WikiParq: A Tabulated Wikipedia Resource Using the Parquet Format

Marcus Klang, Pierre Nugues
Lund University, Department of Computer science,
Lund, Sweden
marcus.klang@cs.lth.se, pierre.nugues@cs.lth.se

Abstract

Wikipedia has become one of the most popular resources in natural language processing and it is used in quantities of applications. However, Wikipedia requires a substantial pre-processing step before it can be used. For instance, its set of nonstandardized annotations, referred to as the wiki markup, is language-dependent and needs specific parsers from language to language, for English, French, Italian, etc. In addition, the intricacies of the different Wikipedia resources: main article text, categories, wikidata, infoboxes, scattered into the article document or in different files make it difficult to have global view of this outstanding resource. In this paper, we describe WikiParq, a unified format based on the Parquet standard to tabulate and package the Wikipedia corpora. In combination with Spark, a map-reduce computing framework, and the SQL query language, WikiParq makes it much easier to write database queries to extract specific information or subcorpora from Wikipedia, such as all the first paragraphs of the articles in French, or all the articles on persons in Spanish, or all the articles on persons that have versions in French, English, and Spanish. WikiParq is available in six language versions and is potentially extendible to all the languages of Wikipedia. The WikiParq files are downloadable as tarball archives from this location: http://semantica.cs.lth.se/wikiparq/.

Keywords: Wikipedia, Parquet, query language.

1. Introduction

1.1. Wikipedia

Wikipedia is a collaborative, multilingual encyclopedia with more than 35 million articles across nearly 300 languages. Anyone can edit it and its modification rate is of about 10 million edits per month[1]; many of its articles are categorized; and its cross-references to other articles (links) are extremely useful to help name disambiguation.

Wikipedia is freely available in the form of dump archives[2] and its wealth of information has made it a major resource in natural language processing applications that range from word counting to question answering (Ferrucci, 2012).

1.2. Wikipedia Annotation

Wikipedia uses the so-called wiki markup to annotate the documents. Parsing this markup is then a compulsory step to any further processing. As for the Wikipedia articles, parts of this markup are language-dependent and can be created and changed by anyone. For example, the {{Citation needed}} template in the English Wikipedia is rendered by {{Citation nécessaire}} in French and {{citazione necessaria}} in Italian. Many articles in French use a date template in the form of {{Year|Month|Day}}, which is not used in other Wikipedias.

Moreover, the wiki markup is difficult to master for the millions of Wikipedia editors and, as a consequence, the articles contain scores of malformed expressions. While it is relatively easy to create a quick-and-dirty parser, an accurate tool, functional across all the language versions is a major challenge.

In this paper, we describe WikiParq, a set of Wikipedia archives with an easy tabular access. To create WikiParq, we reformatted Wikipedia dumps from their HTML rendering and converted them in the Parquet format. In addition to the Wikipedia original content, WikiParq makes it easy to add any number of linguistic layers such as the parts of speech of the words or dependency relations.

The WikiParq files are available for download as tarball archives in six languages: English, French, Spanish, German, Russian, and Swedish, as well as the Wikidata content relevant to these languages, from this location: http://semantica.cs.lth.se/wikiparq/.

2. Related Work

There are many tools to parse and/or package Wikipedia. The most notable ones include WikiExtractor (Attardi and Fuschetto, 2015), Sweble (Dohrn and Riehle, 2013), and XOWA (Gnosygnu, 2015). In addition, the Wikimedia foundation also provides HTML dumps in an efficient compression format called ZIM (Wikimedia CH, 2015).

WikiExtractor is designed to extract the text content, or other kinds information from the Wikipedia articles, while Sweble is a real parser that produces abstract syntactic trees out of the articles. However, both WikiExtractor and Sweble are either limited or complex as users must adapt the output to the type of information they want to extract. In addition, they do not support all the features of MediaWiki. A major challenge for such parsers is the template expansion. Dealing with these templates is a nontrivial issue as they can, via the Scribunto extension[3], embed scripting languages such as Lua.

XOWA and ZIM dumps are, or can be converted into, HTML documents, where one can subsequently use HTML parsers to extract information. The category, for instance, is relatively easy to extract using HTML CSS class information. However, neither XOWA nor ZIM dumps make the extraction of specific information from Wikipedia as easy as database querying. In addition, they cannot be easily extended with external information.

[1] https://stats.wikimedia.org/
[2] http://dumps.wikimedia.org/
[3] https://www.mediawiki.org/wiki/Extension:Scribunto

3. The Wikipedia Annotation Pipeline

The Wikipedia annotation pipeline consists of five different steps:

1. The first step converts the Wikipedia dumps into HTML documents that we subsequently parse into DOM abstract syntactic trees using jsoup.

2. The second step consists of a HTML parser that traverses the DOM trees and outputs a flattened unformatted text with multiple layers of structural information such as anchors, typeface properties, paragraphs, etc. At the end of this step, we have the text and easily parseable structural information.

3. The third step consists of a linking stage that associates the documents and anchors with unique identifiers, either Wikidata identifiers (Q numbers) or, as a fallback, Wikipedia page titles.

4. The fourth step annotates the resulting text with grammatical layers that are entirely parametrizable and that can range from tokenization to semantic role labelling and beyond. The linguistic annotation is provided by external language processing tools. This step is optional.

5. The fifth and final stage links mentions of proper nouns and concepts, possibly ambiguous, to unique entity identifiers. As in the third step, we use the Wikidata nomenclature to identify the entities. This fifth step is also optional.

3.1. Wiki Markup and HTML Parsing

The first and second steps of the annotation pipeline parse and convert the dumps into an intermediate document model. As input data, the markup parser uses either HTML documents from ZIM archives or XOWA outputs. It then uses the jsoup HTML parser[4] to build the document object model (DOM) of every page, where we extract the text, sections, paragraphs, infoboxes, anchors (the wiki links), tables, and lists.

The parser recursively traverses the HTML DOM and uses heuristic hints based on the CSS classes and HTML tags such as `<table>`, `<p>`, `<ol>`, and `<ul>` to carry out the information extraction. It outputs the flattened text and nine independent sequences of ranges, where the sequences describe respectively the tokens, sections, paragraphs, list items, list sections, anchors, headings, italic and bold characters. This structured data is an intermediate format that we call the Multilayer Document Model (MLDM), which is similar to a property graph model. Figure 1 shows the conversion pipeline from the Wikimedia dumps to the abstract syntactic trees (AST) and MLDM layers.

3.2. Anchor Resolution

The third step links the documents and anchors to their entity *id*, a unique entity identifier across Wikipedia language editions available from Wikidata.

[4] `http://jsoup.org/`

Prior to the linking step, we collected a set of entities from Wikidata, a freely available graph database that connects the Wikipedia pages across languages. Each Wikidata entity has a unique identifier, a Q-number, that is shared by all the Wikipedia language versions on this entity.

The Wikipedia pages on *Beijing* in English, *Pékin* in French, *Pequim* in Portuguese, and 北京 in Chinese, are all linked to the Q956 Wikidata number, for instance, as well as 190 others languages. In total, Wikidata covers a set of more than 16 million items that defines the search space of entity linking. In the few cases where a Wikipedia page has no Q-number, we replace it by the page name. In addition to the Q-numbers, Wikidata structures its content in the form of a graph, where each node is assigned a set of properties and values. For instance, Beijing (Q956) is an instance of a city (Q515) and has a coordinate location of 39°54'18"N, 116°23'29"E (P625).

We implemented the anchor resolver through a lookup dictionary that uses all the Wikipedia page titles, redirects, and Wikidata identifiers. The page labels, *i.e.* the page titles and all their synonyms, form the entries of this dictionary, while the Wikidata identifiers are the outputs. In case a page label has no Wikidata number, the dictionary uses its normalized page title.

3.3. Grammatical Annotation

Once we have extracted and structured the text, we can apply a grammatical annotation that is language specific. Depending on the language, and the components or resources available for it, the annotation can range from a simple tokenization to semantic-role labels or coreference chains.

Multilinguality. We implemented the grammatical annotation so that it has a multilingual support at the core. All the language-dependent algorithms are stored in separate packages and tied to the system through abstractions that operate on the Multilayer Document Model. More specifically, the token annotations include a dictionary loosely inspired by the CoNLL format (Buchholz and Marsi, 2006) extended with naming conventions from Exner and Nugues (2014).

Dependency Injection. The annotators apply the *dependency injection* (Fowler, 2004) pattern that solves the problem of hard-coded dependencies by making the code only depend on a dependency injector. The dependency injector is constructed once and reused multiple times. The injector can be configured to provide different concrete implementations which allow a high-level way of switching the implementation of an abstraction. The role of the injector is to provide instances of requested abstractions as well as concrete classes. The injector also injects the dependencies needed to construct these instances. We used Guice (Google, 2011) as base library on top of which we developed thread-safe constructions to be able to process indices and storage.

Tool chains. The tool chains are instances of annotators and are specific to the languages we process. For English, Spanish, and German, we use CoreNLP (Manning et al., 2014). For French, we use CoreNLP for tokenizing the text and MATE for parsing (Björkelund et al., 2010). For Swedish, we use Stagger (Östling, 2013) and MaltParser

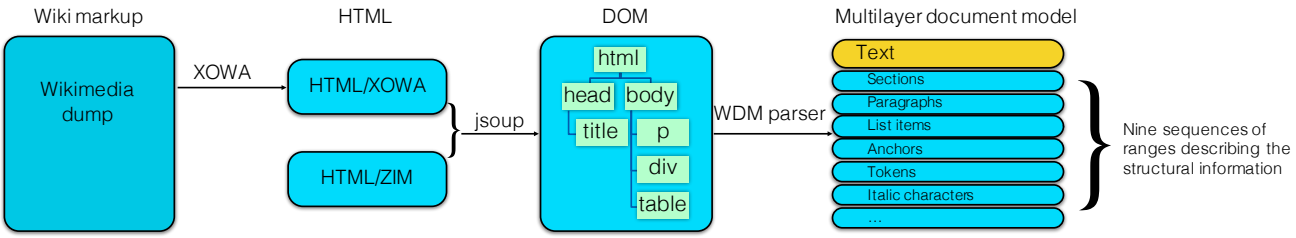

Figure 1: Conversion of Wikipedia dumps into abstract syntactic trees and the Multilayer Document Model (MLDM)

(Nivre et al., 2006). For Russian, there is no linguistic annotation and the tool chain is reduced to nothing as of today.

3.4. Entity Linking

The final step links mentions of named entities and concepts to unique Wikidata identifiers. This last operation is an optional step. While Wikidata defines unambiguous mappings between a Q-number and language-dependent strings, strings may have ambiguous Q-numbers. For example, the mention *Göran Persson* in the Swedish Wikipedia refers to at least four different entities with three different Q-numbers: A former Swedish prime minister (Q53747), a progressive musician (Q6042900), a politician (Q5626648), and a Swedish statesman from the 16th century (Q2625684). The latter is also being spelled *Jöran Person*.

To carry out the mention disambiguation, we reimplemented a variant of TagMe (Ferragina and Scaiella, 2010). TagMe requires minimal grammatical information as it only needs a dictionary of mention-entity pairs and of incoming links. We can then apply it to any of our language versions.

3.5. Visualizing the Annotations

We created a tool to visualize the annotations of the Multilayer Document Model. The layers are selectable from a dropdown menu and their visualization uses brat components[5]. Figures 2, 3, and 4 show screenshots with different layers. They all refer to the Wikipedia article *Carl von Linné* in English.

Figure 2 shows the first paragraph of the article with the token and named entity layers, while Fig. 3 shows the entity links with those manually marked as anchors in the text superimposed to the automatically assigned entity links. The numbers refer to the Wikidata nomenclature. Finally, Fig. 4 shows the dependency relations. This layer is optional and depends on the availability of a parser for the language version. This visualization tool is available at: `http://vilde.cs.lth.se:8080/`.

4. Parquet: A Storage and Extraction Format

Parquet[6] is a popular column-oriented storage format, i.e. a columnar format, where instead of storing a file by row, the data is structured by column. Table 1 shows a simple example of data tabulated in two columns, where the first one consists of words and the second one, of their parts of speech, and Table 2 shows how the first table is stored using a row-oriented format and a columnar one, where the columns are stored sequentially.

Word	POS
The	dt
boy	nn
fell	vb

Table 1: An example of tabulated data.

Row-oriented	Column-oriented	
The	First column	The
dt		boy
boy		fell
nn	Second column	dt
fell		nn
vb		vb

Table 2: Storage organization in row- and column-oriented formats.

In files with large numbers of columns, the Parquet format enables a program to read the columns as needed and skip the others. In addition to providing a faster access to the selected columns, such a format is also very efficient for compressing redundant data; something extremely useful in our case.

5. The WikiParq Format

We created a program to tabulate and store all the annotation layers we described in Sect. 3. using the Parquet format.

5.1. The WikiParq Columns

In its current version, the WikiParq format consists of ten main columns. Some columns borrow concepts from graph database structures in the form of source nodes, target nodes (values), and predicates (or properties) between these nodes:

uri: The wikidata identifier of the document, for instance `urn:wikidata:Q34`;

lang: The language of the document, for instance `de` for German, `fr` for French;

doc: The part of the document. The values can be `article` (the text), `category`, or `disambiguation`;

[5] `http://brat.nlplab.org/`
[6] `https://parquet.apache.org/`

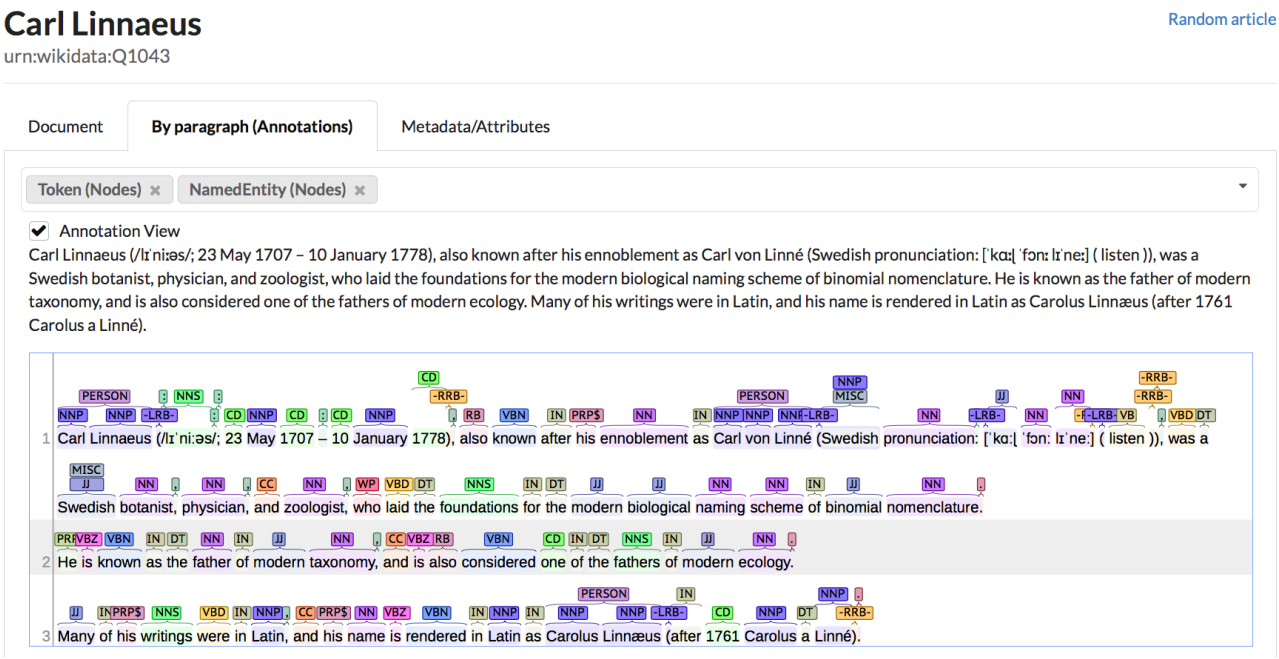

Figure 2: Visualizing tokens and named entities

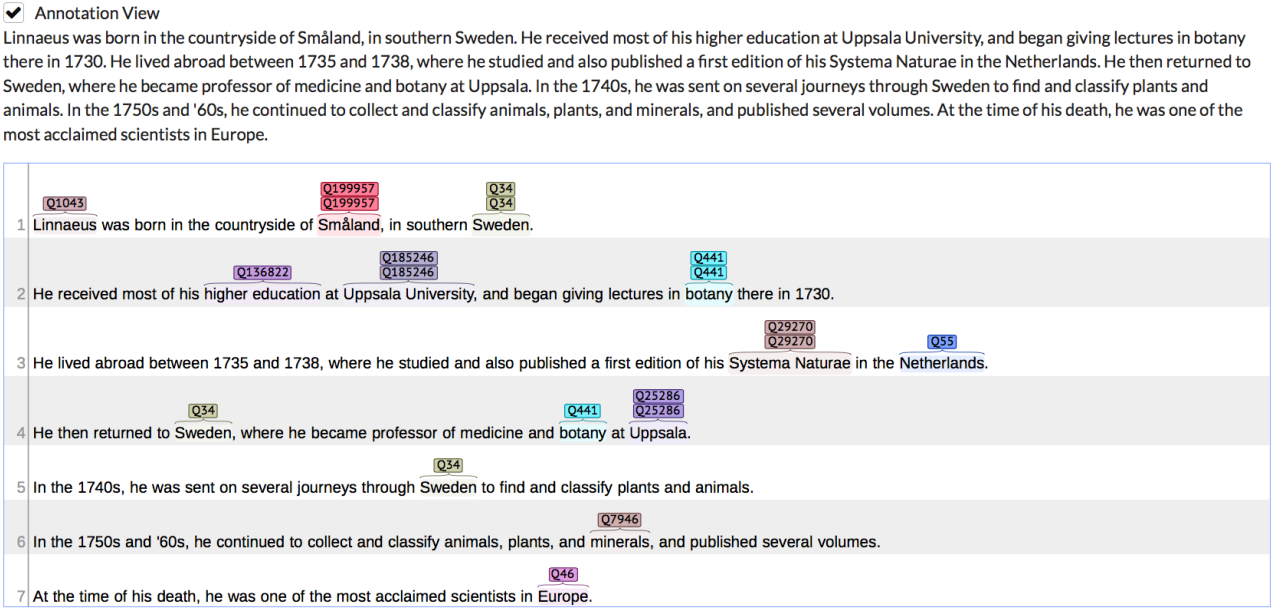

Figure 3: Visualizing the links: Anchors and disambiguated mentions of entities and concepts

source: The layer we are annotating. There are two main types of layers consisting of either nodes or edges. The token layer contains nodes: `node/token`. It is equivalent to a node in a graph, while the dependency layer uses edges;

sourceId: The identifier of an element in the layer, for instance 1 for the first token in a layer;

predicate: The relation annotating the node, for instance the part of speech: `token:pos`. We use similar properties to annotate a token with its lemma, `token:lemma`, a head in a dependency graph, `token:head`, or a resolved Wikipedia link, `link:resolved_target`;

value[1-9]: The values of the target node, normally one: `value1`. For instance, for a token and a part-of-speech property, the value can be a common noun: `NN`. Some relations, such as `link:resolved_target`, have more than one value. In this case, we have `value1` for the target, `value2` for the text, etc.

type: The format of the value, which can either be `string`, `range`, or `reference`, `doclink`, `weblink`, `wikilink` (anchor),

valuei1: Start of the range of the node, if this is relevant, *i.e.* the type is `range` or `reference`. An example of reference that refers to the first token of a document is:

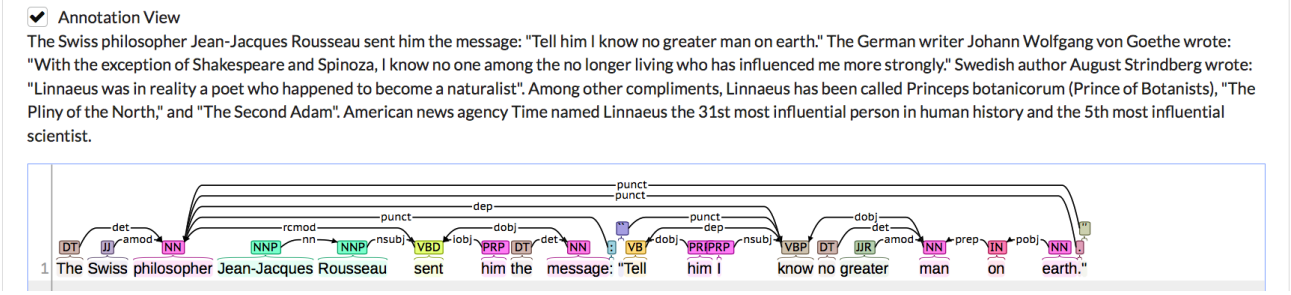

Figure 4: Visualizing the grammatical dependencies

`valuei1 = 1` and `value1 = node/token`;

valuei2: End of the range of the node, if this is relevant (`range`).

Table 3 shows examples of Wikiparq annotations for a whole document, a token, its range and part of speech, as well as an anchor to the *Finland* entity.

5.2. The WikiParq Predicates

We used a set of predicates to describe the relations linking a node to its values. This allows us to represent data in the form of triples consisting of a node, a predicate, and a value. For instance, the token in Table 3 has a text, a range, and a part of speech that conceptually correspond to:

```
wd:Q34/sv/article/node/3614, token:text, "officiellt"
wd:Q34/sv/article/node/3614, range:token, "18-28"
wd:Q34/sv/article/node/3614, token:cpostag, "ADJ"
```

This representation is very flexible. For instance, we designate and annotate token sequences or sentences through the creation of nodes with ranges corresponding to the spans of the groups or the sentences. We can also easily merge multiple editions of Wikipedia by just considering the lang prefix. Table 4 shows the list of all the predicates we are using so far.

We converted the Wikipedia versions of six languages: English, French, Spanish, German, Russian, and Swedish, from archives in the Multilayer Document Model into WikiParq. We also created a WikiParq version of the Wikidata content relevant to these languages. We used Wikipedia dumps from February 3rd or 4th, 2016 and a Wikidata dump from February 22nd, 2016.

The English WikiParq tar file gives an idea of the resulting data volume: It is 15 Gbytes large, has 4.8 million articles, 49.8 million paragraphs, and 175 million resolved links. For Swedish, the total number of triples is of 7.8 billions for the language-annotated version.

6. Querying WikiParq

The Parquet format has been integrated in a number of processing frameworks including map-reduce based databases. It is the default storage format of Spark SQL (Armbrust et al., 2015), a module to query structured data using Spark (Zaharia et al., 2012). Spark is a memory-based implementation of the map-reduce framework (Dean and Ghemawat, 2008) that has become a very popular tool in cluster computing.

Spark SQL makes it very easy to extract information from WikiParq as it follows the familiar SQL syntax and, at the same time, is very fast. In addition, processing can transparently be distributed on a cluster. Spark uses the concept of dataframe, similar to that of R or Python pandas, for loading and saving Parquet files. We can consider dataframes as wrappers to Spark resilient datasets (RDD) with structured schemas.

In the next sections, we provide query examples to show how to extract information from Wikipedia using WikiParq and the Spark (Scala) API.

6.1. Loading a File

The API is straightforward and loading a file, here the English Wikipedia (`en_wiki`), only needs two instructions:

```
val en_wiki = sqlContext.read.parquet(filenames)
en_wiki.registerTempTable("enwiki")
```

which result in a table.

6.2. Extracting Text

To extract all the articles from a Wikipedia version, here Swedish, we use this simple query:

```
SELECT uri, lang, value1 AS text
FROM svwiki
WHERE predicate = 'document:text'
```

inside a `sqlContext.sql()`.

To count all the nouns in a collection, we use:

```
SELECT COUNT(*)
FROM svwiki
WHERE predicate = 'token:cpostag'
  AND value1 = 'NOUN'
```

which results in about 116 millions for Swedish. The corresponding number for verbs is 34 millions.

In the subsequent examples, we used the Parquet files of six languages: de, en, es, fr, ru, and sv, as well as the Wikidata parquet file.

6.3. Counting the Articles per Language

Once the files are loaded, we can extract data or information using SQL queries as for instance for this request:

Extract all the articles on persons in Wikipedia?

Ann. target	uri	doc	source	sourceId	predicate	value1	valuei1	valuei2	type	lang
Document	wd:Q34	article	null	null	document:text	*Sverige*...	null	null	string	sv
Token text	wd:Q34	article	node/token	3614	token:text	officiellt	null	null	string	sv
Token range	wd:Q34	article	node/token	3614	range:token	null	18	28	range	sv
Token POS	wd:Q34	article	node/token	3614	token:cpostag	ADJ	null	null	string	sv
Link	wd:Q34	article	node/anchor	2329	link:resolved_target	wd:Q33	null	null	wikilink	sv

Table 3: Examples of the Wikiparq annotations for a whole document, here the article *Sverige* 'Sweden' in the Swedish Wikipedia, a word in it, here *officiellt*, its range, and part of speech, as well as an anchor (wiki link) to *Finland*. We abridged urn:wikidata in wd

document	link	category	token	range	edge	ne	paragraph	section
alt_title	resolved_target	member-of	cpostag	clean	deprel:label	label	source	title
title	unresolved_target	title	deprel	heading	head			
wiki_page_id			feats	italic	tail			
text			head	link				
category			idx	list_item				
			lemma	list_section				
			norm	named_entity				
			pos	paragraph				
			text	section				
				sentence				
				strong				
				token				

Table 4: List of available predicates organized by prefix. To have the full name, the prefix is concatenated to the relation name, i.e. document:title or link:resolved_target

To carry this out, we first extract the persons from the Wikidata ontology. We use the *instance of* property (P31) and we keep the entities having the property of being an instance of a human (Q5). This is translated in SQL as:

```
sqlContext.sql("""
   SELECT uri
   FROM wikidata
   WHERE predicate = 'wd:P31'
   AND value1 = 'urn:wikidata:Q5'
""").cache().registerTempTable("persons")
```

and results in a table called `persons`.
We then extract the language versions associated with each of these entities (persons). We run the extraction using this query:

```
sqlContext.sql("""
   SELECT wikidata.uri AS uri, lang
   FROM wikidata
   JOIN persons
   ON persons.uri = wikidata.uri
   WHERE predicate = 'wd:sitelink'
""").registerTempTable("person_sitelinks")
```

that produces a table of entity identifiers (Q-numbers) and languages. The `wd:sitelink` predicate enables us to find the language versions of an entity according to Wikidata.
Finally, we count the persons per language using this query:

```
sqlContext.sql("""
   SELECT lang, COUNT(lang) as count
   FROM person_sitelinks
   GROUP BY lang
   ORDER BY lang
""").show()
```

This results in a table with the number of persons per language version:

```
+----+-------+
|lang|  count|
+----+-------+
|  de| 597515|
|  en|1339313|
|  es| 274878|
|  fr| 462839|
|  ru| 313566|
|  sv| 183926|
+----+-------+
```

6.4. Counting the Language Versions of an Article

Going on with this dataset, a second question we may pose is:

> For a given article, how many language versions are there?

which is translated in SQL as:

```
sqlContext.sql("""
   SELECT uri, COUNT(lang) AS numLangs
   FROM person_sitelinks
```

```
   GROUP BY uri
""").registerTempTable("lang_person")
```

The excerpt below shows a subset of the first answers to this question:

```
+--------------------+--------+
|                 uri|numLangs|
+--------------------+--------+
|urn:wikidata:Q100250|       2|
|urn:wikidata:Q100412|       3|
|urn:wikidata:Q100863|       1|
|urn:wikidata:Q101097|       1|
|urn:wikidata:Q101141|       2|
|urn:wikidata:Q101259|       1|
|urn:wikidata:Q101303|       2|
|urn:wikidata:Q101754|       2|
|urn:wikidata:Q101916|       2|
|urn:wikidata:Q102032|       2|
|urn:wikidata:Q102483|       6|
|urn:wikidata:Q102645|       3|
|urn:wikidata:Q10287 |       6|
+--------------------+--------+
```

The Q-number is the Wikidata identifier of a Wikipedia entity. For instance, Q101916 is the identifier of Friedrich von Weech, a German regional historian and archivist, who has German and Swedish versions.

6.5. Counting Articles with a Constraint on the Language Versions

At this point, we may wonder:

> How many articles are available in the six versions?

This question is rendered by the following SQL query:

```
sqlContext.sql("""
  SELECT COUNT(uri)
    FROM lang_person
    WHERE numLangs = 6""")
```

and the answer is: 41,509 articles.

6.6. Extracting the Text of the Articles

So far, we carried out extractions that could have also been done from Wikidata using the SPARQL language. WikiParq merges Wikidata and Wikipedia and extends the query possibilities to span both resources seamlessly as for instance with this request whose goal could be to build loosely parallel corpora in six languages:

> Extract the text of all the articles with six language versions

Such a request is translated by these two following SQL queries, where the first one selects the articles with six language versions:

```
sqlContext.sql("""
    SELECT * FROM lang_person
    WHERE numLangs = 6
""").cache().registerTempTable("lang6_person")
```

and the second one outputs the text, here in Swedish:

```
sqlContext.sql("""
    SELECT lang6_person.uri, value1 AS text
    FROM svwiki
    JOIN lang6_person
    ON svwiki.uri = lang6_person.uri
    WHERE svwiki.predicate = 'document:text'
""").show()
```

6.7. Extracting all the Mentions of an Entity

A last example shows how to build dictionaries of the words or phrases used in Wikipedia to name an entity: A dictionary of mentions. To carry this out, we need to extract all the labels of a entity in Wikipedia. This operation is easy to carry out, for instance for Barack Obama in the English Wikipedia. Barack Obama has Q76 as Wikidata identifier. This leads to this query:

```
sqlContext.sql("""
   SELECT value1 AS target, value2 AS mention,
    COUNT(*) AS freq
    FROM enwiki
     WHERE enwiki.predicate =
       'link:resolved_target'
    AND value1 = 'urn:wikidata:Q76'
    GROUP BY value1, value2
    ORDER BY value1, freq DESC
""").show()
```

that results in a table, where we show the first lines below:

```
+---------------+--------------------+-----+
|          target|             mention| freq|
+---------------+--------------------+-----+
|urn:wikidata:Q76|        Barack Obama|14960|
|urn:wikidata:Q76|               Obama|  887|
|urn:wikidata:Q76|     President Obama|  546|
|urn:wikidata:Q76|President Barack ...|  142|
|urn:wikidata:Q76|     Barack H. Obama|   64|
|urn:wikidata:Q76|       Obama, Barack|   45|
|urn:wikidata:Q76|      Barack Obama's|   40|
|urn:wikidata:Q76|   President Obama's|   21|
|urn:wikidata:Q76|              Barack|   21|
|urn:wikidata:Q76|President Barack ...|   13|
|urn:wikidata:Q76|Obama administration|    9|
|urn:wikidata:Q76|             Obama's|    8|
|urn:wikidata:Q76|President-elect O...|    8|
|urn:wikidata:Q76|           President|    7|
|urn:wikidata:Q76|   Sen. Barack Obama|    7|
|urn:wikidata:Q76|Barack Hussein Obama|    7|
|urn:wikidata:Q76|Barack Hussein Ob...|    6|
|urn:wikidata:Q76|U.S. President Ba...|    6|
|urn:wikidata:Q76|Senator Barack Obama|    6|
|urn:wikidata:Q76|     Barack Obama' s|    5|
+---------------+--------------------+-----+
```

7. Conclusion

We have described WikiParq, a unified tabulated format that uses the Parquet standard to package the Wikipedia corpora. In combination with Spark, a map-reduce computing

framework, and the SQL query language, this format makes it easy to write concise database queries to extract specific information from Wikipedia and have the answer in a few minutes.

Currently, six versions of Wikipedia are available as tarball archives in the WikiParq format from this location: `http://semantica.cs.lth.se/wikiparq/`. We also provide a Parquet version of Wikidata, as well as a Scala program and a Jupyter notebook to run the examples described in this paper. We ran and tested all the examples on a laptop with an Intel i7 processor and 16 Gbytes of memory.

Acknowledgments

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

8. Bibliographical References

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., and Zaharia, M. (2015). Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1383–1394.

Attardi, G. and Fuschetto, A. (2015). Wikiextractor. https://github.com/attardi/wikiextractor/.

Björkelund, A., Bohnet, B., Hafdell, L., and Nugues, P. (2010). A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstration Volume*, pages 33–36, Beijing, August 23-27. Coling 2010 Organizing Committee.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Dohrn, H. and Riehle, D. (2013). Design and implementation of wiki content transformations and refactorings. In *Proceedings of the 9th International Symposium on Open Collaboration*, WikiSym '13, pages 2:1–2:10.

Exner, P. and Nugues, P. (2014). KOSHIK: A large-scale distributed computing framework for NLP. In *Proceedings of ICPRAM 2014 – The 3rd International Conference on Pattern Recognition Applications and Methods*, pages 464–470, Angers, March 6-8.

Ferragina, P. and Scaiella, U. (2010). Fast and accurate annotation of short texts with wikipedia pages. In *Proceedings of CIKM'10*, Toronto.

Ferrucci, D. A. (2012). Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1:1–1:15, May-June.

Fowler, M. (2004). Inversion of control containers and the dependency injection pattern. Last accessed: 2013-12-20.

Gnosygnu. (2015). Xowa. https://github.com/gnosygnu/xowa.

Google. (2011). Guice (version 3.0). Last accessed: 2013-11-10.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Nivre, J., Hall, J., and Nilsson, J. (2006). MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219.

Östling, R. (2013). Stagger: an open-source part of speech tagger for Swedish. *Northern European Journal of Language Technology*, 3:1–18.

Wikimedia CH. (2015). Openzim. http://www.openzim.org.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*.