An FCA classification of durations of time for textual databases

Ulrik Sandborg-Petersen

Department of Communication and Psychology
Kroghstræde 3, DK – 9220 Aalborg East, Denmark
ulrikp@hum.aau.dk

Abstract. Formal Concept Analysis (FCA) is useful in many applications, not least in data analysis. In this paper, we apply the FCA approach to the problem of classifying sets of sets of durations of time, for the purposes of storing them in a database. The database system in question is, in fact, an object-oriented text database system, in which all objects are seen as arbitrary sets of integers. These sets need to be classified in textually relevant ways in order to speed up search. We present an FCA classification of these sets of sets of durations, based on linguistically motivated criteria, and show how its results can be applied to a text database system.

1 Introduction

Formal Concept Analysis (FCA)[1, 2] has many applications, not least of which is aiding a human analyst in making sense of large or otherwise incomprehensible data sets. In this paper, we present an application of FCA to the problem of classifying classes of linguistic objects that meet certain linguistically motivated criteria, with the purpose of storing them in a text database system.

We have developed a text database system, called Emdros[1], capable of storing and retrieving not only text, but also *annotations* of that text [3, 4]. Emdros implements the EMdF model, in which all textual objects are seen as sets of sets of durations of time with certain attributes.

The rest of the paper is laid out as follows. In Sect. 2, I describe four properties of language as it relates to time. In Sect. 3, I describe the EMdF model. In Sect. 4, I mathematically define a set of criteria which may or may not hold for a given object type. This results in a Formal Context of possible classes of objects, having or not having these criteria. In Sect. 5, I use FCA to arrive at a set of criteria which should be used as indexing mechanisms in Emdros in order to speed up search. In Sect. 6, I discuss the implementation of the criteria arrived at in the previous section, and evaluate the performance gains obtained by using them. Finally, I conclude the paper and give pointers to further research.

[1] http://emdros.org

2 Language as durations of time

Language is always heard or read in time. That is, it is a basic human condition that whenever we wish to communicate in verbal language, it takes time for us to decode the message. A word, for example, may be seen as a duration of time during which a linguistic event occurs, viz., a word is heard or read. This takes time to occur, and thus a message or text occurs in time.

In this section, we describe four properties of language which have consequences for how we may model linguistic objects such as words or sentences.

First, given that words occur in time, and given that words rarely stand alone, but are structured into sentences, and given that sentences are (at one level of analysis) sequences of words, it appears obvious that *sequence* is a basic property of language. We will therefore not comment further on this property of language.

Second, language always carries some level of structure; for example, the total duration of time which a message fills may be broken down into shorter durations which map to words. Intermediate between the word-level and the message-level, we usually find sentences, clauses, and phrases. Thus, linguistic units *embed* within each other. For a lucid discussion of the linguistic terms involved, please see [5, 6].

Third, language carries the property of being *resumptive*. By this we mean that linguistic units are not always contiguous, i.e., they may occupy multiple, disjoint durations of time. For one such opinion, see [7].

A fourth important property of linguistic units is that they may "violate each other's borders." By this we mean that, while unit A may start at time a and end at time c, unit B may start at time b and end at time d, where $a < b < c < d$. Thus, while A overlaps with B, they cannot be placed into a strict hierarchy.

3 The EMdF model

In his PhD thesis from 1994 [8], Crist-Jan Doedens formulated a model of text which meets the four criteria outlined in the previous section. Doedens called his model the "Monads dot Features" (MdF) model. We have taken Doedens' MdF model and extended it in various ways, thus arriving at the Extended MdF (EMdF) model. In this section, we describe the EMdF model.

Central to the EMdF model is the notion that textual units (such as books, paragraphs, sentences, and even words) can be viewed as *sets of monads*. A monad *is* simply an integer, but may be viewed as an indivisible duration of time.[2]

Objects in the EMdF model are pairs (M, F) where M is a set of monads, and F is a set of pairs (f_i, v_i) where f_i is the i^{th} *feature* (or attribute), and v_i is the value of f_i for this particular object. A special feature, "self" is always

[2] Please note that we use the term "monad", *not* in the well-established algebraic sense, but as a synonym for "integer in the context of the EMdF model, meaning an indivisible duration of time".

present in any F belonging to any object, and provides an integer ID which is unique across the whole database. The inequality $M \neq \emptyset$ holds for all objects in an EMdF database.

Since textual objects can often be classified into similar kinds of objects with the same attributes (such as words, paragraphs, sections, etc.), the EMdF model provides *object types* for grouping objects.

4 Criteria

In this section, we introduce some linguistically motivated criteria that may or may not hold for the objects of a given object type T. This will be done with reference to the properties inherent in language as described in Sect. 2.

In the following, let $\mathrm{Inst}(T)$ denote the set of objects of a given object type T. Let a and b denote objects of a given object type. Let μ denote a function which, given an object, produces the set of monads M being the first part of the pair (M, F) for that object. Let m denote a monad. Let $f(a)$ denote $\mu(a)$'s first (i.e., lowest) monad, and let $l(a)$ denote $\mu(a)$'s last (i.e., highest) monad. Let $[m_1 : m_2]$ denote the set of monads consisting of all the monads from m_1 to m_2, both inclusive.

Range types:

single monad(T): means that all objects are precisely 1 monad long.
$\forall a \in \mathrm{Inst}(T) : f(a) = l(a)$

single range(T): means that all objects have no gaps (i.e., the set of monads constituting each object is a contiguous stretch of monads).
$\forall a \in \mathrm{Inst}(T) : \forall m \in [f(a) : l(a)] : m \in \mu(a)$

multiple range(T): is the negation of "single range(T)", meaning that there exists at least one object in $\mathrm{Inst}(T)$ whose set of monads is discontiguous. Notice that the requirement is not that all objects be discontiguous; only that there exists at least one which is discontiguous.
$\exists a \in \mathrm{Inst}(T) : \exists m \in [f(a) : l(a)] : m \notin \mu(a)$
$\equiv \neg(\forall a \in \mathrm{Inst}(T) : \forall m \in [f(a) : l(a)] : m \in \mu(a))$
$\equiv \neg(\text{single range(T)})$

Uniqueness constraints:

unique first monad(T): means that no two objects share the same starting monad.
$\forall a, b \in \mathrm{Inst}(T) : a \neq b \leftrightarrow f(a) \neq f(b)$
$\equiv \forall a, b \in \mathrm{Inst}(T) : f(a) = f(b) \leftrightarrow a = b$

unique last monad(T): means that no two objects share the same ending monad.
$\forall a, b \in \mathrm{Inst}(T) : a \neq b \leftrightarrow l(a) \neq l(b)$
$\equiv \forall a, b \in \mathrm{Inst}(T) : l(a) = l(b) \leftrightarrow a = b$

Notice that the two need not hold at the same time.

Table 1. All the possible classes of object types. Legend: sm = single monad, sr = single range, mr = multiple range, ufm = unique first monad, ulm = unique last monad, ds = distinct, ol = overlapping, vb = violates borders.

Class name	sm	sr	mr	ufm	ulm	ds	ol	vb
1.000	X	X					X	
1.300	X	X		X	X	X		
2.000		X					X	
2.001		X					X	X
2.100		X			X		X	
2.101		X			X		X	X
2.200		X		X			X	
2.201		X		X			X	X
2.300		X		X	X		X	
2.301		X		X	X		X	X
2.310		X		X	X	X		

Class name	sm	sr	mr	ufm	ulm	ds	ol	vb
3.000			X				X	
3.001			X				X	X
3.100			X		X		X	
3.101			X		X		X	X
3.200			X	X			X	
3.201			X	X			X	X
3.300			X	X	X		X	
3.301			X	X	X		X	X
3.310			X	X	X	X		

Linguistic properties:

distinct(T): means that all pairs of objects have no monads in common.
$\forall a, b \in \text{Inst}(T) : a \neq b \rightarrow \mu(a) \cap \mu(b) = \emptyset$
$\equiv \forall a, b \in \text{Inst}(T) : \mu(a) \cap \mu(b) \neq \emptyset \rightarrow a = b$

overlapping(T): is the negation of distinct(T).
$\neg(\text{distinct}(T))$
$\equiv \exists a, b \in \text{Inst}(T) : a \neq b \wedge \mu(a) \cap \mu(b) \neq \emptyset$

violates borders(T): $\exists a, b \in \text{Inst}(T) : a \neq b \wedge \mu(a) \cap \mu(b) \neq \emptyset \wedge ((f(a) < f(b)) \wedge (l(a) \geq f(b)) \wedge (l(a) < l(b)))$

Notice that violates borders(T) $\rightarrow$ overlapping(T), since violates borders(T) *is* overlapping(T), with an extra, conjoined term.

It is possible to derive the precise set of possible classes of objects, based on logical analysis of the criteria presented in this section. For details, please see [9]. The possible classes are listed in Table 1.

The context resulting from these tables is then processed by the Concept Explorer software (ConExp)[3]. This produces a lattice as in Fig. 1.

5 Application

It is immediately noticeable from looking at Fig. 1 that "ds" is quite far down the lattice, with several parents in the lattice. It is also noticeable that "ol" is quite far up in the lattice, with only the top node as its parent. Therefore, "ds" may not be as good a candidate for a criterion on which to index as "ol". Hence, we decided to experiment with the lattice by removing the "ds" attribute.

[3] See http://conexp.sourceforge.net. Also see [10].

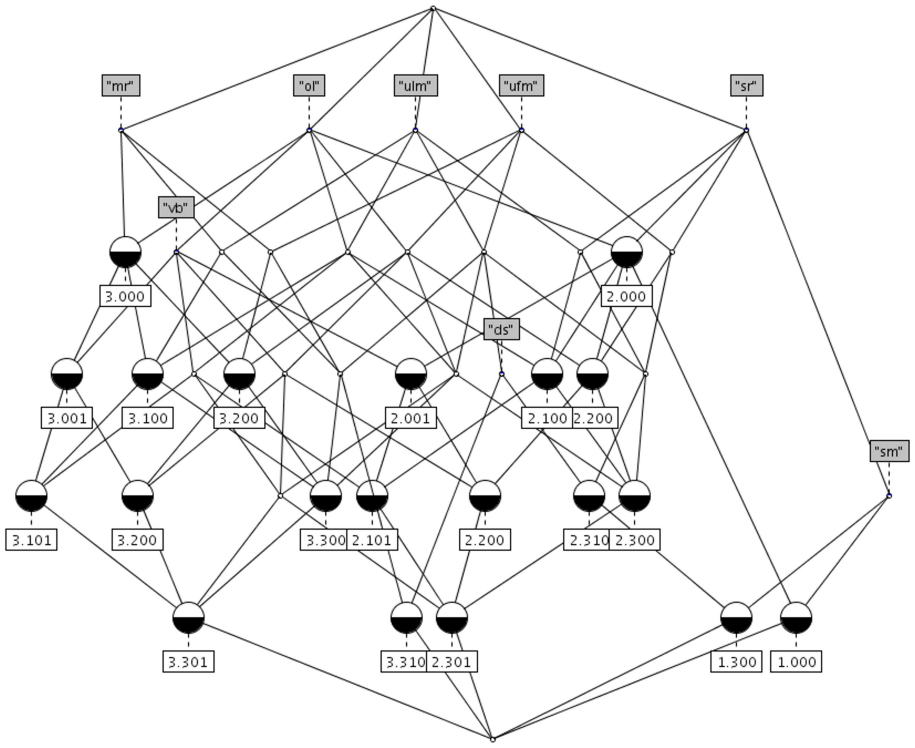

Fig. 1. The lattice drawn by ConExp for the whole context.

By drawing this new lattice with ConExp, it is noticeable that the only dependent attributes are "sm" and "vb": All other attributes are at the very top of the lattice, with only the top node as their parent. This means we are getting closer to a set of criteria based on which to index sets of monads.

The three range types should definitely be accommodated in any indexing scheme. The reasons are: First, "single monad" can be stored very efficiently, namely just by storing the single monad in the monad set. Second, "single range" is also very easy to store: It is sufficient to store the first and the last monad. Third, "multiple range", as we have argued in Sect. 2, is necessary to support in order to be able to store resumptive (discontiguous) linguistic units. It can be stored by storing the monad set itself in marshalled form, perhaps along with the first and last monads.

This leaves us with the following criteria: "unique first monad", "unique last monad", "overlapping", and "violates borders" to decide upon.

In real-life linguistic databases, "unique first monads" and "unique last monads" are equally likely to be true of any given object type, in the sense that if one is true, then the other is likely also to be true, while if one is false, then the other is likely also to be false. This is because of the embedding nature of

language explained in Sect. 2: If embedding occurs at all within a single object type, then it is likely that both first and last monads are not going to be unique.

Therefore, we decided to see what happens to the lattice if we remove one of the two uniqueness criteria from the list of attributes. The criterion chosen for removal was "unique last monads". Once this is done, ConExp reports that "unique first monads" subsumes 11 objects, or 55%. This means that "unique first monads" should probably be included in the set of criteria on which to index.

Similarly, still removing "ds" and "ulm", and selecting "overlapping", we get the lattice drawn in Fig. 2. ConExp reports that "overlapping" subsumes 17 objects, or 85%, leaving only 3 objects out of 20 not subsumed by "overlapping". This indicates that "overlapping" is probably too general to be a good candidate for treating specially.

It is also noticeable that "violates borders" only subsumes 4 objects. Hence it may not be such a good candidate for a criterion to handle specially, since it is too specific in its scope.

Thus, we arrive at the following list of criteria to handle specially in the database: a) single monad; b) single range; c) multiple range; and d) unique first monads.

6 Implementation and evaluation

The three range types can be easily implemented in a relational database system along the lines outlined in the previous section.

The "unique first monads" criterion can be implemented in a relational database system by a "unique" constraint on the "first monad" column of a table holding the objects of a given object type. Notice that for multiple range, if we store the first monad of the monad set in a separate column from the monad set itself, this is possible for all three range types. Notice also that, if we use one row to store each object, the "first monad" column can be used as a primary key if "unique first monads" holds for the object type.

We have run some evaluation tests of 124 diverse Emdros queries against two versions of the same linguistic database, each loaded into four backends (SQLite 3, SQLite 2, PostgreSQL, and MySQL). One version of the database did not have the indexing optimizations arrived at in the previous section, whereas the other version of the database did. The version of Emdros used was 3.0.1. The hardware was a PC with an Intel Dual Core 2, 2.4GHz CPU, 7200RPM SATA-II disks, and 3GB of RAM, running Fedora Core Linux 8. The 124 queries were run twice on each database, and an average obtained by dividing by 2 the sum of the "wall time" (i.e., real time) used for all 2×124 queries. The results can be seen in Table 2.

As can be seen, the gain obtained for MySQL and PostgreSQL is almost negligible, while it is significant for the two versions of SQLite.

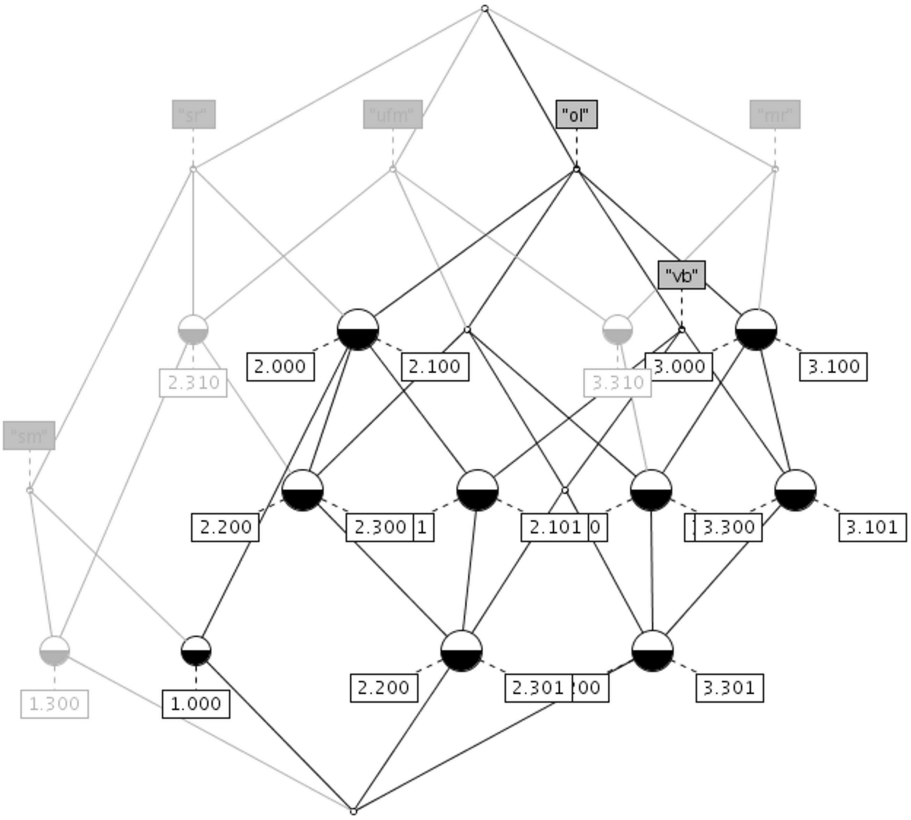

Fig. 2. The lattice drawn without the "ds" and "ulm" attributes, and with "ol" selected.

7 Conclusion

We have presented four properties that natural language possesses, namely sequence, embedding, resumption, and non-hierarchic overlap, and we have seen how these properties can be modeled as sets of durations of time.

We have presented the EMdF model of text, in which indivisible units of time (heard or read) are represented by integers, called "monads". Textual units are then seen as objects, represented by pairs (M, F), where M is a set of monads, and F is a set of attribute-value assignments. An object type then gathers all objects with like attributes.

We have then presented some criteria which are derived from some of the four properties of language outlined above. We have formally defined these in terms of objects and their monads. We have then derived an FCA context from these criteria, which we have then converted to a lattice using the Concept Explorer Software (ConExp).

Table 2. Evaluation results on an Emdros database, in seconds.

Backend	SQLite 3	SQLite 2	PostgreSQL	MySQL
Avg. time for DB **without** optimizations	153.92	130.99	281.56	139.41
Avg. time for DB **with** optimizations	132.40	120.00	274.20	136.65
Performace gain	13.98%	8.39%	2.61%	1.98%

We have then analyzed the lattice, and have arrived at four criteria which should be treated specially in an implementation.

We have then suggested how these four criteria can be implemented in a relational database system. They are, in fact, implemented in ways similar to these suggestions in the Emdros corpus query system. We have also evaluated the performance gains obtained by implementing the four criteria.

Thus FCA has been used as a tool for reasoned selection of a number of criteria which should be treated specially in an implementation of a database system for annotated text.

Future work could also include: a) Derivation of more, pertinent criteria from the four properties of language; b) Exploration of these criteria using FCA; c) Implementation of such criteria; and d) Evaluation of any performance gains.

References

1. Lehmann, F., Wille, R.: A triadic approach to formal concept analysis. In Ellis, G., Levinson, R., Rich, W., Sowa, J.F., eds.: Proceedings of ICCS'95. Volume 954 of LNAI., Springer Verlag (1995) 32–43
2. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1997) Translator-C. Franzke.
3. Petersen, U.: Emdros — a text database engine for analyzed or annotated text. In: Proceedings of COLING 2004. (2004) 1190–1193 http://emdros.org/petersen-emdros-COLING-2004.pdf.
4. Petersen, U.: Principles, implementation strategies, and evaluation of a corpus query system. In: Proceedings of the FSMNLP 2005. Volume 4002 of LNAI., Springer Verlag (2006)
5. Van Valin, Jr., R.D.: An introduction to Syntax. Cambridge University Press, Cambridge, U.K. (2001)
6. Horrocks, G.: Generative Grammar. Longman, London and New York (1987)
7. McCawley, J.D.: Parentheticals and discontinuous constituent structure. Linguistic Inquiry **13**(1) (1982) 91–106
8. Doedens, C.J.: Text Databases: One Database Model and Several Retrieval Languages. Editions Rodopi Amsterdam (1994) ISBN 90-5183-729-1.
9. Sandborg-Petersen, U.: Annotated Text Databases in the Context of the Kaj Munk Corpus: One database model, one query language, and several applications. PhD thesis, Aalborg University, Denmark (2008)
10. Yevtushenko, S.A.: System of data analysis "concept explorer". (in russian). In: Proceedings of the 7th national conference on Artificial Intelligence KII-2000, Russia. (2000) 127–134