Framework for Effective Representation of Wikipedia and Graph-based Distance Calculation

Alexander Ulanov, Dmitry Ryashchentsev

HP Laboratories
HPL-2010-153

Keyword(s):

Wikipedia, semantic similarity, effective graph representation

Abstract:

We consider the problem of using Wikipedia as an external knowledge base in real-time applications. In particular, present tools don't allow making fast computations on Wikipedia graph. This is essential for such tasks as word sense disambiguation or term clustering. To address this issue we propose the framework for effective representation of Wikipedia and graph-based distance calculation.

External Posting Date: October 21, 2010 [Fulltext] Approved for External Publication
Internal Posting Date: October 21, 2010 [Fulltext]

Framework for Effective Representation of Wikipedia and Graph-based Distance Calculation

Alexander Ulanov, Dmitry Ryashchentsev
HP Labs Russia
{alexander.ulanov;dmitry.ryashchentsev}@hp.com

Abstract. We consider the problem of using Wikipedia as an external knowledge base in real-time applications. In particular, present tools don't allow making fast computations on Wikipedia graph. This is essential for such tasks as word sense disambiguation or term clustering. To address this issue we propose the framework for effective representation of Wikipedia and graph-based distance calculation.

1. Introduction

Wikipedia used in various applications: as thesaurus, ontology, for word sense disambiguation, co-reference resolution, query expansion, named entity extraction etc. [Medelyan et al., 09].

Term lookup and computing similarity between them are ones of the most basic procedures, used for more complex methods such as keyterm extraction, term disambiguation, term clustering, taxonomy extraction, named entity recognition. In these tasks, term lookup and similarity are computed huge amount of times, since they imply pair vise term comparison. Although there are available several tools for solving the basic needs mentioned, they have quite low performance, and allow computing only several basic operations per second. This is mainly due to use of relational databases, which cannot deliver good performance in extensive ad-hoc lookup of pages and links between them. This paper presents the framework for effective representation of Wikipedia and graph-based distance calculation which allows storing structure of Wikipedia categories and articles with redirects and corresponding senses for disambiguation pages. Its use is not limited for: term and term senses lookup, term similarity measurement, categories taxonomy extraction etc.

There are several approaches to computing term similarity in Wikipeadia. Wikirelate! [Ponzetto & Strube, 07] is based on category structure. Explicit semantic analysis (ESA) [Gabrilovich & Markovitch, 07] is based on text features and link within articles. Method proposed in [Milne & Witten, 08] uses incoming and outgoing links. It is hard to estimate the performance of Wikirelate! and ESA since they are not available for downloading. There is a service for the method proposed in [Milne & Witten, 08]. It is server-based, so cannot be used for stand-alone applications and large experiments.

We will use the following terminology. Wikipedia page – is any object that has an id in Wikipedia. Wikipedia category is the page which name begins with "Category" prefix. Wikipedia article is the page representing a notion in Wikipedia. Redirect is the page that redirects to another page. Ambiguous page is such a page that has disambiguation categories or "Other uses" link at the top.

The paper is organized as follows. Section 2 describes problem statement and proposed approach. Section 3 describes how and which data were extracted from Wikipedia database. Section 4 is devoted to data preprocessing and loading in memory. Section 5 represents the experiments conducted with the developed library. Section 6 outlines the conclusion.

2. Problem statement

The goal of this work is to develop a library for effective lookup of Wikipedia terms and distance calculation. Requirements are:

- Term lookup – milliseconds
- Distance calculation – tens of milliseconds

Several sub-problems will be solved to achieve the goal:

- Extracting data from Wikipedia database, including extraction of articles names, redirects, disambiguation pages and categories
- Developing structures for effective data representation
- Preprocessing data for in-memory representation
- Implementation of term lookup and term distance

3. Extract data from Wikipedia XML database

We have downloaded and parsed Wikipedia dump and loaded it into XML Database [Wikipedia]. Then we extract data from it. Two tables were used to extract data. First describes categories and article with redirects structure. Second describes only ambiguous articles and corresponding senses. The described structures are represented on the fig.1 and fig.2.

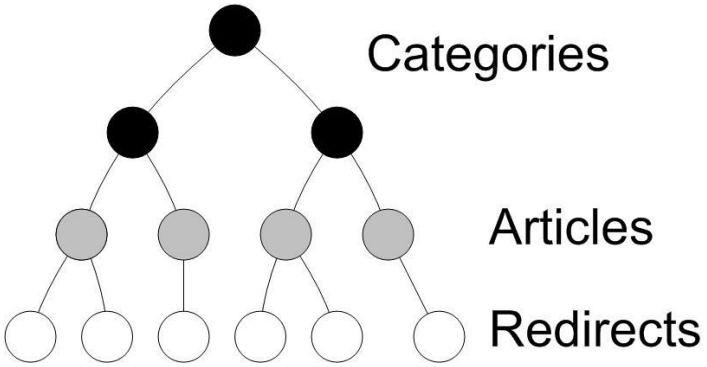

Fig. 1. Wikipedia categories and articles structure with redirects

The table for representing Wikipedia categories and articles structure with redirects has the following structure:

Page name	Page id	Page id, if redirect	Categories ids (if not redirect)

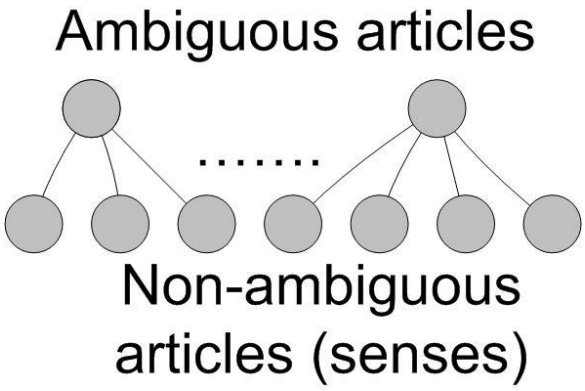

Fig. 2. Wikipedia disambiguation articles structure

The table for representing Wikipedia disambiguation articles structure has the following fields:

Ambiguous article name	Ambiguous article id	Ids of articles that are various senses of ambiguous one

Two queries were implemented to extract data. The query for Wikipedia categories and articles structure with redirects is rather straight-forward. Query processed all Wikipedia pages in database, extracted name and id of it and id of an actual page if it was redirect. If it was an actual page then all categories were extracted.

The query for extracting disambiguation articles structure was more complicated. Query processed all Wikipedia pages and if it was an ambiguous page and not redirect then query extracted its name and id and ids of articles that are other senses of the current article. The list of senses contained senses listed on the disambiguation page plus id of the given page if it had "For other uses" link. Only such senses were extracted that do not correspond to named entities, such as person or location names etc.

4. Preprocess and load into memory

Wikipedia structure is represented in memory as:

- Wikipedia articles and categories titles dictionary
- Categories graph
- Graph of disambiguation references
- Redirects references
- Exclude list

The extracted tables are loaded into the memory, converted into compact binary format by *com.hp.hplabs.lim2.util.webgraph.GraphFunctions* class methods and stored on disk:

- *terms.csa* – Wikipedia terms dictionary
- *hash.graph-off, hash.graph-ref* - Wikipedia terms dictionary hash index
- *parents.graph-off, parents.graph-ref* – Wikipedia categories graph
- *children.graph-off, children.graph-ref* – Wikipedia categories transposed graph
- *redir.ids* – redirects index
- *disamb.graph-off, disamb.graph-ref* – disambiguation graph
- *allow.idx* – allow/exclude index
- *allowcat.graph-off, allowcat.graph-ref* – Wikipedia categories graph with removed refs to excluded nodes
- *allowchild.graph-off, allowchild.graph-ref* - Wikipedia categories transposed graph with removed refs to excluded nodes

Wikipedia articles and categories dictionary is represented as a big byte sequence that contain UTF8 encoded titles and indexes array. That allows storing all Wikipedia titles in 220Mb of memory directly without compression. For the fast terms search it as also build a hash table that takes 100Mb of memory (fig.3).

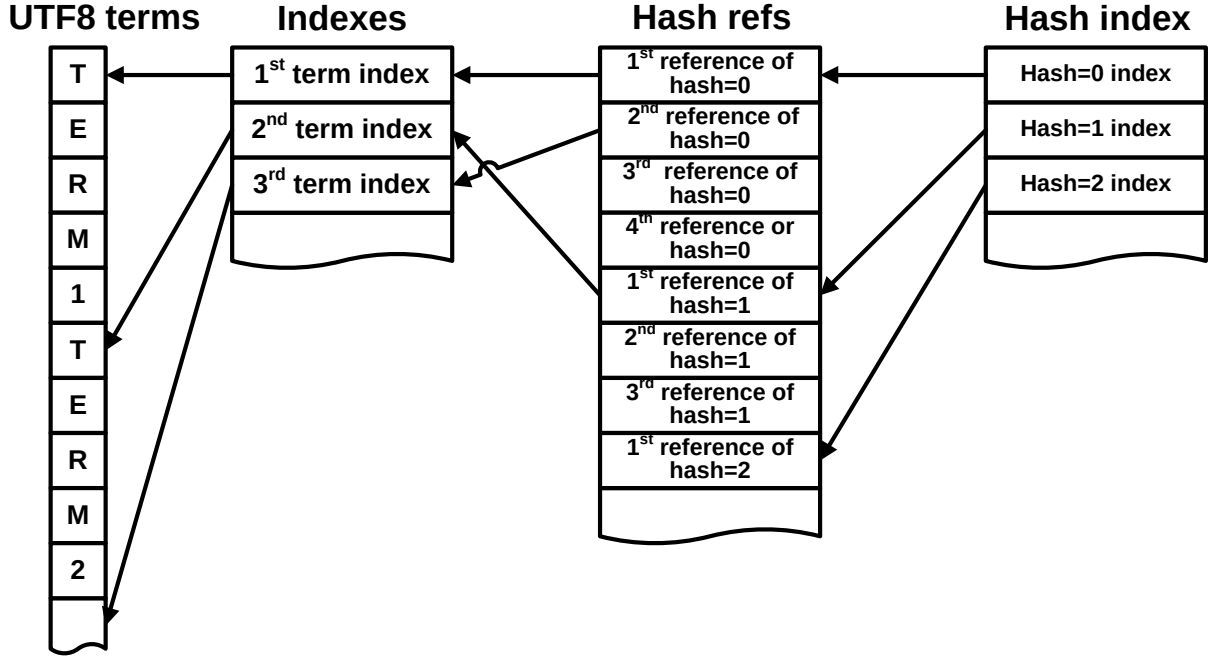

Fig. 3. Structure for terms representation.

Categories, disambiguation graphs are represented as two integer arrays: graph references and nodes indexes – that also allows to provide fast access without any decompression with minimal data extraction stuff (fig.4).

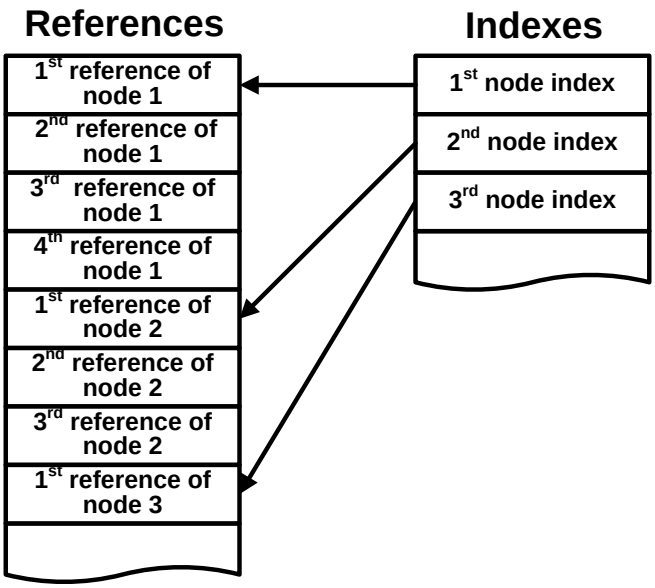

Fig. 4. Structure for categories and redirects representation.

The allow/exclude list index marks the following nodes:

- All nodes under the following categories:
  - "Category:Hidden categories"
  - "Category:Wikipedia maintenance",
  - "Category:Thought"
- All nodes started from the following prefixes:
  - "Category:Articles"
  - "Category:Categories",
  - "Category:Wikipedia",
  - "Category:Very large categories",
  - "Category:All Article",
  - "Category:Cleanup",
  - "Category:Hidden categories",
  - "Category:Wikipedia administration",
  - "Category:Categories by topic",
  - "Category:Redirect templates",
  - "Category:Redirects from other template",
  - "Category:Interdisciplinary fields",
  - "Wikipedia:",
  - "File:",
  - "Template:",
  - "Media:",
  - "User:",
  - "MediaWiki:",
  - "Help:",
  - "Portal:",
  - "Book:",
  - "Talk:"

All in-memory structures: dictionary, dictionary hash table, categories direct and transposed graphs, disambiguation and redirects graphs take less then 600Mb of memory. Therefore this framework can be used on common PC. The use of light in-memory compression methods like [Boldi et al., 04] allows to save ~30% of memory, but we did not find it reasonable to use for Wikipedia for the price of performance.

We store on disk the created arrays and can load them back into memory (in lazy way) very quickly and no deployment is needed. All 600Mb of data are loaded in about 7 seconds.

5. Shortest path

In-memory library contains the following methods to calculate shortest path on a graph:

1. Minimal distance calculation on direct graph with equal arc lengths
2. Making minimal paths using minimal distance calculation temp data

Minimal distance can be calculated from node to node, from node to nodes set, and from nodes set to nodes set. The algorithm is based on breadth-first search. On every n-step it finds all nodes that are in n-arcs minimal distance from the initial node (or nodes set), i.e. on every n-step the area is built around the initial nodes (or nodes sets) of nodes with minimal distances <= n. This process is limited by receiving a minimal distance through the areas intersections or by exceeding a maximum deep.

Gradient descent is used to find all shortest paths between initial points or set of points. It is done using the points in the intersections of the points areas mentioned.

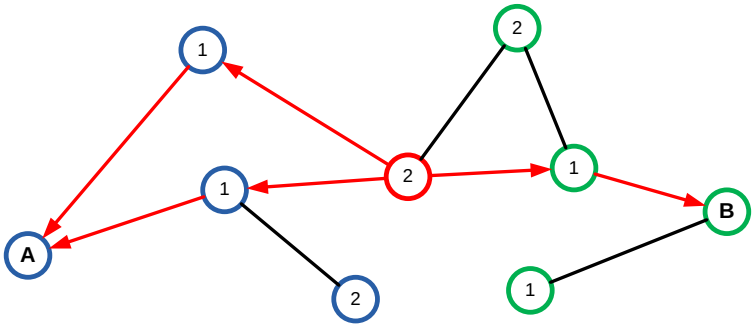

Fig. 5. Example of gradient descent.

With this algorithm a distance from a single node to single node is calculated faster than 1ms, between sets of 5-8 terms is it calculated for about 30ms.

6. Experiments

Two benchmarks were run for evaluation of the proposed framework: Miller&Charles [Miller & Charles, 91] and WordSim353 [Finkelstein et al., 02] testsets. They are based on semantic similarity computation between pairs of nouns. The goal of the benchmark is to show correlation of computer-based similarity with human evaluated. The following similarity function between articles i and j was implemented [Leacock & Chodorow, 98]:

$$Sim_{Jac}(i, j) = \log \frac{s}{2d},$$

where s is the shortest path between articles in a category hierarchy and d is the Wikipedia depth. Due to large categories interconnectivity maximum depth can reach the amount of 30. But such long sequences don't make sense and depth can be approximated as a minimal depth of the deepest node. The implemented function shows very good correlation with human opinions. It is usually used on WordNet. By our experiments we demonstrated that it could be successfully used for Wikipedia categories structure (Table 1). We compared it with WikiRelate! method that

also uses category structure of Wikipedia. Methods by [Gabrilovich & Markovitch, 07] and [Milne & Witten, 08] show better correlation, but they use links in the article body.

Table 1. Similarity benchmarks results

Method	**Miller&Charles**	**WordSim353 (full/set1/set2)**
Leacock & Chodorow	0.63	0.42; 0.48; 0.32
WikiRelate! [Ponzetto & Strube, 07]	0.49	0.49; n/a ; n/a

7. Conclusion

The framework for effective representation of Wikipedia and graph-based distance computation was developed. It was done in several stages: extraction of articles and categories structure from Wikipedia, developing a structure for storing it in memory, preprocessing and loading it in memory. Graph-based distance function was developed for enabling computation of similarity between Wikipedia articles. The developed library was used in experiments for similarity computation in two state of the art benchmarks and it demonstrated high-performance results.

Further work is connected with the use of this framework for Taxonom DTX. The framework is available for download by request and as a service [Service].

References

[Boldi et al., 04] Boldi, P., Vigna, S.: The WebGraph framework I: Compression techniques. In: Proc. Of the Thirteenth International World Wide Web Conference, Manhattan, USA, ACM Press (2004) 595–601

[Finkelstein et al., 02] Finkelstein, L., Gabrilovich, E.,Matias, Y., Rivlin, E., Solan, Z.,Wolfman, G., & Ruppin, E. Placing search in context: The concept revisited. ACM Transactions on Information Systems, 20(1), 116–131, 2002.

[Gabrilovich & Markovitch, 07] Gabrilovich, G. and S. Markovitch. [2007] Computing Semantic Relatedness using Wikipedia-based
Explicit Semantic Analysis. In Proceedings of the 20th International Joint Conference on Artificial
Intelligence, IJCAI'07, Hyderabad, India, January 2007, p.1606–1611.

[Leacock & Chodorow, 98] Leacock, C., Chodorow, M. Combining local context and WordNet similarity for word sense identification. In Fellbaum, C. (Ed.), WordNet. An Electronic Lexical Database, chap. 11, pp. 265–283. Cambridge, Mass.: MIT Press, 1998.

[Medelyan et al., 09] O. Medelyan, D. Milne, C. Legg and I. H. Witten. 2009. Mining meaning from Wikipedia. International Journal of Human-Computer Studies. Volume 67, Issue 9, pp. 716-754

[Miller & Charles, 91] Miller, G. A., & Charles,W. G. Contextual correlates of semantic similarity. Language and Cognitive Processes, 6(1), 1–28, 1991.

[Milne & Witten, 08] Milne, D. and Witten, I.H. (2008) An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In Proceedings of the first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08), Chicago, I.L.

[Ponzetto & Strube, 07] Ponzetto, S. P. and M. Strube. Knowledge Derived from Wikipedia for Computing Semantic Relatedness. Journal of Artificial Intelligence Research 30, pp. 181–212

[Service] Wikipedia similarity service. http://bakeoff-srv-3.hpl.hp.com/wikigraph/similarity.html

[Wikipedia] Wikipedia database download http://en.wikipedia.org/wiki/Wikipedia:Database_download

Appendix. In-memory library interface

The in-memory library consists of two parts:

- Data primitives
- Algorithms

The following data access interfaces are defined in the library:

- *IIndexStorage* – immutable int array
- *IArray* – general array
- *BigByteSequence* – growable byte array
- *IGraph* – minimal graph access interface
- *IFastGraph* – graph access interface with additional methods that allow to reduce memory allocation operations

There is a number of the interfaces implementations that allow:

- Read-Write data into memory in fastest way
- Read-Write data into memory with light compression
- Read-Write data directly to file in fastest way
- Read-Write data directly to file with custom compression (gzip) of arrays content

Any memory structures can be serialized and deserialized from a file.

Class com.hp.hplabs.lim2.wikipedia.CompactWiki provides the following Wikipedia data methods with fast memory access:

- *int getTermID(String)* – get internal Wikipedia term ID by title with hash search
- *String getTermByID(int)* – get Wikipedia term title by internal ID
- *FileGraph getCategoriesGraph()* – get Wikipedia categories graph
- *FileGraph getChildrenGraph()* – get Wikipedia transposed categories graph
- *IIndexStorage getRedirectsIdx()* – get Wikipedia redirects index
- *FileGraph getDisambiguationGraph()* – get Wikipedia disambiguation graph
- *IIndexStorage getWikiIDs()* – get Wikipedia ID by internal one.
- int[] getAllowList() – get allow list flags array: 0 – allow, -1 - exclude
- *FileGraph getAllowCatGraph()* – get Wikipedia categories graph with removed references to excluded nodes
- FileGraph getAlloChildGraph() - get Wikipedia transposed categories graph with removed references to excluded nodes

Class *com.hp.hplabs.lim2.util.webgraph.GraphFunctions* contains data import stuff.

The second part of the library is shortest paths search algorithms implementations that operate with in-memory library interfaces.
The following classes are defined to store result and intermediate data:

- *DistanceMap* – region (node or list of nodes) with distances to n-deep neighbors, this object can be reused in more than one distance calculations that allows to reuse previous calculations steps
- *DistanceIntersection* – two regions closest intersection
- *DistanceStaticFullPathTree* – two regions closest paths

The following algorithms are implemented in class *com.hp.hplabs.lim2.util.webgraph.DistanceEngine*:

- *getIntersection(IFastGraph iGraph, int nMaxDeep, DistanceMap map1, DistanceMap map2, DistanceIntersection iIntersection)* – calculates closest intersection from region *map1* to *map2* on graph *iGraph* with max deep *nMaxDeep*. Result is put to iIntersection.

- *getPaths(IFastGraph iTransGraph, DistanceMap map, DistanceStaticFullPathTree tree)* – make closest path from intersection to points defined in *map* on graph *iTransGraph* by gradient descent method
- *dumpPath(ElementDumper dumper, DistanceStaticFullPathTree.PathInfo path, int nFrom)* – dumps *nFrom*-path contained in *path* into *dumper*

Example of the library usage is in *com.hp.hplabs.lim2.util.webgraph.DistanceEngine.pathsTest()*
Performance evaluation is provided in class *com.hp.hplabs.lim2.util.webgraph.GraphTest*