Shanghai Jiao Tong University

Co-occurrence Analysis System in Wikipedia

SE305 Database System Technology

Hao Xu, Xingya Zhao

2015-12-1

1. INTRODUCTION

Wikipedia is a free-access, free-content Internet encyclopedia, supported and hosted by the non-profit Wikimedia Foundation.Of Wikipedia in all languages, the English Wikipedia is the largest with 5,020,669 articles (having reached 5,000,000 articles in November 2015).Wikipedia is ranked among the ten most popular websites and constitutes the Internet's largest and most popular general reference work.

Wikipedia is widely used as source in text mining due to itslarge amounts of data,high accuracy and free access. It offers well-structured free copies of all available contents to interested users. In this project, we designed a database system to show the relationships between anchors and links, between wiki titles and anchors, and between links and words. Our system can serve as the dataset for further text mining research based on Wikipedia.

2. PROJECT OVERVIEW

2.1. Database Analysis

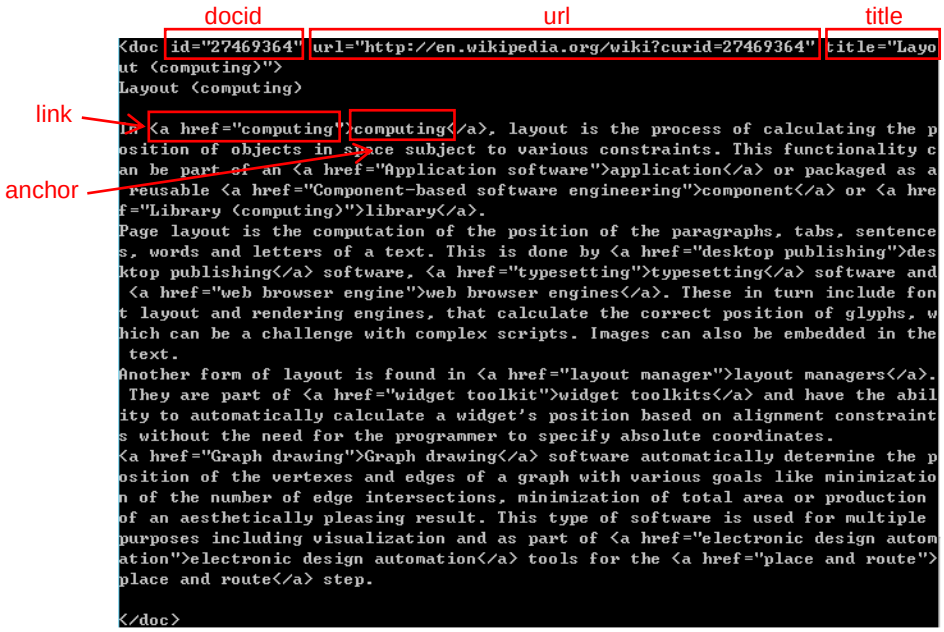


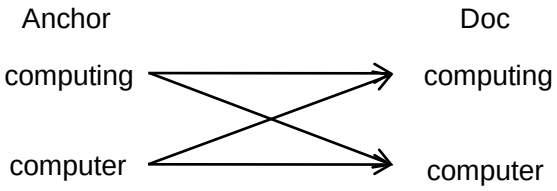
Figure 1 Example format of wiki plain text

For this project, we are provided with plain text of wiki articles as the above format. The articles are marked with HTML tags. Each article has an id of 8 digits, an url of the web page link, and an article title. The url link and article title are strings with uncertain length. Each anchor is marked up with a pair of tags <a><\a>, in which the link is given in href.

2.2. System Functions

- Given an anchor, return all the link and their sentence-level co-occurrence count. For example, if we have the following four anchor-doc(link) pairs

computing
computing
computer
computer



The expected result is

Input: computer
Output:

Doc	Co-occurrence count
Computer	1
Computing	1

- Given a wiki title, return all the anchor linked to it and their sentence-level co-occurrence

count. For example, with the dataset given before, the expected result is

Input: computer
Output:
Doc Co-occurrence count
Computer 1
Computing 1

- Given a link and word pair, return their sentence-level co-occurrence count. For example,

  In computing, layout is the process of calculating the position...

  with this example sentence, the expected result should be,

Input: Computer, layout
Output:
Co-occurrence count
1

Input: Computer, calculating
Output:
Co-occurrence count
1

2.3. System Architecture

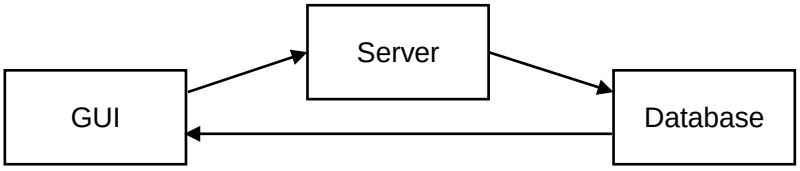


Figure 2 System Architecture

Figure 2 shows the system architecture for our program. The graphic user interface (GUI) is implemented with HTML and CSS. In the GUI, users can select which function mentioned above they want to use, and input the required parameters. Each function corresponds one request type. Received by server, the request type and parameters are combined to generate MySQL requests. The database responds to the requests, gets the query results and sends them to GUI directly. Thus users can see the results from user interface.

3. DATABASE DESIGN

For function 1 and 2, we establish the relationship between anchors and docs.

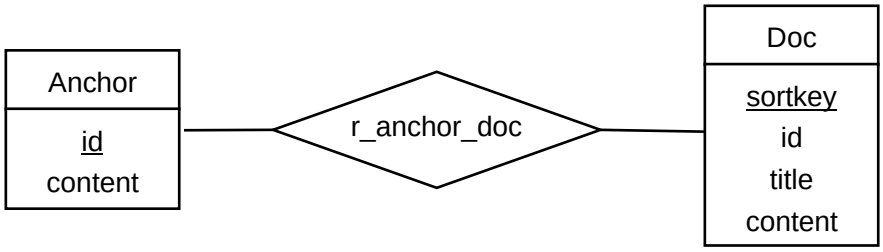


Figure 3 Database design for function 1 and 2

Since both function 1 and function 2 are based on the relationship between anchors and doc links, we design a database as Figure 3 shows, Note that both anchor and doc are used as search key in the two functions, so in the r_anchor_doc relationship, both anchor and link id are sorted for a faster search.

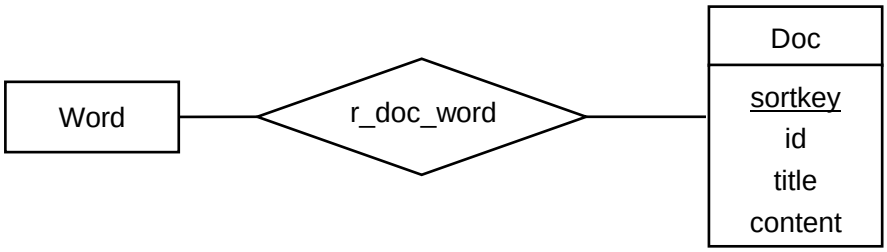


Figure 4 Database design for function 3

For function 3, we establish the relationship between words and docs,

4. SYSTEM IMPLEMENTATION

4.1. Initial Data Handling

The given initial data is a large xml file containing all the pages, so the first step is to store the file in a proper way in the database for convenience. We had created a table "doc" to store the file:

字段	类型
sortkey	int(11)
id	int(11)
title	varchar(256)
content	mediumtext

where a sample record is:

sortkey	id	title	content
1	27469019	Cinta Fitri	<doc id="27469019" url="http://en.wikipedia.org/wi...
2	27469038	Emerge (song)	<doc id="27469038" url="http://en.wikipedia.org/wi...
3	27469091	John D. Foley	<doc id="27469091" url="http://en.wikipedia.org/wi...
4	27469364	Layout (computing)	<doc id="27469364" url="http://en.wikipedia.org/wi...
5	27469376	Winfield Ervin, Jr.	<doc id="27469376" url="http://en.wikipedia.org/wi...
6	27469479	Margot Seitelman	<doc id="27469479" url="http://en.wikipedia.org/wi...
7	27469481	Ravenswood (Bunceton, Missouri)	<doc id="27469481" url="http://en.wikipedia.org/wi...

In this table, attribute "id" is enough to be a primary key, but we still assign a new attribute "sortkey" to be the primary key instead of "id". This is because sometimes we need to extract the record from a specific location, for example, select * from the 100,0000th record. If we use "id" as primary key and make index on it, we can write the query like:

显示行 0 - 29 (30 总计, 查询花费 26.3272 秒)

SQL 查询:

```
SELECT *
FROM `doc`
WHERE 1
LIMIT 1000000 , 30
```

We can see from above that it take about 26 seconds. However, if we assign a new attribute "sortkey", we can rewrite the query like:

显示行 0 - 29 (30 总计, 查询花费 0.0420 秒)

SQL 查询:

```
SELECT *
FROM `doc`
WHERE sortkey >1000000
LIMIT 0 , 30
```

It takes less than 0.1 second.

4.2. Extracting Information.

4.2.1 Extracting doc's title

In this step, we need to extract the id and title of a doc, that is the second and the third column of table "doc". This is also a duplicate table but in order to increase the query speed, we need to do it anyway. This is the schema of this table "doc_name":

	字段	类型
☐	id	int(11)
☐	content	varchar(256)

The reason to build this table is that sometimes we need to load all the 4 millions record of (id, title) in memory so that we can "translate" id to title or "translate" title to id quickly, but loading them from "doc_name" is faster than loading them from "doc".

4.2.3 Extracting anchor

Anchor is a word or some words that surrounding by "" and "", we used python to origin text and then extracting all the anchors and save them in database. This is the table "anchor":

字段	类型	整理	属性	Null	默认	额外	操作
id	int(11)			否		auto_increment	
content	varchar(1000)	utf8_bin		否			

全选 / 全部不选 选中项:

打印预览 规划表结构

添加 1 字段 于表结尾 于表开头 于 id 之后 执行

索引:

键名	类型	基数	操作	字段	
PRIMARY	PRIMARY	7555028		id	
content	INDEX	7555028		content	333

在第 1 列创建索引 执行

已使用空间

类型	用法	
数据	211,400	KB
索引	261,573	KB
统计	472,973	KB

行统计

语句	值
格式	动态
整理	utf8_unicode_ci
行数	7,555,028
行长度 ø	28
行大小 ø	64 字节
下一个 Autoindex	7,555,029

There are about 7 million anchors, associated with about 4 million docs.

4.3. Constructing relationship

4.3.1 Constructing relationship between doc and anchor

A doc and anchor pair has the form: computing

It's easy to identify all this pair from the origin data using Python:

Firstly select * from doc_name and anchor, and then save them in memory, then every time we encounter a doc and anchor pair, translate them from these table to ids pair and save this relation in table "r_anchor_doc":

字段	类型	整理	属性	Null	默认	额外	操作
anchor_id	int(11)			否			
doc_id	int(11)			否			
times	int(11)			否			

全选 / 全部不选 选中项:

打印预览 规划表结构

添加 1 字段 于表结尾 于表开头 于 anchor_id 之后 执行

索引:

键名	类型	基数	操作	字段
anchor_id	INDEX	无		anchor_id
doc_id	INDEX	无		doc_id

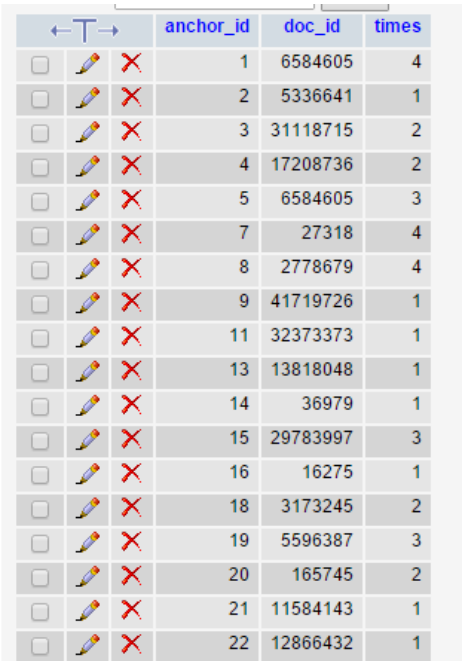
在第 1 列创建索引 执行

已使用空间

类型	用法	
数据	468,231	KB
索引	873,298	KB
统计	1,310	MB

行统计

语句	值
格式	固定
整理	utf8_unicode_ci
行数	36,882,203
行长度 ø	13
行大小 ø	37 字节
创建时间	2015 年 11 月 27 日 21:14
最后更新时间	2015 年 11 月 28 日 20:13



anchor_id	doc_id	times
1	6584605	4
2	5336641	1
3	31118715	2
4	17208736	2
5	6584605	3
7	27318	4
8	2778679	4
9	41719726	1
11	32373373	1
13	13818048	1
14	36979	1
15	29783997	3
16	16275	1
18	3173245	2
19	5596387	3
20	165745	2
21	11584143	1
22	12866432	1

Where times is the co-occurrence times of the pair in one doc.

There are about 40 million that relation in the whole corpus, so the B+ tree index is too large that it may be hard to insert a new record in it. So we create difference table to store all the relation.

4.3.2 Constructing relationship between doc and word

If we constructing this relation just like the relation between anchor and doc, the number of records may become too large (about 400millions or more), so we optimize it by saving records with same doc_id as array. For example, record (1, 2) and record (1, 3) are stored by (1, [2,3]), where the latter one should be encoded by json. And as a result, the number of records is reduced to about 10 million. Similarly, the memory cannot handle that large amount of data, we need to split the table. And the result is shown below:

	docid	words
	3910610	{"later": 1, "Star": 1, "Big": 1, "months": 1, "th...
	6116695	{"old": 1, "process": 1, "To": 1, "scanned": 1, "K...
	11534348	{"FIR": 1, "west": 1, "To": 1, ";": 2, "east": 1, ...
	4194319	{"re-elected": 1, "veteran": 1, "MP": 1, "defeated...
	1048592	{"months": 1, "old": 1, "When": 1, "feature": 1, "...
	25165841	{"week": 1, "Champion": 1, "champion": 1, "behind"...
	14734621	{"Pimoreanca": 1, "Seaca": 1, "Patrani": 1, "Dani"...
	15728662	{"eventually": 1, "Israel": 1, "From": 1, "extend"...
	2097176	{"coach": 2, "Fuller": 1, "fired": 3, "one": 1, "a...
	25	{"founder": 1, "developed": 1, "Breaking": 1, "sou...
	37748766	{"Pink": 1, "screenplay": 2, "\u014ckura": 1, "cla...

5. GUI DESIGN

We implement our GUI by a php website, which is available at

http://radlab.sjtu.edu.cn:8440/wiki_my/ui.html

(Attention: please contact us if you cannot access to the system website.)

The screen shots for our system is shown as following:

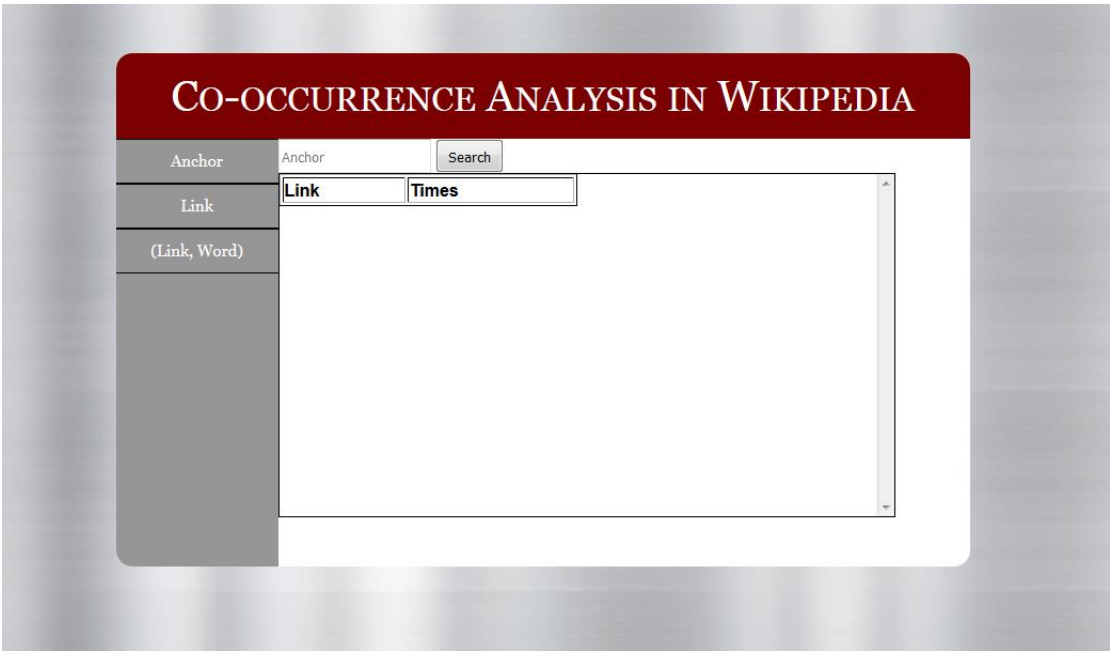


Figure 5 GUI Design

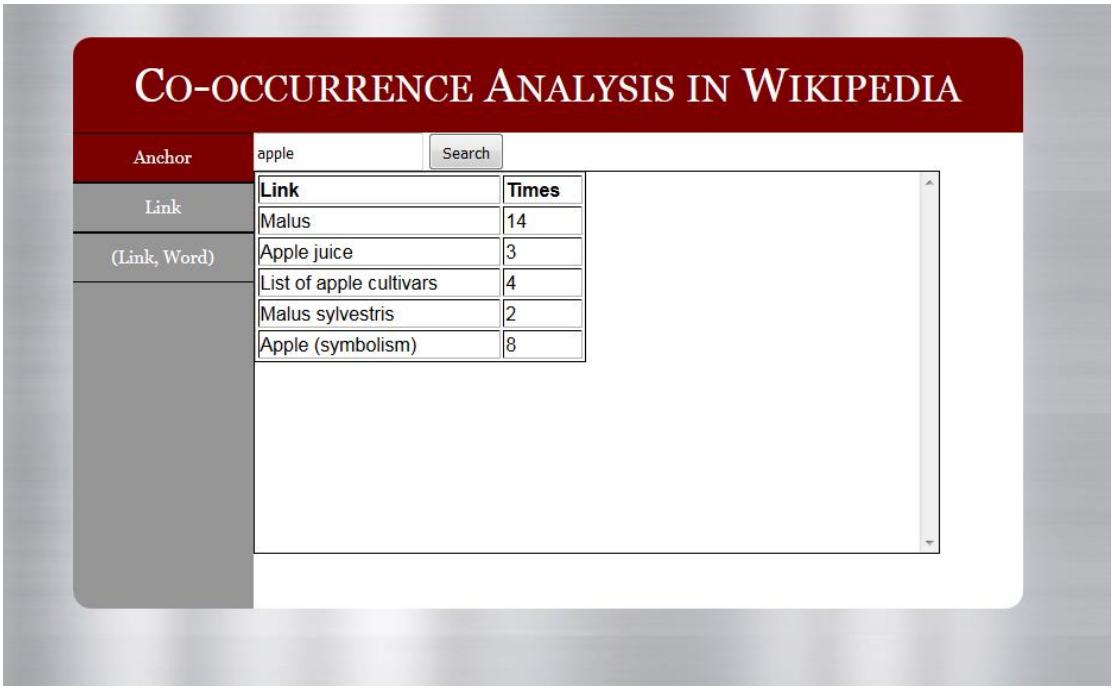


Figure 6 Anchor-link search

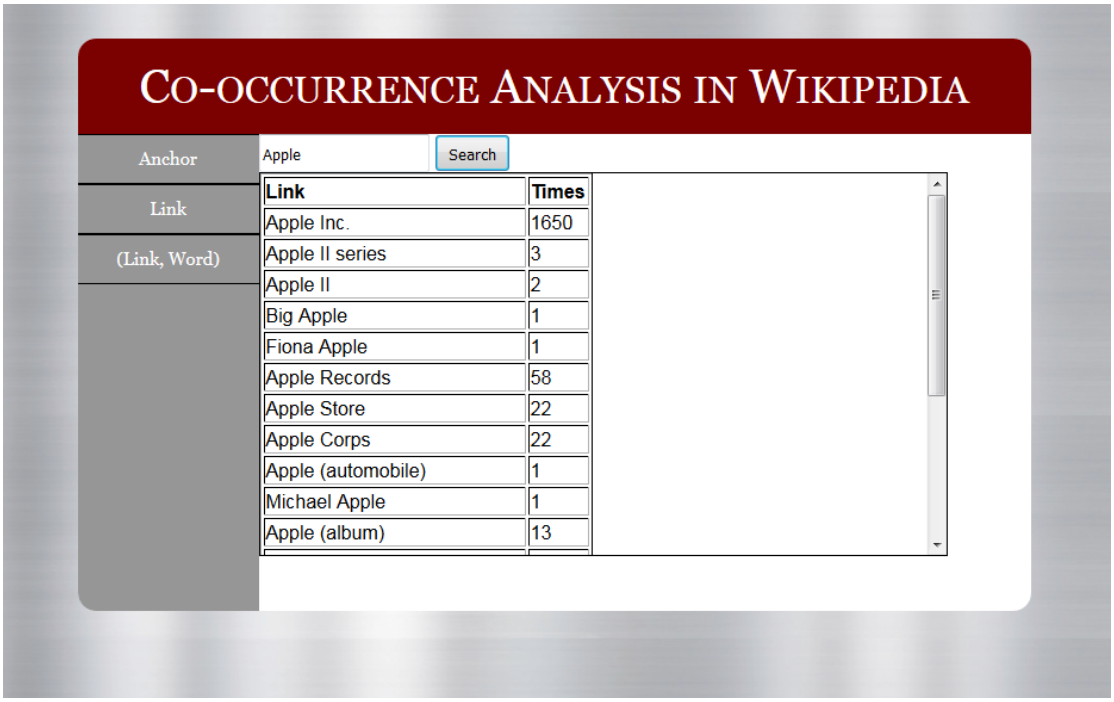


Figure 7 Anchor-link search (case sensitive)

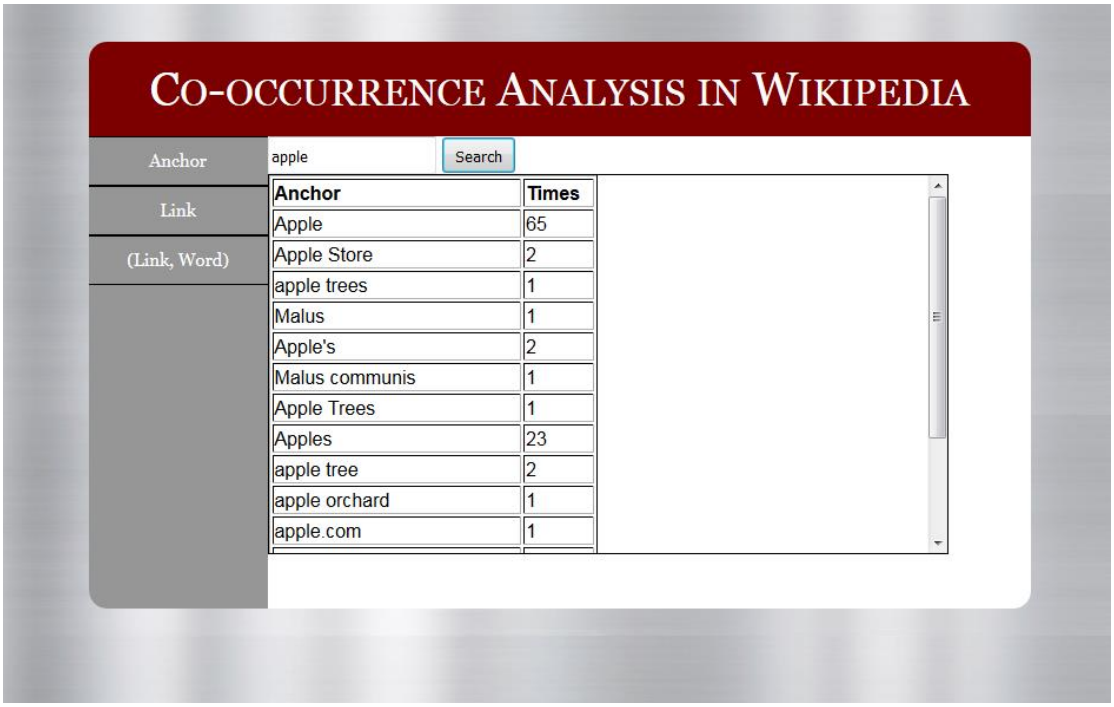


Figure 8 Link-anchor search

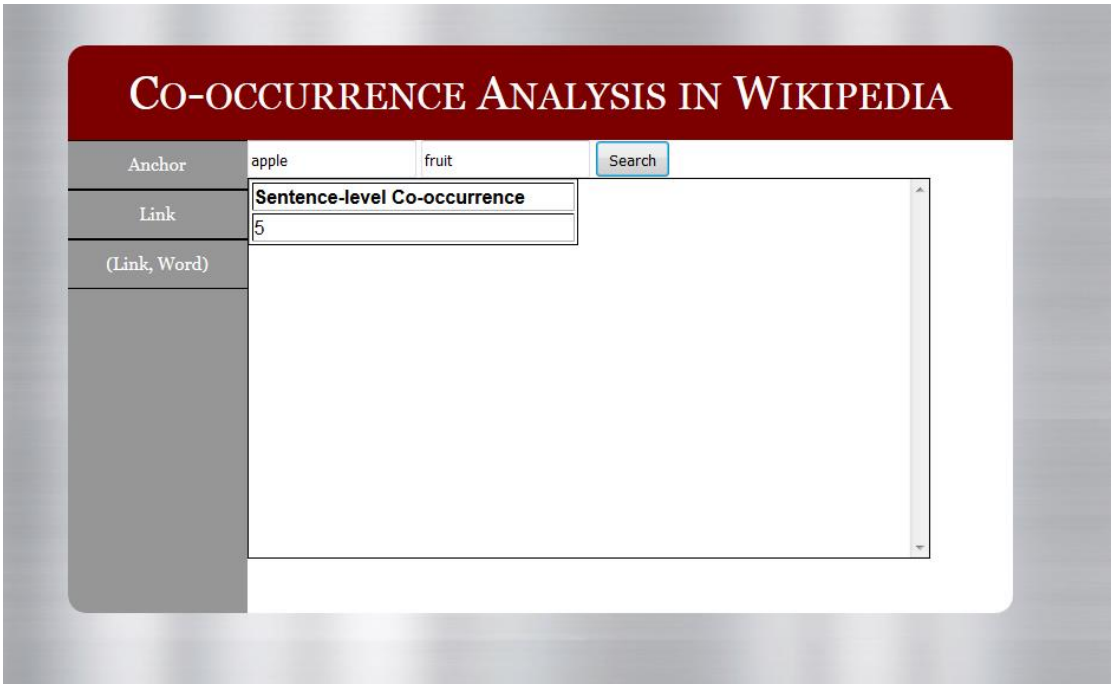


Figure 9 (Link, word) search

Note that the search functions of our system is case-sensitive, which can be seen from Figure 6 and 7.

6. CONCLUSION

In this work, we implemented a system of co-occurrence analysis based on articles of Wikipedia. Our system has mainly three functions: 1) For a given anchor, return all the link and their sentence-level co-occurrence count; 2) for a given link, return all the anchor and their sentence-level co-occurrence count; 3) for a given link and word pair, return their sentence-level co-occurrence count. We also considered about the search time and did some optimization. Our system can be used as a fast and convenient tool for users who want get more insights about the relationships of concepts, as well as those interested in text mining.