Published in: Bański, Piotr; Biber, Hanno; Breiteneder, Evelyn; Kupietz, Marc; Lüngen, Harald; Witt, Andreas (eds.) (2015): Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora (CMLC-3). Mannheim: Institut für Deutsche Sprache, pp. 21-27

Ziggurat: A new data model and indexing format for large annotated text corpora

Stefan Evert
Friedrich-Alexander-Universität Erlangen-Nürnberg
stefan.evert@fau.de

Andrew Hardie
Lancaster University
a.hardie@lancaster.ac.uk

Abstract

The IMS Open Corpus Workbench (CWB) software currently uses a simple tabular data model with proven limitations. We outline and justify the need for a new data model to underlie the next major version of CWB. This data model, dubbed *Ziggurat*, defines a series of types of *data layer* to represent different structures and relations within an annotated corpus; each such layer may contain variables of different types. Ziggurat will allow us to gradually extend and enhance CWB's existing CQP-syntax for corpus queries, and also make possible more radical departures relative not only to the current version of CWB but also to other contemporary corpus-analysis software.

1 Introduction

With recent technological advances, it has become possible – and increasingly practical – to compile huge corpora (of 10 billion tokens and more) with complex linguistic annotation (token-level annotation such as part-of-speech tags, lemmatization, semantic tags; logical and typographical text markup encoded by XML tags; phrase structure trees; syntactic dependency graphs; coreference chains; …) and rich metadata (at text, paragraph or speaker level). At the same time, emerging international standards have begun to account for such richly annotated corpora – defining data models and serialization formats, as in the Linguistic Annotation Framework (LAF, ISO 24612: Ide & Suderman 2014); as well as different levels of query languages for complex linguistic annotations, as in the Corpus Query Lingua Franca (CQLF, ISO/CD 24623-1). Defined in a (currently draft) ISO standard, the CQLF metamodel distinguishes three levels of analysis, which correspond to linguistic annotations of different complexity:

- Level 1: plain-text search and token-level annotations
- Level 2: hierarchical structures and dependency graphs
- Level 3: multiple concurrent annotations

The current generation of software tools for querying large corpora – such as the IMS Open Corpus Workbench (CWB: Evert & Hardie 2011), Manatee/SketchEngine (Rychlý 2007) and Poliqarp (Janus & Przepiórkowski 2007) – are still based on a simple tabular data model that corresponds to CQLF Level 1 and was developed in the 1990s (Witten et al. 1999). This data model represents a text corpus as a sequence of tokens annotated with linguistic features coded as string values. It is equivalent to a data table where rows correspond to tokens and columns to the different annotations – similar to a relational database table, but with an inherent ordering of the rows.

This tabular data model was applied to linguistic corpus indexing by the first release of CWB (Christ 1994). CWB also extended the basic text-indexing structure outlined by authors such as Witten et al., by adding special provisions for simple structural annotation and sentence alignment. These were stored in the form of token ranges (pairs of integer corpus positions). The approach pioneered by the early versions of CWB was later embraced by many other software packages, including those cited above. The current release of CWB and its *Corpus Query Processor* (CQP), that is version 3, is widely used, especially through the user-friendly, browser-based CQPweb interface (Hardie 2012);

it still builds on the same data model and maintains full backwards compatibility. Though the data model has no “official” name, we will refer to it in this paper as the *CWB3 data model*. As lead maintainers and developers of CWB – now an open-source project – we have become increasingly acutely aware of a number of limitations in CWB’s basic design. In addition to its simplistic data model, CWB3 is limited to corpora of at most 2.1 billion tokens, because it stores token positions as signed 32-bit. This design decision, while perhaps justifiable in the early 1990s, no longer makes real sense (as explained in Evert & Hardie 2011).

A number of indexing and query tools do in fact go beyond a data model parallel to CWB3, and can thus support more complex linguistic annotation. Examples include TIGERSearch, (Lezius 2002), ANNIS (Zeldes et al. 2009), and ICECUP (Quinn & Porter 1994). However, such software is usually designed for small, manually annotated data sets and fails to scale up to billion-word corpora harvested from the Web or other sources. This tendency is well-exemplified by ICECUP, which is distributed alongside the corpora it is intended to be used with, namely ICE-GB and the Diachronic Corpus of Present-Day spoken English (DCPSE), densely-annotated corpora on the order of one million tokens in extent.

There is an urgent need, therefore, for efficient corpus query tools that go beyond the limitations of the CWB3 data model, providing compact storage and efficient search over complex linguistic structures. The work of the CWB development team over the past two years has turned to the development of a new data model that can support complex annotation, and can do so at scale.

2 Introducing Ziggurat

We present a novel data model, and associated indexing format, which will underlie the next major version of CWB (version 4). Rather than refer to this as the “CWB4” model, we propose the name *Ziggurat* for the data model, the file format, and the database engine software that implements them. The name is inspired by the shape of the data model, which – as the remainder of this paper will illustrate – consists conceptually of a pile of rectangular layers on top of one another.

The design goals of Ziggurat are that it should (i) scale to corpora of arbitrary size; (ii) support rich linguistic annotation, in particular XML hierarchies, phrase-structure trees, dependency graphs and parallel-corpus alignment; and (iii) provide efficient indexed access to the data, enabling complex linguistic queries in reasonable time. In the long term, by defining the Ziggurat engine as a conceptually-separate entity to the CWB software and the query language that it provides (known as *CQP-syntax*), our aim is to be able to use Ziggurat as the underpinning for more than one (kind of) query language. Towards the end of this paper, we will speculate on the new types of query languages that the enriched data model supported by Ziggurat will enable. Let us first, however, survey some related work, justifying the need for a new database engine.

3 Related work and motivation

In recent years, researchers have explored several alternative approaches to efficient queries for large text corpora:

- A standard relational database with redundant representation of the corpus (e.g. n-gram tables), a large number of indexes and fine-tuning of the database server and SQL queries (as outlined by Davies 2005, although Davies’ current architecture[1] is much-revised from this now somewhat outdated outline). It is unclear whether this approach can be generalized to more complex linguistic data structures and sophisticated query needs.
- A native XML or graph database used off-the-shelf, with built-in indexing and query facilities. Mayo et al. (2006) show that this approach is inefficient using XML databases; Proisl & Uhrig (2012) make the same observation for a popular graph database.
- An information retrieval or Web search engine such as Lucene, with custom modifications to support linguistic annotation and the kinds of query patterns supported by CQP-syntax. A recent example of this approach is the BlackLab[2] software. While it is difficult to assess the potential of the system due to a lack of scientific publications, a small number of blog posts about its internals suggest that it may be very

[1] Accessible at `http://corpus.byu.edu`
[2] `https://github.com/INL/BlackLab`

difficult to extend BlackLab to full tree structures and dependency graphs.

- *Corpuscle* (Meurer 2012) proposes new indexing structures based on suffix trees in order to optimise the performance of regular expressions and CQP-syntax queries. Having a focus on indexing and query algorithms, it does not attempt to go beyond the tabular CWB3 data model.

Despite introducing various innovations, none of these approaches has resorted to a ground-up rethink of the data model: all attempt to extend some existing data model. While such efforts have had notable short-term successes, we believe that ultimately they are self-limiting, for the reasons discussed above. We are convinced that it is necessary to go beyond the CWB3 data model; however, we are likewise convinced that working around other standard data models, whether those of XML databases or web-query engines, is not the best way to do it, especially for a community-driven effort with limited resources. This motivates our proposal of Ziggurat.

Ziggurat *does* represent a ground-up rethink of the CWB3 data model, keeping its basic idea – a tabular data model with implicitly-ordered rows representing sequence positions – but extending it considerably, and like CWB3 using custom index structures and file formats. We believe that this offers better support for the highly successful brute-force corpus search of CWB and similar query tools than a standard off-the-shelf backend such as a SQL RDBMS or Web search engine. Recognizing that it is better to have a simple but flexible tool that is available, well-maintained and actively developed by its user community than to design the "Perl 6" of corpus query engines – that is, a perfect redesign which remains unreleased and unavailable to most users for years on end – we resolved to keep the data model, index structures and file formats as simple and straightforward as possible. Thus, the entire Ziggurat data model builds on a small set of easily implemented data structures.

Further key requirements for the new data model are (i) full Unicode support, (ii) (nigh-)unlimited corpus size, (iii) logical backward compatibility with the CWB3 data model, (iv) full support for hierarchical XML annotation and other tree structures, (v) representation of dependency graphs, (vi) support for sentence (and preferably also word) alignment, and (vii) concurrent annotation layers forming independent or intersecting hierarchies. The Ziggurat data model thus encompasses all three levels of the CQLF metamodel.

4 The data model

In order to ensure a compact representation, efficient access and a simple implementation of the data model, a number of limitations are accepted:

- Corpora are "horizontally" static, i.e. no modification of the tokenization, annotation units or annotated values is allowed in an indexed corpus, and documents can neither be added nor deleted. However, corpora are "vertically" flexible, i.e. individual annotated features or entire annotation layers may be added and deleted.
- Individual physical corpora cannot be collected into a single "virtual" corpus, but queries can be restricted to subsets of a large physical corpus without loss of efficiency.
- The data format is token-based, without support for full-text representation and search.

In the proposed data model, a corpus is a collection of sequential *data layers*, which are connected into one or more annotation hierarchies over the primary text data. Each data layer consists of a sequence of annotation units annotated with one or more variables (i.e. linguistic features). Thus, a data layer in Ziggurat fundamentally has the same tabular format as the annotated token sequence in a CWB3 corpus, and the established representation and indexing approaches for such data structures (similar to Witten et al. 1999) can be used. A key difference between Ziggurat and CWB3 is that all Ziggurat data layers can be annotated with variables, not just the primary token sequence. Moreover, unlike CWB3, Ziggurat will support different *types* of variables:

- *Indexed strings* = string values where all distinct strings are collected in a lexicon and associated with numeric IDs (equivalent to CWB3 token-level annotations)
- *Raw strings* = string values stored without indexing, mainly used for free-form metadata (such as URLs) or unique IDs
- *Integers* = signed 64-bit integer values (which can also be interpreted by client software as fixed-point decimals), used for storing numeric information

- *Pointers* = references to a single parent annotation unit in the same layer, which can be used to structure the sequence of annotation units into a forest of unordered trees (e.g. a simple dependency parse without multiple parents); these will be stored as integers, and thus the maximum corpus size will be the positive limit of a 64-bit signed integer (somewhat over 9.2 quintillion)
- *Hashes* = indexed key-value stores with a lexicon similar to indexed strings, useful for storing variable metadata and the attributes of XML start tags.

Structural information is conveyed by the way in which different data layers are connected. In a Ziggurat index, a basic token sequence together with all token-level annotations forms the so-called *primary annotation layer*. All other types of data layers reference one or more *base layers*. These layers can in turn act as base layers of further data layers, forming a hierarchy of annotation layers. (This is the source of the name *Ziggurat*: the multiple rectangular data layers that are built on top of one another may be visualized in a shape reminiscent of a Mesopotamian ziggurat.)

Annotations are fully concurrent, allowing multiple independent or intersecting annotation hierarchies over the primary layer. In principle, a corpus may also contain multiple *primary* layers, e.g. representing different transcriptions of the same audio signal.

Ziggurat will have the following types of data layers (see appendix for an illustration):

- *Segmentation layer*: Each unit represents an uninterrupted range of base layer units (usually the tokens of a primary layer). Different ranges may neither overlap nor be nested within each other. This layer type extends the structural attributes used to represent multi-token structures in the CWB3 data model, but more flexibly; these layers are useful for storing a simple segmentation of the corpus (into sentences, texts, files, speaker turns, …) and the associated metadata.
- *Tree layer*: Each unit also represents an uninterrupted range of base layer units, but these ranges may be nested hierarchically, forming an ordered tree over the base layer sequence. An important application of tree layers is to represent XML annotation, with each annotation unit corresponding to one XML element. Empty ranges are expressly allowed by the data model for this purpose. Tree layers can also, however, represent the tree structures of constituency-parsing.
- *Graph layer*: Each unit represents a directed edge between two annotation units in the base layer, thus forming a directed graph over the base layer, where both edges (in the tree layer) and nodes (in the base layer) may be annotated with variables. Unlike other layers, graph layers may have two different base layers for the tails and heads of the edges. A graph between two different base layers represents an alignment of the base layers: a sentence alignment if they are sentence segmentation layers, or a word alignment if they are primary layers. This type of layer thus supports both dependency-parsing annotation (with a single base layer) and parallel-corpus alignment (with two base layers: the equivalent of a CWB3 alignment-attribute).

Ziggurat data structures are designed to be as simple and uniform as possible. The only value types are strings in UTF-8 encoding and signed 64-bit integers. Indexing is based on two simple generic structures: a sort index with integer sort keys, and a postings list similar to that used by Web search engines. The Ziggurat file formats are also simplified relative to CWB3, trading off compactness for simplicity and decompression speed. CWB3 uses bit-oriented Huffman and Golomb coding schemes, as proposed by Witten et al. (1999). However, through experiments using CQP we have found that these compression methods, though maximally economical of disk space, require an excessive amount of processor time when the system is running complex queries. Ziggurat instead utilizes variable-length byte encodings (without a codebook) and delta compression. A Ziggurat-encoded corpus will therefore take up more disk space, but will require less CPU time to decompress.

5 New corpus query approaches

The Ziggurat data model's greater expressiveness relative to CWB3 will allow, and therefore ultimately call for, more sophisticated query languages than CWB3 could support. While a concrete specification is not possible at this time, we believe that the following three approaches are promising.

Approach 1 extends the CWB3-style “linear” queries based on regular expression notation, i.e. the kind of query language typified by CQP-syntax. It allows query paths to follow other axes than the token sequence (similar to XPath), in particular along the edges of a graph layer and to parents, children and siblings in a tree layer. Experience from Treebank.info (Proisl & Uhrig 2012) suggests that many linguistically plausible searches can be flattened into a single linear path; otherwise “branching” queries will be needed. This approach will be implemented in version 4 of CWB – the first application using Ziggurat. CWB version 4 will at first simply implement the existing CQP-syntax in terms of calls to the Ziggurat engine; but subsequently it will gradually extend the CQP-syntax query language over time to exploit more of the affordances of Ziggurat.

In Approach 2, a query specifies a finite set of anchor points (tokens or annotation units from a specified data layer), constraints on annotated variables, and relations between different anchors (such as co-occurrence, dominance or precedence). Similar to XQuery, this approach is used by many existing query engines for CQLF levels 2 and 3, including TIGERSearch, ANNIS (Krause & Zeldes in press) and the NXT Query Language (Evert & Voormann 2003).

Approach 3 derives from the following observation by Geoffrey Sampson:

> […] there are usually two possibilities when one wants to exploit corpus data. Often, one wants to put very obvious and simple questions to the corpus; in that case, it is usually possible to get answers via general-purpose Unix commands like grep and wc, avoiding the overhead of learning special-purpose software. Sometimes, the questions one wants to put are original and un-obvious; *in those cases, the developer of a corpus utility is unlikely to have anticipated that anyone might want to ask them, so one has to write one's own program to extract the information*. (Sampson 1998:365; our emphasis).

The most sophisticated corpus query requirements can only be satisfied by a Turing-complete query language. We therefore envisage corpus queries as programs for a virtual machine (VM) that interfaces closely with the corpus data model and index structures. High-level languages (such as JavaScript, Python or Lua) or parser generators can then be used to implement various simplified query languages with relative ease, compiling the queries written in these query languages into VM programs. This approach, then, ultimately will enable “power users” – those with an understanding of the data model and some coding ability – to write their own programs to carry out virtually every imaginable search.

By making the Ziggurat data model and database engine extremely flexible in the ways outlined above, we will establish a foundation on which any or all of these three approaches can be developed, within the same or different pieces of software.

References

Christ, Oliver (1994). A modular and flexible architecture for an integrated corpus query system. In *Papers in Computational Lexicography (COMPLEX '94)*, pages 22–32, Budapest, Hungary.

Davies, Mark (2005). The advantage of using relational databases for large corpora: Speed, advanced queries and unlimited annotation. *International Journal of Corpus Linguistics*, 10(3), 307–334.

Evert, Stefan and Hardie, Andrew (2011). Twenty-first century corpus workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics 2011 Conference*, Birmingham, UK.

Evert, Stefan and Voormann, Holger (2003). NQL – a query language for multi-modal language data. Technical report, IMS, University of Stuttgart. Version 2.1.

Hardie, Andrew (2012). CQPweb – combining power, flexibility and usability in a corpus analysis tool. *International Journal of Corpus Linguistics*, 17(3), 380–409.

Ide, Nancy and Suderman, Keith (2014). The linguistic annotation framework: a standard for annotation interchange and merging. *Language Resources and Evaluation*, 48(3), 395–418.

ISO 24612 (2012) Language resource management – linguistic annotation framework. Technical report, ISO.

ISO/CD 24623-1 (2014). Language resource management – corpus query lingua franca (CQLF) – part 1: Metamodel. Technical report, ISO.

Janus, Daniel and Przepiórkowski, Adam (2007). Poliqarp: An open source corpus indexer and search engine with syntactic extensions. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Posters and Demonstrations Sessions*, pages 85–88, Prague, Czech Republic. Association for Computational Linguistics.

Krause, Thomas and Zeldes, Amir (in press). ANNIS3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities.* Advance access.

Lezius, Wolfgang (2002). TIGERSearch – ein Suchwerkzeug für Baumbanken. In S. Busemann (ed.), Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002), Saarbrücken, Germany.

Mayo, Neil; Kilgour, Jonathan; Carletta, Jean (2006). Towards an alternative implementation of NXT's query language via XQuery. In *Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006)*, pages 27–34, Trento, Italy.

Meurer, Paul (2012). Corpuscle – a new corpus management platform for annotated corpora. In *Exloring Newspaper Language: Using the web to create and investigate a large corpus of modern Norwegian*, number 49 in Studies in Corpus Linguistics. John Benjamins.

Proisl, Thomas and Uhrig, Peter (2012). Efficient dependency graph matching with the IMS open corpus workbench. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC '12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Quinn, Akiva and Porter, Nick (1994). Investigating English usage with ICECUP. *English Today*, 10(3), 19–24.

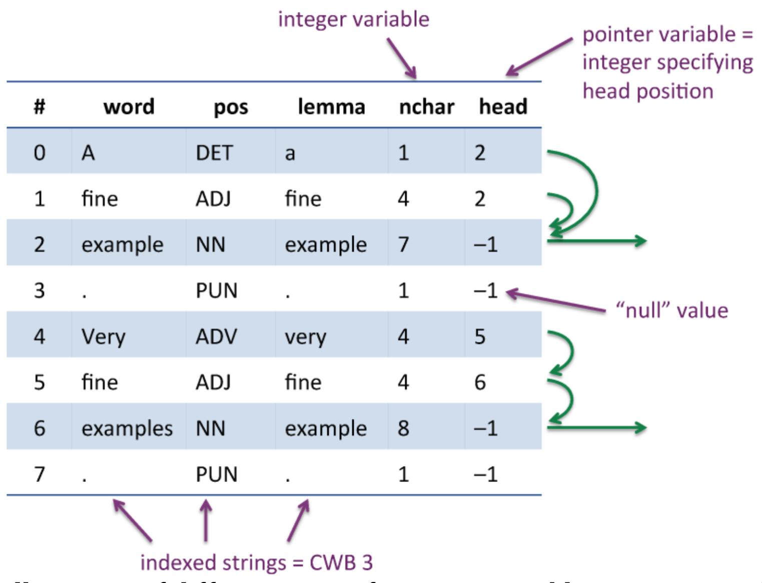
Rychlý, Pavel (2007). Manatee/Bonito - a modular corpus manager. In *Proceedings of the 1st Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 65–70, Brno. Masaryk University.

Sampson, Geoffrey (1998). Review of Sidney Greenbaum (ed.), *Comparing English worldwide: The international corpus of English*. Oxford: Clarendon Press, 1996. ISBN 0-19-823582-8, xvi+286 pages. *Natural Language Engineering*, 4, 363–382.

Witten, Ian H.; Moffat, Alistair; Bell, Timothy C. (1999). *Managing Gigabytes*. Morgan Kaufmann Publishing, San Francisco, 2nd edition.

Zeldes, Amir; Ritz, Julia; Lüdeling, Anke; Chiarcos, Christian (2009). ANNIS: A search tool for multi-layer annotated corpora. In M. Mahlberg, V. González-Díaz, and C. Smith (eds.), *Proceedings of the Corpus Linguistics 2009 Conference*, Liverpool, UK. Article #358.

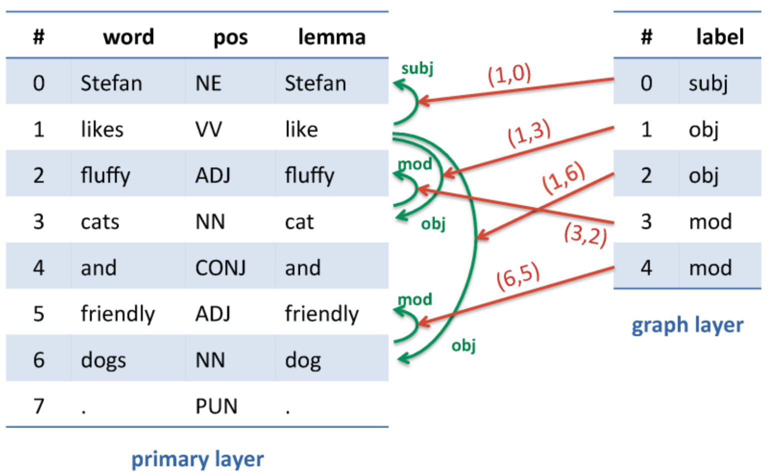
Appendix: Illustrations of different Ziggurat layer types



#	word	pos	lemma	nchar	head
0	A	DET	a	1	2
1	fine	ADJ	fine	4	2
2	example	NN	example	7	–1
3	.	PUN	.	1	–1
4	Very	ADV	very	4	5
5	fine	ADJ	fine	4	6
6	examples	NN	example	8	–1
7	.	PUN	.	1	–1

integer variable
pointer variable = integer specifying head position
"null" value
indexed strings = CWB 3

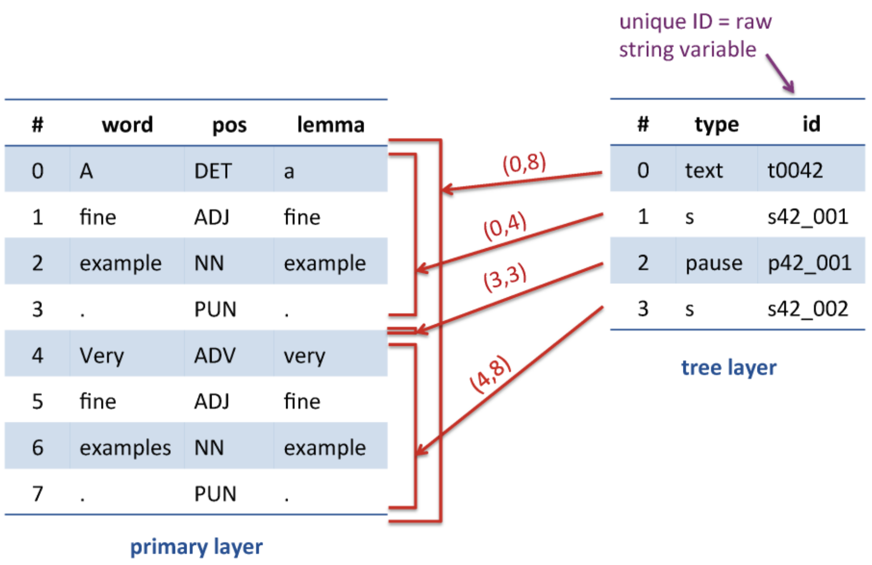
Fig 1. Illustration of different types of Ziggurat variables on a primary layer.
(Note that the simple tree structures defined by the pointer variable in the last column are less general than the graph layer in Fig. 2 and edges cannot be annotated with labels)



#	word	pos	lemma
0	Stefan	NE	Stefan
1	likes	VV	like
2	fluffy	ADJ	fluffy
3	cats	NN	cat
4	and	CONJ	and
5	friendly	ADJ	friendly
6	dogs	NN	dog
7	.	PUN	.

#	label
0	subj
1	obj
2	obj
3	mod
4	mod

Fig 2. A graph layer (representing a dependency parse) and its base layer



#	word	pos	lemma
0	A	DET	a
1	fine	ADJ	fine
2	example	NN	example
3	.	PUN	.
4	Very	ADV	very
5	fine	ADJ	fine
6	examples	NN	example
7	.	PUN	.

#	type	id
0	text	t0042
1	s	s42_001
2	pause	p42_001
3	s	s42_002

Fig 3. A tree layer (representing an XML hierarchy) and its base layer