Kshitij: A Search and Page Recommendation System for Wikipedia

Phanikumar Bhamidipati Kamalakar Karlapalem

Center for Data Engineering
International Institute of Information Technology
Hyderabad, India
phanikumar@research.iiit.ac.in, kamal@iiit.ac.in

Abstract

Semantic information helps in identifying the context of a document. It will be interesting to find out how effectively this information can be used in *recommending* related documents in a partially annotated knowledge base such as Wikipedia. In this paper, we present a generic recommendation system that utilizes the stored as well as dynamically extracted semantics from Wikipedia. The system generates two kinds of recommendations - for search results and for each page viewed by the user. It explores different meta-information such as links and categories in this process. Our experiments show that the system is able to yield good quality recommendations and help in improving the user experience. Though the algorithms are tested on Wikipedia, external systems that do not have access to structured data can benefit from the recommendations.

1 Introduction

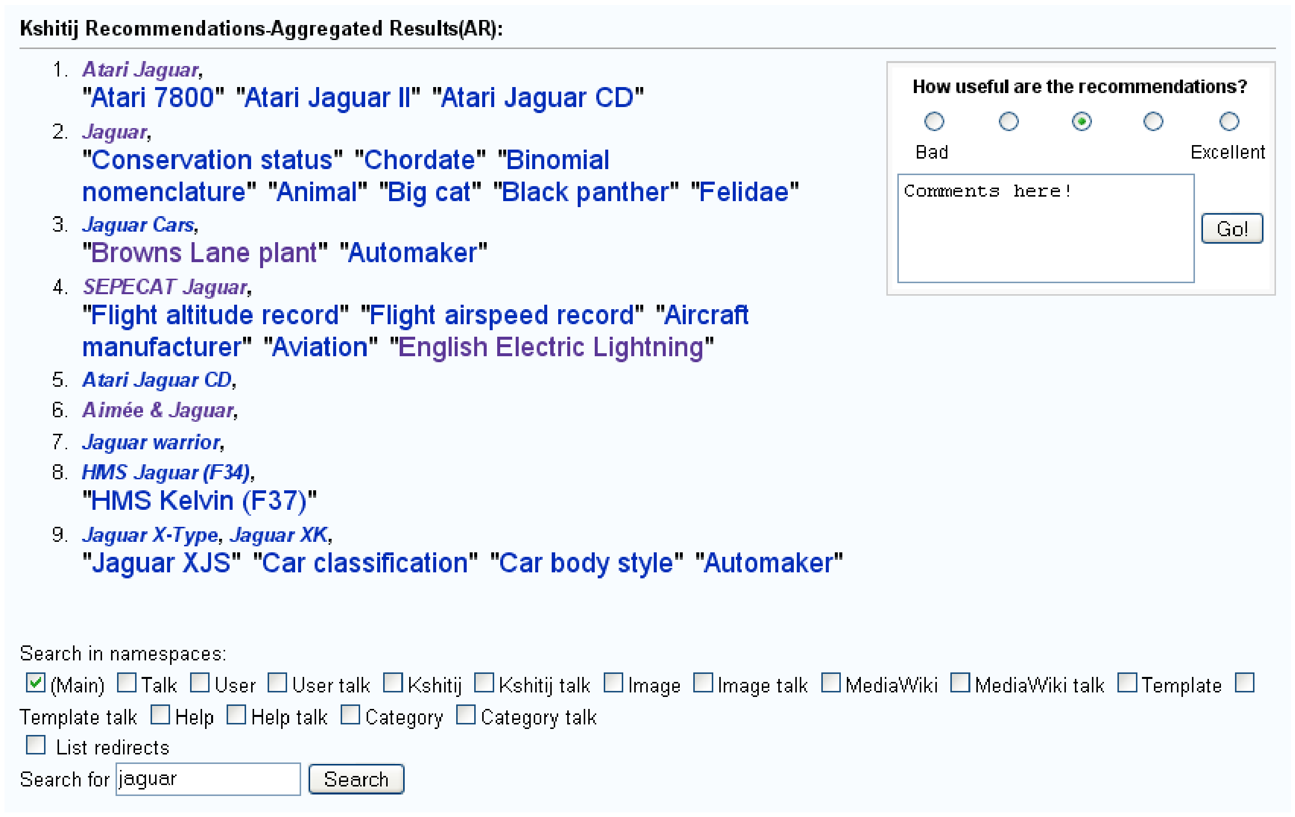
Wikipedia[27] is arguably one of the most popular and extensive knowledge bases available today. Certain aspects of semantics are already present within Wikipedia in the form of Categories, Info-boxes, etc. The fixed and clean Wiki format helps in storing and presenting the data in a consistent way. The links present within each Wikipedia page represent some kind of relation with the base page. This opens up opportunities to be able to mine both the semantics and data from Wikipedia.

Searching for information is one of the most common tasks performed by the end users of a system today, be it an e-commerce application or a knowledge base such as Wikipedia or the World Wide Web. Recently, many efforts have been made towards identifying new paradigms in search, especially with the emergence of new technologies such as the Semantic Web and Web 2.0/3.0. One such new dimension in search is recommendations while search is in progress by making use of semantics present in the data. As the user starts a search task, with a set of keywords, it would be a good value addition if a set of pages that are related can be recommended along with the normal results. This should present the user with various perspectives and bring out different topics to which the results could belong to. Traditional recommendation systems can be applied here, but they work on a specific domain where the semantics are pre-defined and unambiguous. A generic recommendation system that can leverage the semantics to yield high quality recommendations precisely fits in this paradigm. Our efforts have been concentrated towards building such a system. Note that this is different from the search relevance problem, as the system generates recommendations on top of the search results. These need not be part of the search results.

Many efforts were made in the past to use the Wikipedia semantics to formulate an ontology. Yago[25] is one such light weight, extensible ontology constructed by mining Wikipedia and unifying it with WordNet. It is a set of facts in the form of a 3-tuple $< E1, R, E2 >$, indicating that entities E1 and E2 are related to each other by the relation R.

In this paper, we present Kshitij, a recommendation system that leverages certain aspects of Wikipedia semantics and provides two services: search recommendations and page recommendations. It uses Yago as the stored knowledge base and extracts additional knowledge dynamically from the Wiki pages. A screenshot of the search recommendations is shown in figure 1. The user supplies keyword(s) as input, which is sent to a search engine to obtain result pages. For each result page, our algorithms are applied to obtain a related page set. The result pages are then grouped based on the related page sets. As shown in the figure, the result pages are displayed first (in *italics*), followed by the group's related pages/recommendations. Note

International Conference on Management of Data
COMAD 2008, Mumbai, India, December 17–19, 2008
©Computer Society of India, 2008



Kshitij Recommendations-Aggregated Results(AR):

1. *Atari Jaguar*,
   "Atari 7800" "Atari Jaguar II" "Atari Jaguar CD"
2. *Jaguar*,
   "Conservation status" "Chordate" "Binomial nomenclature" "Animal" "Big cat" "Black panther" "Felidae"
3. *Jaguar Cars*,
   "Browns Lane plant" "Automaker"
4. *SEPECAT Jaguar*,
   "Flight altitude record" "Flight airspeed record" "Aircraft manufacturer" "Aviation" "English Electric Lightning"
5. *Atari Jaguar CD*,
6. *Aimée & Jaguar*,
7. *Jaguar warrior*,
8. *HMS Jaguar (F34)*,
   "HMS Kelvin (F37)"
9. *Jaguar X-Type*, *Jaguar XK*,
   "Jaguar XJS" "Car classification" "Car body style" "Automaker"

How useful are the recommendations?

Bad ○ ○ [✓ ●] ○ ○ Excellent

Comments here!

Go!

Search in namespaces:

[✓ (Main)] ☐ Talk ☐ User ☐ User talk ☐ Kshitij ☐ Kshitij talk ☐ Image ☐ Image talk ☐ MediaWiki ☐ MediaWiki talk ☐ Template ☐ Template talk ☐ Help ☐ Help talk ☐ Category ☐ Category talk

☐ List redirects

Search for jaguar Search

Figure 1: A screen-shot of the Kshitij Search Recommendations

that multiple result pages can have the same page in their related page list. A screen-shot of the page recommendations is shown in figure 3. When the user visits a page, its identifier is sent as input to our algorithms to obtain recommendations. They will be displayed on top of the page as shown in the figure. We built our modules on top of Mediawiki[14], the software that runs Wikipedia.

The rest of this paper is organized as follows: We first explain the Wikipedia structure and our main idea in section 2.1. We then give the details of our algorithms in sections 2.2 and 2.3, followed by a detailed analysis of the results in section 3. Later, we present the related work in section 4, and conclude in section 5 summarizing the contributions of the paper and future work.

2 Kshitij Recommendations

2.1 Wikipedia Structure

Wikipedia has many built-in semantics. The category structure is one of its simple yet powerful features, and is manually built by its contributors. Each page in Wikipedia belongs to one or more categories. Each category can have multiple pages in it, and the category itself can belong to multiple categories. Apart from this, the traditional link information is accessible in a tuple $< srcid, destid >$ format. Other structures such as Info-boxes, external links, templates, see-also, etc. are easily accessible. Each of these meta-structures can be used in expressing a relationship between different pages. For example, when two pages belong to the same category, it implies that they are related (E.g., *London* and *Berlin* belong to category *Capitals in Europe* and hence are *related* to each other). This way, entire Wikipedia can be visualized as a huge graph, where each page is a node. Relations between pages can be represented by edges between the nodes. As each relation is supported by one type of structure, we can visualize all edges supported by a structure by one edge type/color.

Figure 2 shows a small part of the graph around Jaguar and its related pages. The filled nodes represent the results of a search task performed by the end user using the keyword *'jaguar'*. These nodes are related to other pages in the system that could potentially be of interest to the user. As these relations could be from different semantics, the graph shows the relations and pages as edges and nodes with different types/colors. Pages that are related via categories are shown as dashed and blue, via links are shown as normal and red, and via Yago ontology are shown as bold and green. Out of these, some are most relevant to the end user.

Now, the problem of recommendations can be defined as following: Given a set of nodes S in the

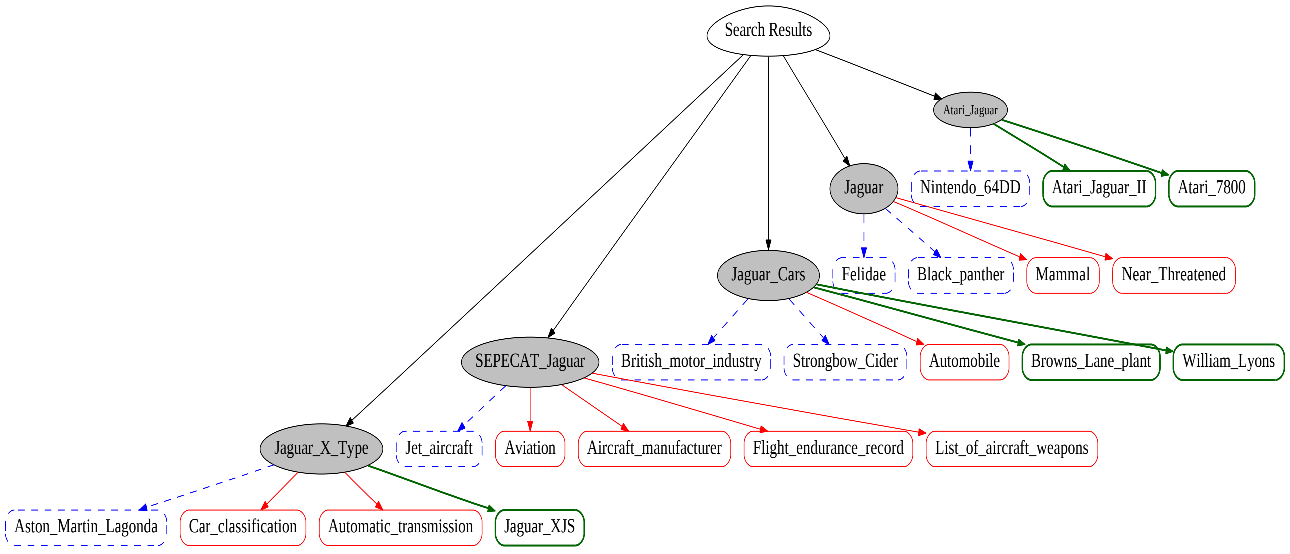
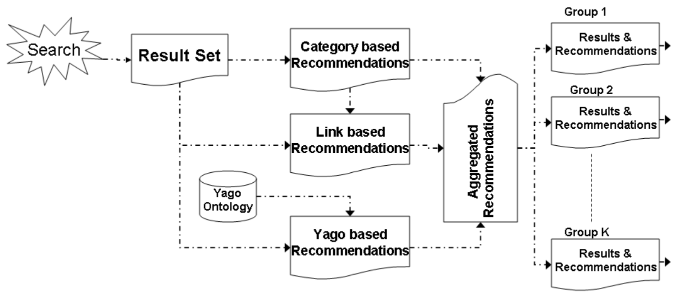


Figure 2: Graph View of Wikipedia Pages

Wikipedia graph, extract the top k nearest nodes to S. The nearness value will be determined based on both the edge weight and edge type/color. This is precisely the problem addressed by Kshitij. The process of identifying recommendations is explained in figure 4. The output of a search task performed by the end user is given to the three algorithms. Additionally, Link Based Recommendations (LBR) uses output of Category Based Recommendations (CBR), and Yago Based Recommendations (YBR) uses the stored knowledge of Yago ontology in identifying the recommendations. The individual recommendations by the three algorithms are aggregated, grouped and presented to the user as shown in the figure 1. The system recommends pages in two places. Search recommendations are given along with the traditional search results when a search task is initiated by the user. Page recommendations are given when the user browses through a page. Three recommendation algorithms are explained in the next section, and the details of cumulative recommendations are presented in the subsequent section. Please note that we deliberately did not depend on the content of a page while defining the edge types because our intention is to emphasize on the importance of the meta-information in identifying the recommendations. As can be seen in our work, the quality of the recommendations is good even without the page content.

2.2 Recommendation Algorithms

In this section, we present three algorithms that are designed to recommend related pages for any given set of result pages. We explain how the individual recommendations of these algorithms are aggregated based on the topics they represent, in the next section.

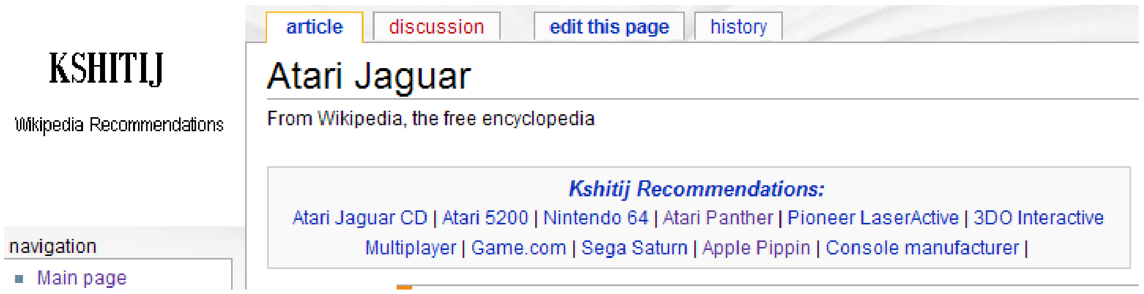


Search
Result Set
Category based Recommendations
Link based Recommendations
Yago Ontology
Yago based Recommendations
Aggregated Recommendations
Group 1
Results & Recommendations
Group 2
Results & Recommendations
Group K
Results & Recommendations

Figure 4: Process of Algorithm Execution in Kshitij

2.2.1 Category Based Recommendations (CBR)

Our first algorithm uses the Category structure present within Wikipedia pages to extract related entities for a given set of search results. Categories are manually entered by Wikipedia contributors and are of two types: categories of the first type are used for management purposes and hold no semantic meaning in our context (For example, the category *All Articles to be Merged*). These are eliminated from our consideration, as they do not hold any semantic meaning. We used certain heuristics to eliminate some of them (For example, categories whose name has an year followed by word "births" or "deaths"). The second type of categories represent the intrinsic relationship present among Wikipedia pages. The main idea here is that if two pages are in multiple categories together, the probability that they belong to the same topic increases. For example, *London* and *Berlin* belong to - *Capitals In Europe* and *Host cities of the Summer Olympic Games*. So, *London* and *Berlin* are consid-



Atari Jaguar

From Wikipedia, the free encyclopedia

Kshitij Recommendations:

Atari Jaguar CD | Atari 5200 | Nintendo 64 | Atari Panther | Pioneer LaserActive | 3DO Interactive | Multiplayer | Game.com | Sega Saturn | Apple Pippin | Console manufacturer |

Figure 3: Header of a Wikipedia Page with Kshitij Recommendations

ered *related*.

Input to the algorithm is a set of pages, returned as a result of any search task performed by the user (indicated as RS). Its output is a set of related pages in the descending order of closeness to the result pages. The steps are explained in algorithm 1.

Algorithm 1 Category Based Recommendations (CBR)

```
Input: $RS$
1: $MPL \leftarrow \emptyset$
2: for all $page$ in $RS$ do
3:   $CL \leftarrow getAllCategoriesOf(page)$
4:   for all $category$ in $CL$ do
5:     $CP \leftarrow getPagesInCategory(category, page)$
6:     $mergeWithMasterPageList(MPL, CP)$
7:   end for
8: end for
9: $CBR \leftarrow \emptyset$
10: for all $page$ in $MPL$ do
11:   for all $resultPage$ in $RS$ do
12:     $count \leftarrow getCount(resultPage, page, MPL)$
13:     $s \leftarrow findSimilarity(page, resultPage, count)$
14:     if $s >= T1$ then
15:       $mergeWithRelatedPages(CBR, page, s)$
16:     end if
17:   end for
18: end for
19: for all $(page, s)$ in $CBR$ do
20:   if $s < T2$ then
21:     $unset(CBR[page])$
22:   end if
23: end for
24: return $CBR$
```

We first construct a master page list (MPL) containing all candidate pages from the categories of the result pages, in steps 1 to 7. In steps 8 to 16, the similarity values between individual pages in MPL and RS are calculated, and those that are below $T1$ are pruned. *count* is the number of categories shared by two pages and will be used in computing the similarity between the pages. The threshold $T1$ represents the minimum similarity value needed for a candidate page to be considered as close to a result page. The pages remaining after pruning are considered as related pages (CBR). The method $mergeWithRelatedPages$ stores the aggregated similarity value for each page in MPL with pages in RS. In steps 17 to 21, the set CBR is pruned further, and the pages whose similarity values are less than $T2$ are removed. The threshold $T2$ represents the minimum similarity value required to be considered in the set CBR. Note that $T1$ refers to similarity between pages in MPL and RS, where as $T2$ refers to the normalized similarity value for a page in CBR. The final output CBR is returned back.

We use the Jaccard coefficient on category set to find the similarity between two pages. In our experiments, we typically use a value of 0.5 for $T1$, which indicates that 50% of the categories that the pages belong to should match, and various values in the range (0,1] for $T2$. The final output of the algorithm is a sorted set of related pages based on the similarity, which indicates the level of closeness with the result pages.

2.2.2 Link Based Recommendations (LBR)

The outward links in each Wikipedia page represent some amount of relation with the page. They could possibly represent different topics that the documents detail about. If multiple pages in a result set are pointing to the same link, it probably means that the link is potentially useful in the current search context. Similarly, if two pages are referred together from the same set of pages, they could be considered as related (competing sports persons, countries in same alliance, etc.). To capture these relations, an Apriori-like algorithm with prioritized transactions is designed. It is explained in algorithm 2. We first construct the Transaction set as $T = RS \bigcup CBR$, where RS is the set of Result pages, and CBR is the output of Category based Recommendations. Adding CBR into the transaction set provides increased support to pages that are linked both from RS and its related pages (CBR). For each page $t \in T$, a priority value p is associated such that $p(t_i) > p(t_j) \ \forall \ t_i \in RS \ and \ t_j \in CBR$. Given this bias, we consider all outward links from each transaction as items, and apply our algorithm to get frequent itemsets.

The k-itemsets that are output by the algorithm

Algorithm 2 Link Based Recommendations (LBR)

```
Input: $T, minSupport, maxLength$
1: $itemsets \leftarrow find-1-itemsets(T)$
2: $k \leftarrow 1$
3: while $true$ do
4:   for all $(txn, priority)$ in $T$ do
5:     for all $itemset$ in $itemsets$ do
6:       if $isInTxn(itemset, txn)$ then
7:         $mergeWithCandidates(ci, itemset, priority)$
8:       end if
9:     end for
10:  end for
11:  $reset(itemsets)$
12:  for all $(itemset, support)$ in $ci$ do
13:    if $support > minSupport$ then
14:      $addItemset(itemsets, itemset)$
15:    end if
16:  end for
17:  if $k == maxLength$ then
18:    $break$
19:  end if
20:  $itemsets = find-k-itemsets(itemsets, ++k)$
21: end while
22: return $itemsets$
```

represent a set of pages that are linked *together* most frequently from the pages in Transaction set T. The input parameter *minSupport* determines the minimum number of pages in T that should support the itemset. We have imposed another constraint to get better focus: each itemset should be supported by at least one page t where $t \in RS$. As mentioned before, priority $p(t)$ also plays a key role in determining the support value. Pages in RS get more priority, thus more support. Another input parameter *maxLength* determines the maximum itemset length to be considered while finding frequent itemsets. Throughout our experiments, we used values of 2 and 3 for this parameter. We consider the individual items (pages) in the topmost frequent k-itemsets as the final output for this algorithm.

2.2.3 YAGO Based Recommendations (YBR)

YAGO[25] is an ontology compiled from Wikipedia's semantic information and unified by WordNet. It consists of a large set of facts that are stored in a particular format. Each fact conveys a relationship between two entities. The entities are Wikipedia pages or WordNet entries or numerical figures. For example, *New Delhi* and *India* are two entities that are related to each other by the *isCapitalOf* relation. This is represented as a single fact by YAGO. Out of the ninety eight unique types of relations in the YAGO version we have used, we removed some relationship types that are not useful in our context (such as *type*) and ranked the remaining based on their strength. Our intention in using this ontology is to find a prioritized set of entities that are related to a given set of Wikipedia pages. We also filtered out entities that do not have an entry in Wikipedia and sorted the list according to a simple weight measure based on the relation strength and match count. The algorithm returns pages that are related most to the *Result Pages* based on the weight measure. This is explained in algorithm 3. It invokes $getYagoPages$ (algorithm 4) for each result page, and merges all results to form a single list of recommendations. The function $getPagesFromFacts(E1, R)$ retrieves all pages P such that the triplet $< E1, R, P >$ is a fact. Similarly, the function $getPagesFromFacts(R, E2)$ retrieves all pages P such that the triplet $< P, R, E2 >$ is a fact.

Algorithm 3 YAGO Based Recommendations (YBR)

```
Input: $RS$
1: $initialize(YBR)$
2: for all $page$ in $RS$ do
3:   $CP \leftarrow getYagoPages(page)$ //Algorithm 4
4:   $mergeWithMasterPageList(CP, YBR)$
5: end for
6: return $YBR$
```

Algorithm 4 Get Yago Pages

```
Input: $page$
1: $initialize(ML)$
2: for all $R$ in $YagoRelations$ do
3:   $list1 \leftarrow getPagesFromFacts(page, R)$
4:   $list2 \leftarrow getPagesFromFacts(R, page)$
5:   $mergeWithMasterList(list1, list2, ML)$
6: end for
7: return $ML$
```

2.2.4 Discussion

Table 1 shows the output of each of the algorithms for a set of pages (indicated by RS) that are obtained as a result of search tasks for various keywords. It can be observed from the results that the algorithms discussed above explore different types of knowledge spaces, leading to different results in most cases. If we visualize these results in the graph as discussed in section 2.1, we see that each algorithm mostly explores the graph along the edges of a specific color to identify related nodes. We now need an algorithm that combines and prioritizes the results yet keeps the semantic information that is brought by individual results. The AR algorithm does the same and is explained in the next section.

2.3 Aggregated Recommendations (AR)

The results of three algorithms – CBR, LBR, and YBR – are combined together to form a final set of recommendations, and we denote this set as Aggregated Recommendations (AR). We take the sorted results

Table 1: Output of individual recommendations

Result Type	Results
amazon	
[RS]	Amazon River, Amazon Rainforest, Amazon.com, Survivor: The Amazon, Volvo Amazon, Rio Negro (Amazon), HMS Amazon, HMS Amazon (F169)
[CBR]	HMS Alacrity (F174), HMS Ambuscade (F172), HMS Arrow (F173), HMS Avenger (F185), Survivor: Pearl Islands, Survivor: Thailand, Survivor: Africa
[LBR]	Website, Industry, Product (business), NASDAQ, Revenue, Public company
[YBR]	Brazil, Volvo 140 Series, Colombia, South America, Peru
Analysis	Many topics are related to Amazon: the river, TV show, company, cars, ship etc. [CBR] explores two topics: ships and the TV Show. [LBR] lists corporate terminology, as two companies are involved here. [YBR] lists countries through which the river Amazon flows.
firefox	
[RS]	Mozilla Firefox, Firefox (disambiguation), Firefox (film)
[CBR]	ViolaWWW, Linux From Scratch, Arena (web browser), Links (web browser), ABrowse, WorldWideWeb, Mac OS X v10.2
[LBR]	Website, Web browser, Operating system, Microsoft Windows, Portal:Free software, List of web browsers, Software license, Mac OS X, GNU General Public License
[YBR]	Clint Eastwood, Freddie Jones
Analysis	Deals with two topics: Web browser and film. The browser dominates in most search results, but [YBR] lists the director and actor of the movie. [CBR] lists different web browsers, and [LBR] returns pages detailing specifications and nature of the browser.
federer	
[RS]	Roger Federer
[CBR]	Theodor Zwinger, Beat Raaflaub, Martina Hingis, Kim Clijsters, Johann Jakob Wettstein, Emil Frey, Ernst Brenner, Edwin Fischer
[LBR]	Tennis, U.S. Open (tennis), Australian Open, The Championships, Wimbledon, French Open
[YBR]	Oberwil, Basel-Country, Laureus World Sports Awards
Analysis	The search has only one topic: a tennis star. All three behave differently in this case. [CBR] returns related people, i.e., other tennis stars and people from Federer's hometown and city. [LBR] lists different Tennis tournaments as these are played/won by him. [YBR], however, just returns an Awards page and Federer's hometown.

for each algorithm as input. The three algorithms discussed above explore three different types of semantics, leading to different results in most cases. One way of aggregating these is to group them based on the topic each result belongs to. We propose a link based approach to accomplish this, which is explained in algorithms 5 and 6. The input to the algorithm is search output (RS), and the three individual recommendations, denoted by CBR, LBR and YBR respectively. First, a cumulative list CL will be formed such that $CL = CBR \cup LBR \cup YBR$.

Given a page $p \in RS$, the pages that are reachable from p, and are in CL might represent the same topic as p. This is the intuition behind looking at the links of each result page. As explained in steps 2 to 7, we explore the neighborhood of each result page p using $expNH$ procedure, and if we find any of the pages in CL, we mark them with its nearness to the result page. We define *nearness* as inverse of the link distance from the result page. Once this procedure finishes, we will have nearness values calculated for each page from all related pages. Its value will be 0 for unrelated/unreachable pages. Now, we define a threshold T on the nearness value to filter the related pages, such that all pages with nearness $< T$ are removed.

Each page in RS can be considered as a point in multi-dimensional space, where each dimension is represented by a recommended page $q \in AR$. If a page $p \in RS$ has $AR_p = \{q_1, q_2 ... q_j\}$ as its recommendations, and $AR = \{q_1, q_2 ... q_j, q_{j+1} ... q_k\}$ is the set of total recommendations ($AR_p \subset AR$), p can be visualized as a point in k-dimensional space, which can be represented as a k-dimensional vector $(d_1, d_2 ... d_j, 0, .., 0)$, where d_i is the distance of the page p from recommendation q_i. Now, these points can be grouped using any clustering algorithm. We apply Agglomerative Nesting (AGNES - A hierarchical clustering algorithm) on the above representation of the results to group the pages. We observed that the value of threshold T determines the quality of the recommendations as well as the groups. We used various values for the threshold T. A detailed analysis is presented in section 3.

Algorithm 5 Aggregated Recommendations

```
Input: $RS, CBR, LBR, YBR$
1: $CL \leftarrow mergeLists(CBR, LBR, YBR)$
2: for all $page$ in $RS$ do
3:     $pageRec \leftarrow CL$
4:     $outLinks \leftarrow getOutLinks(page)$
5:     $expNH(outLinks, pageRec, 1)$
6:     $pageList[page] \leftarrow pageRec$
7: end for
8: return $pageList$
```

Algorithm 6 expNH procedure

```
Input: $outLinks, CL, depth$
1: if $depth > MAX\_DEPTH$ then
2:     return
3: end if
4: for all $page$ in $outLinks$ do
5:     if $exists(page, CL)$ then
6:         $CL[page] \leftarrow 1.0/depth$
7:     end if
8:     $nextList \leftarrow merge(nextList, getOutLinks(page))$
9: end for
10: $expNH(nextList, CL, depth + 1)$
```

3 Results and Analysis

In this section, we present some of the final results of the recommendation system and evaluate their effectiveness. We first explain the metrics used to evaluate the system and then the results for search recommendations, followed by page recommendations.

3.1 Evaluation Methodology

The effectiveness of a recommendation system can be evaluated using metrics like mean squared error, mean absolute error, and precision. In our experiments on Kshitij, we chose mean absolute error (MAE) as the primary metric for estimating the effectiveness of the recommendations. Given a result set RS and corresponding recommendation set AR, the MAE is defined as the following:

$$MAE = \frac{1}{N * K} \sum_{i}^{N} \sum_{j}^{K} |r_{ij} - \hat{r}_{ij}|$$

where $\hat{r_{ij}}$ is the relevance given by the system and r_{ij} is the actual relevance of a given recommendation $q_j \in AR$ to a particular page $p_i \in RS$. N is the total number of result pages and K is the total number of recommendations. In case of page recommendations, the value of N is 1. To benchmark the results given by the system for a keyword or a page, the actual relevance value r_{ij} for each recommendation pair (p, q) is rated manually, where $p \in RS$ and $q \in AR$. Each pair is rated based on whether the recommendation is *relevant, partially relevant*, or *irrelevant* indicating scores of 1.0, 0.5, and 0 respectively. These scores are used to calculate the MAE. A lower MAE value for a keyword or page implies high quality recommendations. We use this metric for both search and page recommendations.

We downloaded the Wikipedia dump of articles, which is freely available for research purposes. The compressed version, without page history and images, was 3.6 GB as on October 2007. We used various tools to load the pages into the MySQL database and used Mediawiki [14] for managing the user interface. We implemented our algorithms in PHP. We also downloaded Yago[25], which was around 780MB as on February 2008, and loaded the ontology to MySQL.

3.2 Search Recommendations

The starting point for search recommendations is the result set given by any search engine. We use Wikipedia's title search feature for obtaining this. For evaluation, a set of keywords are chosen such that they represent different topics in different contexts. Each keyword is sent to the search engine to get the result set, which is then sent as input to the algorithms. The recommendations are displayed to the user along with the search results. Final aggregated recommendations for a set of keywords are shown in table 2. The results and recommendations are grouped together based on their topics. As shown in the table, the result group and the corresponding recommendation group are labeled with the same number.

To evaluate the effectiveness of the recommendations, we obtained the relevance values given by the system for each pair (p, q), where $p \in RS$ and $q \in AR$, and compared them with the manually rated values. Table 3 shows MAE values for different keywords.

There are two perspectives in the evaluation: quality of the recommendations and quality of the groupings. The parameter T has a direct impact on both of them. We use different values of T to calculate MAE for different keywords. The results are plotted in figures 4. First one shows the MAE against T for different keywords, and the second one shows the number of recommendations against T. We observed that the MAE decreases as T increases, which implies that the quality improves. However, the total number of recommendations go down, which implies that we might miss some of the important recommendations. Lower thresholds give many recommendations, but we get many unrelated pages for consideration. Hence there is a trade-off in choosing T here. From our experiments, we observed that a value of 0.4 for T balances both, by fetching moderate number of recommendations while keeping good quality of results.

Apart from obtaining recommendations, the AR algorithm groups the results based on the recommended pages. The groupings are shown in table 2. Pages in the same group are labeled with the same number.

Each group is supported by its recommendations.

Keyword	MAE	Keyword	MAE
real madrid	0.13	amazon	0.22
graph	0.11	jaguar	0.15
obama	0.22	berlin	0.13
king kong	0.19	hyderabad	0.23
casino	0.199	yahoo	0.19
nasdaq	0.167	google	0.14
jazz	0.2	tendulkar	0.25
database system	0.24	operating system	0.13
wikipedia	0.19	india cricket	0.29

Table 3: MAE for various keywords

(For example, the group containing Jaguar X-Type and Jaguar XK is supported by Jaguar XJS, car classification, etc). It can be observed that the pages in a group and their corresponding recommendations represent the same topic. This way, an indirect grouping for widely separated topics is achieved. As the recommendations depend on the semantic information in the data set, the semantics have direct impact on the quality of the groups.

3.3 Page Recommendations

Table 4 lists some page recommendations along with their MAE values and analysis. Page recommendations are shown on top of each page. Whenever the user opens a page in the system, a single element result set is constructed containing the page id. This is sent as input to the recommendation algorithms, and the results are merged using the aggregation algorithm. There is no concept of grouping here because the result set has only one element. The most relevant aggregated results are displayed as hyperlinks in the page, as shown in figure 3. The MAE values are consistently good in this case, mainly because the algorithms extract recommendations from a focused set of pages.

3.4 Discussion

Tables 2 and 4 give an overview of some results to understand the utility of the page and search recommendations. The results can be evaluated to infer the goodness of our recommendations. A more qualitative evaluation is provided for more keywords in table 3. As far as our knowledge goes, there is no existing recommendation system with which we can directly compare ours. The results will be useful to the user if he is looking for an assistance from the system. The system can be designed as an 'on-demand' recommendation generator, i.e., suggesting keywords only if the user requires.

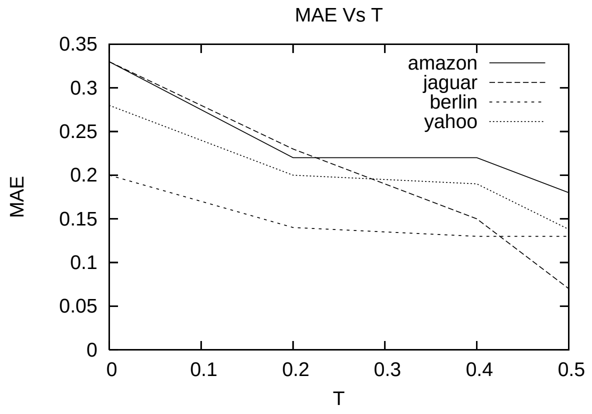
4 Related Work

Since its inception, Wikipedia has been a source of interest for researchers. A content driven reputation system was built on top of Wikipedia in [1], that highlights the content based on its credibility. SuggestBot[5] makes it easy for contributors to find work in Wikipedia by recommending related articles based on similarity of text, connections (through links), and co-editing. It finds similarity between people, using the edit history. In [3], a semantic schema was proposed, to be extracted by analyzing the links between Wikipedia categories. This schema is used to give meaningful suggestions for editing the pages and improve search capabilities. A Wiki-based, community maintained, generic recommender system was proposed in [8], to be useful to build new personalized recommender systems rapidly, without needing to worry about the algorithms, software infrastructure, etc. In [29], the authors presented a new dimension to Wiki – collaborative software development by multiple contributors using Wikis. A prototype called Galaxy Wiki was developed where different contributors can do coding simultaneously. Attempts to automatically identify and enhance semantics in Wikipedia were very successful. In [28], various types of structures were enhanced by using machine learning methods. As the Wikipedia category structure is an important source for semantic information, the category network is carefully visualized in [9] to find its distribution over different topics, edit history, and authors.

WikiRelate! [24] uses Wikipedia knowledge (structure and data) to extract semantic relatedness among different concepts. A different approach was taken in [15] to find semantic relatedness. Only the link structure was mined in the process, which reduces the pre-processing overhead. Similarly, [18] uses sub-tree mining for relation extraction among entities. On the same lines, a huge ontology was built from Wikipedia knowledge, using semantic relation extraction methodologies, in [17].

Recommendation systems are present in many commercial e-commerce systems and are very successful. A good amount of research is done in this area as well, with many recommendation algorithms such as Collaborative Filtering [22]. But not much effort is made to obtain recommendations that depend primarily on semantics. There are some search suggestion algorithms, but they are mostly driven by usage, popularity, etc. of the pages, and not on semantics.

The Web has semantics mostly in the form of hyperlinks that connect a page with others. Finding related pages with such information is interesting. Emerging topics in the Web were identified based on the analysis of co-citations between web pages in [31]. Web communities were identified first using an extended version of KeyGraph [19], and then emerging topics were identified as pages relevant to multiple communities.



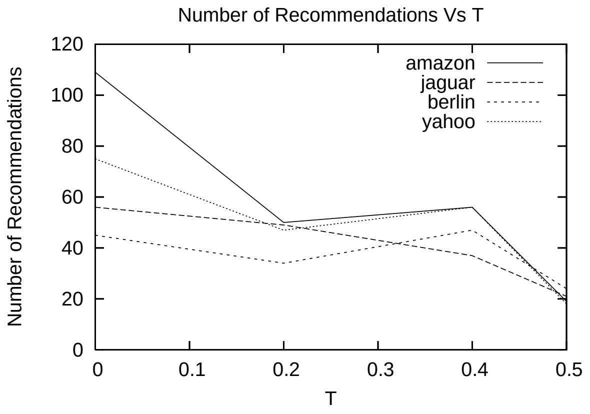


Figure 5: MAE, Recommendation count vs T

[7] introduced scalable algorithms useful to explore the hyperlink structure for similarity information. The authors proposed an extension to SimRank[11] and suggested that vertices within four to five steps provide adequate information for similarity search. In [4], the concept of authorities and hubs was extended to finding related pages. An algorithm called HubFinder was proposed that uses the link structure of the Web to find Hubs around and related to the initial set of pages. [6] proposed algorithms based on concepts of co-citation and hubs to find related pages using the connectivity information.

There were attempts to completely redefine the search task. Koru[16] is a new search interface that can use Wikipedia knowledge base to identify topics and expand queries. The structure of Wikipedia is explored to find document semantics. Similarly, [21] is designed as a natural language search engine that can make use of semantics to discover articles in Wikipedia.

Shashank et al. in [20] defined a new search paradigm called Navigation Aided Retrieval (NAR), where in the user is presented a set of related documents to start with, instead of directly starting with the result pages. Search tasks such as orienteering and open ended search benefit from this new paradigm. Sun J. et al. defined and studied a new search problem: Comparative Web Search (CWS) [26], which helps users to compare pages among a set of topics. However, the topics to be compared need to be supplied by the user. Knowledge of user intent will be useful in enhancing the search process. In [10], three broad classifications of search tasks and an algorithm to classify the user queries to one of the search tasks were studied. [13] proposed three algorithms that proactively capture the information need of the user and augment the search query to get results specific to the context. In [30], a query specific web recommendation system was proposed that identifies the user's unfulfilled needs by analyzing the history and retroactively answers the queries as new results arise.

The concept of search recommendations matches closely with the feature explained in [23]. The authors coined the terms *find similar* and *similarity browsing* to describe the feature, used it as a search tool, and evaluated the retrieval performance of this use-case with the traditional similarity algorithms as the base. However, this differs from our approach as we concentrated on utilizing more of the built-in semantics of the documents for improving the quality of recommendations.

The Apriori algorithm [2] is an important contribution to the data mining research community. The frequent k-itemsets are identified first, and then are pruned based on the minSupport and minConfidence values. A similar idea was applied on text corpora to find out top phrases using n-Grams by Johannes Frnkranz et al. in [12]. This is a very useful result, as it can identify the key terms representing a document. As can be seen in our work, we build our algorithms on these research ideas and apply them to recommend pages for Wikipedia search.

5 Conclusion

Kshitij provides recommendations using the Wikipedia structure. Our experiments show that good quality recommendations can be obtained from a knowledge base such as Wikipedia, even with simple extraction methods and only meta-information.

As part of the future work, we would like to take more structures into consideration for better results. The system currently calculates the recommendations on-demand, so there is good scope to improve the performance. We plan to come up with a strategy that pre-calculates and stores the recommendation set for each page in an additional table. Whenever a user searches for a keyword or views a page, the recommendations get picked from this table and aggregation is applied on them. A separate daemon process runs periodically to update these recommendations so that all page updates are considered. Also, using Wikipedia knowledge to build new recommendation systems or improve existing systems that work on unstructured data adds a lot of value. We believe that this has scope in various application areas, such as the generic World Wide Web search and enterprise search.

References

[1] B. T. Adler and L. de Alfaro. A content-driven reputation system for the Wikipedia. In *WWW*, 2007.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.

[3] S. Chernov, T. Iofciu, W. Nejdl, and X. Zhou. Extracting semantic relationships between Wikipedia categories. In *SemWiki*, 2006.

[4] P. A. Chirita, D. Olmedilla, and W. Nejdl. Finding related pages using the link structure of the WWW. In *International Conference on Web Intelligence*, 2004.

[5] D. Cosley, D. Frankowski, L. Terveen, and J. Riedl. SuggestBot: using intelligent task routing to help people find work in Wikipedia. In *IUI*, 2007.

[6] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks*, 1999.

[7] D. Fogaras and B. Racz. Scaling link-based similarity search. In *Technical report - MTA SZTAKI*, 2004.

[8] D. Frankowski. The WikiLens community-maintained recommender system. In *WikiSym*, 2007.

[9] T. Holloway, M. Bozicevic, and K. Börner. Analyzing and visualizing the semantic coverage of wikipedia and its authors: Research articles. *Complex*, 2007.

[10] B. J. Jansen, D. L. Booth, and A. Spink. Determining the user intent of web search engine queries. In *WWW*, 2007.

[11] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *SIGKDD*, 2002.

[12] F. Johannes. A Study Using n-gram features for text categorization. In *Technical Report - Austrian Research Institute for Artificial Intelligence*, 1998.

[13] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar. Searching with context. In *WWW*, 2006.

[14] MediaWiki. http://www.mediawiki.org.

[15] D. Milne. Computing semantic relatedness using Wikipedia link structure. In *New Zealand Computer Science Research Student Conference*, 2007.

[16] D. N. Milne, I. H. Witten, and D. M. Nichols. A knowledge-based search engine powered by Wikipedia. In *CIKM*, 2007.

[17] K. Nakayama, T. Hara, and S. Nishio. Wikipedia link structure and text mining for semantic relation extraction. In *SemSearch*, 2008.

[18] D. P. T. Nguyen, Y. Matsuo, and M. Ishizuka. Relation extraction from Wikipedia using subtree mining. In *AAAI*, 2007.

[19] Y. Ohsawa, N. E. Benson, and M. Yachida. KeyGraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *ADL*, 1998.

[20] S. Pandit and C. Olston. Navigation-aided retrieval. In *WWW*, 2007.

[21] PowerSet. http://www.powerset.com.

[22] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[23] M. D. Smucker and J. Allan. Find-similar: similarity browsing as a search tool. In *SIGIR*, 2006.

[24] M. Strube and S. P. Ponzetto. WikiRelate! - Computing semantic relatedness using Wikipedia. In *AAAI*, 2006.

[25] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *WWW*, 2007.

[26] J. T. Sun, X. Wang, D. Shen, H. J. Zeng, and Z. Chen. CWS: A comparative web search system. In *WWW*, 2006.

[27] Wikipedia. http://www.wikipedia.org.

[28] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *CIKM*, 2007.

[29] W. Xiao, C. Y. Chi, and M. Yang. On-line collaborative software development via wiki. In *WikiSym*, 2007.

[30] B. Yang and G. Jeh. Retroactive answering of search queries. In *WWW*, 2006.

[31] N. M. Yutaka. Discovering emerging topics from WWW. In *Journal of Contingencies and Crisis Management*, 2002.

Table 2: Results and Aggregated Recommendations

Result from Search Engine	Kshitij Recommendations
yahoo {Yahoo (literature)}^{1}, {Yahoo! Internet Life,}^{2}, {Yahoo!}^{3}, {Yahoo! Messenger, Yahoo! Widgets}^{4}, {Yahoo! Groups}5, {Rogers Yahoo! Hi-Speed Internet}^{6}, {Yahoo Serious}^{7} ***Analysis*** - *Apart from the company Yahoo!, other meanings of the word (such as the Australian movie director name), are well identified and grouped. The company products are also grouped based on their style of operation. We can observe some overlap between them but that is justified, given that certain products (like the messenger) fall into both groups.*	{Australia, Humanoid}^{1} {Yahoo!}^{2}, {Product (business), Revenue, Toronto, NASDAQ}^{3}, {LAUNCHcast, Operating system, Microsoft Windows}^{4} {Yahoo! Voice, Yahoo! Search LAUNCHcast, Yahoo! Messenger, Yahoo! Answers}^{5}, {Industry, Bell Sympatico}^{6} {Young Einstein, Reckless Kelly, Australia}^{7}
amazon {Amazon.com}^{1}, {Amazon Standard Identification Number}^{2}, {Amazon parrot}^{3}, {Survivor, The Amazon}^{4}, {Volvo Amazon}^{5}, {Amazon River Dolphin}^{6}, {Rio Negro (Amazon)}^{7}, {HMS Amazon, HMS Amazon (F169)}^{8}, {Amazon Rainforest, Amazon River, Amazon Basin}^{9} ***Analysis*** - *The results of "amazon" have many topics: river, company, TV show and ship. TV Show and ship are separated well. The river is subdivided into species around the river and the geographical qualities. However, pages belonging to one company are scattered into two groups, due to the too generic recommendations.*	{Public Company, Brazil, Colombia, Industry, United States, Employment, NASDAQ}^{1}, {Industry, United States, Website}^{2}, {South America, Chordate, Bird, Scientific classification, Brazil}^{3}, {United States, Survivor: All-Stars, Brazil, Survivor: Africa, Survivor: Pearl Islands, Survivor: Thailand}^{4}, {Car classification, Automaker, Car body style, Automobile layout}^{5}, {Colombia, Peru, Venezuela, Brazil, Scientific classification, Chordate, Animal}^{6}, {Colombia, Brazil, Venezuela, South America}^{7}, {Royal Navy, HMS Alacrity (F174), HMS Ambuscade (F172), HMS Avenger (F185), HMS Arrow (F173)}^{8}, {Brazil, Peru, Colombia, Bird, South America, Scientific classification, Venezuela}^{9}
berlin {Berlin International Film Festival}^{1}, {Treaty of Berlin}^{2}, {Irving Berlin}^{3}, {Funeral in Berlin}^{4}, {Berlin wool work}^{5}, {German Museum of Technology (Berlin), Berlin, East Berlin, Berlin Wall, West Berlin}^{6} ***Analysis*** - *Places around Berlin city are grouped together. The wool work is a separate topic and is well grouped. The writer Irving is also separated correctly with no German references. Other groupings are fine, but there is no strong support from their recommendations.*	{Berlin, Europe, East Germany, Area, United States, France}^{1}, {Europe, France, German language, Germany, United States}^{2}, {France, Area, United States}^{3}, {World War II, German language, Germany, United States}^{4}, {Hardanger embroidery, Sampler (needlework), Germany, Berlin, Needlepoint, Jacobean embroidery, Machine embroidery, Assisi embroidery, Crewel embroidery, Canvas work}^{5}, {Berlin, Germany, Cold War, Geographic coordinate system, Arms race, Berlin Wall, East Germany}^{6}
jaguar {Jaguar Cars}^{1}, {SEPECAT Jaguar}^{2}, {Aimee & Jaguar}^{3}, {HMS Jaguar (F34)}^{4}, {Atari Jaguar, Atari Jaguar CD}^{5}, {Jaguar X-Type, Jaguar XK}^{6}, {Jaguar, Jaguar warrior}^{7}, ***Analysis*** - *Out of all topics, the game, ship, movie, animal and flight are correctly classified into different groups. Jaguar Cars as a company is distinguished from its models. However, the movie doesn't have any recommendations, and Jaguar warrior isn't grouped correctly. This is due to the lack of enough semantic information.*	{Browns Lane plant, Automaker}^{1}, {Flight altitude record, Flight airspeed record, Aircraft manufacturer, Aviation, English Electric Lightning}^{2}, {HMS Kelvin (F37)}^{4}, {Atari 7800, Atari Jaguar II}^{5}, {Jaguar XJS, Car classification, Car body style, Automaker}^{6}, {Binomial nomenclature, Conservation status, Chordate, Felidae, Animal, Big cat, Black panther}^{7}

Table 4: Output of Page Recommendations

Page	Recommendations	MAE	Analysis
Lufthansa	Germany, Aviation, Airline hub, Airline call sign, Austrian Airlines, Air Canada, Adria Airways, BMI, Airline alliance, All Nippon Airways, Asiana Airlines, Air India, Air China, Ansett Australia, Air New Zealand	0.1	The page is about Lufthansa, a German Airline company. All Airline related pages and Germany are shown as related pages.
Amazon River	Branco River, Rio Negro (Amazon), Brazil, Andes, Peru, Tributary, Colombia, Atlantic Ocean, Ecuador	0.23	Various related pages: Tributaries of Amazon, countries through which the river flows, etc.
Weka	Kea, Conservation status, Scientific classification, Binomial nomenclature, IUCN Red List, Chordate, World Conservation Union, Vulnerable species, Bird, New Zealand, Australia, Carolus Linnaeus	0.22	Weka is an endangered bird species that lives in New Zealand. Recommendations include the bird's country, other similar species, scientific literature, etc.
Pacific Ocean	Pacific Islands, Mariana Trench, Challenger Deep, International Date Line, Australia, Chile, Bathyscaphe Trieste, Alaska, Aleutian Islands	0.21	Fetches countries and islands in the Pacific Ocean.
Salzburg	Vienna, Archbishopric of Salzburg, Augsburg, Austria, Geographic coordinate system, Anschluss, Amsterdam	0.17	Salzburg is a tourist city in Austria. The recommendations fetched majorly include other tourist cities in Austria and Europe.
Jaguar	Big cat, Felidae, Felis, Panthera, Black panther, Animal, Binomial nomenclature, Conservation status, Chordate, Mammal, Carnivora, Carolus Linnaeus	0.18	The page is about the animal Jaguar. The recommendation set includes other similar species and pages detailing its scientific classification.
Hyderabad State	Kolhapur, Delhi Sultanate, List of Indian Princely States, British India, India, Bengal, Deccan, Gujarat	0.17	The page is about the princely state of Hyderabad before Indian independence. Recommendations include other princely states in India, references to the colonial India etc.
Godavari River	Krishna River, Kaveri River, Beas River, Eastern Ghats, Ganges, Brahmaputra River, Indus River, Ganges Delta, Bay of Bengal, Chilka Lake	0.2	Godavari is a river in South India. Recommendations include other rivers in various parts of India and the Bay of Bengal.
DAX	Stock market index, List of stock market indices, CAC 40, Hang Seng Index, Nikkei 225, NASDAQ-100, Dow Jones Industrial Average, Employment, BMW, Allianz	0.19	DAX is a German stock market index. Recommendations include other indices, companies that are listed in DAX etc.
Horlicks	Ovaltine, Hot chocolate, Ribena, Nestle Milo, Maxim's, United Kingdom, World War II, Malted milk, London, GlaxoSmithKline	0.18	It is the name of an energy drink. Related pages include other energy drinks, owning company, etc.