WikiCat

A graph-based algorithm for categorizing Wikipedia articles

Benjamin Perez
bperez@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Cristoforo Feo
cfeo@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Andrew G. West
westand@cis.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Insup Lee
lee@cis.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

ABSTRACT

In 2005 Wikipedia implemented a category system for the purposes of facilitating navigation throughout the site. Since then, it has grown to 1.5 million categories covering over four million articles. Currently, all categorization of articles on Wikipedia is done by human editors—a task which involves an enormous amount of repetitive work.

This paper details the results of building an automated system that suggests missing category-article links to editors through a browser-based user-interface. This system utilizes the existing corpus of human-categorized articles, as well as available metadata, to provide a ranked list of suggested categorizations. All suggested categorizations are subject to final approval or rejection by a human editor.

1. INTRODUCTION

Fundamentally, Wikipedia consists of a number of content pages, or articles, and a series of categories to which articles might be linked. Categories may have sub categories and pages may belong to several different categories. In a traditional knowledge base this format works well for classifying many different types of objects. If a new object is to be added, the editors will follow whatever criteria they have laid out for placing items into their correct categories. However, the unique nature of Wikipedia as a massively collaborative resource presents an interesting challenge for categorization [16]. In particular, all editing is performed by volunteers who are wildly diverse in terms of geography, education, and professional interests. While this is great for the collection of knowledge in general, it is not ideal for categorization—where there are many opportunities for misunderstanding [24]. Figure 1 shows a subset of the complex web of categories and sub-categories applied to the Wikipedia article on George Washington.

Take, for example, the Wikipedia page for "George Washington". Suppose an editor writes this article and categorizes it under "Presidents of the United States". Later, another author may write an article on "James Monroe" and decide to create a new category for "People from Westmoreland County, Virginia". Unless the author of the "George Washington" article later visits the "James Monroe" page, she will not know that the "Westmoreland County" category even exists and so it will be missing from the "George Washington" page . These kinds of missing links exist in many places on Wikipedia and are difficult to detect through casual browsing. The category system itself suffers from a number of ambiguities that make this a difficult problem to solve. Perhaps the greatest is a fundamental disagreement over how categorizations should work first identified by Thornton [23]. She noted that the two ways of looking at categories, hierarchical and relational, are both currently used by Wikipedia. This makes automated category suggestion relatively difficult with traditional machine learning techniques that attempt to extract a single type of semantic meaning from article categorizations.

In spite of this, it is clear that the category system serves its purpose as a navigational tool tying together articles that share some measure of similarity. Therefore, we plan to utilize an approach that leverages the existing pseudo-hierarchical category structure of Wikipedia as a guide. In fact this is exactly what the recommended procedure for categorizing new articles suggests: "One way to determine if suitable categories already exist for a particular page is to check the categories of pages concerning similar or related topics." [26] The proposed goal of this system is to create a tool which will find these missing category links and bring them to the attention of editors so that they might be quickly and easily added. This will benefit the category system as a whole and potentially simplify the categorization process for a wider audience of editors [8].

2. RELATED WORK

Because Wikipedia is such a rich, diverse corpus of information, a lot of areas of machine learning research use Wikipedia as a training set for a variety of learned models. In this way, Wikipedia is used to improve the quality of machine learning research. This paper proposes the opposite: to use machine learning techniques to improve the quality of Wikipedia.

Automated classification of Wikipedia categories is an area that has been studied from several different angles. The category system of Wikipedia already contains a large amount of extractable information, although it is very loosely structured [14, 19]. This loose structure arises naturally in Wikipedia, since articles are written and edited collaboratively. Furthermore, category links are not applied in a consistent or sys-

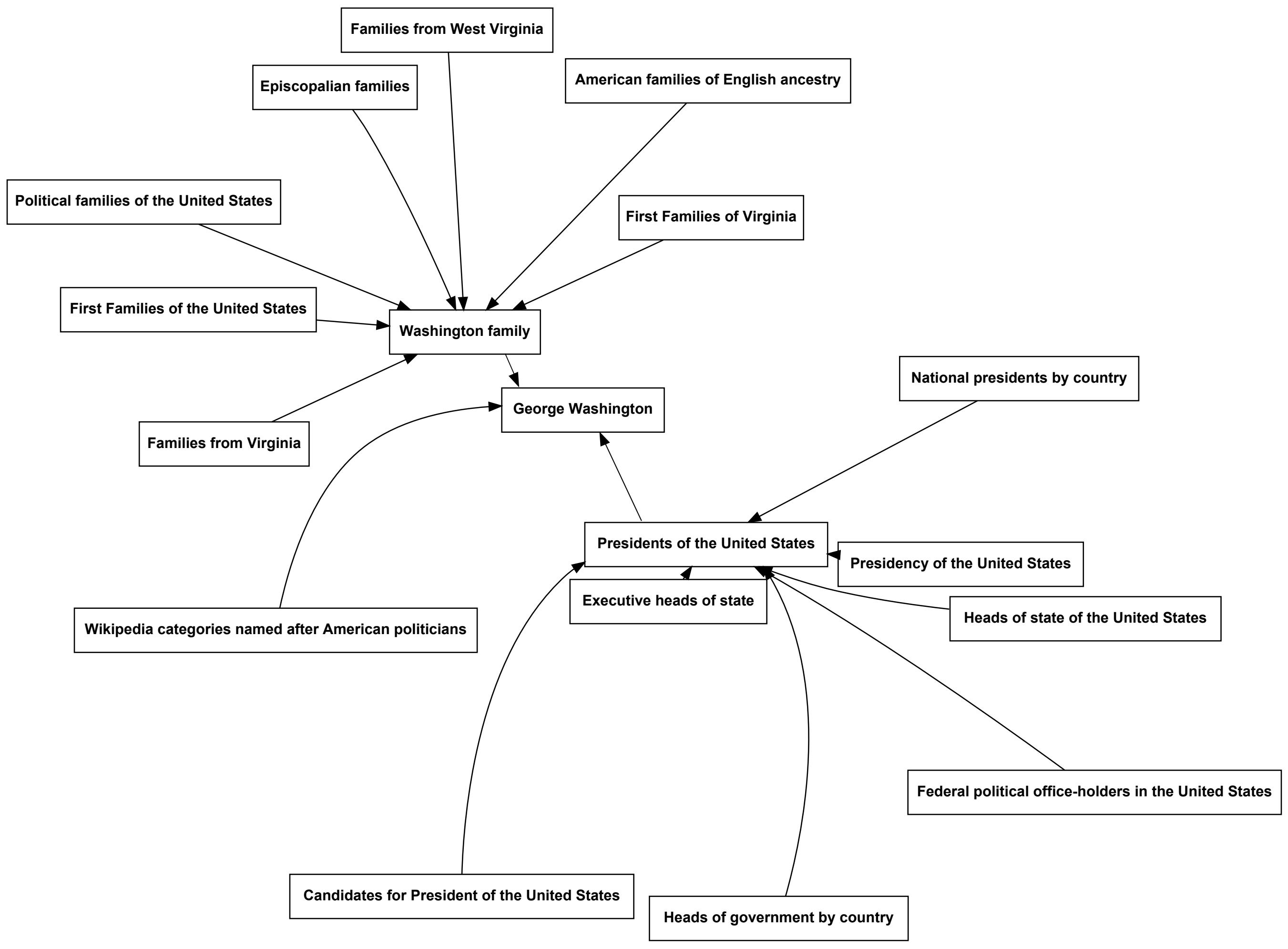
Families from West Virginia
Episcopalian families
American families of English ancestry
Political families of the United States
First Families of Virginia
First Families of the United States
Washington family
Families from Virginia
George Washington
National presidents by country
Wikipedia categories named after American politicians
Presidents of the United States
Presidency of the United States
Executive heads of state
Heads of state of the United States
Federal political office-holders in the United States
Candidates for President of the United States
Heads of government by country

Figure 1: A small part of the category heirarchy for the Wikipedia article on "George Washington".

tematic manner across all of Wikipedia. The most simplistic approaches to categorizing Wikipedia utilize supervised learning. Supervised learning takes as input a set of training examples and their associated labels. In our case, these training examples are Wikipedia articles and their labels are the categories associated with them. Supervised learning uses these labeled training examples to learn a model which can predict the labels of previously unseen examples [22].

There are two broad groups of features which are used as input to supervised learning algorithms: content features and network features. Content features reflect information directly contained within a given page. Examples of content features include textual keywords, infobox information, and metadata associated with a specific article [25]. Network features describe the hyperlinked graph of Wikipedia articles and may contain information such as the number of outgoing links, the number of incoming links, and properties of neighbors in this hyperlinked graph [17].

The use of content features or network features alone has yielded impressive results [9, 22]. Using a combination of both feature types has been shown to be even more effective [9]. Although supervised learning techniques are highly accurate for a few very specific categories [9, 8, 22, 25], they suffer from problems of scale. This is primarily because a classifier must be trained for each and every category. This approach will clearly not scale across the entire category system of Wikipedia, with its large and constantly changing set of categories [8].

Furthermore, previous work demonstrated that supervised learning is best for distinguishing between a few very distinct categories. This assumption does not hold across the vast majority of the category system, where categories may overlap or contain subtle differences [23]. All of these limitations arise because supervised learning works primarily at the level of individually labeled articles, fundamentally limiting the amount of information that can be incorporated from the entire Wikipedia graph, even when network features are included.

Another approach to categorizing Wikipedia is to use semi-supervised learning algorithms. Semi-supervised learning differs from supervised learning in that it uses unlabeled data in the learning process. Many of these methods rely on label-propagation across a graph. The input to a semi-supervised learning algorithm is generally a set of examples. This set of examples is divided into labeled and unlabeled data. Using the assumption that the data are distributed in a way which correlates with their labels, the missing labels in the data can be estimated [6].

Azran [4] describes a generic example of an algorithm—known as the rendezvous algorithm—which can estimate a multi-class distribution of labels across examples. The rendezvous algorithm works by treating each example as a node in a fully connected graph. Each edge in this graph has an associated weight which is given by its entry in a pairwise similarity matrix between all examples called the transition matrix. Labeled example nodes are set to be absorbing states and unlabeled example nodes are set to be emitting states in a Markov random walk. This means that each unlabeled example will emit a particle that chooses a neighbor to move to based on the probabilities defined in the transition matrix. At each new node, this particle chooses its next step based only on the probabilities defined for its current location—the "Markov" aspect of the random walk. Upon reaching a labeled node, the particle is absorbed and the label of the node is recorded. After running a number of random walks for each unlabeled node, the distribution of labels for that node can be inferred by looking at the distribution of absorbing states for its random walks.

Chidlovskii [7] describes the results of a related random walk algorithm on Wikipedia. One of the important concerns he raises is the difficulty of computing a pairwise similarity matrix for a dataset as large as Wikipedia. One way to avoid this problem is to use the graph of hyperlinks between pages as a starting point for the transition matrix. Although the specifics of Chidlovskii's algorithm differ from the rendezvous algorithm, they demonstrate that label-propagation approaches have enormous potential in sparse graph applications such as Wikipedia.

Graph based approaches such as these avoid the scalability bottleneck of needing to train an individual classifier for each category on Wikipedia. Furthermore, they allow us to incorporate all of the information inherent in the hyperlink structure of Wikipedia in our inference [3]. This link structure between articles has been found to be very highly correlated with the link structure of categories [14, 11], leading us to believe that it will be quite useful.

This paper plans to build on the preliminary results described here to develop a label propagation technique which can scale to the entire Wikipedia dataset. This approach will combine the robust, multiclass nature of a technique like the rendezvous algorithm with approaches for dealing with scalability similar to those Chidlovskii describes to produce a superior classification system.

3. SYSTEM MODEL

The input to the WikiCat system is a dump of the article, article links, category, and category links tables of Wikipedia. This raw data is then fed into the first portion of the system: filtering.

The first task of the algorithm is to filter out articles and categories which are administrative. Much of the content on Wikipedia is not directly viewable by end users and is dedicated to organizing the large number of people and ideas engaged in its maintenance. While Wikipedia has a notion of the "Main namespace" to differentiate content articles from administrative ones, there is no similar notion for categories. Suggesting administrative categories is outside the scope of this project and is already performed to some extent by the automated services run by Wikipedia. Since its creation Wikipedia‘s editors have enforced strict naming conventions for articles and with the advent of the category system, this practice was continued. The algorithm therefore utilizes simple heuristics that filter out any categories whose title matches one of the identified "administrative prefixes". For example, categories which begin with "Articles with", "Articles lacking", "Articles needing", or "All articles with" are safely removed from the dataset before proceeding. Once the data needed for generating suggestions has been sufficiently filtered, individual articles are queued up for categorization by the graph algorithm.

The input to the graph algorithm, shown in Figure 2, is an article which needs category suggestions. The first step in the algorithm is to gather all existing categories for this article. Next, all articles which are also part of this category are found. This set of related articles can vary drastically in size and an attempt to enumerate all of them results in unacceptable performance. Therefore, a random walk of depth one is used to select into this set of articles and return all categories of the chosen article. These found categories are then tallied and the frequency with which they were seen becomes their score. The reasoning for this is that thanks to the work that many Wikipedia editors have already put in, similar articles will be more likely to share categories than two otherwise independent articles.

Once a large enough sample of category suggestions has been generated, this data is sent to a server running a web application which can display these suggestions to Wikipedia annotators. The web application presents annotators with a list of categories suggested by the graph algorithm, as well as a view of the Wikipedia article they are attempting to categorize. The annotators can label each suggestion as either good, bad, or an administrative category.

4. SYSTEM IMPLEMENTATION

Filtering out administrative content from the raw Wikipedia data is necessary not only to remove unwanted items, but also for performance reasons. While it is possible to screen the data as it is read into the graph algorithm, this results in long delays as the database searches through a large number of unneeded entries. Instead, the data is prefiltered in the database by creating new tables that are populated based on the determined heuristics. The results of this process are listed in the table provided below. The number of categories and pages in the dataset decreased by reasonable amounts after filtering (3.4 and 1.6 times, respectively) . However, the category links table, which encodes links between articles and the categories they belong to, became 11 times smaller. This results in significantly faster database access, which is important when processing large numbers of articles in parallel.

Table	Unfiltered Size	Filtered Size
Category	1,553,957	462,060
Page	3,233,030	2,014,379
Category Links	65,797,333	5,916,824

The main drawback of this method is that it results in significant downtime when swapping out the data set. As time progresses, new dumps of Wikipedia are generated and it therefore becomes necessary to update the data. Because this data needs to be filtered before it can be used by WikiCat, there is more downtime for the system. This is in contrast to simply loading data and filtering it as it is accessed. While this task must be performed relatively infrequently,

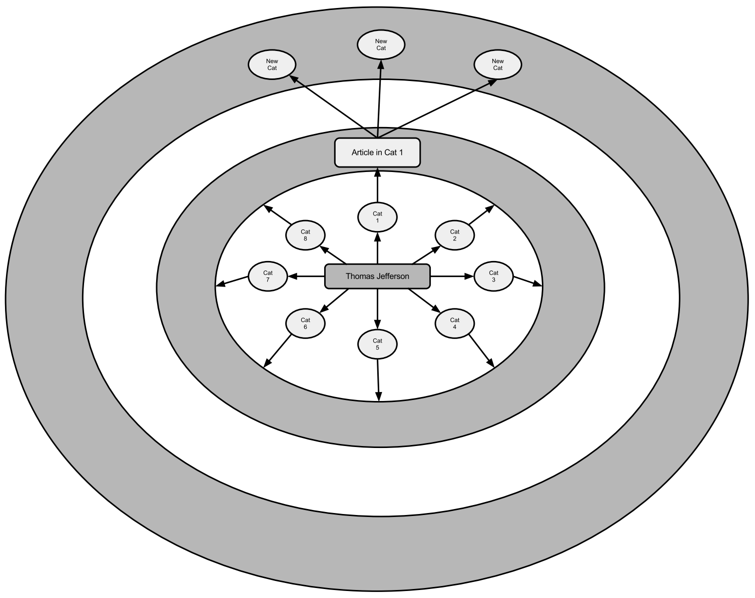

Figure 2: The steps involved in the graph algorithm to generate category suggestions

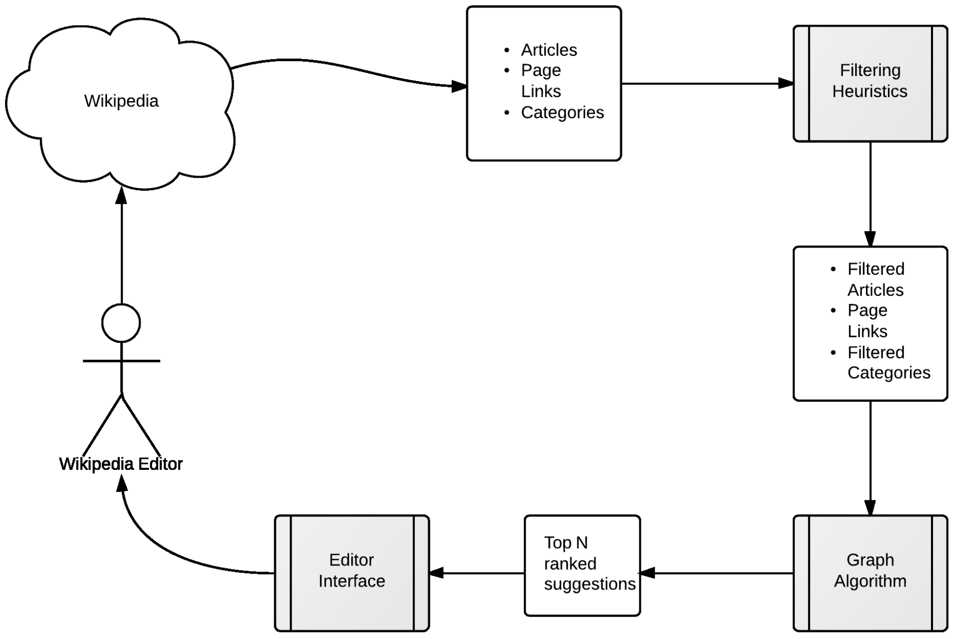

Figure 3: System architecture. Gray blocks are processes, while arrows and white blocks represent data interchance between components.

it is still of concern when running the tool for an extended period, such as in a production environment.

The graph algorithm portion of the system is where the majority of the computational work is done. For this reason, scalability and performance were primary concerns when designing and implementing the graph algorithm.

Due to the very large size of the Wikipedia dataset, the graph algorithm needed an efficient way to store and access data. A relational database was an ideal candidate for this task, since many relational databases are optimized for storing and accessing millions of different data points. WikiCat uses MySQL [1] as its relational database, due to its performance, stability, and widespread use throughout open-source projects.

In addition to relying on a relational database to make data access efficient, WikiCat also uses lazy-loading and caching of articles and categories to only select certain pieces of data as they are needed and to keep previously loaded items in memory. Lazy-loading refers specifically to deferring execution of database queries until just before data is actually needed. This reduces the loading of unnecessary articles or categories and allows database queries to be more efficient. When an article or category is loaded from the database, it is kept in memory to be used again. This caching optimization reduces the need for redundant calls to the database for articles and categories that have been recently loaded.

Aside from storing and accessing the required data, the graph algorithm also needs to efficiently execute the logic required to perform random walks. WikiCat's random walk logic is implemented in Java. Java was chosen because it is a mature programming language with many libraries for accessing databases and making programs scalable. WikiCat uses Java's JDBC library to connect to a MySQL database and execute queries and updates to this database. The lazy-loading and caching optimizations mentioned earlier are implemented in the Java code portions of WikiCat.

WikiCat's random walk logic is multithreaded. Multithreading serves two main functions in WikiCat. The first is to allow WikiCat to reduce the effects of latency when accessing the MySQL database. The second is to parallelize the work being done by the graph algorithm. WikiCat uses a producer-consumer model to process individual articles for categorizations. Each producer and consumer is its own thread and all of these threads are synchronized on a blocking queue of articles to be categorized. Producer threads query a database table of articles needing to be categorized. Then these producer threads create article objects and place them onto a blocking queue. Articles are taken off of the blocking queue by consumer threads.

Consumer threads do all of the actual work involved in categorizing an article. As mentioned in the system model section, this requires performing random walks to find categories of similar articles. Because these random walks are mostly comprised of making successive queries to the database for information, many consumer threads are run in parallel for optimal performance. While one thread is waiting on the results of its query to the database, another thread can be doing computational work. Additionally, both producer and consumer threads take advantage of JDBC's connection pooling features to maximize parallel performance when accessing the database. The number of individual producer and consumer threads can also be varied. In practice, using many more consumer threads than producer threads resulted in the best performance, since producer threads were doing the majority of the computational work.

5. EVALUATION CRITERIA

Evaluation of the algorithm's results may be broken down into two cases: When is it suggesting good categories and when is it suggesting bad categories? To verify that good suggestions are being made, an article is taken and several categories are removed from it. Then, the algorithm is run to see if these removed categories are suggested. To verify that bad suggestions are not taking precedence over good ones, the algorithm is run on a sample of categories and the output is inspected by hand to check whether good suggestions are being made.

6. RESULTS

To check that the algorithm works properly a sample of 100 articles was taken from the data set and several categories were removed from each article. On Wikipedia there is an average of about 12 categories per article, so for each article 3 categories were removed. This left each article with enough remaining categories for the algorithm to run effectively while still providing useful insight into whether good suggestions were being made. 10 trials were performed with the algorithm providing 10 suggestions per article. The result was that of the 300 categories removed per trial, an average of 103 removed categories were suggested. Looking into the results further, it became clear that many of the missed categories belonged to articles which had significantly fewer total categories than the average. This leads to the conclusion that the algorithm is ineffective for articles which have fewer than 4 existing categories, which is to be expected as the algorithm depends on the current category structure to function properly.

In addition to the above automated testing, manual evaluation of results was performed. The algorithm was run for a large number of articles and results were loaded into a web application that would display the article as well as the list of suggestions for it. After annotating these by hand and analyzing the results it became apparent that articles which received good suggestions tended to receive a lot, while it was relatively uncommon for an article to receive only one good suggestion. This indicates that there may be some property which makes an article a good candidate for this algorithm.

7. FUTURE WORK

While the results are promising, this algorithm could be improved in many ways, both in terms of performance and in the quality of results. The easiest way to improve performance would be to increase the number of heuristics used in the filtering stage. Currently there are only seven different title prefixes which are indicative of an administrative category. By increasing this number and removing more administrative categories the graph algorithm will see much faster access times and better performance.

In terms of suggestion quality, the results could be greatly refined through the use of machine learning. To accomplish this, a subset of suggestions from the graph algorithm could be taken and given to a set of annotators. This would be considered a training set. These annotators would look through suggestions and label them as either good or bad. This could be easily accomplished using the web application we developed for analyzing results. Once enough of this feedback is collected it could be paired with features to build a model of what makes a good suggestion. One obvious feature to use in learning the model would be the suggestion's score from the graph algorithm. Other features might include document similarity measures between the article and articles with the suggested category, metadata about both the article and suggested category, and properties of the category graph such as subcategory or supercategory relationships.

8. ETHICS

One of Wikipedia's core principles is that all editing is performed by humans. This tool has the potential to violate this rule or at least come close to doing so. While any production system would make editors manually reject or accept suggestions, it still has the potential to oversimplify the process of categorization. This could cause the category system to change in ways that some Wikipedians believe it should not, as discussed by Thornton [23].

9. CONCLUSIONS

This algorithm describes the basis for a system which assists Wikipedia's editors in performing a relatively time consuming and difficult task, that of categorization. Building off of previous work in the field, it does so by utilizing the existing semantic structure of Wikipedia to infer connections between articles and categories. If anything, the results detailed above show that there is a large amount of untapped information contained in the category system and that further research has the potential to greatly ease the task of editing and encourage more active contribution to the Wikipedia project.

10. REFERENCES

[1] Mysql. `http://www.mysql.com`.

[2] Sisay Fissaha Adafre and Maarten de Rijke. Discovering missing links in wikipedia. In *Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05, pages 90–97, New York, NY, USA, 2005. ACM.

[3] Konstantin Avrachenkov, Paulo Gonçalves, Alexey Mishenin, and Marina Sokol. Generalized optimization framework for graph-based semi-supervised learning. *CoRR*, abs/1110.4278, 2011.

[4] Arik Azran. The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 49–56, New York, NY, USA, 2007. ACM.

[5] Sharon Ann Caraballo. *Automatic construction of a hypernym-labeled noun hierarchy from text*. PhD thesis, Providence, RI, USA, 2001. AAI3006696.

[6] Andrew Carlson, Justin Betteridge, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, SemiSupLearn '09, pages 1–9, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[7] Boris Chidlovskii. Advances in focused retrieval. chapter Semi-supervised Categorization of Wikipedia Collection by Label Expansion, pages 412–419. Springer-Verlag, Berlin, Heidelberg, 2009.

[8] Linyun Fu, Haofen Wang, Haiping Zhu, Huajie Zhang, Yang Wang, and Yong Yu. Making more wikipedians: facilitating semantics reuse for wikipedia authoring. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 128–141, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] Zeno Gantner and Lars Schmidt-Thieme. Automatic content-based categorization of wikipedia articles. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, People's Web '09, pages 32–37, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[10] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[11] Todd Holloway, Miran Bozicevic, and Katy Börner. Analyzing and visualizing the semantic coverage of wikipedia and its authors: Research articles. *Complex.*, 12(3):30–40, January 2007.

[12] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[13] Magnus Manske. Hotcat. `http://en.wikipedia.org/wiki/Wikipedia:HotCat`.

[14] Simone Paolo Ponzetto and Michael Strube. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI'07, pages 1440–1445. AAAI Press, 2007.

[15] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[16] Matthew Richardson and Pedro Domingos. Building large knowledge bases by mass collaboration. In *Proceedings of the 2nd international conference on Knowledge capture*, K-CAP '03, pages 129–137, New York, NY, USA, 2003. ACM.

[17] Prithviraj Sen and Lise Getoor. Link-based classification. Technical Report CS-TR-4858, University of Maryland, February 2007.

[18] Rion Snow. Semantic taxonomy induction from heterogenous evidence. 2006.

[19] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1419–1424. AAAI Press, 2006.

[20] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. 2007.

[21] Omer Sunercan and Aysenur Birturk. Wikipedia missing link discovery: A comparative study, 2010.

[22] Julian Szymański. Towards automatic classification of wikipedia content. In *Proceedings of the 11th international conference on Intelligent data engineering and automated learning*, IDEAL'10, pages 102–109, Berlin, Heidelberg, 2010. Springer-Verlag.

[23] Katherine Thornton. Contentious categories: Discussions of the design of the category system in wikipedia. In *North American Symposium on Knowledge Organization*, 2011.

[24] Katherine Thornton and David W. McDonald. Tagging wikipedia: Collaboratively creating a category system. In *Proceedings of the 2012 ACM Conference on Supporting Group Work*, 2012.

[25] Maksim Tkachenko, Alexander Ulanov, and Andrey Simanovsky. Fine grained classification of named entities in wikipedia. Technical report, HP Laboratories, 2010.

[26] Wikipedia. Wikipedia: Categorization. `http://en.wikipedia.org/wiki/Wikipedia:Categorization#Categorizing_pages`.