q_i}^2 \cdot \sum_{i=1}^{N} w_{d_i}^2}} \tag{1}$$

By this definition, similarity values are real numbers in the interval $[0, 1]$.

Let us now proceed to give the syntax of the query language for $\mathcal{AWPS}$. Since $\mathcal{AWPS}$ extends $\mathcal{AWP}$, a query in the new model is given by Definition 10 with one more case for atomic queries:

- $A \sim_k s$ where $A \in \mathcal{A}$, s is a text value over $\mathcal{V}$ and k is a real number in the interval $[0, 1]$.

Example 11. The following are some queries in $\mathcal{AWPS}$ using the schema of Example 8:

$$BODY \sim_{0.6} \text{“}Milos\ is\ the\ ideal\ place\ for\ holidays\ by\ the\ beach\text{”},$$
$$(SENDER \sqsupset (John \prec_{[0,2]} Brown)) \wedge$$
$$(TITLE \sim_{0.9} \text{“}Hotels\ and\ resorts\ in\ Greece\text{”}),$$
$$BODY \sim_{0.9} \text{“}Stock\ options\ during\ Easter\ holidays\text{”}$$

We now give the semantics of our query language, by defining when a document satisfies a query. Naturally, the definition of satisfaction in $\mathcal{AWPS}$ is as in Definition 11 with one additional case for the similarity operator:

- If ϕ is of the form $\mathcal{A} \sim_k s_q$ then $d \models \phi$ iff there exists a pair $(A, s_d) \in d$ and $sim(s_q, s_d) \geq k$.

[4] The IR literature gives us several very closely related ways to define the notions of weight and similarity [3, 23, 31]. All of these weighting schemes come by the name of $tf \cdot idf$ weighting schemes. Generally a weighting scheme is called $tf \cdot idf$ whenever it uses word frequency in a monotonically increasing way, and document frequency in a monotonically decreasing way.

The reader should notice that the number k in a similarity predicate $A \sim_k s$ gives a *relevance threshold* that candidate text values s should exceed in order to satisfy the predicate. This notion of relevance threshold was first proposed in an information dissemination setting by [17] and later on adopted by [33]. The reader is asked to contrast this situation with the typical information retrieval setting where a ranked list of documents is returned as an answer to a user query. This is not a relevant scenario in an information dissemination system because very few documents (or even a single one) enter the system at a time, and need to be forwarded to interested users (see the architecture sketched in Figure 1).

A low similarity threshold in a predicate $A \sim_k s$ might result in many irrelevant documents satisfying a query, whereas a high similarity threshold would result in very few achieving satisfaction (or even no documents at all). In an implementation of our ideas, users can start with a certain relevance threshold and then update it using relevance feedback techniques to achieve a better satisfaction of their information needs. Recent techniques from adaptive IR can be utilised here [7].

Example 12. The first query of Example 11 is likely to be satisfied by the document of Example 8 (of course, we cannot say for sure until the exact weights are calculated in the manner suggested above). The second query is not satisfied, since attribute $TITLE$ does not exist in the document. Moreover the third query is unlikely to be satisfied since the only common word between the query and Example 8 is the word "holiday".

5 The Complexity of Satisfaction and Filtering

For an information dissemination architecture like the one in Figure 1 to become a reality, two very important problems need to be solved efficiently. The first problem is the *satisfaction* (or *matching*) *problem*: Deciding whether a document satisfies (or matches) a profile. The second problem is the *filtering problem*: Given a database of profiles db and a document d, find all profiles $q \in db$ that match d. This functionality is very crucial at each middle-agent and it is based on the availability of algorithms for the satisfaction problem. We expect deployed information dissemination systems to handle hundreds of thousands or millions of profiles.

In this section we present PTIME upper bounds for the satisfaction problem and filtering problem in models $\mathcal{WP}$ and $\mathcal{AWP}$. The reader is reminded that profiles in $\mathcal{WP}$ and $\mathcal{AWP}$ are defined by languages containing all Boolean connectives and not just conjunction as in virtually all previous work in the area of event dissemination systems (a notable exception of this rule is [4]).

5.1 Algorithms for Satisfaction

In previous research, [9] have presented a method for evaluating positive word patterns with proximity operators kW and kN on sets of text values (Fig. 2 of

function $eppf(wp, s)$
if wp is a word of $\mathcal{V}$ **then**
  return $\{\ [x, x] :\ s(x) = wp\ \}$
else if wp is of the form $wp_1 \wedge wp_2$ **then**
  return $\{\ [min(l_1, l_2), max(u_1, u_2)] :\ [l_1, u_1] \in eppf(wp_1, s)$ and
    $[l_2, u_2] \in eppf(wp_2, s)\ \}$
else if wp is of the form $wp_1 \vee wp_2$ **then**
  return $\{\ [l, u] :\ [l, u] \in eppf(wp_1, s)\ \cup eppf(wp_2, s)\ \}$
else if wp is of the form (wp_1) **then**
  return $eppf(wp_1)$

function $prox(wp, s)$
if wp is a positive proximity-free word pattern **then**
  return $eppf(wp, s)$
else
  Let wp be $wp_1 \prec_i rest$ where $rest$ is a proximity word pattern
  return $\{\ [l_1, u_1] :\ [l_1, u_1] \in eppf(wp_1, s)$ and there exists
    a position interval $[l_2, u_2] \in prox(rest, s)$
    such that $l_2 - u_1 - 1 \in i\ \}$
end

Fig. 2. Some useful functions for deciding whether $s \models wp$

[9]). This method is intended to provide semantics to word patterns, and nothing is said about the computational complexity of evaluation. In this paper we have followed the more formal route of *separating* the definition of semantics from the algorithms and complexity of deciding satisfaction.

We start with the satisfaction problem for proximity-free word patterns of the model $\mathcal{WP}$.

Lemma 1. *Let s be a text value and wp a proximity-free word pattern. We can decide whether $s \models wp$ in $O(\delta + \rho)$ time on average where δ is the number of words and ρ is the number of operators in wp.*

We now turn to proximity word patterns. We first need the following lemma.

Lemma 2. *Let s be a text value and wp a positive proximity-free word pattern. Function $eppf(wp, s)$ shown in Figure 2 returns a non-empty set of position intervals O iff $s \models wp$. Additionally, for every set of positions P such that $s \models_P wp$ there exists an interval $[l, u] \in O$ such that*

$$P \subseteq [l, u],\ min(P) = l \text{ and } max(P) = u.$$

The set O can be computed in $O((\delta + \rho)\,|s|^4)$ time where δ is the number of words and ρ is the number of operators in wp.

We now use the above lemma to compute an upper bound on the complexity of satisfaction for proximity word patterns.

Lemma 3. *Let s be a text value and wp a proximity word pattern. We can decide whether $s \models wp$ in $O(n(\delta_{max} + \rho_{max})\,|s|^4)$ time where n is the number of proximity free subformulas of wp, δ_{max} is the maximum number of words in a proximity-free subformula of wp and ρ_{max} is the maximum number of operators in a proximity-free subformula of wp.*

We can now show that satisfaction can be decided in PTIME for all formulas in $\mathcal{WP}$.

Theorem 1. *Let s be a text value and wp a word pattern. The problem of deciding whether $s \models wp$ can be solved in $O(\mu n_{max}(\delta_{max} + \rho_{max})\,|s|^4)$ time where μ is the number of operators $\wedge$ and $\vee$ in wp, n_{max} is the maximum number of proximity-free subformulas in a proximity word pattern of wp, δ_{max} is the maximum number of words in a proximity-free subformula of wp and ρ_{max} is the maximum number of operators in a proximity-free subformula of wp.*

We will now show that satisfaction can be decided in PTIME for all formulas of $\mathcal{AWP}$ as well.

Theorem 2. *Let d be a document and ϕ be a query in $\mathcal{AWP}$. Deciding whether $d \models \phi$ can be done in $O((E+H)(\alpha+V)MN(\Delta+P)^2S^4)$ time where E is the number of atomic subqueries in ϕ, H is the number of operators in ϕ, α is the number of attributes in d, V is the maximum size of a text value appearing in ϕ, M is the maximum number of operators $\wedge$ and $\vee$ in a word pattern of ϕ, N is the maximum number of proximity-free subformulas in a proximity word pattern of ϕ, Δ is the maximum number of words in a proximity-free subformula of ϕ, P is the maximum number of operators in a proximity-free subformula of ϕ, and S is the maximum size of a text value appearing in d.*

Now we turn to the complexity of deciding the similarity between two text values.

Lemma 4. *Let s, v be text values. We can decide whether $s \sim_k v$ in $O(\max(|s|, |v|))$ time on average, where $|s|, |v|$ is the size of text values s and v respectively.*

We can now show that the satisfaction problem for $\mathcal{AWPS}$ can also be solved in PTIME.

Theorem 3. *Let d be a document and ϕ be a query in $\mathcal{AWPS}$. To decide whether $d \models \phi$ we need $O((E+H)(\alpha+\max(S,V))MN(\Delta+P)^2S^4)$ time, where all the parameters are as in Theorem 2.*

Let us now consider the filtering problem introduced at the beginning of this section. Let us assume that we have a database of profiles *db* and a published event e, how can we find all the elements of *db* that match e efficiently? In a brute-force fashion could solve a filtering problem by solving $|db|$ satisfaction problems where $|db|$ is the size of the database of profiles *db*. Thus the filtering problem for models $\mathcal{WP}$ and $\mathcal{AWP}$ can also be solved in PTIME (the exact upper bounds are omitted because they can be easily computed using Theorems 1 and 2).

In practice one would create *indices* over the database of profiles *db* to solve the matching problem more efficiently. This approach has been pioneered in SIFT [32, 33] where queries are conjunctions of keywords interpreted under the boolean or vector space model. Similar indexing algorithms for simple arithmetic constraints have also been presented in [26, 14, 5].

The algorithms of [32] have recently been re-evaluated extensively in [13] and some properties overlooked in the original study of [32] were pointed out. Additionally, it was shown that a main memory version of the most sophisticated algorithm of [32] (called *Key*) can solve the filtering problem for millions of profiles very efficiently. Currently, we are extending this study to the more expressive languages of this paper.

6 Conclusions

In this paper we presented the models $\mathcal{WP}$, $\mathcal{AWP}$ and $\mathcal{AWPS}$ for textual information dissemination in distributed agent systems. We laid down the logical foundations of these models and their corresponding languages, we formalized two fundamental problems arising in information dissemination environments utilizing them, and studied the computational complexity of these problems.

Our current work concentrates on extending the results of [13] to obtain efficient algorithms for solving the filtering problem for queries in the model $\mathcal{AWPS}$. We are also implementing a prototype information dissemination system using the architecture briefly discussed in Section 1 and the API developed in the DIET project and discussed in [24, 30, 19].

Acknowledgements

This work was carried out as part of the DIET (Decentralised Information Ecosystems Technologies) project (IST-1999-10088), within the Universal Information Ecosystems initiative of the Information Society Technology Programme of the European Union. We would like to thank the other participants in the DIET project, from Departmento de Teoria de Senal y Comunicaciones, Universidad Carlos III de Madrid, the Intelligent Systems Laboratory, BTexact Technologies and the Intelligent and Simulation Systems Department, DFKI, for their comments and contributions.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
2. M. Altinel and M.J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference*, 2000.
3. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

4. A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith. Efficent filtering in publish-subscribe systems using binary decision diagrams. In *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001.
5. A. Carzaniga, J. Deng, and A. L. Wolf. Fast forwarding for content-based networking. Technical report, Dept. of Computer Science, University of Colorado, 2001.
6. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC'2000)*, pages 219–227, 2000.
7. U. Cetintemel, M.J. Franklin, and C.L. Giles. Self-adaptive user profiles for large-scale data delivery. In *ICDE*, pages 622–633, 2000.
8. C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Boolean Query Mapping across Heterogeneous Information Sources. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):515–521, 1996.
9. C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. *ACM Transactions on Information Systems*, 17(1):1–39, 1999.
10. T. T. Chinenyanga and N. Kushmerick. Expressive retrieval from XML documents. In *Proceedings of SIGIR'01*, September 2001.
11. William W. Cohen. WHIRL: A word-based information representation language. *Artificial Intelligence*, 118(1-2):163–196, 2000.
12. K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of IJCAI-97*, 1997.
13. M. Koubarakis et. al. Project DIET Deliverable 7 (Information Brokering), December 2001.
14. F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proceedings of ACM SIGMOD-2001*, 2001.
15. D. Faensen, L. Faulstich, H. Schweppe, A. Hinze, and A. Steidinger. Hermes – A Notification Service for Digital Libraries. In *Proceedings of the Joint ACM/IEEE Conference on Digital Libraries (JCDL'01), Roanoke, Virginia, USA*, 2001.
16. T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press.
17. P.W. Foltz and S.T. Dumais. Personalised information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):29–38, 1992.
18. M. J. Franklin and S. B. Zdonik. "Data In Your Face": Push Technology in Perspective. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 516–519, 1998.
19. A. Galardo-Antolin, A. Navia-Vasquez, H.Y. Molina-Bulla, A.B. Rodriquez-Gonzalez, F.J. Valvarde-Albacete, A.R. Figueiras-Vidal, T. Koutris, A. Xiruhaki, and M. Koubarakis. I-Gaia: an Information Processing Layer for the DIET Platform . In *Proceedings of the 1st International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2002)*, July 15–19 2002.
20. M. Koubarakis. Boolean Queries with Proximity Operators for Information Dissemination. Proceedings of the workshop on Foundations of Models and Languages for Information Integration (FMII-2001), Viterbo,

Italy , 16-18 September, 2001. In LNCS (forthcoming). Available from: http://www.intelligence.tuc.gr/~manolis/publications.html.
21. M. Koubarakis. Textual Information Dissemination in Distributed Event-Based Systems. Proceedings of the International Workshop on Distributed Event-Based systems (DEBS'02), July 2-3, 2002, Vienna, Austria. Available from: http://www.intelligence.tuc.gr/~manolis/publications.html.
22. D. R. Kuokka and L. P. Harada. Issues and extensions for information matchmaking protocols. *International Journal of Cooperative Information Systems*, 5(2-3):251–274, 1996.
23. C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.
24. P. Marrow, M. Koubarakis, R.H. van Lengen, F. Valverde-Albacete, E. Bonsma, J. Cid-Suerio, A.R. Figueiras-Vidal, A. Gallardo-Antolin, C. Hoile, T. Koutris, H. Molina-Bulla, A. Navia-Vazquez, P. Raftopoulou, N. Skarmeas, C. Tryfonopoulos, F. Wang, and C. Xiruhaki. Agents in Decentralised Information Ecosystems: The DIET Approach. In M. Schroeder and K. Stathis, editors, *Proceedings of the AISB'01 Symposium on Information Agents for Electronic Commerce, AISB'01 Convention*, pages 109–117, University of York, United Kingdom, March 2001.
25. C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
26. J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/subscribe systems. In *Proceedings of COOPIS-2000*, 2000.
27. M.F. Porter. An Algorithm for Suffix Striping. *Program*, 14(3):130–137, 1980.
28. K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic Service Matchmaking Among Agents in Open Information Environments. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(1):47–53, 1999.
29. K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5:173–203, 2002.
30. F. Wang. Self-organising Communities Formed by Middle Agents. In *Proceedings of the 1st International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2002)*, July 15–19 2002.
31. I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kauffman Publishing, San Francisco, 2nd edition, 1999.
32. T.W. Yan and H. Garcia-Molina. Distributed selective dissemination of information. In *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems (PDIS)*, pages 89–98, 1994.
33. T.W. Yan and H. Garcia-Molina. The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4):529–565, 1999.