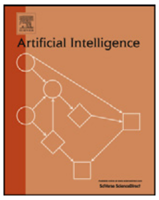
Artificial Intelligence ••• (••••) •••–•••

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



An open-source toolkit for mining Wikipedia

David Milne *, Ian H. Witten

Computer Science Department, The University of Waikato, Private Bag 3105, Hamilton, New Zealand

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Wikipedia
Toolkit
Ontology extraction
Semantic relatedness
Disambiguation
Annotation

ABSTRACT

The online encyclopedia Wikipedia is a vast, constantly evolving tapestry of interlinked articles. For developers and researchers it represents a giant multilingual database of concepts and semantic relations, a potential resource for natural language processing and many other research areas. This paper introduces the Wikipedia Miner toolkit, an open-source software system that allows researchers and developers to integrate Wikipedia's rich semantics into their own applications. The toolkit creates databases that contain summarized versions of Wikipedia's content and structure, and includes a Java API to provide access to them. Wikipedia's articles, categories and redirects are represented as classes, and can be efficiently searched, browsed, and iterated over. Advanced features include parallelized processing of Wikipedia dumps, machine-learned semantic relatedness measures and annotation features, and XML-based web services. Wikipedia Miner is intended to be a platform for sharing data mining techniques.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The online encyclopedia Wikipedia is a vast, constantly evolving tapestry of richly interlinked textual information. To a growing community of researchers and developers it is an ever-growing source of manually defined concepts and semantic relations. It constitutes an unparalleled and largely untapped resource for natural language processing, knowledge management, data mining, and other research areas.

Those who wish to draw on Wikipedia as a source of machine-readable knowledge have two options. They can either base their work on secondary structures that others have extracted from it, such as Freebase [2] and Yago [26], or they can start from scratch and build their own algorithms to mine Wikipedia directly. The first approach is the easiest. However—as this special issue demonstrates—new innovations and mining techniques are introduced regularly, potentially rendering obsolete pre-built resources that rely on the current state of the art. Furthermore, unless they are studiously maintained, such resources forego one of Wikipedia's greatest strengths: its propensity to grow rapidly and keep abreast of world events. The second option—working directly from the source—allows researchers to continue to innovate and find new ways to mine knowledge from Wikipedia. Moreover, any new information added by Wikipedians flows directly through. Wikipedia's entire content is readily available under a Creative Commons license, with regular releases in the form of large XML and HTML dumps.[1] Unfortunately, substantial effort is needed to mine these dumps: they are enormous and replete with cryptic markup.

This paper introduces a third option: to share algorithms and code rather than secondary resources. We introduce Wikipedia Miner,[2] an open-source toolkit that allows its users to sidestep the laborious effort needed to mine Wikipedia's

* Corresponding author. Tel.: +61 29 372 4328.
E-mail addresses: d.n.milne@gmail.com (D. Milne), ihw@cs.waikato.ac.nz (I.H. Witten).

[1] The content of Wikipedia and other MediaWiki projects is available for download from http://download.wikipedia.org.

[2] Code, data and online demonstrations of the Wikipedia-Miner toolkit are available at http://wikipedia-miner.sourceforge.net.

0004-3702/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.artint.2012.06.007

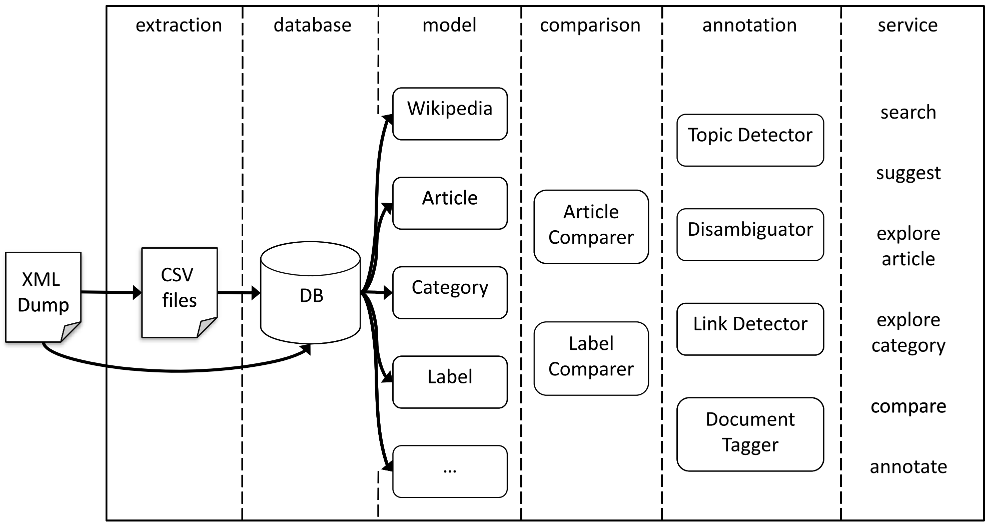


Fig. 1. Architecture of the Wikipedia Miner toolkit.

riches. It also provides a platform for sharing mining techniques, and for taking advantage of powerful technologies like the distributed computing framework Hadoop [27] and the Weka machine learning workbench [28].

The paper is structured as follows. Section 2 provides a broad overview of the toolkit. The following two sections elaborate on two of its unique features: Section 3 describes and evaluates its algorithms for measuring semantic relatedness using Wikipedia as background knowledge, and Section 4 does the same for detecting and disambiguating Wikipedia topics when they are mentioned in documents. The application of these features is illustrated in Section 5, which demonstrates how a simple thesaurus browser and document annotator can be constructed with minimal code. Section 6 reviews related work, including alternative and complementary resources for mining Wikipedia, and projects that apply the toolkit to various research problems. The paper concludes with a discussion of Wikipedia Miner's features and limitations, and points out directions for future development.

Before continuing, it may be valuable to clarify some of the terminology that will be used in the remainder of the paper. Because Wikipedia articles are typically homogenous—they are each dedicated to describing a single topic—they can be treated the same as descriptors in a thesaurus or concepts in an ontology. Consequently we use the terms *article*, *topic* and *concept* interchangeably throughout the paper. Articles are referred to in textual documents by words or phrases that we call *terms* or *labels*; again, we use these interchangeably. Where a label may refer to multiple concepts (i.e. it is ambiguous), we refer to these concepts as *senses*.

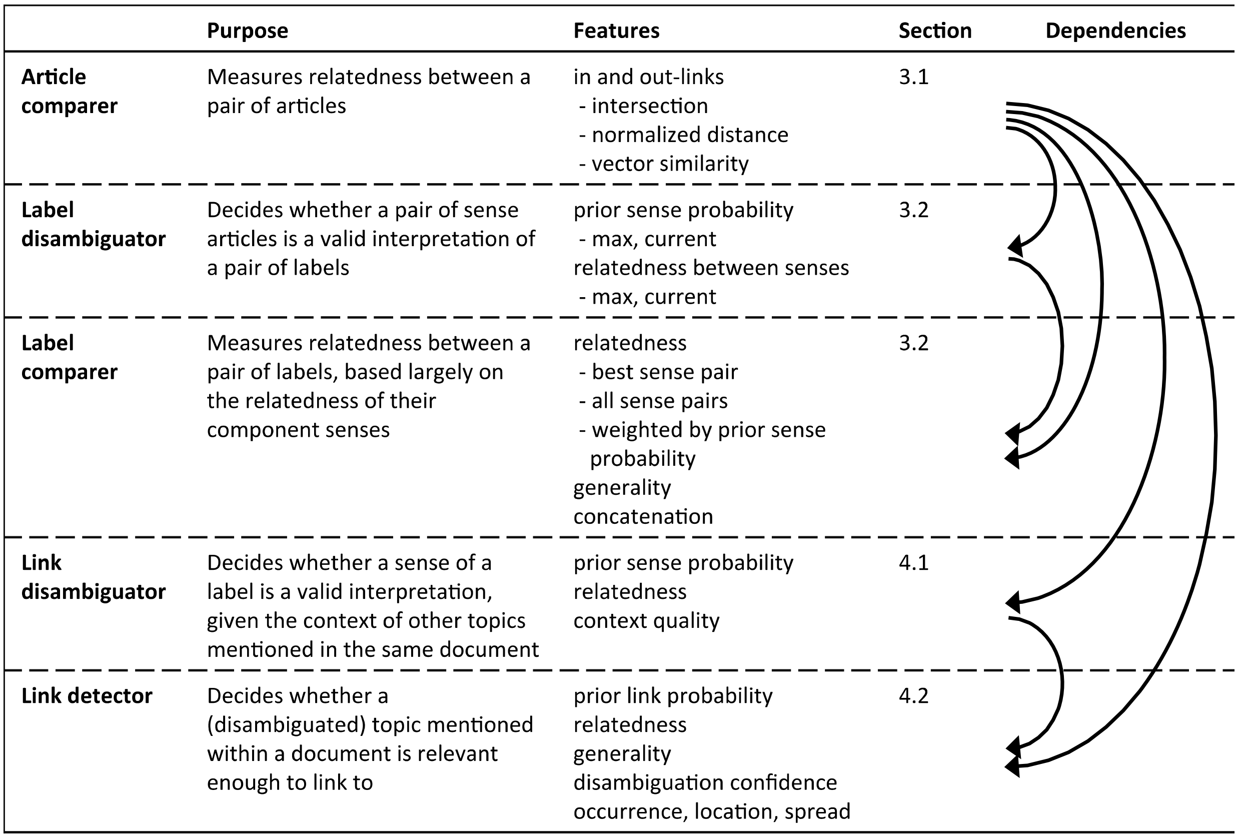
2. The Wikipedia Miner toolkit

Fig. 1 illustrates the overall architecture of the Wikipedia Miner toolkit, which is implemented entirely in Java. The figure begins on the left with a single large XML file containing the full content of a particular edition of Wikipedia, obtained directly from the Wikimedia Foundation, excluding revision history, background discussion and multimedia content. This file is fed into a run-once extraction process (found within the *extraction* package described in Section 2.1), which produces a series of flat-file summaries of Wikipedia's structure.

These summaries are simply delimited text files, and developers could construct programs to read them directly. This would, however, require significant computer time and memory—the link-graph summary alone occupies more than 1 GB. Instead, both the summaries and the original XML dump are fed into a database environment, managed by the *database* package described in Section 2.2, so that they can be indexed persistently. As Section 2.3 explains, the *model* package of the toolkit simplifies access to the stored data by wrapping it with easy to understand, thoroughly documented classes, such as *Wikipedia*, *Article* and *Category*.

Section 2.4 describes the *comparison* package for measuring relatedness, both between pairs of concepts and between pairs of terms. Section 2.5 describes the *annotation* package for detecting and disambiguating concepts when they are mentioned in textual documents. Both packages rely heavily machine-learned classifiers, which are listed in Fig. 2. The figure also summarises the features these classifiers draw on, the chain of how the output of one classifier flows on to the next, and the sections in the paper that discuss them in detail.

Using the features described up to this point requires a significant commitment from the user: one has to download Wikipedia in its entirety and invest many machine hours preprocessing it. The toolkit's final component is the *service* package, a suite of web services that allows casual users to explore its functionality. Section 2.6 gives a brief overview of this package.



	Purpose	Features	Section	Dependencies
Article comparer	Measures relatedness between a pair of articles	in and out-links - intersection - normalized distance - vector similarity	3.1	
Label disambiguator	Decides whether a pair of sense articles is a valid interpretation of a pair of labels	prior sense probability - max, current relatedness between senses - max, current	3.2	
Label comparer	Measures relatedness between a pair of labels, based largely on the relatedness of their component senses	relatedness - best sense pair - all sense pairs - weighted by prior sense probability generality concatenation	3.2	
Link disambiguator	Decides whether a sense of a label is a valid interpretation, given the context of other topics mentioned in the same document	prior sense probability relatedness context quality	4.1	
Link detector	Decides whether a (disambiguated) topic mentioned within a document is relevant enough to link to	prior link probability relatedness generality disambiguation confidence occurrence, location, spread	4.2	

Fig. 2. Machine-learned classifiers used within the toolkit.

2.1. Extraction

The toolkit's *extraction* package is responsible for gathering summary data from Wikpedia's XML dumps. We will not describe the process or the data it produces in detail, as users are unlikely to interact with it directly.

The extraction process exploits Hadoop [27], an open source implementation of Google's proprietary GFS file system [11] and MapReduce technology [6]. The former implements a reliable file system distributed across clusters of machines, while the latter supports a programming paradigm that greatly simplifies sharing work between multiple machines. Combining the two yields a powerful platform for processing "big data" that is endorsed by Internet giants Yahoo, Facebook, Twitter, and many others.

This Hadoop-powered extraction process is extremely scalable. Given a cluster of 30 machines, each with two 2.66 GHz processors and 4 GB of RAM, it processes the latest versions of the full English Wikipedia—3.3 million articles, 27 GB of uncompressed markup—in a little over 2.5 hours.[3] The process scales roughly linearly with the size of Wikipedia and the number of machines available.

2.2. Storage, indexing and caching

The *database* package provides persistent, appropriately indexed access to the summarized data and original markup. It depends heavily on Berkeley DB JE [29], an open-source Java-based storage engine maintained by Oracle. Users interact with the database via the *model* package that wraps it (Section 2.3), so we will not elaborate on its content.

The performance of this file-based database is a bottleneck for many applications. Although Berkeley DB caches data to memory if it is accessed repeatedly, it will be very slow for applications that require millions of lookups; such as the semantic relatedness and annotation experiments that follow. Fortunately, the toolkit allows any of its databases to be cached to memory in their entirety, greatly reducing access time. Users can choose which databases are cached, depending on the tasks that need optimizing and the amount of memory available. They can also specify articles that should not be cached, in which case the toolkit does not waste time retrieving them from disk, but instead behaves as if these articles do not exist. Common strategies are to avoid articles that are extremely short or receive few links from other articles.

[3] All statistics are derived from a version of Wikipedia released on July 22, 2011.

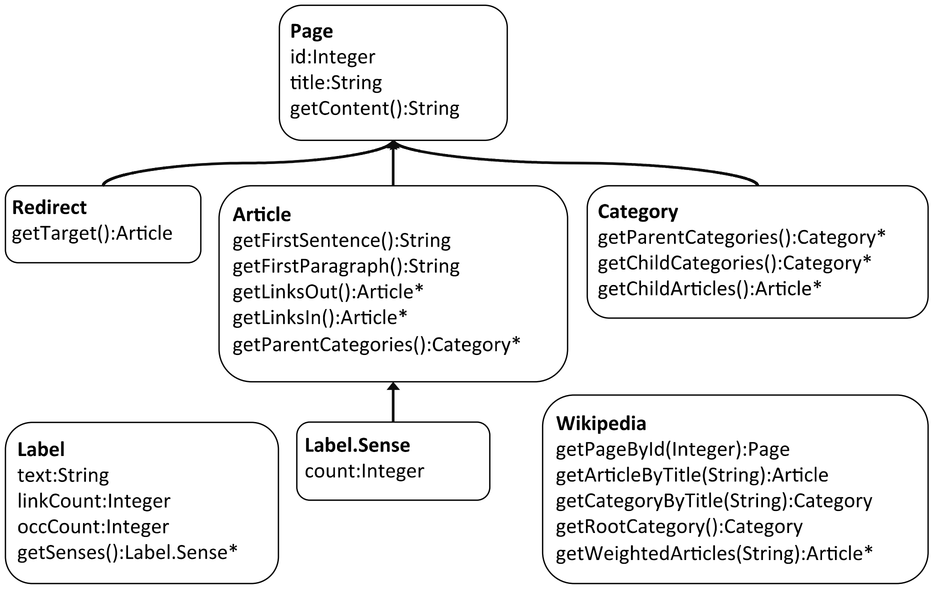

Page
id:Integer
title:String
getContent():String
Redirect
getTarget():Article
Article
getFirstSentence():String
getFirstParagraph():String
getLinksOut():Article*
getLinksIn():Article*
getParentCategories():Category*
Category
getParentCategories():Category*
getChildCategories():Category*
getChildArticles():Article*
Label
text:String
linkCount:Integer
occCount:Integer
getSenses():Label.Sense*
Label.Sense
count:Integer
Wikipedia
getPageById(Integer):Page
getArticleByTitle(String):Article
getCategoryByTitle(String):Category
getRootCategory():Category
getWeightedArticles(String):Article*

Fig. 3. Classes, properties and methods from the *model* package.

Wikipedia contains many of these: they would require significant space to cache and are likely to be of limited use. Users can also specify whether databases should be cached directly, prioritizing speed, or in compressed form, prioritizing space.

2.3. Modeling

The main access point to the toolkit is the *model* package, which abstracts away from the data to provide simplified, object-oriented access to Wikipedia's content and structure. Fig. 3 gives an overview of its most important classes, along with their inheritance hierarchy and some selected properties and methods.

2.3.1. Wikipedia

Wikipedia itself is, of course, one of the more important objects to model. This class provides the central point of access to most of the toolkit's functionality. Among other things, here users can gather basic statistics about the encyclopedia, or access the pages within it through iteration, browsing, and searching.

2.3.2. Page

All of Wikipedia's content is presented on pages of one kind or another. The toolkit models every page as a unique id, a title, a type, and some content expressed as MediaWiki markup. More specific functionality depends on the type of page.

2.3.3. Article

Articles supply the bulk of Wikipedia's informative content. Each article describes a single concept or topic, and their titles are succinct, well-formed phrases that can be used as descriptors in ontologies and thesauri. For example, the article about domesticated canines is entitled *Dog*, and the one about companion animals in general is called *Pet*. Articles follow a predictable layout, which allows the toolkit to provide short and medium length definitions of concepts by extracting the corresponding article's first sentence and paragraph.

Once a particular article is identified, related concepts can be gathered by mining the articles it links to, or those that link to it. However, many of the links do not correspond to semantic relations, and it is difficult to separate useful links from irrelevant ones. Section 3.1 describes how this is resolved by considering links in aggregate, rather than individually.

Articles often contain links to equivalent articles in other language versions of Wikipedia. The toolkit allows the titles of these pages to be mined as a source of translations. For example, the article *Dog* links to *Chien* in the French Wikipedia, *Haushund* in German, 犬 in Chinese, and many others.

2.3.4. Redirect

Redirects are Wikipedia pages whose sole purpose is to connect articles to alternative titles that correspond to synonyms and other variations in surface form. For example, *dogs*, *canis lupus familiaris*, and *domestic dog* redirect to the article entitled *Dog*. Redirects may also represent more specific topics that do not warrant separate articles, such as *male dog* and *dog groups*. The toolkit allows redirects to be mined for their intended target, and articles to be mined for all redirects that refer to them.

2.3.5. Category

Almost all Wikipedia's articles are organized within one or more categories, which can be mined for hyponyms, holonyms and other broader, more general, topics. *Dog*, for example, belongs to the categories *domesticated animals*, *cosmopolitan species*, and *scavengers*. If a topic is broad enough to warrant several articles, the central article may be paired with a category of the same name: the article *dog* is paired with the category *dogs*. This equivalent category can be mined for more parent categories (*canines*) and subcategories (*dog breeds*, *dog sports*). Child articles and other descendents (*puppy*, *fear of dogs*) can also be mined for hypernyms, meronyms, and other more specific topics.

All Wikipedia's categories descend from a single root. The toolkit uses the distance from the root to a particular article or category to provide a measure of its generality or specificity. According to this measure, *dog* is more specific than *carnivores*, which has the same specificity as *omnivores* and is more specific than *animals*.

2.3.6. Label

Wikipedia provides several structural elements that associate articles with terms or surface forms that can be used to denote them. The most obvious elements are article titles and redirects: the article about dogs is given the title *Dog* and has about 30 redirects, including *canis familiaris*, *domestic dog* and *man's best friend*. The links made from other Wikipedia articles to this one provide additional surface forms, because authors tailor anchor text to suit the surrounding prose. A scientific article may contain a link to *Dog* that is labeled with the anchor text *canis lupus familiaris*, while a more informal article may refer to *doggy*.

Article titles, redirects and link anchors are all combined in the toolkit as *Labels*: terms (including phrases) that have been used to refer to the article in some way. Labels encode synonomy and polysemy, because it is possible for an article to be referred to by many labels, and for a label to refer to multiple articles. Because these labels are mined from an extensive corpus of text—that is, the full content of Wikipedia articles—they also have associated usage statistics: 76% of *dog* labels refer to the pet, 7% to the Chinese star sign, and less than 1% to *hot dogs*. These are useful prior probabilities when performing automatic disambiguation (Sections 3.2.1 and 4.1). The toolkit also tracks how often labels are used within links, and how often they are found in plain text. The ratio between these statistics—*prior link probability*—helps to identify terms in new text that are likely to refer to concepts. For example, the probability that the term *dog* is linked in Wikipedia is 6%, while for the term *the* it is only 0.0006%. These statistics are useful for automatically detecting salient topics when they are mentioned in documents (Section 4.2).

By default, labels are indexed and searched without modifying them in any way. They already encode many of the desired variations in letter case (*Dog* and *dog*), pluralism (*dogs*), and punctuation (*US* and *U.S.*), so automatic term conflation is often unnecessary and may introduce erroneous matches—returning *digital on-screen graphic* (or *DOG*) as a match to *dog*, for example. When modification is desirable, the toolkit provides several text processors—case-folders, stemmers, and punctuation cleaners—to re-index the labels. Users can also develop and apply their own text processors.

2.4. Comparison

The toolkit's *comparison* package contains algorithms for generating semantic relatedness measures, which quantify the extent to which different words or concepts relate to each other. According to these algorithms, *dog* is 84% related to *canine*, 72% related to *pet*, and 66% related to *animal*. These measures have a wide range of applications—including word-sense disambiguation [14], spelling correction [3], and document clustering [12]—because they allow terms and concepts to be compared, organized, and perhaps even reasoned about.

The package contains two main classes: *ArticleComparer*, which measures relatedness between pairs of Wikipedia articles, and *LabelComparer*, which measures relatedness between pairs of terms and phrases. Section 3 explains how these classes work and evaluates their performance.

2.5. Annotation

The *annotation* package provides tools for identifying and tagging Wikipedia topics when they occur in textual documents. Documents are processed in five main steps: cleaning them, detecting terms (including phrases) that could refer to topics, resolving ambiguous terms to create a functional mapping from term occurrences to topics, predicting the salience of each topic, and marking up the document with references to these topics. The toolkit is intended to be modular, so separate classes handle each step.

The annotation process begins by feeding a document into a *DocumentPreprocessor*. This abstract class is responsible for identifying regions of the document that should not be altered and from which Wikipedia topics should not be mined. The toolkit provides code for processing HTML and MediaWiki markup, and allows users to develop new preprocessors.

Preprocessors produce *PreprocessedDocuments*. These are copies of the original document that distinguish markup from content, so that the latter can be manipulated and tagged without invalidating the former. Processed documents keep track of text that is ineligible for tagging but may be useful for understanding or disambiguating content, such as the content of HTML *title* and *meta* tags and the anchor text of existing links.

The *TopicDetector* gathers all labels in the document whose prior link probability (Section 2.3.6) exceeds a configurable threshold. This acts as a rough filter to discard terms and phrases that are rarely used within Wikipedia articles to refer to other Wikipedia articles.

The labels gathered by the topic detector are often ambiguous in that they refer to multiple articles—that is, multiple concepts. To resolve ambiguity, each candidate concept from each detected label is fed in turn into a *Disambiguator*, which produces a probability that the concept is relevant for that label in the context of the given document. This disambiguator uses machine learning, and is trained using the article-to-article links that Wikipedia already contains. Each existing link provides one positive example, namely its chosen destination, and several negative examples, namely the destinations that have been chosen for this link text in other articles but not this one. Section 4.1 gives further details of the disambiguation algorithm and its performance.

The combination of *TopicDetector* and *Disambiguator* produces a list of *Topics*. A *Topic* is an extension of the *Article* class which also encapsulates the locations of terms and phrases within the document that refer to it, and various features (described in Section 4.2.1) for judging its significance.

Many of the detected topics will be questionable. At this stage, no effort has been made to distinguish those that are central to the document from ones that are only mentioned in passing. For example, the topic *Dog* has the same status in a document about pet registration as it does in one that describes the weather as "raining cats and dogs". In the latter document, the family movie *Cats & Dogs* is considered no less important than the topic *Rain*.

The *TopicWeighter* abstract class provides a blueprint for classes that are responsible for identifying the importance or relevance of topics within documents. Currently the toolkit provides only one topic weighter: the *LinkDetector*. This class is based on the idea that every existing Wikipedia article is an example of how to separate relevant topics—ones that authors chose to link to—from irrelevant ones. For each topic in a new document, the link detector calculates a weight based on how well it fits the model of what Wikipedians would choose to link to if the document were a Wikipedia article. Section 4.2 gives details of this algorithm.

By this point, many users of the toolkit will have achieved what they need: a list of Wikipedia topics for any given document, weighted by their relevance to it. This list could be used as a concept-based representation of the document in applications like categorization, clustering, retrieval, and so on.

For other applications, the detected topics should be injected back into the original document. To achieve this, the *PreprocessedDocument* and the list of detected *Topics* can be fed into a *DocumentTagger,* which produces a new version of the original document, marked up to identify the topics. The tagger assumes that tags should not be nested, and resolves collisions or overlapping mentions of topics. Imagine, for example, that a document mentioning "cats and dogs" was processed and *Cat*, *Dog*, and *Cats & Dogs* were all given to the tagger as relevant topics. The tagger decides whether to create a single tag that refers to the family movie or two tags that refer to each animal separately.

Like the preprocessor, the tagger can be tailored to generate different styles of markup. *HTMLDocumentTagger* creates HTML links to Wikipedia, while *MediaWikiDocumentTagger* identifies links using standard MediaWiki markup. Users can also develop their own document taggers.

This section has glossed over many details concerning Wikipedia Miner's topic detection and disambiguation algorithms and their performance. These are left for Section 4. Section 5.1.2 gives further details of how to use the package, in the form of a code example.

2.6. Service

The *service* package provides web-based access to a subset of the toolkit's functionality, via REST-style XML-over-HTTP web services. These services are hosted as a publicly available demonstration—we strongly recommend readers try them out—and can be redeployed by anyone who hosts the toolkit. The services are briefly described below. Further details (and hands-on experience) are available at the toolkit's website (see footnote 2).

The *exploreArticle* service takes either the title or unique id of an article, and returns details such as textual definitions, alternative labels, in- and out-links, and representative image icons. The *exploreCategory* provides similar functionality for categories.

The *search* service matches queries against Wikipedia's label vocabulary—breaking complex queries into their component terms as necessary—and lists the different senses each label can refer to. The *suggest* service takes a set of article ids (such as a selection of the senses returned by the search service) and returns lists of related articles, organized by the categories they belong to.

The *compare* service takes a pair of terms, a pair of article ids, or a set of ids. It returns measures of how strongly the terms or articles relate to each other, and can optionally return details of how ambiguous terms have been interpreted, and lists of articles and textual snippets that help to explain the relation.

The *annotate* service takes a snippet of HTML or MediaWiki markup, or a web-accessible URL, and returns the markup augmented with links to the relevant Wikipedia articles. It can optionally return details of how relevant each topic is to the rest of the document.

3. Measuring relatedness

How are *cars* related to *global warming*? What about *social networks* and *privacy*? One of the toolkit's key goals is to supply accurate and efficient algorithms for measuring the strength of such relations by drawing on the background knowledge

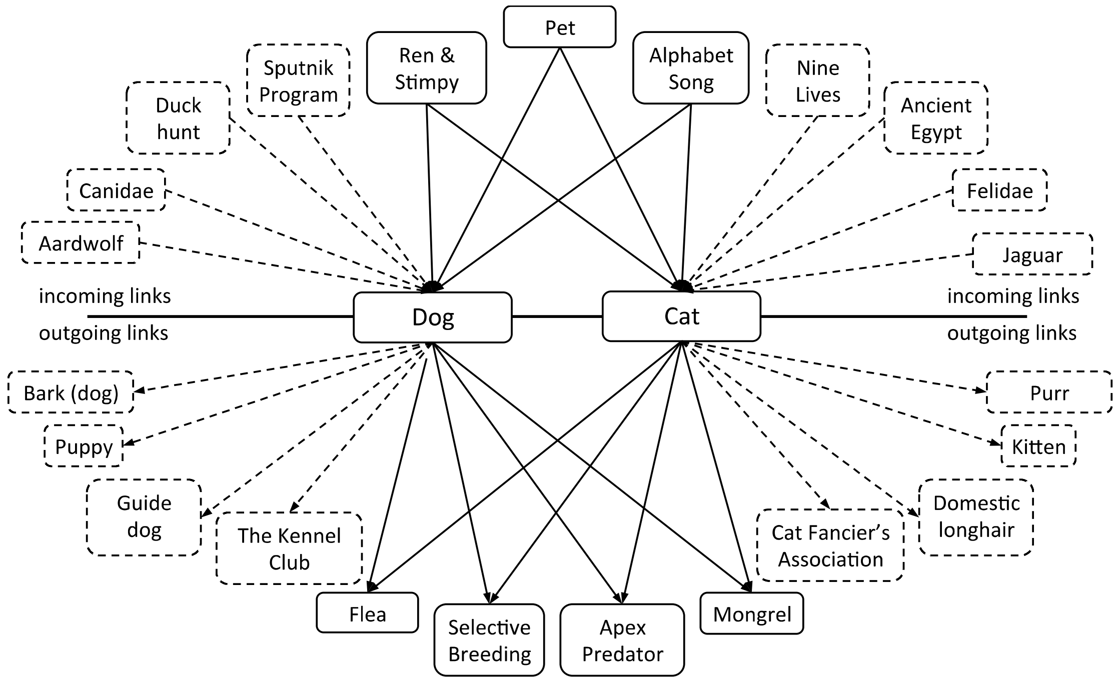


Fig. 4. Measuring the relatedness of *Dog* and *Cat*.

that Wikipedia provides. The measures can be applied to a host of different tasks, including disambiguation, topic detection, document clustering and categorization.

The work described in this section builds directly on Milne and Witten [21], and readers are directed to that paper for further discussion of the task and related work. Section 3.1 explains how relatedness is measured between concepts—that is, Wikipedia articles—while Section 3.2 addresses the more difficult task of extending this to measure relatedness between terms, that is, words and phrases.

3.1. Measuring relatedness between concepts

Wikipedia articles reference each other extensively, and at first glance the links between them appear to be promising semantic relations. *Dog* links to broader concepts like *mammal* and *pet*, to narrower topics such as *working dog* and *chihuahua*, and to related concepts like *domestication* and *dog breeds*. Unfortunately, the article also contains links to many irrelevant concepts, such as *inch*, *color,* and *suntan lotion*. An individual link between two Wikipedia articles cannot be trusted to represent any particular semantic relation. Links are only useful for judging semantic relatedness when they are aggregated together.

Fig. 4 gives a rough impression of how this aggregation works when comparing the article about *dogs* with another about *cats*. If we gather the links made by these articles, shown in the lower half of the diagram, we see some overlap, which indicates that the two concepts are related: both link to *selective breeding, flea,* and around 40 other shared concepts. There are also links to that are unique to each of the concepts (*puppy* and *bark*, or *kitten* and *purr*), which indicate that they are not related—or at least not the same thing. The same is true for the links that are made to each article, shown in the upper half of the diagram.

Our basic approach is to use these sets of common and distinct links to generate features, and combine them using a classifier trained over manually defined ground truth. The classifier learns how to most effectively combine the individual features into a single measure. Section 3.1.1 describes the features, while Section 3.1.2 evaluates their utility for predicting relatedness.

3.1.1. Features

Direct comparison of the sets of links obtained from each of the two pages provides two simple features, *intersection size* and *union size*. Another feature, *normalized link distance,* is modeled after the normalized Google distance measure [4]. The original approach measures the relatedness of terms by using the Google search engine to obtain web pages that mention them, on the basis that pages containing both terms indicate relatedness, while pages with only one or the other suggest the opposite. Our measure does the same with the Wikipedia articles that link to the two articles of interest. Formally, the measure is:

$$nld(a,b) = \frac{\log(\max(|A|,|B|)) - \log(|A\ B|)}{\log(|W|) - \log(\min(|A|,|B|))}$$

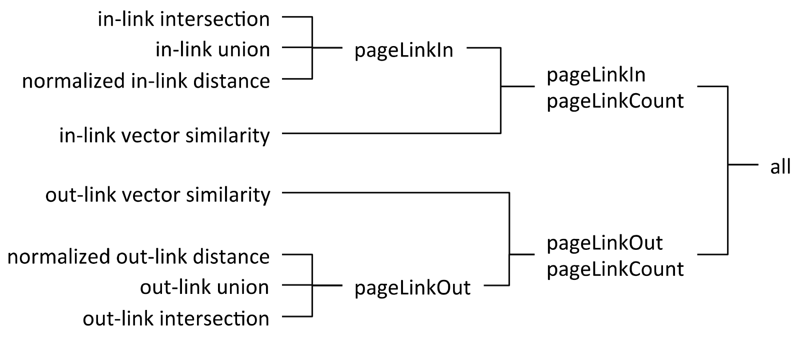


Fig. 5. Features (and their dependencies) for measuring article relatedness.

where a and b are the two articles of interest, A and B are the sets of all articles that link to a and b respectively, and W is the set of all Wikipedia articles.

A fourth feature, *link vector similarity*, is inspired by the near-ubiquitous use of $tf \times idf$ vectors to measure document similarity in tasks like clustering and information retrieval. It uses link occurrences within articles, called $lf \times iaf$ vectors, instead of term occurrences within documents. The *link frequency* (lf) component gives a measure of the importance of a link l_i in an article a_j:

$$lf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where $n_{i,j}$ is the number of times a_j links to l_i, and the denominator is the total number of links in a_j. The *inverse article frequency* (iaf) component measures the general significance of a link:

$$iaf = \log\left(\frac{|W|}{|\{a\colon l_i \in a\}|}\right)$$

where W is the set of all articles in Wikipedia and the denominator is the total number of articles that contain links to l_i. Each article is represented as a vector of $lf \times iaf$ values for all the links within it, and the similarity score between two articles is given by the angle between these vectors. This ranges from 0° if the articles contain identical lists of links to 90° if there are no shared links.

The two measures described above use different sets of links. The normalized distance measure is based on an approach that looks for documents that mention the terms of interest, and has been adapted to use the links made to articles. The vector similarity measure is based on an approach that looks for terms mentioned within two documents of interest, and has been adapted to use the links contained within articles. However, there is no reason why each measure should not be applied to the other link direction. Thus each of the measures described above yields two features, one for in-links and the other for out-links.

Fig. 5 summarizes the full feature set used for measuring relatedness between pairs of Wikipedia articles. Also shown are the databases these features depend on: the *normalized in-link distance*, for example, requires only the *pageLinkIn* database, while the *in-link vector similarity* also requires the *pageLinkCount* database. Practical application of these measures involves a tradeoff between obtaining accurate measures and using as little data as possible, because there is a limit to how much time can be invested and how much data can be cached in memory. This tradeoff is explored next.

3.1.2. Training and evaluation

It should be noted that there are no language dependent components—no part of speech tagging, stemming, or lexicalization—in the algorithm described above. All that is required is an edition of Wikipedia of sufficient size, in the desired language. To demonstrate this, we evaluate the measures in English using the WordSimilarity-353 collection [8], and in German using the Gur 350 dataset [30].

The English dataset contains 353 pairs of terms, each annotated with similarity scores obtained from between 13 and 16 human subjects. These were manually disambiguated by the authors to obtain 293 *article* pairs—as opposed to *term* pairs—and manually defined measures of relatedness between them.[4] Sixty term pairs had to be discarded because at least one of the terms involved did not correspond to an article. The term *defeat*, for example, is only ever used in Wikipedia to refer to specific military encounters; there is no article that corresponds to the general concept.

The German dataset contains 350 pairs of terms, annotated by 8 human subjects. These were manually disambiguated by two native German speakers to obtain 134 pairs of articles; 116 pairs had to be discarded because at least one of the terms involved could not be disambiguated to a suitable sense article, and 100 due to terms that could not be matched to a label at all.

[4] The manually disambiguated WordSimilarity 353 dataset is available at http://www.nzdl.org/wikipediaSimilarity.

Table 1
Performance of the article comparer.

Dependencies	Correlation	1M comparisons (min:s)	Comparisons/s	Cache time (min:s)	Cache space (MB)
English (WordSimilarity 353)					
pageLinksIn	0.71	**00:01**	**784 314**	01:12	275
pageLinksIn+linkCounts	**0.74**	00:15	66 912	01:26	357
pageLinksOut	0.61	00:02	592 417	**01:03**	**269**
pageLinksOut+linkCounts	0.62	00:17	57 844	01:27	352
all	0.72	00:31	32 562	02:00	638
German (Gur 350)					
pageLinksIn	0.56	**00:02**	**624 610**	00:38	154
pageLinksIn+linkCounts	0.63	00:15	65 428	00:46	171
pageLinksOut	0.63	**00:02**	472 813	**00:21**	**92**
pageLinksOut+linkCounts	**0.64**	00:19	53 634	00:24	122
all	**0.64**	00:33	30 600	00:35	226
German nouns (Gur 350 subset)					
pageLinksIn	0.58	**00:02**	**638 162**	00:23	153
pageLinksIn+linkCounts	0.65	00:15	65 612	00:29	170
pageLinksOut	0.67	**00:02**	485 673	**00:19**	**92**
pageLinksOut+linkCounts	0.65	00:18	54 535	00:24	122
all	**0.69**	00:32	31 047	00:35	226

Table 1 investigates the accuracy of the automatically generated relatedness measures using 10-fold cross-validation over the manually disambiguated word similarity collections. It reports the Spearman correlation between the automatic and manually defined ground-truth provided in the dataset. The Gaussian processor classifier [15] provided by the Weka workbench was used throughout, configured with default parameters.

The table also reports the efficiency of the measures, calculated by systematically making 1M pairwise comparisons between 1000 randomly selected articles. For each measure, all the required data—for all articles in Wikipedia, not just the selected ones—was cached in memory, prioritizing speed over space, and the time and memory required to build the cache is also reported. These speeds were attained on a machine with a quad-core 3 GHz processor and 8 GB of RAM. Note that results of previous comparisons were not cached and reused, even though it would be very sensible to do so given that the measures are symmetric. Each of the 1M comparisons was calculated separately.

The greatest efficiency is 780 000 comparisons per second in English, and 625 000 comparisons per second in German. In both languages, this is obtained using only the *pageLinkIn* database—in other words, the *in-link intersection*, *in-link union*, and *normalized in-link distance* features. These features provide reasonably accurate measures in English (71% correlation with human judgements), but comparatively poor results in German (56% correlation). Higher accuracy (up to 74% correlation in English, and 64% in German) can be achieved using additional features, at the cost of lower performance.

The poorer results in German can be attributed to a lack of data (more than half of the terms had to be discarded), and at least partly due to the difficulty and subjectivity of the dataset. On the English dataset, each individual annotator achieves 78% correlation on average with the group. This is reduced to 0.69 correlation for German annotators. Additionally, the dataset includes comparisons across nouns, verbs and adjectives. Our algorithms perform significantly better when the dataset is restricted to noun–noun pairs. Using the full suite of features it achieves 69% correlation to human judgments on this 173 pair subset of the Gur 350 dataset.

3.2. Measuring relatedness between terms

The next step is to adapt the article relatedness measure described above to measure relatedness between terms. Two tasks are involved: deciding which articles should be chosen to represent each term, and combining the relatedness measures between articles, along with some additional features, into a relatedness measure for the original terms. These are tackled in Sections 3.2.1 and 3.2.2 respectively, while Section 3.2.3 provides an overall evaluation.

3.2.1. Features for disambiguating term pairs

Terms are often ambiguous. The *LabelComparer* resolves such ambiguity by selecting a single pair of concepts (that is, Wikipedia articles) to represent each term pair. It takes every possible interpretation—every pair of concepts—and uses a learned classifier to predict a binary class label that indicates whether or not the interpretation is valid.

As Section 2.3.6 explained, Wikipedia provides statistics about how often the associations between *Labels* and *Senses* are made: 96% of *dog* labels refer to the animal, 0.7% to the sign of the zodiac, and so on. These statistics yield three features for the classifier: the *average*, *maximum* and *minimum prior probabilities* of the two concepts. A fourth and final feature is the *semantic relatedness* between the concepts, computed by the classifier described in Section 3.1.

3.2.2. Features for measuring term relatedness

One last classifier is used to combine the relatedness scores of all the potential concepts into a final relatedness score for the original term pair. Three features are the *maximum relatedness, average relatedness* and *weighted average relatedness*

Table 2
Performance of the label comparer.

Dependencies	Correlation	Disambig accuracy	1M comparisons (min:s)	Comparisons/s	Cache time (min:s)	Cache space (MB)
English (WordSimilarity 353)						
pageLinksIn	0.66	**0.81**	01:16	13 235	**02:53**	1281
pageLinksIn+linkCounts	0.72	**0.81**	03:22	4955	03:14	1337
pageLinksOut	0.57	0.71	**00:59**	**17 057**	02:55	**1272**
pageLinksOut+linkCounts	0.65	0.72	01:47	9379	03:00	1340
all	**0.74**	**0.81**	04:10	3999	04:52	1619
German (Gur 350)						
pageLinksIn	0.57	**0.57**	01:06	15 263	01:03	460
pageLinksIn+linkCounts	0.62	0.55	03:52	4309	01:06	481
pageLinksOut	0.61	**0.57**	**00:42**	**24 004**	**00:52**	**458**
pageLinksOut+linkCounts	**0.64**	0.55	01:29	11 216	00:57	482
all	0.62	**0.57**	04:43	3538	01:09	591
German nouns (Gur 350 subset)						
pageLinksIn	0.54	0.54	00:43	23 489	00:56	463
pageLinksIn+linkCounts	0.65	0.56	03:36	4625	00:57	481
pageLinksOut	**0.68**	0.59	**00:27**	**36 711**	**00:48**	**462**
pageLinksOut+linkCounts	0.64	0.58	01:13	13 695	00:59	485
all	0.65	**0.61**	04:22	3815	01:12	587

(weighted by average prior probability) of all the interpretations. Another is the relatedness of the *best interpretation*; the concept pair for which the classifier from Section 3.2.1 had the greatest confidence—in other words, the most skewed probability distribution.

Finally, the classifier also considers cases where two words are closely related because they are frequently concatenated together. For example, *family planning* is a well-known phrase, and consequently *family* and *planning* are given a high semantic relatedness by humans even though their respective concepts are somewhat disjoint. To identify these cases, terms are concatenated and the corresponding label is retrieved. This label's *prior link probability* and *occurrence count* (normalized by taking its logarithm) provide the final two features.

3.2.3. Training and evaluation

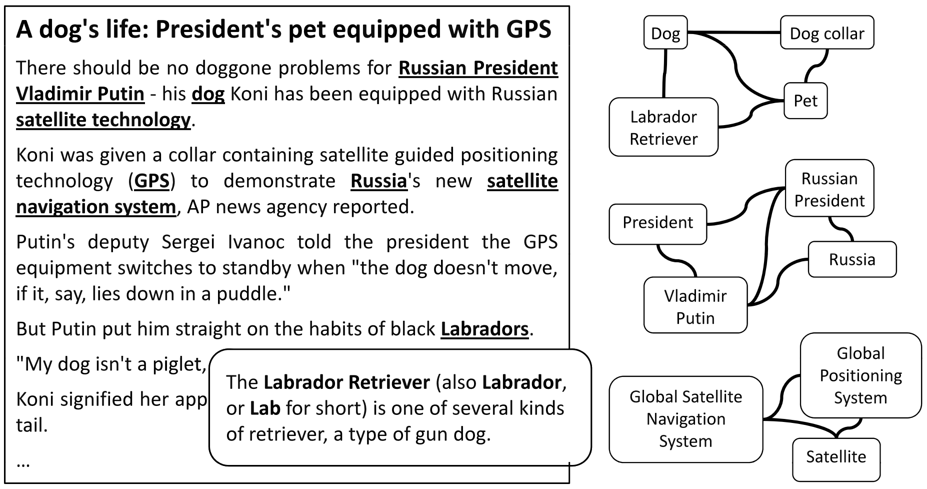
The datasets described in Section 3.1.2 are used to train and evaluate the label disambiguation and comparison classifiers in both German and English, again using 10-fold cross-validation. The results are reported in Table 2, in terms of (a) how often the first classifier chooses the correct sense pair, and (b) the Spearman correlation between the original relatedness and the output of the second classifier. Disambiguation accuracy is only reported over the 293 English and 134 German term pairs that could be manually disambiguated to corresponding Wikipedia articles, whereas the correlation figure is reported over all terms that are found in Wikipedia's label vocabulary; all 353 English term pairs, and 250 of the German pairs. A Bagged C4.5 classifier is used for disambiguation, and a Gaussian processor for comparison; both are configured with the default parameters provided by Weka.

It is important to note the chain of dependency between these classifiers. The label comparer requires a label disambiguator, and both require an article comparer. All three classifiers are retrained on every fold to preserve independence between training and test data. This process is repeated for each of the data dependencies described in Section 3.1.2, in order to again explore the tradeoff between accuracy and the amount of data used. Efficiency is reported by systematically making 1M pairwise comparisons—using the same machine as in Section 3.1.2—between 1000 randomly selected labels. For each set of dependencies, all the required data has been cached in memory, prioritizing speed over space, and the time and memory required to build the cache is also reported. As before, the results of previous comparisons were not cached and reused; each of the 1M comparisons was calculated separately.

For both English and German, highest performance (17 000 and 24 000 comparisons per second, respectively) is achieved using only the links extending out from articles. Taking advantage of additional features improves the accuracy of the resulting measures to 74% correlation with human judgements for English, and 65% for German. As in the previous evaluation, the latter results improve (to 68% correlation) when the dataset is restricted to noun–noun pairs.

In terms of accuracy, our approach is competitive with the current state of the art in applying Wikipedia to measuring relatedness. Over the English dataset, the best performing Wikipedia-based system so far—the Explicit Semantic Analysis [10]—achieves 75% correlation. Zesch et al. [30] utilize the same approach on the German dataset, and report 65% correlation when using Wikipedia as a source of background knowledge, and 68% when using Wiktionary.

The technique utilized by Gabrilovich and Markovitch and Zesch et al. works by distilling the background knowledge provided by Wikipedia (or Wiktionary) into a large matrix in which terms are associated with a weighted list of the articles in which they appear. Our approach, in contrast, relies on up to four separate structures; one that associates labels with a weighted list of articles that represent candidate senses, one or two structures that associate articles with unweighted lists of the articles they link to (or vice versa), and another structure that associates articles with summaries of the number


A dog's life: President's pet equipped with GPS

There should be no doggone problems for **Russian President Vladimir Putin** - his **dog** Koni has been equipped with Russian **satellite technology**.

Koni was given a collar containing satellite guided positioning technology (**GPS**) to demonstrate **Russia**'s new **satellite navigation system**, AP news agency reported.

Putin's deputy Sergei Ivanoc told the president the GPS equipment switches to standby when "the dog doesn't move, if it, say, lies down in a puddle."

But Putin put him straight on the habits of black **Labradors**.

"My dog isn't a piglet,

Koni signified her app

tail.

...

The **Labrador Retriever** (also **Labrador**, or **Lab** for short) is one of several kinds of retriever, a type of gun dog.

Fig. 6. A news article augmented with Wikipedia topics.

of incoming and outgoing links. As Table 2 demonstrates, all of these structures can comfortably fit within a few GB of memory, for even the largest editions of Wikipedia. Consequently, relatedness measures between arbitrary terms can be extracted extremely quickly.

Neither Gabrilovich and Markovitch [10] nor Zesch et al. [30] provide timings in their experiments, but if the term–concept matrices they utilize are too large to be cached to memory—as we would expect—then their approach would be significantly slower. Efficiency is an important consideration, because practical applications for these measures—e.g., clustering large corpora, or the annotation experiments described in Section 4—can involve millions of comparisons. Pre-computing these is simply not practical: even the fastest measure reported in Table 2 would require approximately 2M machine-hours to exhaustively compare each of the 11M labels in the English Wikipedia against each other.

A limitation of our approach is that it is only able to compare terms that have been used as labels for Wikipedia concepts. ESA and similar approaches are able to compare terms that occur anywhere in the Wikipedia corpus, and consequently achieve greater coverage. For example, when Zesch et al. [30] apply ESA to the German dataset, they find it covers 96% of terms. Our own approach achieves only 71% coverage.

4. Understanding and augmenting documents

For most documents, Wikipedia probably knows something about the topics discussed and could likely add additional information. Fig. 6, for example, shows a short news article in which several Wikipedia topics can be identified. These could be made into links to the appropriate articles—or definitions mined from them, as shown for *Labrador*—so that users could easily investigate the topics further. The detected topics can also improve how the document is modeled and understood by automatic systems. These systems—search engines, recommender systems, and so forth—typically represent documents as bags of the words they contain. By consulting Wikipedia, they could instead draw on the concepts these words represent and the rich information that the resource provides to describe and connect them. This is illustrated on the right of the figure, which graphs some of the detected concepts and the relations between them.

The challenge is to detect these topics and create the appropriate links accurately and efficiently. This section describes algorithms to do just that. It briefly explains and evaluates the two steps involved: *disambiguation*, or determining a link's destination, and *detection*, or deciding whether a link should be made at all. Readers are directed to [22] for more information, including comparison with related work and a broader evaluation using news articles.

4.1. Disambiguation

The annotation process begins by gathering overlapping word n-grams (where n ranges between 1 and the maximum label length in Wikipedia) from the document and consulting the label vocabulary described in Section 2.3.6 to ascertain which terms and phrases correspond to concepts in Wikipedia. This presents an immediate problem, because labels are often ambiguous: the same label denotes multiple articles. Ambiguity must be resolved by selecting a single most-appropriate article to represent each label.

Fortunately, there is an abundance of training data for this task. For every existing link within every Wikipedia article, a human editor has manually—and probably with some effort—selected the correct destination to represent the intended sense of the anchor text. This provides millions of manually defined ground-truth examples to learn from. The toolkit provides a machine-learned disambiguator, which can be trained on existing Wikipedia articles and applied to any textual document. The features that inform this disambiguator are described in Section 4.1.1; its performance is evaluated in Section 4.4.

4.1.1. Features

The document topic disambiguator resembles the term pair disambiguator described in Section 3.2.1, and works with similar features. There is a subtle difference, however, in what these two classifiers attempt to predict. The term pair disambiguator considers two senses at a time and produces a probability that their combination is a valid interpretation of the original pair of terms. The document topic disambiguator, however, considers just one sense of one label in the document, and produces a probability that it is valid. Moreover, the document topic disambiguator can bring far more context to bear on the problem.

The two main features of the document topic disambiguator are again *prior probability* and *relatedness*. The first feature remains unchanged, except that there is now only one sense to consider and thus no need for average, maximum and minimum values. The second feature is expanded to take into account the rich context provided by the surrounding document.

The context is obtained by gathering all concepts that relate to unambiguous labels within the document. These are weighted by the average of their link probability, to emphasize terms that are more often used to represent concepts, and their average relatedness to other context terms, to emphasize concepts that relate strongly to the document's central thread. The *relatedness* feature of a candidate concept is its weighted average relatedness to all the context concepts.

A third feature—*quality of context*—is used to adjust the balance between *prior probability* and *relatedness* from document to document. It is simply the sum of the weights of context concepts, as described above.

The disambiguation classifier does not actually choose the best sense for each term. Instead it considers each sense independently, and produces a probability that it is valid. If strict disambiguation is required, as when choosing the destination for a link, the sense with the highest probability should be selected. If more than one sense may be useful, as when representing the document as a graph, all senses whose probability of being valid exceeds 50% (senses that are more likely to be valid than not) can be used.

4.2. Detection

The disambiguator described above produces associations between terms in the document and the Wikipedia articles that describe them. Many of the associations will be questionable, because no effort has been made to separate topics that are central to the document from those that are mentioned only in passing. The final task is to choose which of these associations are sufficiently relevant to retain.

As before, there is an abundance of training data. For each article in Wikipedia, authors have manually selected other articles that are likely to be sufficiently interesting to the reader that they are worth linking to. The toolkit provides a link detector that learns to replicate this selective cross-linking behavior.

To train the detector, a Wikipedia article is stripped of markup and automatically processed by the steps described earlier. This produces a set of automatically identified Wikipedia articles, some of which worth linking to, which provide training instances for a classifier. Positive examples are the articles that were manually linked to, while negative ones are those that were not. Features of these articles—and the places where they were mentioned—are used to inform the classifier about which topics should and should not be linked.

4.2.1. Features

The *prior link probabilities* (Section 2.3.6) of the labels from which topics are mined provide a general indication of the likelihood that each label represents a concept. Each training instance may involve several labels, because a single topic can be mentioned multiple times using different synonyms, and these are combined to produce two features: *average* and *maximum prior link probability*.

Two further features come from the disambiguation classifier described earlier. First, the disambiguator yields a *disambiguation confidence* for each candidate concept, rather than a simple a yes/no judgment of whether the concept is a valid sense of the term, and this is used as a feature to give the most likely senses a greater chance of being linked. Second, the *weighted average relatedness to context* feature calculated previously is used to gain a sense of how strongly the candidate links relates to the document's central thread.

Generality measures the length of the path from the root of the category hierarchy to the concept. This allows the classifier to distinguish specialized topics that the reader may not know about from general ones that do not require explanation.

The remaining features are based on the locations where topics are mentioned, that is, the labels from which they were mined. *Frequency* is an obvious choice, since the more times a topic is mentioned the more important and link-worthy it is. *First occurrence*, *last occurrence*, and *spread*—the distance between first and last occurrence—are also helpful, since important topics tend to be discussed in introductions, conclusions, and consistently throughout documents.

4.3. Configuration

Both the disambiguation and detection classifiers use an initial parameter as a rough first pass to remove candidate senses and links that are unlikely to be valid.

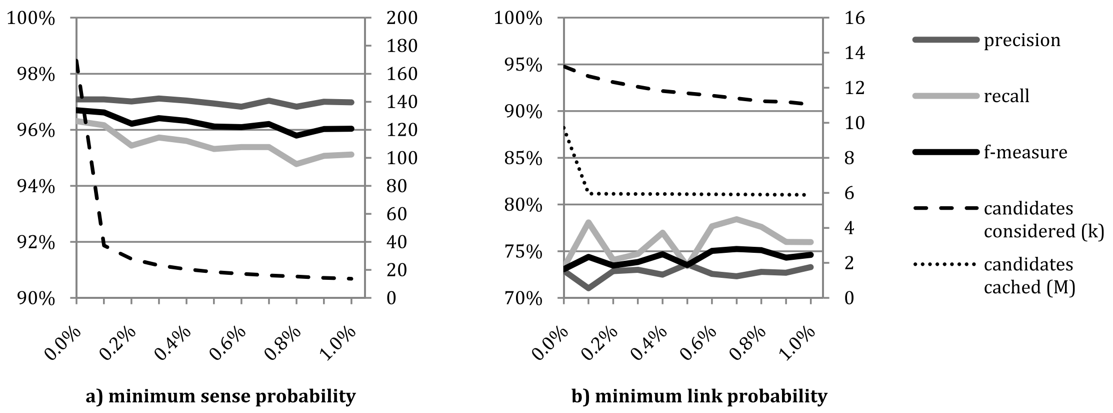


Fig. 7. Impact of thresholds for disambiguation and detection.

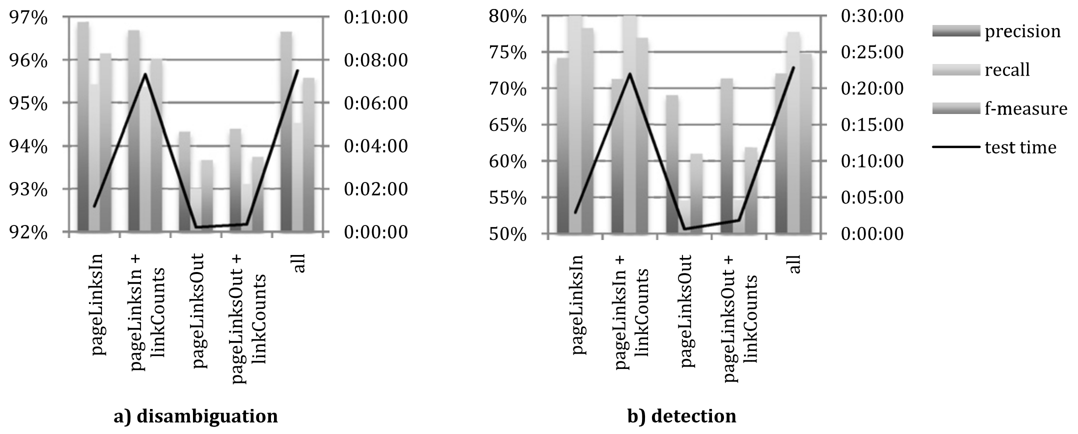


Fig. 8. Impact of relatedness dependencies.

The disambiguation algorithm uses the prior probability of a sense. As illustrated in Section 2.3.6 for *dog*, terms often have extremely unlikely senses (e.g. *hot dog*) that can be safely ignored. The distribution follows the power law: the vast majority of links are made to just a few destinations and there is a long tail of extremely unlikely senses. Jackson, for example, has 230 senses, of which only 31 have more than 1% chance of occurring. If all these are considered they must each be compared to all the context terms. Much speed is gained by imposing a threshold below which all senses are discarded.

Fig. 7a plots the precision, recall, and f-measure of the disambiguation classifier as the minimum sense probability threshold is adjusted between 0% and 1%. At all points, the classifier is trained on a constant set of 100 random English articles, and tested on 50 articles. Also plotted—on a secondary y-axis—is the number of senses considered by the algorithm during the testing phase. The precision of the classifier is robust against changes to the threshold, with recall dropping only slightly as it increases. In return, the work the classifier must perform is dramatically reduced, with the number of senses considered during testing dropping from 170 000 to 40 000 when a threshold of 0.1% is applied. Given that further increases to the threshold provide diminishing returns, future experiments use a minimum sense probability of 0.1%

In a similar fashion, the detection classifier uses prior link probability (Section 2.3.6) to discard unlikely candidate labels. Fig. 7b plots the performance of the detection classifier as the minimum link probability threshold is adjusted between 0% and 1%. As before, the classifier is trained on a constant set of 100 random English articles, and tested on 50 articles. Also plotted is the number of candidate labels (in thousands) considered by the algorithm during the testing phase, and size of Wikipedia's full vocabulary of labels (in millions), which must be cached to memory. As was the case for disambiguation, the performance of the detection classifier remains reasonably consistent. There is no dramatic decrease in the number of candidates it must consider; after all, the only candidates that are being discarded are those that the classifier is unlikely to see anyway. Consequently the threshold provides only modest gains to speed. It does, however, greatly reduce the number of candidate labels (and their senses) that must be cached to memory. For this reason, future experiments use a minimum link probability of 0.5%.

Both classifiers rely extensively on inter-article relatedness measures for features, and their complexity is another factor to consider. Figs. 8a and 8b plot the performance of the disambiguation and detection classifiers as different dependencies are utilized for generating the measures. In both figures, the time taken to test each classifier is plotted on a secondary

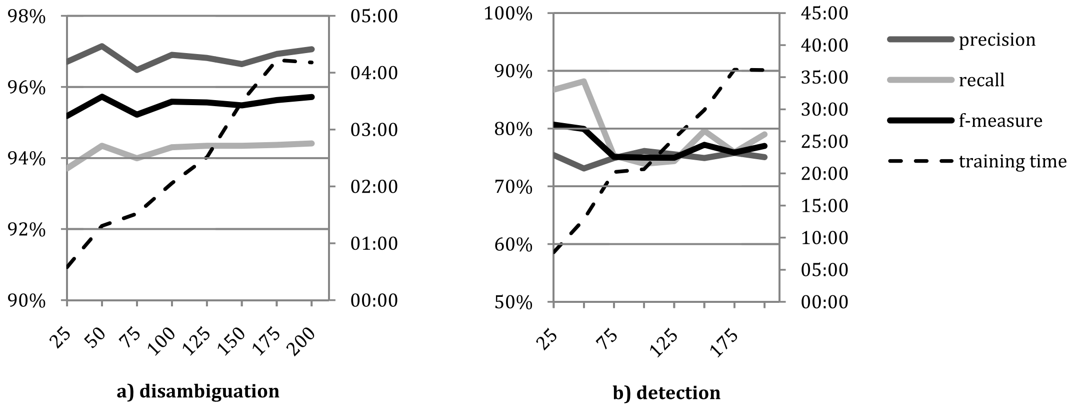


Fig. 9. Impact of training data.

Table 3
Performance of the disambiguator.

	Precision	Recall	f-Measure	Training time	Testing time
English (random)	96.6%	94.9%	95.8%	02:43	02:25
German (random)	95.3%	94.4%	94.8%	02:17	01:50
English (featured)	97.0%	95.7%	96.3%	04:38	04:28
German (featured)	97.8%	96.5%	97.2%	04:03	03:54

Table 4
Performance of the detector.

	Precision	Recall	f-Measure	Training time	Testing time
English (random)	72.9%	74.8%	73.8%	29:01	18:09
German (random)	78.6%	87.5%	82.8%	15:02	13:18
English (featured)	80.1%	91.6%	85.5%	28:15	28:16
German (featured)	83.1%	87.1%	85.0%	23:12	20:25

y-axis. Again, at all points, the disambiguation and detection classifiers are trained on a constant set of 100 random English articles, and tested on another 50 articles. The article comparison classifiers they rely on are trained on the entire WordSimilarity 353 dataset. Increasing the complexity of the relatedness measure has little effect on recall or precision, and a large detrimental effect on efficiency. Consequently, future experiments use relatedness measures that depend only on *pageInLinks*.

One last factor to consider is the amount of data to utilize for training. Although Wikipedia provides hundreds of thousands of manually constructed links, there is little point in utilizing all of them if they do not result in more accurate classifiers. Figs. 9a and 9b plot the performance of the disambiguation and detection classifiers on 50 randomly selected articles, as successively more articles are used for training. Also plotted is the amount of time required to train the classifier. This increases roughly linearly, as one would expect. The majority of time is spent generating features; a process that is independent from one article to the next. Recall, precision and f-measure stabilize with about 100 training articles, so there appears to be little point in using more.

4.4. Evaluation

Tables 3 and 4 demonstrate the performance of disambiguation and detection in both English and German. The disambiguation and detection classifiers are both trained on 100 random articles from the appropriate language version of Wikipedia, and each tested on separate sets of 100 random articles. To maintain independence between training and testing, none of the articles in these test sets have been used in any of the experiments described so far. The disambiguation and detection classifiers are configured using the thresholds and relatedness dependencies established in the previous section, and utilize an article comparison classifier that was trained on all available article pairs from the appropriate language dataset.

The disambiguator achieves approximately 96% precision and 95% recall in both languages. On the same machine as described in Section 3.1.2, the English version requires less than 3 minutes for training, and a similar time for testing. The German version is slightly faster.

The English link detector achieves 73% precision and 75% recall. The German version performs significantly better, with 79% precision and 88% recall. We can only assume that German Wikipedians are more careful, and consequently provide more consistent training and testing data. There is a marked drop in performance from disambiguation to detection in both languages. This is to be expected, because the detector inherits all of the errors made by the earlier disambiguation

```
 1 Wikipedia w = new Wikipedia(new File("path/to/conf.xml")) ;
 2
 3 Label lblDog = w.getLabel("Dog", null) ;
 4 System.out.println("Senses for Dog:") ;
 5 for(Label.Sense sense: lblDog.getSenses())
 6    System.out.println(" -" + sense.getTitle()) ;
 7
 8 Article artDog = lblDog.getSenses()[0] ;
 9 System.out.println(artDog.getSentenceMarkup(0)) ;
10
11 System.out.println("Synonyms: ") ;
12 for (Article.Label synDog: artDog.getLabels())
13    System.out.println(" -" + synDog.getText()) ;
14
15 System.out.println("Translations: ") ;
16 Map<String, String> trans = artDog.getTranslations() ;
17 for (Map.Entry<String,String> e: trans.entrySet())
18    System.out.println(" -" + e.getValue() + " (" + e.getKey() + ")") ;
19
20 Article[] relatedTopics = artDog.getLinksOut() ;
21 ArticleComparer comparer = new ArticleComparer(w);
22 for (Article rt:relatedTopics)
23    rt.setWeight(comparer.getRelatedness(artDog,rt)) ;
24
25 Arrays.sort(relatedTopics) ;
26 System.out.println("Related Topics: ") ;
27 for (Article rt: relatedTopics)
28    System.out.println(" -" + rt.getTitle()) ;
```

```
Senses for Dog:
 - Dog
 - Dog (zodiac)
 - Canidae
 ...

The dog (Canis lupis familiaris) is a
domestic subspecies of the wolf, a
mammal of the Canidae family of the
order Carnivora.

Synonyms:
 - Canis familiaris
 - Man's Best Friend
 - Doggy
 ...

Translations:
 - Chien (fr)
 - Haushund (de)
 - 犬 (zh)
 ...

Related Topics:
 - Dog Breed
 - American Kennel Club
- Pet
 ...
```

Fig. 10. Java code and truncated output of a simple thesaurus browser.

step. Additionally, determining a link's target is far less subjective than deciding whether the link should be made at all. The time required is also significantly increased, with the English link detector now requiring almost half an hour to train.

The lower two rows in Tables 3 and 4 report on the accuracy of our disambiguation and detection classifiers when trained and evaluated on *Feature articles*, which have been peer-reviewed by trusted editors and elevated to a more reputable status. These high-quality articles can be readily identified in English and German by the presence of the *Featured article* and *Exzellent* templates, respectively. Not only are these articles more factually accurate, but they also provide more constant data for our machine-learned approaches to capitalize on. In both languages, the disambiguator achieves an f-measure of approximately 95%, while the detector achieves an f-measure of approximately 85%.

5. Writing Java applications

To give the reader a more concrete description of the toolkit, this section demonstrates it can be used to craft two applications: a thesaurus browser that allows users to explore the concepts and semantic relations encoded in Wikipedia, and an HTML annotator that automatically augments web pages with links to relevant Wikipedia articles.

5.1.1. A thesaurus browser

Fig. 10 shows the Java code and output of a simple thesaurus browser. The first line initializes an instance of Wikipedia, using an XML configuration file that specifies where the Berkeley database environment is located, which databases to cache in memory, and so on. Line 3 queries this instance to retrieve the label for the term *Dog*. The second argument of this method call is an optional text processor.

Lines 4–6 display all the articles to which the term "*Dog*" can refer. As described in Section 2.3.6, the concepts returned by this call are obtained by locating all the occurrences of the term in titles, redirects, and link anchors. These concepts are sorted by the proportion of references that target them, the most likely sense appearing first—in this case, the domestic pet. Line 8 stores this in a new variable, and 9 outputs a brief definition.

Lines 11–13 output the different labels that refer to the article about *Dogs*, which correspond to synonyms in a thesaurus. Lines 15–18 output links to equivalent articles in other language versions of Wikipedia, which correspond to translations of the concept. Line 15 mines links within the *Dog* article for related topics. Not all of them represent useful semantic relations, so lines 20–23 instantiate an *ArticleComparer* to sort articles according to their semantic relatedness to *Dog* (Section 3.1). Lines 25–28 output the result.

```
 1 DocumentPreprocessor preprocessor = new HtmlPreprocessor() ;
 2 ArticleComparer comparer = new ArticleComparer(wikipedia) ;
 3 Disambiguator disambiguator = new Disambiguator(wikipedia, comparer) ;
 4 TopicDetector topicDetector = new TopicDetector(wikipedia, disambiguator) ;
 5 LinkDetector linkDetector = new LinkDetector(wikipedia) ;
 6 DocumentTagger tagger = new HtmlTagger() ;
 7
 8 String html = "<p>This piece of <em>HTML markup</em> has been automatically augmented with links to
   the relevant Wikipedia articles.</p>" ;
 9 PreprocessedDocument doc = preprocessor.preprocess(html) ;
10
11 Collection<Topic> allTopics = topicDetector.getTopics(doc, null) ;
12 System.out.println("All detected topics:") ;
13 for (Topic t:allTopics)
14   System.out.println("\t" + t.getTitle()) ;
15
16 ArrayList<Topic> bestTopics = linkDetector.getBestTopics(allTopics, 0.5) ;
17 System.out.println("Topics that are probably good links:") ;
18 for (Topic t:bestTopics)
19   System.out.println("-" + t.getTitle() + "[" + t.getWeight() + "]" ) ;
20
21 String newHtml = tagger.tag(doc, bestTopics,RepeatMode.ALL) ;
22 System.out.println("Augmented markup:\n" + newHtml) ;
```

```
All detected topics:
  Article (publishing)   Augmented reality   Has Been           HTML        HTML element
  Hyperlink              Markup language     Relevance (law)    Wikipedia

Topics that are probably good links:
  HTML [0.87]            Wikipedia [0.78]          HTML element [0.66]
  Hyperlink [0.63]       Markup language [0.62]

Augmented markup:
 <p>This piece of <em><a href="http://www.en.wikipedia.org/wiki/HTML">HTML</a>
 <a href=" http://www.en.wikipedia.org/wiki/Markup language">markup</a></em> has been automatically
 augmented with <a href="http://www.en.wikipedia.org/wiki/Hyperlink">links</a> to the relevant
 <a href="http://www.en.wikipedia.org/wiki/Wikipedia">Wikipedia</a> articles.</p>
```

Fig. 11. Java code and output of an HTML Wikifier.

The thesaurus browser could take advantage of many other features of the toolkit. It could navigate the article and category links to gather more related topics. These could be classified as broader, narrower, or associated topics by inspecting the hierarchical relationships between them, or clustered using the article comparer. However, Fig. 10 already does a great deal with just 28 lines of code. A little more work to provide input and selection facilities would yield a useful—and large-scale—thesaurus browser.

5.1.2. An HTML annotator

Fig. 11 provides a concrete description of the workflow described in Section 2.5 for annotating documents against Wikipedia's topic vocabulary. It assumes that an instance of the Wikipedia class is ready for use with the appropriate database tables cached to memory.

Lines 1–6 instantiate the classes required to process documents: an *HtmlPreprocessor* for cleaning out markup, an *ArticleComparer*, *Disambiguator* and *TopicDetector* for mining topics from documents, a *LinkDetector* for weighting them, and a *DocumentTagger* for annotating mentions of the topics.

Line 8 specifies a snippet of HTML markup for the purposes of this demonstration. Line 9 uses the *DocumentPreprocessor* to ensure that topics will not be detected from invalid parts of the markup, such as with tags or inside existing links.

Line 11 invokes the *TopicDetector* to gather terms and phrases from the document, match them to Wikipedia topics, and discard irrelevant senses using the *Disambiguator*. The detector is a first-pass effort only, and is quite generous in what it will accept as a topic. The output shown at the bottom of Fig. 11 lists several questionable items, notably *Has Been*, a musical album from William Shatner. Line 16 filters the topics with the *LinkDetector* to include only those that are at least 50% likely to be a link, and lines 17–19 output the result, a much cleaner list of topics.

Line 21 uses the *DocumentTagger* to insert the link-worthy topics into the original markup. Note that one of the link-worthy topics—*HTML Element*—has been discarded during the tagging process. It was obtained from the phrase “HTML markup” and overlaps with the *HTML* and *Markup* topics. The tagger resolves such collisions, choosing the latter two topics because their average link probability outweighs that of *HTML Element*.

6. Related work

This section describes related work on mining Wikipedia. We first review related resources, then summarize the various research problems to which Wikipedia Miner has been applied.

6.1. Alternative and complementary resources

The software system that is most directly related to this work is the Java Wikipedia Library (JWPL) produced by Darmstadt University [30]. The architectures of JWPL and Wikipedia Miner are similar: both comprise databases containing Wikipedia’s summarized content and structure, and a Java API to provide access to it. Much of the API’s functionality is also similar: articles, categories and redirects are represented as classes, and can be efficiently searched, browsed, and iterated over. JWPL also allows the article content to be parsed for finer-grained information. In contrast, Wikipedia Miner does little with article content except to make it available as original markup and plain text. However, the parallelized dump processing, machine-learned semantic relatedness measures and annotation features, and XML-based web services—essentially everything described from Section 2.4 onwards—are unique to Wikipedia Miner.

Another related system is Wikipedia-Similarity, produced by EML Research [25]. This is more concerned with computing semantic relatedness than with modeling Wikipedia. It communicates with MediaWiki, leaving most of the overhead of serving Wikipedia to the original software that produced it. Libraries of Perl and Java code provide access to Wikipedia pages and calculate semantic relatedness. However, the libraries provide less than either of the other toolkits, and the semantic relatedness measure is less accurate than Wikipedia Miner’s [21].

Another software system that mines Wikipedia directly is the WikiPrep PERL script [9]. It produces much the same data—category hierarchy, anchor statistics, etc.—as Wikipedia Miner’s extraction package; it also feeds this information back to produce an augmented version of the original Wikipedia dump. The script is not parallelizable, however, and the use of these files is left entirely to the user.

If one requires a thesaurus or ontology derived from Wikipedia rather than direct access to the original structure, there are several options. Dbpedia [1], Freebase [2], Yago [26], BabelNet [24] and Menta [5] are all large-scale ontologies that have been either partially or fully derived from Wikipedia. The Wikipedia Lab, a special interest group centered at the University of Tokyo, has also produced several resources, including a thesaurus [13] and a bilingual Japanese–English dictionary [7].

6.2. Case studies and applications

The Wikipedia Miner toolkit has been applied to several different research problems. Obviously it has been used for the calculation of semantic relatedness measures [21] and link detection [22]; these are integral components of the toolkit. This section describes some other research that draws directly upon this work.

Section 2.3 identified several similarities between the structure of Wikipedia and the relations expressed in thesauri and ontologies. Milne et al. [20] investigate the extent to which these similarities hold for Agrovoc, a hand-built thesaurus for the agriculture domain, and concluded that Wikipedia had much to offer despite its relatively chaotic structure. This work is continued by Medelyan and Milne [17], who use semantic relatedness measures and label statistics to map descriptors in Agrovoc to articles in Wikipedia with an average accuracy of 92%. The matched articles are used to augment the thesaurus with extended definitions and new synonyms, translations, and other relations. A similar mapping strategy is used by Medelyan and Legg [16] to perform large-scale alignment between Wikipedia and the common-sense ontology Cyc. Their ultimate aim is to create a resource combining the principled ontological structure of the latter with the messier but more abundant information of the former. An evaluation on 10 000 manually mapped terms provided by the Cyc Foundation, as well as a study with six human subjects, shows that performance of the mapping algorithm compares with the efforts of humans.

Wikipedia Miner’s ability to identify topics within documents and generate measures of semantic relatedness between them has been applied to several natural language processing problems. Medelyan et al. [19] apply it to topic indexing, using Wikipedia articles as index terms to describe the document. Huang et al. [12] use similar techniques to improve how documents are clustered. Nastase et al. [23] use the toolkit to build models of the topics that should be discussed in multi-document summaries, and to identify the best sentences to extract to construct them.

7. Conclusions

This paper has presented a highly functional, multilingual toolkit for mining the vast amount of semantic knowledge encoded in Wikipedia. Given the hundreds of papers that have been published in this area over the last few years—Medelyan et al. [18] provide a comprehensive survey—the research community surrounding Wikipedia is clearly in a healthy state.

Unfortunately it is also somewhat fragmented and repetitive. We hope that Wikipedia Miner will allow developers and researchers to avoid re-inventing the wheel, and instead invest their time on the novel aspects of their work.

The toolkit already offers a broadly applicable API for accessing Wikipedia, and state-of-the-art functionality for measuring semantic relatedness and annotating text. We acknowledge that there are gaps: it makes little use of the textual content of Wikipedia, and no use at all of its templates or revision history—both of which have proved fruitful in related research. Our aim in releasing this work open source is not to provide a complete and polished product, but rather a resource for the research community to collaborate around and continue building together.

Readers are encouraged to download the toolkit or draw on the web services it provides. More importantly, we invite them to join the project and help us identify and resolve Wikipedia Miner's shortcomings. Wikipedia itself grew, and continues to grow, out of a community of collaborative authors. We—the researchers and developers who benefit from this shared effort—would do well to follow the same model.

8. Downloads, demos and services

Code, data and online demonstrations of the Wikipedia-Miner toolkit are available at http://wikipedia-miner.sourceforge.net.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: 6th International Semantic Web Conference, Busan, South Korea, 2007, pp. 715–728.
[2] K. Bollacker, R. Cook, P. Tufts, Freebase: A shared database of structured general human knowledge, in: 22nd National Conference on Artificial Intelligence (AAAI), Vancouver, Canada, 2007, pp. 1962–1963.
[3] A. Budanitsky, G. Hirst, Evaluating WordNet-based measures of lexical semantic relatedness, Computational Linguistics 32 (1) (2006) 13–47.
[4] R.L. Cilibrasi, P.M.B. Vitanyi, The Google similarity distance, IEEE Transactions on Knowledge and Data Engineering 19 (3) (2007) 370–383.
[5] G. de Melo, G. Weikum, Menta: Inducing multilingual taxonomies from Wikipedia, in: 19th ACM Conference on Information and Knowledge Management, Toronto, Canada, 2010, pp. 1099–1108.
[6] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.
[7] M. Erdmann, K. Nakayama, T. Hara, S. Nishio, An approach for extracting bilingual terminology from Wikipedia, in: 13th International Conference on Database Systems for Advanced Applications, New Delhi, India, 2008, pp. 380–392.
[8] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, E. Ruppin, Placing search in context: The concept revisited, ACM Transactions on Information Systems 20 (1) (2002) 116–131.
[9] E. Gabrilovich, Wikipedia preprocessor (Wikiprep). Retrieved 29 October 2010 from http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep/, 2007.
[10] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using Wikipedia-based explicit semantic analysis, in: 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 2007, pp. 1606–1611.
[11] S. Ghemawat, H. Gobioff, S. Leung, The Google file system, ACM SIGOPS Operating Systems Review 37 (5) (2003) 29–43.
[12] A. Huang, D. Milne, E. Frank, I.H. Witten, Clustering documents using a Wikipedia-based concept representation, in: 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, Bangkok, Thailand, 2009, pp. 628–636.
[13] M. Ito, K. Nakayama, T. Hara, S. Nishio, Association thesaurus construction methods based on link co-occurrence analysis for Wikipedia, in: 17th ACM Conference on Information and Knowledge Management, Napa Valley, CA, 2008, pp. 817–826.
[14] U.S. Kohomban, W.S. Lee, Learning semantic classes for word sense disambiguation, in: 43rd Annual Meeting of the Association for Computational Linguistics, Ann Arbor, MI, 2005, pp. 34–41.
[15] D. Mackay, Introduction to Gaussian Processes, NATO ASI Series: Computer and Systems Sciences, vol. 168, 1998, pp. 133–166.
[16] O. Medelyan, C. Legg, Integrating Cyc and Wikipedia: Folksonomy meets rigorously defined common-sense, in: Wikipedia and Artificial Intelligence: An Evolving Synergy, Chicago, IL, 2008.
[17] O. Medelyan, D. Milne, Augmenting domain-specific thesauri with knowledge from Wikipedia, in: New Zealand Computer Science Research Student Conference, Christchurch, New Zealand, 2008.
[18] O. Medelyan, D. Milne, C. Legg, I.H. Witten, Mining meaning from Wikipedia, International Journal of Human-Computer Studies 67 (9) (2009) 716–754.
[19] O. Medelyan, I.H. Witten, D. Milne, Topic indexing with Wikipedia, in: Wikipedia and Artificial Intelligence: An Evolving Synergy, Chicago, IL, 2008, pp. 10–24.
[20] D. Milne, O. Medelyan, I.H. Witten, Mining domain-specific thesauri from Wikipedia: A case study, in: Proc. IEEE/WIC/ACM International Conference on Web Intelligence, Hong Kong, China, 2006, pp. 442–448.
[21] D. Milne, I.H. Witten, An effective, low-cost measure of semantic relatedness obtained from Wikipedia links, in: Wikipedia and Artificial Intelligence: An Evolving Synergy, Chicago, IL, 2008, pp. 25–30.
[22] D. Milne, I. Witten, Learning to link with Wikipedia, in: 17th ACM Conference on Information and Knowledge Management, Napa Valley, CA, 2008, pp. 509–518.
[23] V. Nastase, D. Milne, K. Filippova, Summarizing with encyclopedic knowledge, in: 2nd Text Analysis Conference, National Institute of Standards and Technology, Gaithersburg, MA, 2009.
[24] R. Navigli, S.P. Ponzetto, BabelNet: Building a very large multilingual semantic network, in: 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 2010, pp. 216–225.
[25] S. Ponzetto, M. Strube, Knowledge derived from Wikipedia for computing semantic relatedness, Journal of Artificial Intelligence Research 30 (1) (2007) 181–212.
[26] F.M. Suchanek, G. Kasneci, G. Weikum, Yago: A large ontology from Wikipedia and WordNet, Web Semantics: Science, Services and Agents on the World Wide Web 6 (3) (2007) 203–217.
[27] T. White, Hadoop: The Definitive Guide, second edition, O'Reilly Media/Yahoo Press, 2010.
[28] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, third edition, Morgan Kaufmann, Burlington, MA, 2011.
[29] H. Yadava, The Berkeley DB Book, Springer-Verlag, New York, 2007.
[30] T. Zesch, C. Müller, I. Gurevych, Extracting lexical semantic knowledge from Wikipedia and Wiktionary, in: Conference on Language Resources and Evaluation (LREC), Citeseer, Marrakech, Morocco, 2008, pp. 1646–1652.