CWB The IMS Open Corpus Workbench

Lancaster University

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
PHILOSOPHISCHE FAKULTÄT UND FACHBEREICH THEOLOGIE

Ziggurat v0.1

A next-generation system for modelling, storing, and retrieving corpus (and other) data

Stefan Evert, FAU Erlangen-Nürnberg
Andrew Hardie, Lancaster University

http://cwb.sf.net/

What is Ziggurat?

- CWB version 3 & CQPweb: limited in support for …
  - very large corpora (> 2.1 billion words)
  - XML / constituency trees
  - dependency annotation

  Codebase is 30 years old and **you can tell**!

- Many of the limiations are baked into the data model & file format, hence **Ziggurat** = new data access layer as self-contained system

- **Project Ziggurat**
  - Define data model and file formats (since about 2015–2020)
  - Define API (2021)
  - Implement Ziggurat library (now!)
  - Build CWB version 4 on top

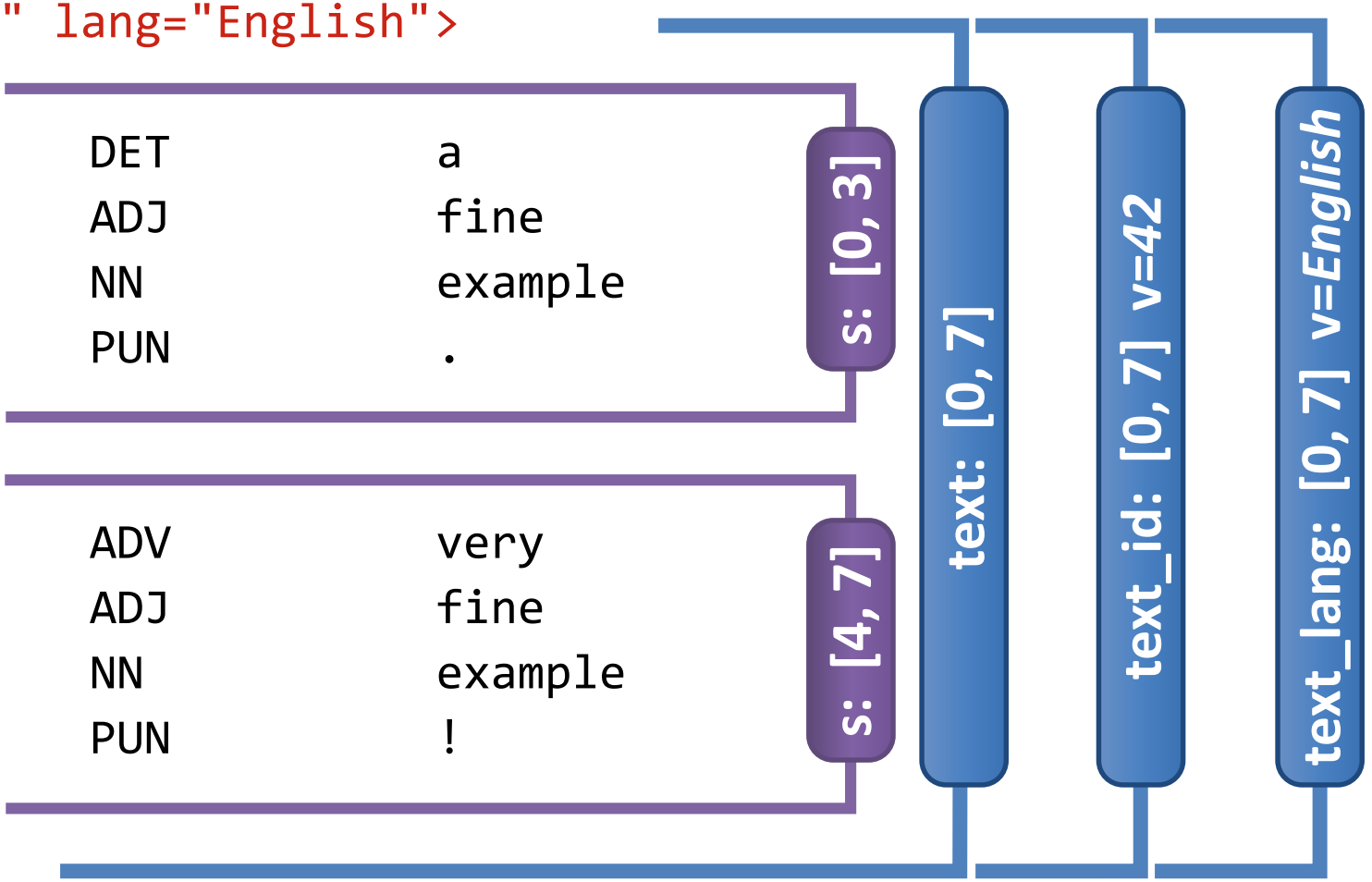
Tabular data for corpus representation

- Tabular data model of CWB version 3 has been very influential (→ SketchEngine, CoNLL, R, …)

```
<text title="The Garden" author="Stefan Evert" author_sex="male">
<p num="1">
<s>
It          PP      it
seemed      VBD     seem
a           DT      a
day         NN      day
much        RB      much
as          IN      as
any         DT      any
...
</s>
</p>
</text>
```

CWB3 data model

#	word	pos	lemma
(0)	<text id="42" lang="English">		
(0)	<s>		
0	A	DET	a
1	fine	ADJ	fine
2	example	NN	example
3	.	PUN	.
(3)	</s>		
(4)	<s>		
4	Very	ADV	very
5	fine	ADJ	fine
6	examples	NN	example
7	!	PUN	!
(7)	</s>		
(7)	</text>		



CWB3 data model

#	word		pos		lemma	
(0)	<text id="42" lang="English">					
(0)	<s>					
0	A	0	DET	0	a	0
1	fine	1	ADJ	1	fine	1
2	example	2	NN	2	example	2
3	.	3	PUN	3	.	3
(3)	</s>					
(4)	<s>					
4	Very	4	ADV	4	very	4
5	fine	1	ADJ	1	fine	1
6	examples	5	NN	2	example	2
7	!	6	PUN	3	!	5
(7)	</s>					
(7)	</text>					

no such indexing for s-attribute values

lexicon IDs for annotation strings (per column)

signed 32-bit integer

K I S S

Ziggurat data model: Layers and variables

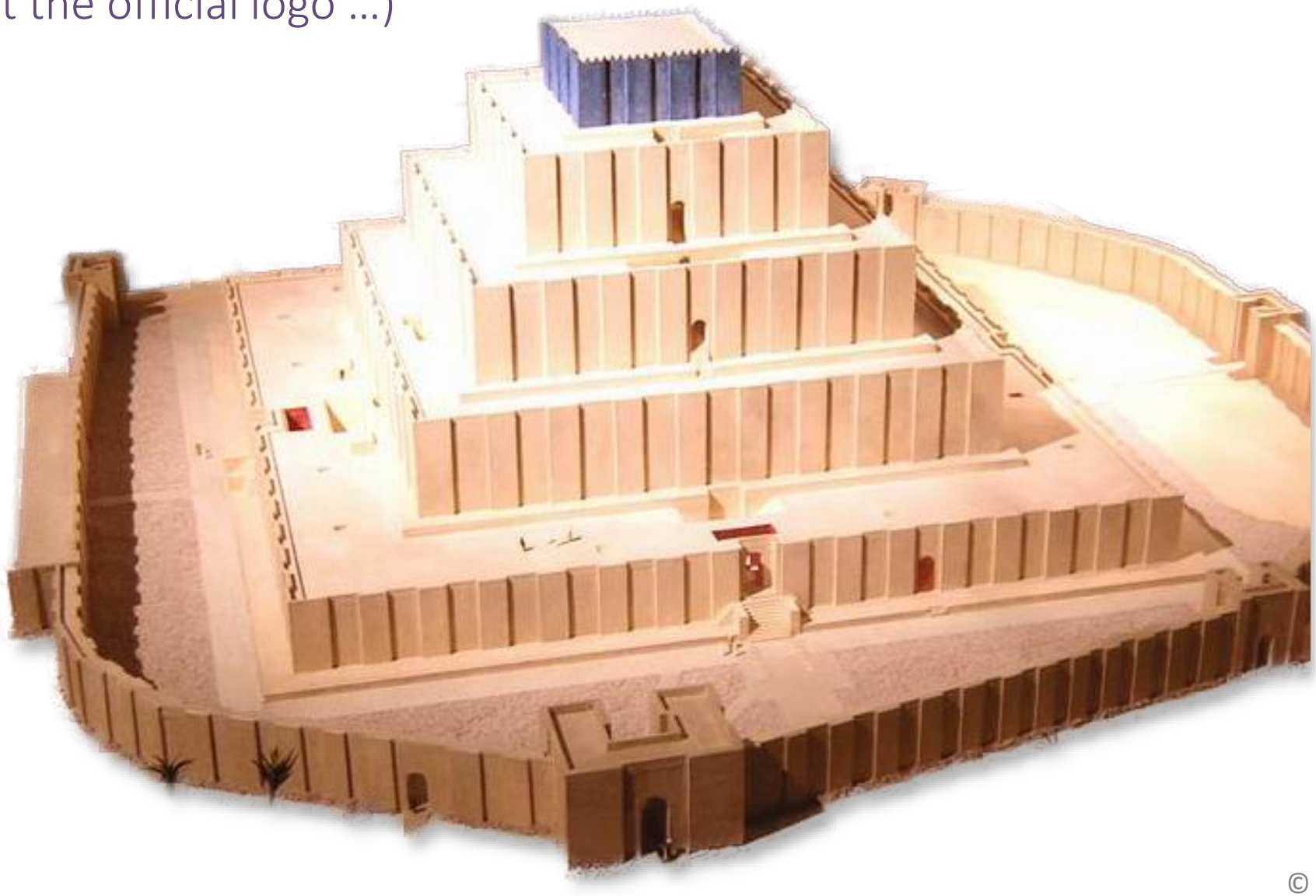
- Generalised from CWB data table idea

- **Layer** = position sequence
  - Primary = object data (usually tokens)
  - Secondary ➞ linked to another layer (usually primary, but not always)
    = structural annotation units (e.g. tree constituents, graph edges)

- **Variable** = set of values associated with a layer
  - One value per position sequence
  - Data types (string, integer, set, pointer) with different search methods

Thus, **Ziggurat**

(this is not the official logo …)



https://www.livius.org/pictures/iran/choga-zanbil/choga-zanbil-ziggurat/choga-zanbil-ziggurat-model/

Layer types

Layer Type	Contains…
Primary layer	Bare sequence of positions
Segmentation layer	A sequence of non-overlapping ranges on the base layer, with begin and end points
Tree layer	A sequence of nodes corresponding to nested ranges on the base layer with begin and end points on the base layer, plus mother/daughter/sister relationships
Graph layer	A sequence of links between source/target base layers (source and target can be the same); each item is a graph edge which points from a position on the source layer to a position on the target layer

Variable types

Variable Type	Contains…
Indexed string	A string for each position on its layer, indexed with lexicon of all unique types; for use with non-unique values
Plain string	A string for each position on its layer without lexicon; for use when values are likely unique
Integer	An integer value (whole number) for each layer position; can also be interpreted as timestamp or fixed-point value
Set	Like indexed string, but multiple values at each position allowed
Hash	Like indexed string, but at each position there is an associative array (aka. *hash* or *dictionary*) of KEY→VALUE mappings
Pointer	A pointer from each position (tail) to some other position on the layer (head), possibly NULL

Example: CoNLL-U

from https://universaldependencies.org/format.html

```
1    They      they     PRON    Case=Nom|Num=Pl                2    nsubj
2    buy       buy      VERB    Num=Pl|Pers=3|Tense=Pres       0    root
3    and       and      CONJ    _                              4    cc
4    sell      sell     VERB    Num=Pl|Pers=3|Tense=Pres       2    conj
5    books     book     NOUN    Num=Pl                         2    obj
6    .         .        PUNCT   _                              2    punct
```

Example: CoNLL-U

from https://universaldependencies.org/format.html

primary layer

#	id	form	lemma	upos	feats	deprel	head
0	1	They	they	PRON	{c:nom, n:pl}	nsubj	1
1	2	buy	buy	VERB	{n:pl, p:3, t:pres}	root	NULL
2	3	and	and	CONJ	{}	cc	3
3	4	sell	sell	VERB	{n:pl, p:3, t:pres}	conj	1
4	5	books	book	NOUN	{n:pl}	obj	1
5	6	.	.	PUNCT	{}	punct	1

plain string variable

indexed string variables

hash variable

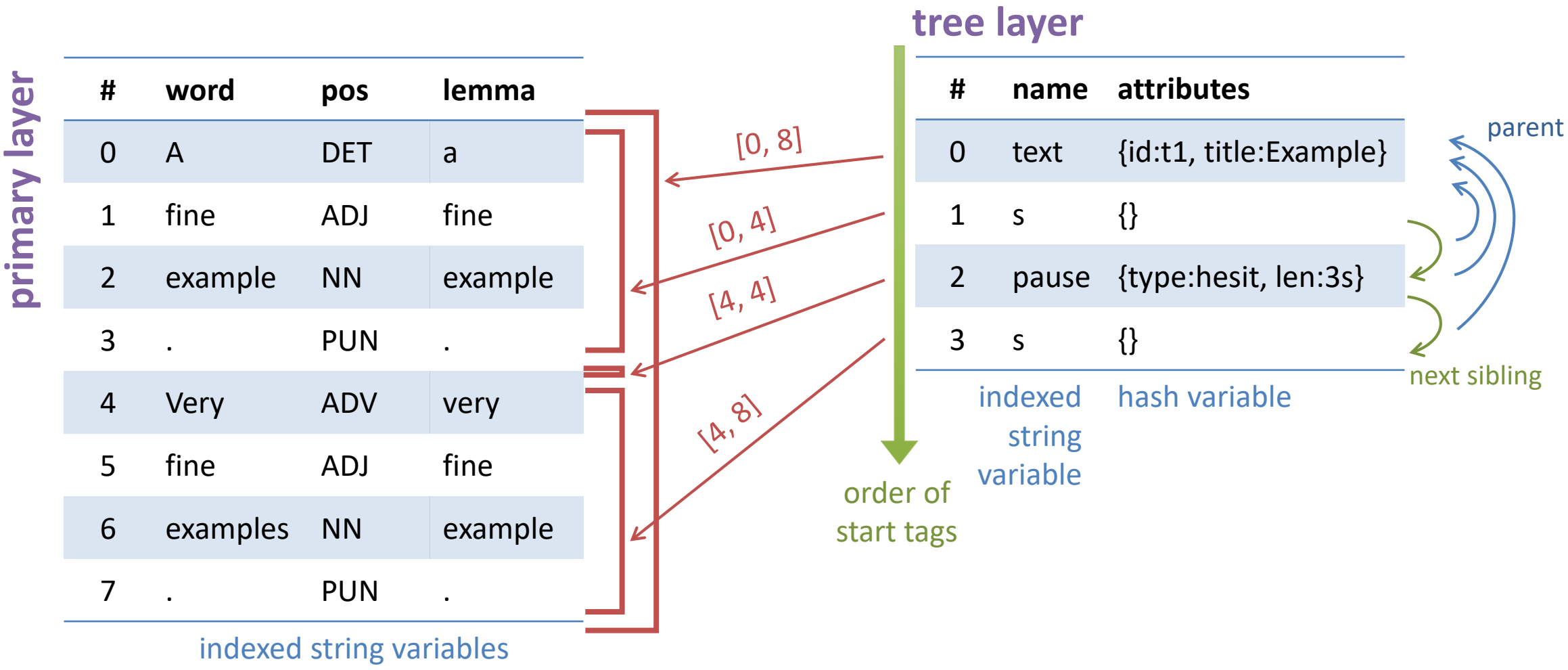
indexed string variable

pointer variable

Example: XML tree

```
<text id="t1" title="Example">
<s>
A       DET   a
fine    ADJ   fine
example NN    example
.       PUN   .
</s>
<pause type="hesit" len="3s" />
<s>
Very    ADV   very
fine    ADJ   fine
examples      NN   example
.       PUN   .
</s>
</text>
```

Example: XML tree



Example: Multiple concurrent tokenisations

It's out of this world!

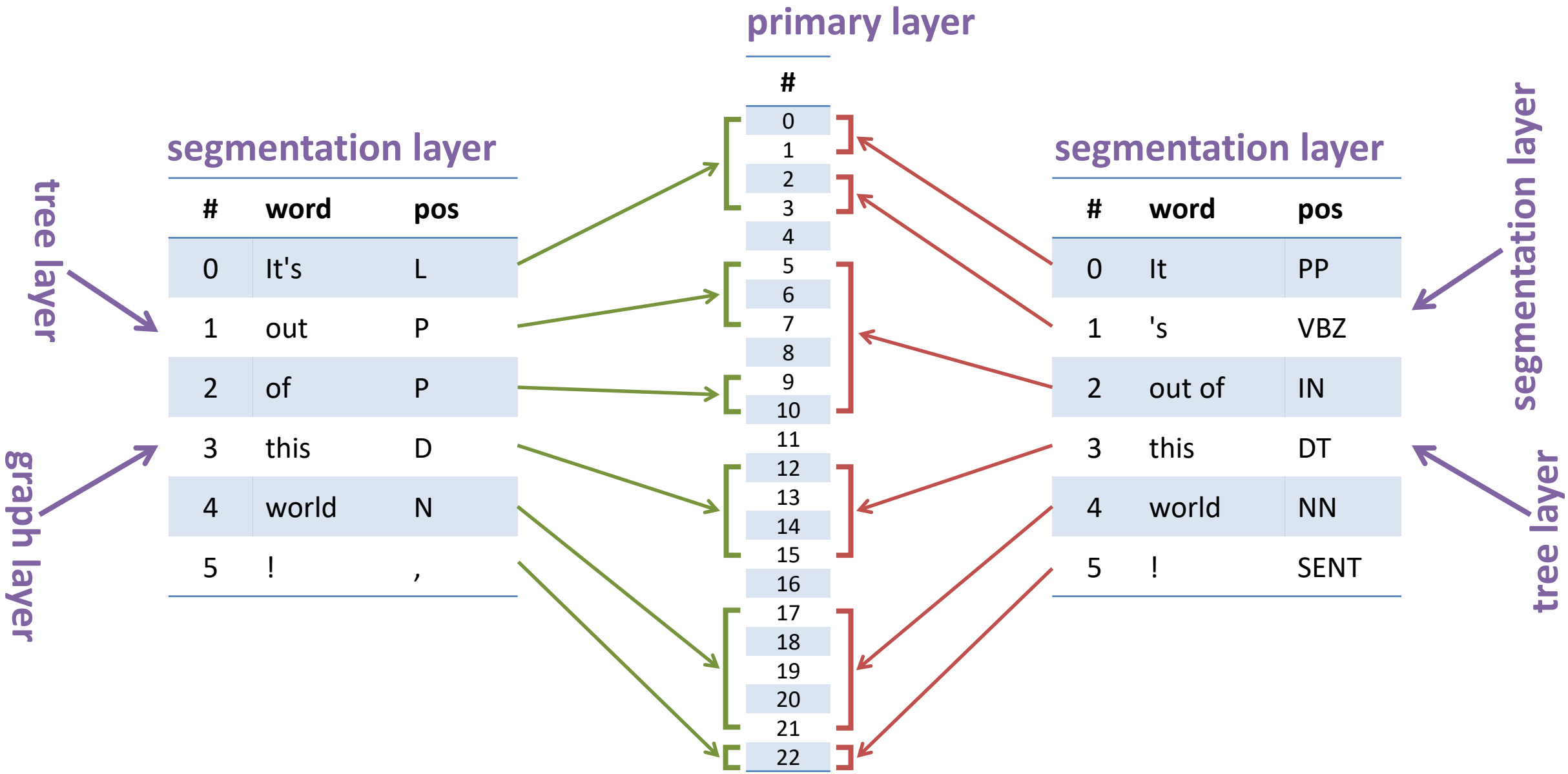
NLP pipeline A

It's	L
out	P
of	P
this	D
world	N
!	,

NLP pipeline B

It	PP
's	VBZ
out of	IN
this	DT
world	NN
!	SENT

Example: Multiple concurrent tokenisations

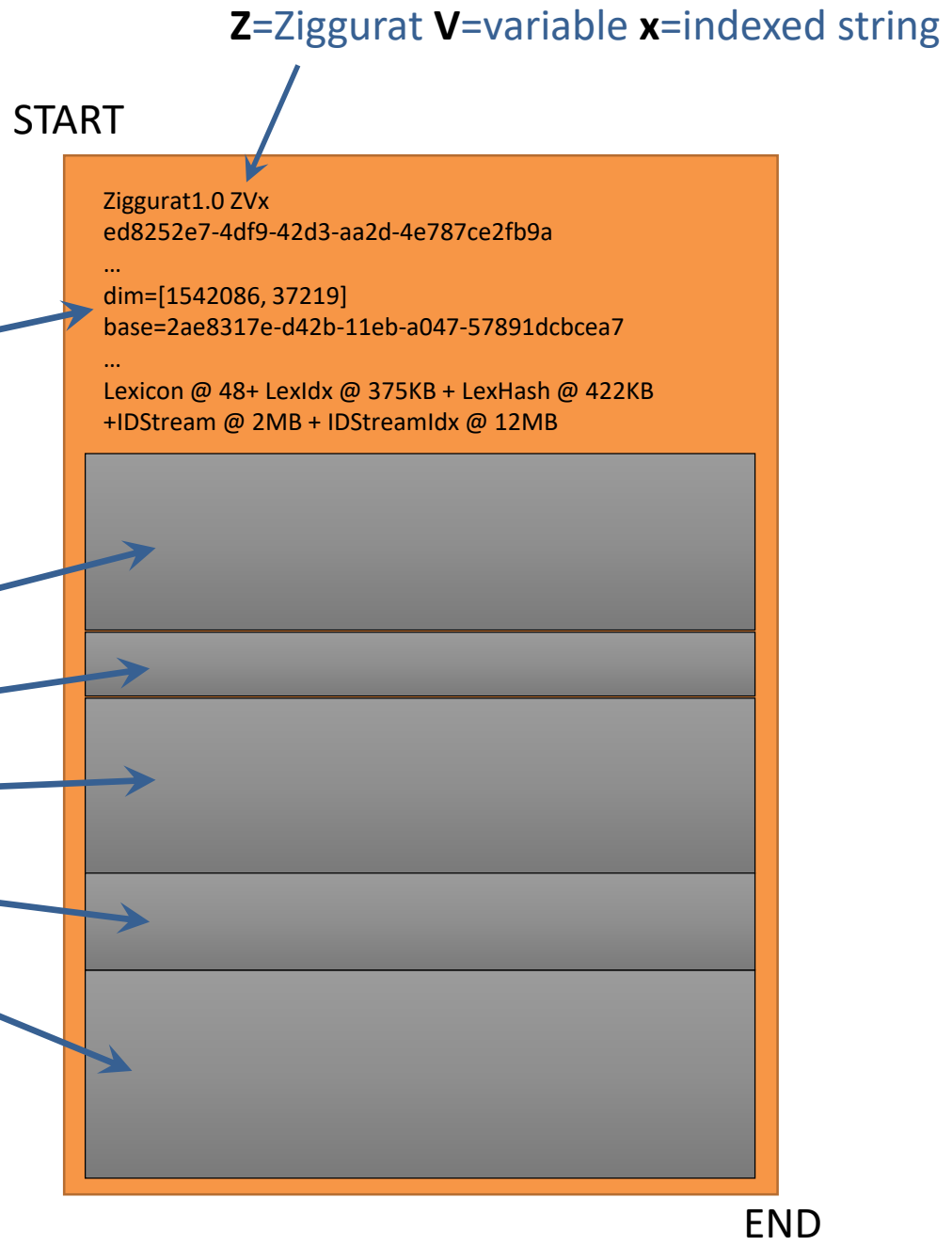


Storage

Container file = layer / variable

- Header/bill of materials
- Components
  - Lexicon of actual type strings
  - Index from Lex IDs to strings
  - Hashed index from strings to IDs
  - Sequence of Lex IDs on layer
  - Index of Lex IDs to layer positions

  (for “indexed string”)



Retrieval

- Users and applications **do not need to know about components**

- Component data for each variable type optimised for swift retrieval (but with KISS in mind → prefer simple data structures & re-use)
  - Indexed string variable: *indexes on types and lexicon IDs means all positions can be retrieved without going sequentially through token data of the corpus*
  - Hash variable: *separate type lexicons for keys and values, plus lexicon of all key-value pairs (as pairs of lexicon IDs)*
  - Integer variable: *sorted index allows efficient search of numeric range*

- All details encapsulated in Ziggurat library

Application Programming Interface

- Ziggurat as embedded library (written in C11, for reasons)
- API bindings for multiple languages (Python, R, …)
- CWB version 4
  - Corpus-management library as Ziggurat “user”
  - Other libraries as needed (CQP syntax parser / compiler, user interface)
  - Actual programs quite small and just use these libraries
  - Other applications can use Ziggurat in more flexible ways

API examples *(in Python-ish)*

Indicative only! Work-in-progress!

```
1
2   ## Some EXAMPLE Python code using Ziggurat
3
4   import os
5   import Ziggurat
6
7   # the Ziggurat class represents the Ziggurat engine.
8   # Each engine instance is "aware" of a certain set of consistent layers & variables.
9   store = new Ziggurat()
10
11  # Z throws exception on error.
12  try:
13      # we open a layer by pointing to its container file
14      store.add_object("/home/andrew/zds/pri-lay-324.zig")
15      store.add_object("/home/andrew/zds/sec-lay-324.zig")
16      store.add_object("/home/andrew/zds/word-var-324.zig")
17
18  except ZiggyException as e:
19      if Ziggurat.ERR_FILE == e.errno:
20          print("Failed to open a container file; please check specified path")
21      if Ziggurat.ERR_INCONSISTENT == e.errno:
22          print("Couldn't add specified layer/variable to Z , it is inconsistent with a known layer.")
23
```

API examples *(in Python-ish)*

Indicative only! Work-in-progress!

```
# exception checking elided for clarity from now on

print(string(store.n_objects))     # "3"

# get the variable for "word" (normal name for the actual tokens)
wordvar = store.seek_layer('primary').seek_variable_by_name('word')

if (Ziggurat.TYPE_INDXSTRING == wordvar.type):
    print("this is a string variable, let's do a regular expression query!")
else
    print("Type is: ", wordvar.type_format())
    print("We can't regular expression this, it's the wrong data type")
    os._exit()

# a query is done with methods on the variable object. What we get back are lexicon ID codes.

word_ids = wordvar.get_ids(Ziggurat.QUERY_TYPE_REGEX, "elephant.*", Ziggurat.RX_FLAG_C)

# word_ids is a list of ID codes, wrapped in an object.
print("Found ", word_ids.size, " word-types for the regex query.")

while None !=  (id = word_ids.fetch_next()):
    print("Lexicon ID # ", id, " for word form " , wordvar.id_to_str(id) , " which has frequency " , wordvar.id_to_freq(id) )
```

Conclusion

- Ziggurat is still in its very early stages
- We welcome comments and suggestions
- See working docs & updates on CWB website
  - http://cwb.sourceforge.net/cwb4.php
- We aim to have a 0.1 version for people to try out by July
  - most likely: **scratchpad protoype** written in PHP
  - will be used to work out API, test suitability of file format & access patterns, etc.
  - settled and tested parts then ported to C11 code at https://github.com/schtepf/ziggurat

https://commons.wikimedia.org/wiki/File:Tchogha_Zanbil.jpg