



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Fachbereich Informatik  
Ubiquitous Knowledge Processing Lab  
Prof. Dr. Iryna Gurevych

---

# **-Wiki-Sniffer- Creating Wiki Page Overview Snippets**

## **Diploma Thesis**

**Supervisor:** Prof. Dr. Iryna Gurevych

**Responsible Staff:** Johannes Hoffart and Joachim Caspar

**Fabian Lagonda Tamin**

Darmstadt, 15 Desember 2009



---

# Wiki-Sniffer – Creating Wiki Page Overview Snippets

---

**Diploma Thesis** by **Fabian Lagonda Tamin**

December 15, 2009

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



Ubiquitous  
Knowledge  
Processing

---

## **Wiki Sniffer – Creating Wiki Page Overview Snippets**

vorgelegte Diploma-Thesis von Fabian Lagonda Tamin

Supervisor: Prof. Dr. Iryna Gurevych

Coordinators: Johannes Hoffart, Joachim Caspar

Tag der Einreichung: 15. Dezember 2009

---

## **Ehrenwörtliche Erklärung zur Diplomarbeit**

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den December 15, 2009

---

(Fabian Lagonda Tamin)

## Zusammenfassung

Die Web-basierte Kollaboration namens Wiki ist in den letzten Jahren enorm gewachsen, wie z.B. das allseits bekannte Wikipedia, welches wir als Online-Enzyklopädie kennen. Die Webseite besitzt eine sehr gute Benutzerfreundlichkeit und ist für viele Menschen schon ein alltäglich genutztes Werkzeug. Viele Artikel werden kontinuierlich erzeugt und mehr Informationen werden ständig in die bestehenden Artikel hinzugefügt. Ein üblicher Nachteil hierbei ist, dass die Benutzerfreundlichkeit wegen der zunehmenden Menge des Inhalts schlechter geworden ist.

In dem Projekt Wikulu arbeiten wir an der Verbesserung der Benutzerfreundlichkeit des Wikis. Hierbei werden die Vorteile seiner Eigenschaften, Wiki-Mining und den Einsatz der Methoden in der natürlichen Sprachverarbeitung genutzt. Das Ziel ist es, einen sinnvollen und verständlichen Einsatz zu finden, so dass für den Benutzer die Informationen gut dargestellt werden.

Eine besondere Eigenschaft der typischen Wiki-Seiten ist die gute Verbindung zu den Artikeln mit ähnlichen Themen. Diese Eigenschaft ist eine gute Möglichkeit, um die Links zu den verwandten Artikeln besser darzustellen.

Um dieses Ziel zu erreichen, wird ein Extra-Werkzeug namens *Wiki-Sniffer* als ein neues intelligentes Benutzerschnittstellenelement entwickelt. Es kann später in der bestehenden Wiki-Umgebung einfach integriert werden. Durch dieses Werkzeug bekommt der Benutzer einen Überblick von einem Artikel zu sehen anstatt sich sofort die ganze Information anschauen zu müssen.

Diese Diplomarbeit konzentriert sich darauf, die wichtigsten Informationsarten in den Wiki-Artikeln zu untersuchen und zu verwalten. Desweiteren wird ein benutzerfreundliches Schnittstellenelement entworfen, das die Informationen dem Nutzer in einer guten Struktur darstellen.

## Abstract

The web-based collaboration called wikis has grown tremendously over the last years, e.g. *Wikipedia* that we know as the famous online encyclopedia. It has very high usability, and it is also used by many people in the daily life. There have been many articles written and much more information has been put within the existing articles. As the common drawback, the usability is decreasing because of the increasing amount of content.

In the *Wikulu* project, we work on improving the usability of wikis by using the benefits of wiki's characteristics and Wiki-Mining. The natural language processing (NLP) methods will also be employed in this area. The goal is to find good ways of presenting information to users in a way which is meaningful and easy to understand.

The special characteristic of typical wiki sites is its good connection among the articles with similar topics. This characteristic can be used as a good possibility for improving how links to other related wiki articles are displayed.

To achieve this goal, an extra tool called *Wiki-Sniffer* is built as a new intelligent user interface element that can be attached easily later on to the existing wiki environments. By using this tool, the users will have a possibility to see an overview of an article first, before they decide to see the whole information.

This diploma thesis focuses on investigating and arranging the most important information parts in the wiki articles. Furthermore, the next main task is to design a usable user friendly interface element which presents the information in a good structure to users.

# Table of Contents

Zusammenfassung .....	i
Abstract .....	ii
Table of Contents .....	iii
1. Introduction .....	1
1.1. History of Wiki .....	1
1.2. Motivation and Problem Description .....	2
1.3. Goal, Limitation, and Tasks .....	4
1.3.1. Investigating Information Structure in Wiki pages .....	4
1.3.2. Create a Design of User Interface Element to Present the Information .....	5
1.3.3. Create Back-End and Front-End as well as their Integration .....	5
1.3.4. Perform User Study for Collecting Data and an Evaluation .....	6
1.4. Thesis Overview .....	6
2. Theoretical Foundation .....	8
2.1. Information Seeking .....	8
2.1.1. Model of Information Seeking Process .....	8
2.1.2. Browsing and Search as Important Tasks of Seeking .....	10
2.1.3. A Multidimensional Typology of Browsing .....	12
2.1.4. Expanding Context: Enhancing Information Scent for Better Navigation ..	15
2.2. Wiki-Mining .....	18
2.2.1. Wiki Characteristics .....	18
2.3. Classifications of Information Sorts .....	20
2.4. Natural Language Processing .....	21
2.4.1. Automatic Keyphrase Extraction .....	21
2.4.2. Summarization .....	30
2.4.3. UIMA Framework .....	35
2.5. Guideline for Designing User Interface .....	36
2.5.1. Keep the Interface Simple .....	36
2.5.2. Offer Efficient and Informative Feedback .....	36
2.5.3. Balance User Control with Automated Action .....	37
2.5.4. Reduce the User's Memory Load .....	37
2.5.5. Recognize the Importance of Small Details .....	37

2.5.6.	Strive for Consistency.....	38
2.5.7.	Provide Shortcuts for Skilled Users .....	38
2.5.8.	Reduce User’s Errors and Offer Simple System’s Error Handling.....	38
2.5.9.	Permit Easy Reversal of Actions .....	38
2.5.10.	Recognize the Importance of Aesthetic in Design .....	38
2.6.	What is a Snippet .....	39
3.	Patents and Related Works.....	40
3.1.	Patents.....	40
3.1.1.	Expanded Snippets .....	40
3.1.2.	Auto-Zoomable Snippets in Multiple Snippet Windows .....	42
3.1.3.	Collapsible Dialog Window .....	43
3.2.	Related Works .....	45
3.2.1.	Google Snippet.....	45
3.2.2.	Firefox CoolPreviews.....	46
4.	Wiki Sniffer .....	48
4.1.	Investigating Information Structure in the Wiki Pages.....	48
4.1.1.	MediaWiki .....	48
4.1.2.	TWiki.....	52
4.1.3.	Information Sort Candidates to Present in Page Overview .....	55
4.2.	Designing Mockups .....	56
4.2.1.	Design Candidates.....	56
4.3.	User Study 1 – Process Data .....	63
4.3.1.	Executive Summary.....	63
4.3.2.	Task.....	64
4.3.3.	Participants .....	64
4.3.4.	Result .....	65
4.3.5.	Feedback.....	66
4.3.6.	Suggested Improvements .....	66
4.4.	Creating Components.....	67
4.4.1.	Creating Back-End .....	67
4.4.2.	Creating Front-End.....	68
4.4.3.	Integration of Back-End and Front-End.....	69
4.5.	Wiki Sniffer’s Design Implementation.....	70



4.5.1.	Wiki-Sniffer Layout and Information Structure .....	70
4.5.2.	Opening and Closing the Snippet.....	71
4.5.3.	Other Usability Features .....	75
4.5.4.	Look and Feel.....	75
5.	Evaluation .....	76
5.1.	User Study 2 – Bottom-line Data.....	76
5.1.1.	Executive Summary and Plot.....	76
5.1.2.	Task.....	77
5.1.3.	Participants .....	77
5.1.4.	Results .....	77
5.1.5.	Feedback and Problems Found .....	82
5.1.6.	Suggested Improvements .....	84
6.	Conclusion.....	85
	Acknowledgment.....	86
	Appendixes.....	87
	Appendix A: Wiki-Sniffer Architecture.....	87
	Appendix B: User Study 1 Documents .....	88
	Introduction .....	88
	Questionnaire .....	89
	1 <sup>st</sup> candidate: Prototype SA.....	97
	2 <sup>nd</sup> candidate: Prototype SD .....	97
	3 <sup>rd</sup> candidate: Prototype AT .....	98
	4 <sup>th</sup> candidate: Prototype HE.....	100
	Appendix C: User Study 2 Documents .....	101
	Introduction .....	101
	Questionnaire .....	102
	Solutions .....	105
	Appendix D: Wiki-Sniffer Screenshots.....	107
	Appendix E: Tools and Libraries.....	108
	Appendix F: Important References.....	109
	Links .....	109
	Bibliography .....	110

# 1. Introduction

## 1.1. History of Wiki

*Wiki* is a web-based collaboration system which allows web pages to be created and collaboratively edited using a common web browser.

Etymologically “wiki” comes from Hawaiian word “wikiwiki” which means quick. It was introduced by Ward Cunningham as the developer of the first wiki software, *WikiWikiWeb*. It was developed in order to make the exchange of ideas between programmers easier. It was based on the ideas developed in HyperCard<sup>1</sup> stacks that he built in the late 1980s (1). On March 25, 1995, he installed his wiki system on the Internet domain c2.com. He described it as “the simplest online database that could possibly work” (2).

After several years of wiki’s invention, a large number of implementations now exist, from very simple implementations to highly satisfying *content management system (CMS)*<sup>2</sup>. However, a content management system is not always a wiki. A wiki tends to focus on the contents for the collaboration and sharing the knowledge. Thus, a wiki tends to have many editors and many readers. By contrast, a typical CMS has only some editors and many readers because a CMS typically focuses on more powerful control over layout seen than the content itself.

It is hard to determine which wiki applications are the most popular ones. The popularity also differs for each country and language, but there are lead candidates of wiki trends among wiki applications, i.e. *MediaWiki*, *TWiki*, *DokuWiki*, *MoinMoin*, and *PmWiki* (3).

These days, wikis are used in many areas. Many wikis are used in private communities and particularly within enterprises. They are often used for internal documentations.

However, there are also many public wikis on the *World Wide Web*<sup>3</sup>. The famous one is for instance *Wikipedia*. It is a free, online, web-based, collaborative, multilingual encyclopedia project supported by the non-profit Wikimedia Foundation. It uses MediaWiki as its wiki environment. In the Alexa.com (4), it ranks in the top 10 among all web sites in term of traffic. It has already more than 3,000,000 articles for the English Wikipedia (5).

---

<sup>1</sup> HyperCard is an application program created by Bill Atkinson for Apple Computer, Inc. that was among the first successful hypermedia systems before the World Wide Web. It combines database capabilities with a graphical, flexible, user-modifiable interface (73).

<sup>2</sup> Content Management System (CMS) is a system which provides features for collaborations of creating and editing the content (text as well as multimedia documents). Generally it is used for World Wide Web. The famous server side CMS are *TYPO3*, *Typolight*, *Drupal*, *Joomla* and *OpenCms*.

<sup>3</sup> The World Wide Web is a system of interlinked hypertext documents contained on the Internet (69).

Other famous wikis in the World Wide Web are for example, *Wiktionary* as a free content, online, and multilingual dictionary, as well as *WikiAnswers* which shares about general knowledge. There are still many other wikis used to share different kinds of information, e.g. music lyric, films, series and dramas, as well as for software and operating system guides and documentations.

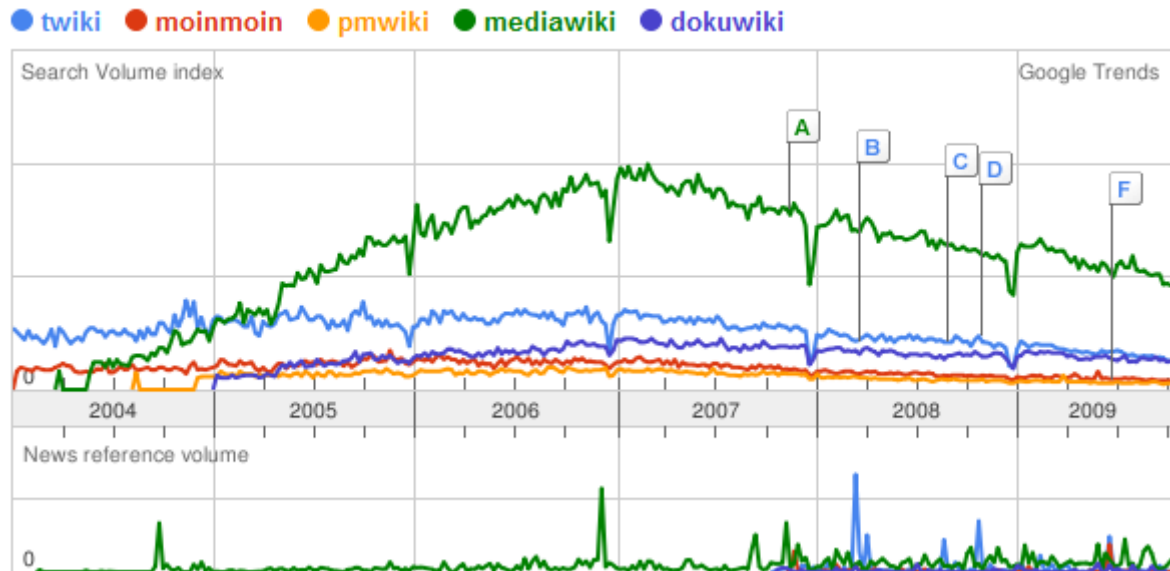


Figure 1: Google trend history comparison for famous wiki environments (October 2009)

## 1.2. Motivation and Problem Description

Since a wiki's usability is initially very high, the amount of its contents grows very fast. Many people collaborate, write many new articles, and also put more details in the existing articles. Articles with similar contents are often created without being aware that there are other people who had written about the topic, for instance, because those are written with meaningless page titles. Moreover, articles which are edited collaboratively are often lack of a coherent structure, and it makes the navigation and orientation within the articles getting more difficult because of the disorganized content. As a result, the wiki is becoming unusable as a drawback of the increasing amount of content (6).

The *Wikulu – Self-Organizing Wikis* project at the UKP Lab employs the latest *Natural Language Processing (NLP)*<sup>4</sup> technologies to manage unstructured information, i.e. to structure the content in corporate wikis (7). The goal of this project is to implement intelligent approaches that assist the user while creating, editing, or searching content. The name “Wikulu” comes from word “wiki” and “kukulu”. Like the word “wiki”, word “kukulu” comes from Hawaiian language. It means “to organize”. All together, Wikulu means self-organizing wikis.

<sup>4</sup> Natural language is a language which is used by humans for communication. It can be a spoken, written, or signed language. Natural language processing (NLP) is a branch of artificial intelligence that deals with analyzing, understanding, and generating the natural languages.

Normally, a user interacts directly with wiki environment. They request any article to the wiki, and the wiki gives a response directly as a *HTTP*<sup>5</sup> response. Wikulu, in addition, applies wiki-mining in order to get more structure from the input data. Wikulu also integrates natural language processing (NLP) techniques to support the system, e.g. for classifying content into several categories as well as for constructing new useful information sorts from a natural text. Finally, Wikulu suggests the enhanced and more structured information to the user by using an enhanced user interface. The following figure shows the interaction among different players in Wikulu project.

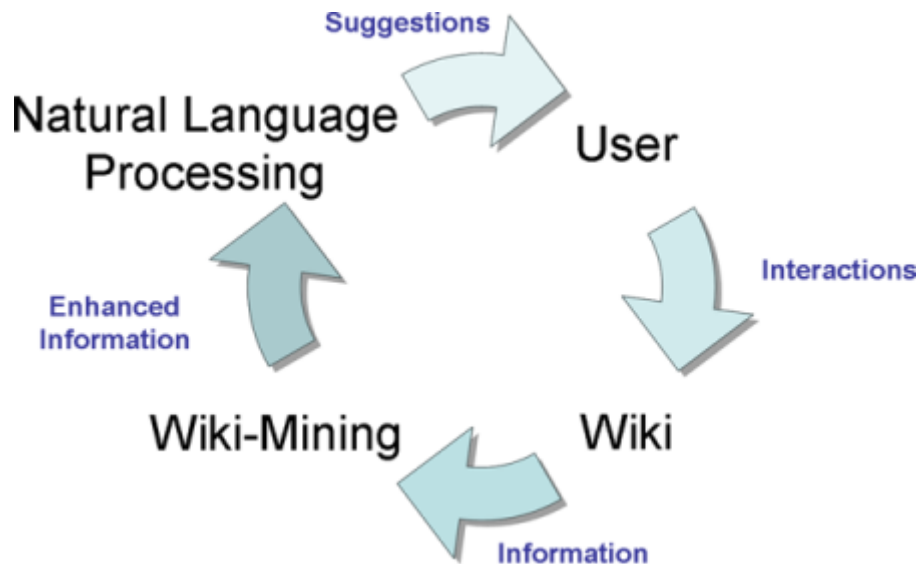


Figure 2: The Self-Organizing Wiki Cycle

In this diploma thesis, we, as a part of the Wikulu project work on improving the usability of Wikis in information seeking area so that users can obtain better possibilities to find any information they need in more comfortable ways.

Generally, wiki articles are compact articles. They have little redundancy in their contents. They use so called “Split it” strategy to write and organize the articles. To expand the concept, the authors tend to create another article and use hypertext link or “See Also” instead of writing all concepts in one article. This characteristic of wiki articles avoids the wiki articles from getting overlong or overcomplicated (8).

In this diploma thesis, we want to use the opportunity of this wiki characteristic to implement an intelligent user interface element to enhance the information seeking process. One possible option is by improving the way how search results or links to related Wiki pages are displayed.

Normally, if a user wants to know about the information behind a link, they have to click the link so that they have to leave the originating page and read it as a new page in the

<sup>5</sup> Hypertext Transfer Protocol (HTTP) is a protocol which is used for transferring data in the network. It is primarily used for the communication between a server in World Wide Web and a user web browser (e.g. Firefox, Internet Explorer, Safari, and Opera) to exchange the data including web pages by sending requests and responses.

web browser. In the new scenario, the system provides some additional information to the links as an overview of the contents behind them. By using this overview, a user could find some parts of the information immediately without leaving the originating page. If the page overview satisfies the user's need, they could continue reading the main article directly without switching to another page. In case they need some more information from the link, they could open the page by clicking the link in the page directly or by clicking the link in the page overview snippet. It means that we put a new iteration in the seeking process to enhance the process itself by giving some parts of information as a scent of the entire information. We call this strategy *sniffing* because the user sniffs a scent of a link by using some part of information as a representation of the entire information. The new interface element itself is called *Wiki-Sniffer*. In the next chapter, we will discuss about *information scent* in more details.

### 1.3. Goal, Limitation, and Tasks

The goal of this diploma thesis is to create a new user interface element which enhances the entire user interface to obtain a better performance and comfort of **information seeking process**. Enhancing the process' performance for *adding new information* is not part of this diploma thesis.

This diploma thesis contains four main important tasks:

- Investigating information structure in wiki pages.
- Creating a design of user interface element to present the information.
- Creating back-end and front-end as well as the integration of the components.
- Performing user study for collecting data and the evaluation.

#### 1.3.1. Investigating Information Structure in Wiki pages

The first task of this thesis is to investigate wiki page structures and properties in order to find which sources and sorts of information can be obtained. One possible source of such information is the natural language content in the wiki page. Hence, there are also possibilities of employing natural language processing (NLP) methods in this area to access the information from the content and construct new sorts of information, e.g. *Summary* or *Abstracts* of an article as a natural text which represents the entire content, as well as *Keywords* or *Keyphrases* which can give users a quick impression about the article by blasting sparks of important information pieces.

It is also very important to find sorts of information which can be extracted directly from the content or the wiki sites since those kinds of information extraction are mostly easier to implement. The possible sorts of such information parts are, for examples:

- The page structure in the form of *table of contents*.
- *Related wiki links* (links to and from other wiki articles).
- *Tags* which are assigned by wiki users.
- Also other possible kinds of information.

### 1.3.2. Create a Design of User Interface Element to Present the Information

Although many kinds of information can be extracted or constructed from wiki sites, there must be a good structure for presenting them to users. Presenting too much information at a single blow can for example distract the user. They would get overloaded and they would find it confusing. By contrast, if the page overview snippet always provides only insufficient information, or if it is too complicated to navigate, the user would prefer to browse without the new tool. As the result, the new tool will become unusable.

Hence, the next part of this diploma thesis is to find good ways to present that information to the user in a way which is meaningful and easy to understand. In order to create a good user interface which is user friendly and comfortable to use, variety of designs should be firstly compared to find the most suitable one.

To achieve this goal, several design mockups which use different approaches to display information should be created and compared. The most suitable design will be implemented to display the information. The implementation can also contain some modifications or combinations of the several mockups.

### 1.3.3. Create Back-End and Front-End as well as their Integration

After some observations and investigations to know which kinds of information should be presented later to users, the next task is to create internal components as the back-end for extracting information parts from the wiki environment. Some components like *TextRank Keyphrase Extractor*<sup>6</sup> and *LexRank Summarizer*<sup>7</sup> are already available in *DKPro*<sup>8</sup>. In order to use them, they must be adjusted. Creating a new keyphrase extractor or summarizer by applying a similar or another NLP algorithm is not part of this thesis. This back-end is implemented by using *Java programming language*<sup>9</sup>.

Afterwards, the most suitable design for the new user interface element is implemented. *JavaScript*<sup>10</sup> is used to create the information container as the front-end. Finally, the front-end should be integrated with the back-end.

---

<sup>6</sup> TextRank Keyphrase Extractor is an automatic keyphrase extractor which applies NLP algorithm called TextRank. More details about keyphrase extraction and TextRank can be found in the next chapter.

<sup>7</sup> LexRank Summarizer is an automatic summarizer which applies LexRank as its NLP algorithm. More details about summarization and LexRank can be found in the next chapter.

<sup>8</sup> DKPro (Darmstadt Knowledge Processing Repository) is a repository of Ubiquitous Knowledge Processing (UKP) Group at Technische Universität Darmstadt which provides the NLP research community with a collection of ready-to-use and robust NLP components.

<sup>9</sup> Java programming language is an object oriented programming language which was originally developed by James Gosling at Sun Microsystems. It was released in 1995 as a core component of Sun Microsystems' Java platform.

<sup>10</sup> JavaScript is an object-oriented scripting language and mostly used for scripting in the web browser. JavaScript is not Java programming language.

### 1.3.4. Perform User Study for Collecting Data and an Evaluation

Overall in this diploma thesis, two user studies have been planned. The first user study is performed in order to find relevant sorts of information to present to users. The design mockups will also be compared in this user study. The goal is to get the *process data* which consists of informal, qualitative observation of what people are thinking and doing to know what works on a design and what does not work (9). This study is performed as soon as the sources of information have been investigated from the wiki environment and after the several mockups have been created, before we begin with the real implementation.

The next user study is for the evaluation after the implementation has been done. The goal is to test the usability of wiki after the new user interface element has been integrated to the wiki environment. In contrast to the first user study, this study is performed to get the *bottom-line data* which consists of formal, quantitative measurement of what happens such as time it takes to learn and complete a task or the number of errors that occurs (9). In this study we want to compare the performance between browsing in the new way and the baseline, i.e. with the new tool and without the new tool.

## 1.4. Thesis Overview

This thesis presents possibilities of improving the usability by integrating a new interface element to the existing user interface so that the user interface after the new element has been integrated becomes more comfortable to use. In this thesis, it describes how a source is analyzed, how to process the information and how to structure the information so that it can be presented well to users. This thesis also discussed some guidelines for designing a user interface. Furthermore, this thesis describes several natural language processing methods for constructing some sorts of information. The chapters in this thesis are described in the following points:

- **Introduction (Chapter 1)**

This chapter gives an overview about what should be done in this thesis. It describes the problem description, motivation, goal, limitation, and the tasks in this thesis.

- **Theoretical Foundation (Chapter 2)**

In this chapter, it discusses some important aspects and several theories that are used in this thesis including the theoretical foundation about information seeking and browsing process. It also describes some wiki characteristics which play important roles in wiki mining area. The classifications of information sorts are also described. Furthermore, several natural language processing methods for automatic keyphrase extraction and summarization are discussed in this chapter. At the end of this chapter, the guidelines for designing a user interface which are also used in our design implementation are discussed. Afterwards, the definition of snippet is described in this chapter.

- **Patents and Related Works (Chapter 3)**

It is important to observe related patents and related works to look what have been done so far before we begin with the design and implementation. This chapter introduces some patents and related works that we used as some sources of our ideas to design the user interface element.

- **Wiki Sniffer (Chapter 4)**

This chapter describes how the process of designing the user interface element has been done from the beginning, i.e. investigating the source, until the implementation of the design. Therefore, this chapter also includes the process of designing the mockups, the first user study to obtain process data, and the process of creating the components of the user interface element. The design of the user interface element itself is also a main topic which is discussed in this chapter.

- **Evaluation (Chapter 5)**

This chapter discusses about the evaluation which has been done by performing the second user study after the new user interface element had been implemented and integrated to the existing user interface.

- **Conclusion (Chapter 6)**

This last chapter summarizes this thesis with a plot of what have been done in this research and also gives a conclusion according to the research results of this project.



## 2. Theoretical Foundation

### 2.1. Information Seeking

Several things can be done in the online life, like interacting or communicating with other people in internet, transferring data or downloading it, opening any website to find any information, as well as writing an article on websites so that other people can read it when they are online on the internet. One of the important parts that we want to discuss is about the *information seeking process*. Browsing a website, reading any article to find information we need, searching for any term by using Google to find websites, searching for any article by using eBay, and searching for any definition of a term on Wikipedia are some examples of what people do while they are involved in seeking information.

Since the amount of information on internet is increasing continually, there are some necessities to improve the ways of finding it. One thing that can be done is to design enhanced user interfaces which are used by people to interact with the platform. However, there are some risks that the new designs are uncomfortable to use so that they become unusable. Therefore, in order to design a successful user interface, it is necessary to firstly understand about the seeking process itself and also the tasks as well as the tactics in the seeking process.

In the next parts, the models of information seeking will be discussed. Afterwards, its important tasks and tactics will be introduced.

#### 2.1.1. Model of Information Seeking Process

A *user interface* is a point of communication between a computer and an external entity, in this case a user. A good user interface should be user friendly, which means that it should be intuitive and also easy to use. In order to design a good one which can help people to interact with machines so that the user can find information faster and more comfortable, firstly, we will have a look on how people operate in the world if they are involved in information seeking process. Two models will be discussed here, i.e. the *standard model of information seeking* as the *static model* and the *berry-peeking model* as the *dynamic model* of the information seeking process. By understanding these two models, our design can be focused on the cognition model in the ways how people interact with the world.

#### *The Standard Model of Information Seeking*

According to the *standard model*, the information seeking process can be assumed as a cycle of iterations to satisfy information needs of the information seeker. The following steps are the outline of the standard model of information seeking process (10):

- **Recognizing and interpreting the information problem**  
In this step a person recognizes and interprets that they have a problem with some information. For example, a person recognizes that they do not understand about a definition of term.
- **Establishing a plan to find some information**  
In order to find some information that the person needs, they establish a plan. For example, the person is going to find the definition in Wikipedia and read some articles to obtain some information.
- **Executing the plan**  
After establishing the plan, the person performs their plan. For example, the person searches the article by using search text field in Wikipedia and reads the first section of the article.
- **Evaluating the results**  
After the plan has been executed, the person evaluates the results of their seeking process, whether the results satisfy their information need or not. For example, after reading the first section, the person thinks about it, whether they have got enough information from what they have read.
- **If necessary, iterating through the process again**  
If the results are not satisfying, the person will recognize that there is still an information problem. A further iteration of seeking process will be executed. For example, if the person is satisfied from the information that they have read, the iteration will stop. It means that they will stop looking for any other information related to the definition of term. However, it is also possible that they encountered another definition that they found in beginning of the article that they do not understand. Therefore, they have to find any information about it to support a better understanding of the first definition. A further iteration is executed to improve the results.

Hence, information seeking can be assumed as a special case of problem solving. In the previous example the problem was to find the definition of a term.

### *The Dynamic (Berry-Picking) Model*

In the standard model of the information seeking, it can be assumed that the user's information need is static because the goal remains unchanged. There are iterations and refinements, but any other information is only for supporting the main goal.

However, the observational studies of information seeking process find that the information needs often change as the information seekers interact with the system. This model is known as the *dynamic model* or the *berry-picking model* (11) (12).

In this model there are two main points of information seeking (11). The first point is that, the seeker's information needs and also their queries shift continually, while they

are interacting with the system by reading and learning from the encountered information. The information that they get can lead them to another goal or an unanticipated direction, so that the original goal is just partly fulfilled because the priority is lowering after they encounter some new information. For instance, the person in the previous example who wants to find any information about a term definition encounters an article about an interesting topic, while they are reading at the beginning of the article. Suddenly they want to know more information about it because they think that the topic is more interesting than the first one. So, they wander from their initial subject to a new one even though they have not finished with finding information about the initial topic.

The second point of this model is that the information needs are not satisfied by a single document or even a set of documents but rather by a series of selections and bits of information that the seekers found along the way. This could occur because there are many cases that there are no perfect match of a set of retrieved document which can directly satisfy the original information need.

### 2.1.2. Browsing and Search as Important Tasks of Seeking

There are two important tasks of information seeking, i.e. *searching* and *browsing*. A number of theories along several dimensions have been formulated to distinguish the definition of searching and browsing.

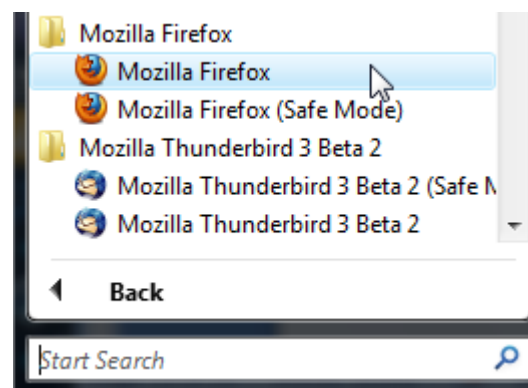
Furnas (13) has distinguished *searching* and *browsing* as two different tasks of information seeking process, and he also distinguishes two tactics which are used for accomplishing these both tasks, i.e. *querying* and *navigating*. *Searching* is the tasks of looking for a known target, whereas *browsing* refers to the task of looking to see what available in the words. It is often that browsing can be a part of searching, e.g. when a person has to see what available in their environment while they are actually searching for a specific target. However, browsing is not always a search, e.g. when a person just wants to see what available in the world but has no specific target to find.

*Querying* is submitting a description of the object being sought (for instance, using keywords) to a search engine which will return relevant content or information. *Navigation* is moving oneself sequentially around an environment, deciding at each step where to go next based on the task and the parts of the environment seen so far. For the both tactics, searching is more closely related to querying tactic, whereas browsing is more closely related to navigation tactic. However, it is also possible to involve navigating in searching or to involve querying in browsing.

An example of search is querying in the Wikipedia by entering a word or a phrase in the text field in order to find some information needed. An example of browsing is reading an article in Wikipedia and navigating from one page to another by using wiki links or categories.

Both navigating and querying have own advantages. Neither of them is the better one. It should be analyzed first, which one should be implemented and more suitable to the condition. Sometimes, it is easier for users to browse the elements by using hierarchical navigation by using a tree or links, but sometimes search field is more suitable. Combining them together if it is possible also can be a better solution.

An example of integrating querying together with navigation is the implementation of start menu in Windows operating system. Initially it implements navigation feature so that a user can search the programs by navigating through the program tree. However, since the number of programs is increasing from time to time, it is getting difficult for the user to find the program that they want. To improve this user interface, an additional search field has been implemented in the start menu which has been introduced since Windows Vista version. By using the text field, the users can search a program by querying. However, if they do not know how to formulate the query, they still have a possibility to find it by using the navigation tree. The example above shows us how the querying functionality is integrated on the place where navigation has been implemented.



**Figure 3: Windows Vista Start Menu – Search Text Field to Support Navigation**

An opposite example is the implementation of Google search suggestion. Google provides a search field, where the users are able to give a text query to search any information. However, it is often difficult for users to formulate the term query which can give promising results. Thus, Google implemented additionally a search suggestion tool in its text field. By using this navigation tactic the users can choose a more promising term without the need for formulating it completely. In this case a navigation feature has been integrated to support querying.



natural lang	
natural language processing	3,440,000 results
natural language	88,800,000 results

Figure 4: Search Suggestions – Integrated Navigation to Support Search

In this diploma thesis we focus ourselves on enhancing the navigation by improving the way how information behind the links can be presented to users. Therefore, it is very important that we put more concern to the browsing strategy since this strategy is very closely related to this tactic.

### 2.1.3. A Multidimensional Typology of Browsing

According to Chang Shan-Ju and Ronald E. Rice (14), there are five general dimensions which are underlying browsing phenomenon, i.e. the *context dimension*, the *behavioral dimension*, the *motivational dimension*, the *cognitive dimension*, and the *resource dimension*.

#### *Context Dimension*

The context dimension indicates where browsing takes place and how the situation/condition there. It includes five important aspects:

- **Organization (structure)**  
How resources are organized influences the type and the ease of browsing. For instance, an alphabetically order of items facilitates a behavioral scanning, and this type of browsing does not occur if the items are not ordered in a logical way. A bad organization of resources may lead to an inefficient browsing because people would find it difficult to find any certain information they need. It is like to find something in a messy room.
- **Interface (display)**  
A user interacts with the platform by using an interface. It includes the design of every user interface element inside it. An interface affects scanning and movement behavior. For example, positioning button in a bad place can change cursor movement behavior which can also affect the efficiency of browsing process.
- **Feedback**  
Browsing depends on the available feedback. Both insufficient feedback and too much feedback can lead to inefficient as well as ineffective browsing. With

insufficient feedback a user can not be sure whether the system responds or works. However, too many feedbacks can be distracting to the user.

- **Economic factor**

Another aspect of this context is economic factor. This aspect may include money, time, and energy. For example, if the internet cost is high and the bandwidth is low, the browsing process will be more time consuming and the successful outcomes will be lower.

### *Behavioral Dimension*

The behavioral dimension is about the actions that people take when they engage in browsing process. It includes two important aspects:

- **Scanning and skimming**

*Scanning* is a technique where the information seeker looks for specific information rather than trying to absorb the entire information (15). *Skimming* is a technique with purpose of gaining a quick overview in order to identify the main points.

Both scanning and skimming is characterized by quick glance and often use organizational clues such as title, bold type, italics, capitalized words, captions, etc. The difference is that people has some specified contents to find while they scan, e.g. find article begin with alphabet "x" in a sorted list, or find all links in an article, whereas in skimming, people may not have a specified contents to find but rather to be directed by a goal to identify the main points of entire information by using some clues from the source, e.g. try to understand about an article by using header, links and table of content instead of reading the whole text. Although scanning and skimming are mostly related to reading, they do not always have to be visual and may involve more than one sense.

- **Movement**

It includes the person's movements, e.g. the eye movement but also the movements that the person does in the interface, e.g. the cursor movement.

### *Motivational Dimension*

Motivation dimension is a dimension which is related to why people get involved with browsing and what they intend to accomplish. There are two important aspects of this dimension:

- **Purpose or motive**

This aspect is about the reason why people browse. The motive can be a cognitive motive, e.g. because they want to find any specific information, or recreational motive, e.g. for enjoying the time by finding some new information.

- **Goal**  
Goal is about what they intend to accomplish, e.g. to buy a specific book. Browsing can be *goal-directed* or *non-goal-directed*. In a goal-directed situation the whole actions that they do are in order to support the overall purpose, e.g. the skimming behavior. In a non-goal-directed situation, the actions are manifested by externally driven activity such as window shopping on the way home. Furthermore, a goal can be either content specific or non-content specific, either path specific or non-path specific, and either location specific or non-location specific. An example for a content specific situation is the scanning behavior.

### *Cognitive Dimension*

The cognitive dimension is a dimension which is related to the mental state of the browser. There are two important aspects in this dimension:

- **Plan**  
The task of accomplishing a goal can be *planned* or *unplanned*. An example of a planned browsing is the previously mentioned standard model of information seeking. However, people can also accomplish some goals without a plan but by taking an advantage of a situation while they interact with the platform. It is because browsing is often a situated action or situated learning process. Many cases in berry-picking model are the examples in which the goal is unplanned.
- **Knowledge and experience**  
A different state of knowledge or experience influences the form of browsing. For example, for some people who know about Wikipedia categories tends to browse to the bottom of the page to click the category links, when they need to find a similar articles, whereas the other people who do not know about it will find the information in different ways.

### *Resource Dimension*

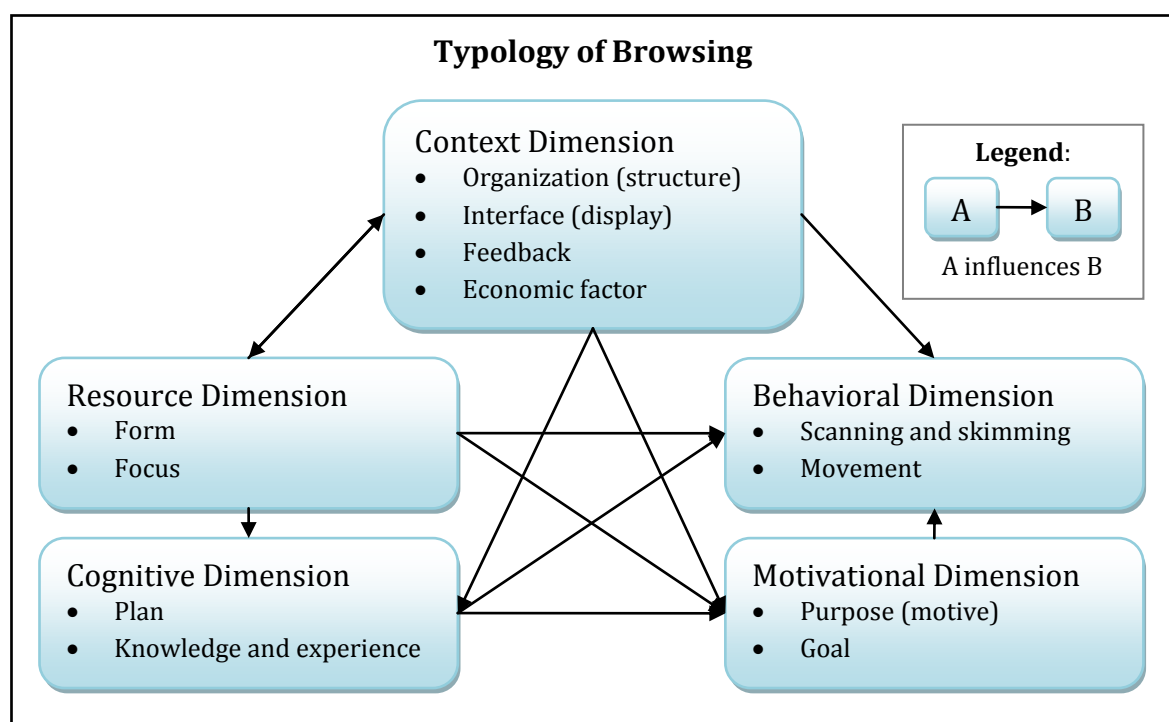
The resource dimension is about on which kind of platform the people browse and about which kind of target the people browse. There are two aspects of this dimension:

- **Form**  
The form of a browsing platform can be either an *object* or a *representation*. Browsing a book as a physical object is not like browsing an e-book as its representation. A physical book offers more sensory experiences like holding some pages with hand or marking some pages in ways which are impossible in an e-book. An e-book by contrast can provide a special navigation such as hypertext link navigation for table of contents or “go to” feature to navigate to a specific page.
- **Focus**  
One can either browse a resource’s *content* or *path*. The techniques which are used for browsing contents are not the same as the techniques for browsing paths. For



example, to browse some content, one can use the skimming or scanning technique whereas to browse a path one can use the navigation technique.

The dimensions in browsing are closely related one to another. One dimension can influence another dimension of browsing. Each dimension affects the entire browsing process. For example, a form of resource influences how the interface looks like. The way how the structure is organized in the interface can also affect the focus of a resource. The form and focus of a resource as well as the entire context affect the browsing behavior and the cognition how people plan. The knowledge and the experience affect the browsing behavior. The cognition may affect the motivation, and the motivation affects the behavior. This dimensional typology of browsing and the relations between dimensions are shown in the following diagram.



**Figure 5: Typology of Browsing – Structural and Relational Diagram**

#### 2.1.4. Expanding Context: Enhancing Information Scent for Better Navigation

Browsing is a complex activity of information seeking. There is a general agreement that “we all browse in various contexts to make sense of the world around us” (14). Thus, browsing or more generally seeking process which also often includes searching is actually a part of a larger process, i.e. *information access*. Information access has two main parts of its cycles:

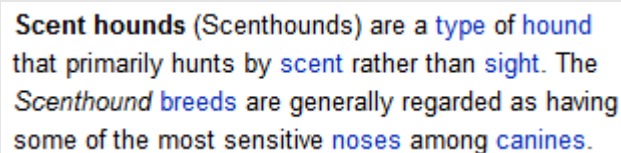
- Information retrieval through searching and browsing
- Analysis and synthesis of results



This larger process is often referred to a term *sensemaking* which can be formulated as an iterative process of formulating a conceptual representation from a large volume of information (10).

As a part of this larger process, browsing is also described as a process of “picking out bits and pieces ... selecting worthwhile information need or interest” (16) (17). It has been recognized that while browsing, it is often that users do not read the whole pages but they rather scan or skim them by using some organizational clues of information. It is done in order to obtain some information parts faster.

One of those organizational clues is links. Links play an important role to facilitate users for better scanning and skimming while they are involved in a browsing process (18). It is because of the way how the links are presented to users. Its attraction takes their attention to use the scanning cognition. Because of this situation, links can be found faster than any other parts of the text.



**Scent hounds** (Scenthounds) are a [type of hound](#) that primarily hunts by [scent](#) rather than [sight](#). The [Scenthound breeds](#) are generally regarded as having some of the most sensitive [noses](#) among [canines](#).

**Figure 6: The blue colored links attract the user to use their scanning ability to find them in a glance. (cited from english Wikipedia: Scent hound (19)).**

In spite of the fact that the links can be recognized very quickly, they often do not offer much more information than the meaning of the hyperlink text itself. Moreover the users often have problems deciding on the link’s context when the link itself is not descriptive enough.

This situation motivates us to use the opportunity of link characteristic by giving them higher *information scent*. Information scent is cues that indicate to an information seeker whether an interface element, in this case a link has the information they seek, and help the seeker navigate to the required information (20).



[Data](#) | [Economy at a Glance](#) | [Keyword Search of BLS Web Pages](#)  
[Surveys & Programs](#) | [Publications & Research Papers](#) | [Regional Information](#)  
[About BLS](#) | [Other Statistical Sites](#) | [What's New](#) | [Contact Information](#)

The Bureau of Labor Statistics is an agency within the [U.S. Department of Labor](#).

Figure 7: [Before] An early version of the home page for the U.S. Bureau of Labor Statistics, which hid most of the content behind graphics, requiring users to make guesses as to what kind of information is available, and where on the site it might reside (21) (10).

Figure 8: [After] The same home page redesigned to have high-quality information scent; intended for heavy users of government statistics (10).

Pirola notes that a small perturbation in the accuracy of information scent can cause the qualitative shifts in the cost of browsing. Some improvements in the information scent can produce better efficiency of the information seeking process (22).

Enhancing information scent in the link can be done by giving more information descriptions directly around the link or by giving an extra iteration of browsing process in between to provide more information about the link, e.g. giving an extra information description as an overlay if a link is pointed by using mouse cursor. However, we have

to keep the new iteration small and simple. The goal is to provide clues to the user about what to click so that they know about what can be found in further exploration.

Simon Harper et al. have done some evaluation about “How much is too much in a hypertext link”. They found that enhancing link descriptions is useful and important. However, information overload may lead to a lack of cognition over link destination. Too much information preview can disrupt the narrative flow. Hence, the target is to modify the link usefully but not to overload the user with too much information (17).

## 2.2. Wiki-Mining

### 2.2.1. Wiki Characteristics

#### *Dense Link Structure*

Wiki sites have some important characteristics. One of the most interesting ones has been mentioned in the first chapter, i.e. the dense link structure. Dense means that a wiki page has a lot of inner links to other wiki pages. Hence, a wiki sites has more topic locality. The connectivity among the articles is much stronger than the ordinary websites because of the dense link structure.

According to Kotaro Nakayama (23), from 1,686,960 pages from English Wikipedia 49,980,910 forward links (excluding redirect links) were extracted. It means a page in Wikipedia contains 29.62 forward links in average. Further, 2,531 pages have more than 500 forward links/page and 94,932 pages have more than 100 forward links/page.

#### *Wikitext*

Another characteristic of wiki sites is that it uses wikitext editor within the browser for creating and editing wiki pages. Wikitext is a markup language like HTML<sup>11</sup> but it is a rather lightweight one. Its syntax is also much simpler than HTML. Its purpose is to make the page creation and page editing easier and nevertheless to keep the content readable. According to a Canadian study among fourth grade students (age eight to nine), wikitext can even be used by children after a 15-minute introduction (24).

The contents in wiki applications are kept in the wiki software database in a form of wikitext. If an article or a page is requested by a user by using their web browser, it will be converted by the wiki engine into HTML text. The web browser will get a HTML response and serve it to the user.

The question is why we do not use a WYSIWYG text editor in this era. Of course, there has been a long standing discussion in the wiki world whether they have to move back to a visual text production in wikis again since users who usually only use WYSIWYG editors must suddenly confronted with wiki markup language and find it disturbing at

---

<sup>11</sup> HTML (HyperText Markup Language) is a standard markup language for web pages which is used to create a structure of contents in a document such as texts, images, hyperlinks, and other objects such as embedded plug-in.

first. Many people wonder why they should use it, because they find editing in that way is old-fashioned.

Some wiki applications provide also WYSIWYG text editor in addition to their wikitext editor. However, there are still reasons why people should not give up on wikitext in favor of WYSIWYG editor. WYSIWYG text editor mix up between content and layout. That's why many people who are only adapted with it usually do not aware of the logical structure of the document since they are too concentrating with the layout.

By using wikitext editor users know the structure better. The writers can give much more concentration to the content, and since the syntax is readable, they still can maintain the content very well. Moreover, they can leave the formatting to the machine which has also a context sensitive display. No matter which medium is used to display the text, the layout will adapt accordingly by the defined design of the medium.

In spite of those benefits of using wikitext, it also comes with drawbacks. One of those problems is so called "wiki markup mess". This problem comes because these days there are numerous popular wiki engines developed, and each of them has own wiki markup language. So far, people have failed to standardize them and it makes it really hard for the users that edit different wikis which run on the different engines.

In this thesis we are also using wikitext as a source of information to extract. However, since there are numerous wikitext developed out there, we focus ourselves only on two popular wiki engines, i.e. MediaWiki and TWiki (9).

Here is an example of a text in different syntaxes:

MediaWiki syntax	Equivalent HTML	Rendered output
"Take some more <span>[[tea]]</span> ," the March Hare said to Alice, very earnestly.	<code>&lt;p&gt;"Take some more &lt;a href="/wiki/Tea" title="Tea"&gt;tea&lt;/a&gt; ," the March Hare said to Alice, very earnestly.&lt;/p&gt;</code>	"Take some more tea," the March Hare said to Alice, very earnestly.
"I've had nothing yet," Alice replied in an offended tone: "so I can't take more."	<code>&lt;p&gt;"I've had nothing yet," Alice replied in an offended tone: "so I can't take more."&lt;/p&gt;</code>	"I've had nothing yet," Alice replied in an offended tone: "so I can't take more."
"You mean you can't take 'less'," said the Hatter: "it's very easy to take 'more' than nothing."	<code>&lt;p&gt;"You mean you can't take &lt;i&gt;less&lt;/i&gt;," said the Hatter: "it's very easy to take &lt;i&gt;more&lt;/i&gt; than nothing."&lt;/p&gt;</code>	"You mean you can't take less," said the Hatter: "it's very easy to take more than nothing."

**Table 1: Syntax Comparison (text from Alice's Adventures in Wonderland by Lewis Carroll)**

## 2.3. Classifications of Information Sorts

According to the way how to obtain the information from its sources, information sorts can be basically classified as *extracted information* or *constructed information*. Extracted information is an information sort which can be fetched directly from the source by using the structure or organizational clues of the document source, e.g. wiki links in a wiki page. Additionally, we need some processes to be done by using defined algorithms to get so called constructed information, e.g. important phrases of an article.

Important phrases are called constructed information because there is no such definition in the wiki environment which directly tells us that a certain phrase is important. We have to define it by ourselves by performing an algorithm to tell the system how they calculate whether a phrase is important. In the next part of this chapter we will discuss several algorithms for constructing some sorts of information. Which sort of information is extracted and constructed for the page overview snippet will be discussed later in the chapter 4.

Sometimes, constructing a new sort of information can be done by extracting the information from the source, for example related articles. Concerning to its term, related articles are not provided directly by Wikipedia, but they can be constructed either by using a complex algorithm or by extracting the wiki links directly from an article.

Hence, constructed information can be assumed as a “black box” whereas extracted information is a “white box”. The evidence suggests that showing information as a black box to enhance the context, or in other words encapsulate the white box, can be useful for designing an effective user interface (10).

The next classification for information sorts is by classifying them according to the author of the information. According to the author, they are classified as *information sorts generated by machine using automation* or the *information sorts suggested manually by humans*. The links in the Google search results are for example an information sort which is automatically constructed by machine. External links provided in Wikipedia are for example an information sort which is suggested manually by human.

Neither of them is known as a better approach. Machines work normally better for a large amount of information which is difficult to manage by human, but humans tend to work better for the topic, for which they are an expert. To obtain the advantages of each approach, we want to combine them together in our new user interface element.

The last classification that we want to discuss is by classifying them according to its form. According to its form, the sorts of information can be classified to *natural language content* and *non-natural language content*. Natural language contents can be more *unstructured* such as a flowing text or more *structured* such as a list or a table. The non-natural language contents include images, sounds, and videos.

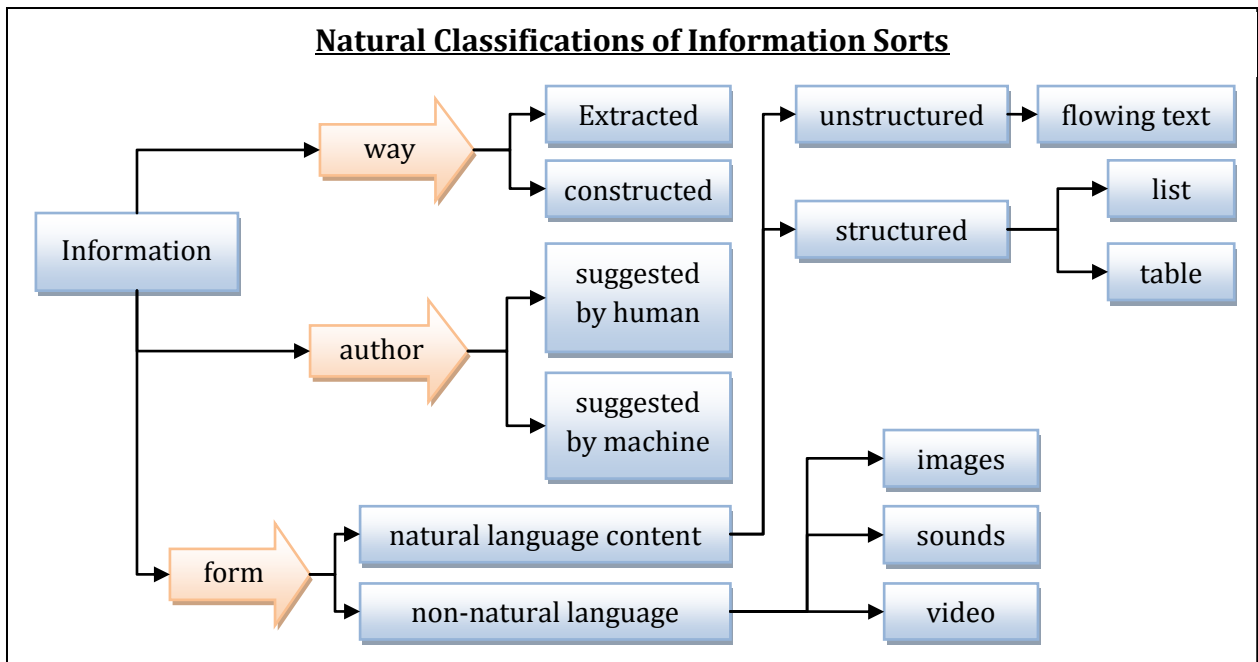


Figure 9: Natural Classifications of Information Sorts

The classifications that we discussed above are the natural classifications of information sorts, where the classification is unaffected by the way how the information is presented. Besides, information sorts can also be classified by the structural or organizational aspects how the information is presented, e.g. italic text, bold faced text, hypertext link, etc. In each wiki application, these sorts of information will be investigated.

## 2.4. Natural Language Processing

### 2.4.1. Automatic Keyphrase Extraction

Many journal articles ask their authors to provide a list of keywords for their articles, so people can get a general view about the document. Since they produce keys with two or more words as phrases rather than a single word, we like to call them *keyphrases* instead of *keywords*.

Keyphrases are small sets of expressions (words or phrases) which represents a document's content. *Automatic keyphrase extraction* is a task of automatically extracting such keyphrases from a document.

The goal of using keyphrases is to get an overview about the main idea of a document without reading the text in the document completely but rather by getting the information from the important phrases which represents the main idea of the document. This idea is also part of skimming behavior.

In this section we will see how natural language processing methods helps us by automatically producing keyphrases for a given document without the need for assigning them manually to a document by ourselves.



## Approaches

There are various methods to automatically extract keyphrases from a document or a set of documents. Basically, they can be classified as *supervised* or *unsupervised approaches*.

### Supervised approaches

*Supervised keyphrase extraction* is a technique of automatically extracting keyphrases by using a corpus of training data in order to classify or rank the keyphrase candidates.

In this approaches the system as a learner first receives training data which has been labeled or sorted in a discrete finite set of classes. The system algorithm is thereafter trained by using this data. Once the system has been trained, it can guess the label of unlabeled objects from the input data and give an output based on the guessed label (25).

Well known supervised approaches for keyphrase extraction are *GenEx* which uses genetic algorithm as its classifier and *Kea* which uses probabilistic algorithm as its classifier.

*GenEx* is a supervised approach introduced by Turney (26). It is a hybrid algorithm based on *Genitor* (steady-state genetic algorithm) and the *extractor* (parameterized keyphrase extraction algorithm). Extractor which also has been introduced by Turney is an approach which uses stems<sup>12</sup> and stemmed n-grams instead of using whole words as its input. The two most popular stemming algorithms for English words are *Porter Stemmer* (1980) and *Lovins Stemmer* (1968). These two algorithms use heuristic rules to remove or transform English suffixes to group words into the same morphological family.

Lovins Stemmer tends to strip the word ending more aggressively than Porter Stemmer. It is more likely to recognize that two words share same stem, but it is also more likely that it maps two distinct words incorrectly to a same stem. Nevertheless, Turney chose it as the stemmer for their algorithm, and they have even made it more aggressive according to the results of their experiments that an aggressive stemming is better for keyphrase extraction than the conservative one. They call their stemmer *Iterated Lovin Stemmer*, because the algorithm repeatedly applies Lovin Stemmer, until the word stops changing.

---

<sup>12</sup> Stemming is an algorithmic approach to strip off the ending of the word. It is intended to group words belonging to the same morphological family. There are several number of stem algorithms. Different algorithms can produce a different stem for a same word.

Word	Porter Stem	Lovins Stem	Iterated Lovin Stem
<b>believes</b>	believ	Belief	belief
<b>belief</b>	belief	Belief	belief
<b>believable</b>	believ	Belief	belief
<b>jealousness</b>	jealous	Jeal	jeal
<b>jealousy</b>	jealoussi	jealous	jeal
<b>police</b>	polic	Polic	pol
<b>policy</b>	polic	polic	pol
<b>assemblies</b>	assembl	Assembl	assembl
<b>assembly</b>	assembl	assemb	assemb
<b>probable</b>	probabl	prob	prob
<b>probability</b>	probabl	prob	prob
<b>probabilities</b>	probabl	probabil	probabil

**Table 2: Samples of the Behaviour of Three Different Stemming Algorithms**

Extractor works by assigning a numerical score to the phrases. The numerical score is determined by twelve numerical parameters. To maximize the performance of Extractor for its learning mechanism, Genitor is used to tune the setting of its numerical parameters by using genetic algorithm.

*Kea* (Keyphrase extraction algorithm) is another unsupervised approach. Its goal is for domain-specific keyphrase extraction. It uses all n-grams<sup>13</sup> of a certain length as its keyphrase candidates. *Kea* ranks them by using the probability based on a *Naïve Bayes classifier* which uses *TF-IDF* and position of candidate's first occurrence as its main baseline features.

The *TF-IDF* (*Term Frequency - Inverse Document Frequency*) is a standard metric for giving weight in information retrieval by using statistical measure. The words which are likely to occur in the document, like stop words (e.g. "a", "are") have the *Inverse Document Frequency* (IDF) value close to zero, whereas rare words including proper nouns typically have higher IDF value. The *TF-IDF* of phrase *P* in the document *D* (*TF-IDF*(*P*, *D*)) is basically defined as follow:

$$TF-IDF(P, D) = TF(P, D) \cdot IDF(P) = TF_{P,D} \cdot \log\left(\frac{N}{N_P}\right)$$

$TF_{P,D}$  is the number of times that Phrase *P* occurs in the document *D*.

*N* is the total number of the documents.

$N_P$  is the number of the documents containing phrase *P*.

**Equation 1**

By using Naïve Bayes classifier,  $\Pr[key|T, D]$  as the probability that a phrase is a keyphrase given that it has discretized TF-IDF value *T* and discretized *distance* *D* of its positional parameter can be formulated as follow:

<sup>13</sup> N-Gram is a subsequence of n items for a given sequence. Example of 2-grams: going to, next level, very nice. Examples of 3-gram: natural language processing, power of love, the next letter.



$\Pr[key T, D] = \frac{\Pr[T key] \cdot \Pr[D key] \cdot \Pr[key]}{\Pr[T, D]}$
<p><math>\Pr[T key]</math> is the probability that a keyphrase has a discretized value T.  <math>\Pr[D key]</math> is the probability that a keyphrase has a discretized distance D.  <math>\Pr[key]</math> is the a priori probability that a phrase is a keyphrase.  <math>\Pr[T, D]</math> is a normalization factor that makes <math>\Pr[key T, D]</math> lie between zero and one.</p>

Equation 2

Roughly Kea and GenEx are known to achieve the same level of performance. In some tests by Eibe Frank et al. (27), Kea is sometimes better and sometimes worse than GenEx. However, kea can be boosted by exploiting domain-specific training data. The advantage of Kea is that Kea uses a simpler algorithm (naïve Bayes) than GenEx genetic algorithm so that Kea can be trained more quickly for a new specific domain.

According to Hulth (28) in her report in 2003, the precision of the extraction method can be enhanced by adding more linguistic knowledge to the algorithm. She reported that using NP-Chunk for her introduced supervised keyphrase algorithm gave a better precision result than using n-gram in the algorithm. Her main idea is adding the linguistic knowledge, rather than only relying on statistics.

Olena Medelyan and Ian H. Witten (29) also reported that they were successful to enhance the performance of Kea by using candidate selection with reference to a controlled vocabulary from thesaurus. They named their implementation *Kea++*.

### Unsupervised Approaches

*Unsupervised keyphrase extraction* is a technique of automatically extracting keyphrases which uses input data as the main source of information. It takes the candidate from the document itself and ranks them by using the information found in the document and the document's structure. It does not use a set of documents or corpus to train the system for classifying the object, but rather to classify the object directly from the input data itself.

There are several numbers of unsupervised keyphrase extraction methods. For example, Barker and Cornacchia (30) extract noun phrases as keyphrase candidates from the document and rank them by using heuristics based on length, term frequency, and head noun frequency. Bracewell et al. (31) use *clustering method* to group noun phrase as their candidates based on likelihood ratio in order to extract keyphrases from a document.

Another unsupervised approach is the *TextRank* proposed by Mihalcea and Tarau (32). This approach is known of producing competitive results opposing the methods of supervised approaches. TextRank is a graph-based ranking algorithm which reflects text units of a document as graph vertices and uses their relations as graph edges. By iterating and doing some calculations, this algorithm ranks each vertex to get the top-n vertices as the keyphrases. In this diploma thesis we use this method as our algorithm

to extract the keyphrases from a document. For better understanding about this approach, we will discuss about it in more details in the next section.

## TextRank

### History

*TextRank* is an algorithm derived from the famous algorithm *Google PageRank*. Like the other graph-based algorithm *HITS* developed at IBM Almaden by Jon Kleinberg (25), *Google PageRank* which was developed at Stanford University by Larry Page and Sergey Brin as the founders of Google is also an algorithm used for ranking the web pages to determine which pages are important in the World Wide Web (25). These both successful web page ranking algorithms rely on the collective knowledge of the web architects rather than individual content analysis of each web page. In other words, they decide the importance of a web page as a vertex in the graph, by taking into account global information recursively computed from the entire graph, rather than relying only on some specific information of a local vertex (32). Similar to PageRank, TextRank uses the relations between tokens to build the graph which are used to rank the candidates. The difference is TextRank uses the relations between tokens from natural language text in the document, whereas PageRank uses the relations between links in the World Wide Web to build the graph.

### Model

TextRank as a derived algorithm of PageRank is a graph-based ranking algorithm which implements *voting* or *recommendation* as its basic idea.

In PageRank, the links of web pages reflect the vertices in the graph. A link that points to or is pointed by another link in the web page shows the relation between two vertices in the graph. Thus, these relations reflect the edges in the graph. When a vertex links to another vertex, it gives a vote or recommendation to that pointed vertex. When a vertex gets more votes from other vertices, it will become more important than the other vertices. Moreover, a more important vertex will give a higher score of voting to the other vertex which is pointed by. Hence, the importance of the vertex depends on the number of the votes that the vertex gets and the score of each vote which are casted to that vertex.

Both the graph in PageRank and TextRank can be described formally as a Graph  $G = (V, E)$ , where  $V$  is the set of vertices in the graph and  $E$  is the set of edges in the graph.  $S(V_i)$  as the score of a vertex  $V_i$  is defined as follow:

$S(V_i) = (1 - d) + d \cdot \sum_{V_j \in In(V_i)} \frac{1}{ Out(V_j) } S(V_j)$
$In(V_i)$ is predecessors of $V_i$ (a set of vertices which point to $V_i$ ) $Out(V_j)$ is successors of $V_j$ (a set of vertices which are pointed by $V_j$ ) $d$ is damping factor that can be set between 0 and 1.

Equation 3

Parameter  $d$  is a damping factor which is initially used to implement *random surfing model* in PageRank by assuming that there is a “random surfer” who opens a web page at random, keeps clicking links at pages randomly and eventually gets bored and starts on another random page. The factor  $d$  is usually set to 0.85 (33). This is also the value that we use in the TextRank.

To perform the algorithm, each vertex  $V_i$  is initially assigned by using an arbitrary value as the initial score. The initial value will not affect the final value, but it can affect the number of iterations. After the initialization, it starts iterating with the calculation of the score for each vertex in the graph by using the formula until the *convergence value* is below a given *threshold*. Threshold is a fixed value which is defined before running the iteration. The convergence value is the difference between the new score and the score in the previous iteration. Our goal is to find  $S_k$  for each vertex so that  $S_k(V_i) - S_{k-1}(V_i) < \text{threshold}$ , where  $k$  is the number of the iterations.

Since the previous formula is just a basic formula, it can also come with some modifications. Traditionally  $G$  is a *directed graph*<sup>14</sup>. However, it can also be applied to *undirected graph*. In an undirected graph, the *in-degree*<sup>15</sup> of each vertex in the graph is equal as its *out-degree*<sup>16</sup>.

Moreover, since the graph model of TextRank comes from natural language texts, where a graph may include multiple or partial links between text units or vertices, it may be useful to applied *weighted graph* to model the strength of the relations between two vertices. The new formula for the score of vertex  $WS(V_i)$  in a weighted graph can be described as follow:

$WS(V_i) = (1 - d) + d \cdot \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} S(V_j)$
<p><math>In(V_i)</math> is predecessors of <math>V_i</math> (a set of vertices which point to <math>V_i</math>).</p> <p><math>Out(V_j)</math> is successors of <math>V_j</math> (a set of vertices which are pointed by <math>V_j</math>).</p> <p><math>w_{ji}</math> is the weight of the edge between <math>V_j</math> and <math>V_i</math>.</p> <p><math>d</math> is damping factor that can be set between 0 and 1.</p>

Equation 4

### The Main Steps for Applying TextRank

Many applications including keyphrase extraction and summarization can implement TextRank as its algorithm. Since we are performing TextRank which works for a natural language text, the first thing to do is to create a graph from the text. The unit we choose as text units are varied, and it depends on the application that we build.

Systematically the main steps for applying TextRank as a graph-based algorithm in the application are:

<sup>14</sup> A directed graph is a graph which the direction of the edges between two vertices is relevant. If the edges are not directed, the graph is an undirected graph.

<sup>15</sup> In-degree of a node is the number of predecessors owned by the node.

<sup>16</sup> Out-degree of a node is the number of successors owned by the node.

- **Identify text units** which most suitable to our application and add them as the vertices.
- **Identify relations** which connect text units and add them as the edges of the graph. The relation can be either directed or undirected and it can be either weighted or unweighted.
- **Set initial score value** of each vertex arbitrarily.
- **Define threshold value** for termination of our iteration.
- **Iterate** using TextRank until the convergence values are below the threshold.
- **Sort vertices** based on their final score.
- **Select our decisions** based on the ranking (post processing).

### Applying TextRank for Keyphrase Extraction

These following subsections describe how the graph components should be built for keyphrase extraction and how to perform the algorithm.

#### *Vertices*

The vertices should correspond what we want to rank. We can actually choose each phrase with certain length as a vertex, e.g. each N-Gram, where N is smaller than four (each unigram, bigram, and trigram of the text document). However, to keep the graph small for better efficiency, and to get better flexibility, unigrams or words are used as our text unit. The same unigrams or words are grouped together to a single *token* which reflects a single vertex. The iterations will become faster because it has fewer nodes to score and it has a nice side effect that we can produce keyphrases by combining some single words in post-processing.

The filter can also be added to enhance the algorithm's performance for producing better keyphrases, e.g. by choosing only open class words or only noun and verbs as the vertices. Mihalcea and Tarau have done some experiments by restricting the words with various kinds of filters, and "only nouns and adjectives"-filter achieved the best result in the experiments (32). This is also the word filter that we use in our TextRank component.

#### *Edges*

Edges are created based on *co-occurrence* relation between words in the document. It is controlled by a sliding window which moves within the document. If any filters are used, the filtered document which only contains the restricted words is used instead of the raw document. Normally the window size can be set between 2 to 10, but according to the Mihalcea's and Tarau's, the windows size of two gives the best precision in the final results (32).

A co-occurrence between two words in a text can occur multiple times. Hence, it is more suitable to use weighted graph as our graph model. Undirected graph model is used. According to the report (32), it produces better final results than the directed one.



*Initializations and Iterations*

After the graph has been built, each vertex is set to an initial score value of one. The threshold is set to 0.0001. Usually the algorithm runs for 20-30 iterations until the threshold is reached. After a final score is obtained for each vertex, the vertices are sorted. A vertex with a higher score gets a better rank than vertices with lower scores.

*Post-processing*

During post processing we use the unfiltered document to get the full text with all lexical units. If  $n$  text units have highest ranks and they occur together as a sequence in the text, this  $n$ -gram will be chosen as a keyphrase. For instance, both words “keyphrase” and “extraction” get the highest ranks, and they occur in the text as adjacent words or as a 2-gram in a sentence, for example in the sentence “TextRank can be used for keyphrase extraction”. So, phrase “keyphrase extraction” is one of keyphrases extracted by this algorithm. In other case, if the sequence does not exist in the text, the word “keyphrase” and the word “extraction” will stay independently as two independent keyphrases which have only one word for each keyphrase. In other words, they are keywords.

**Applicability**

TextRank is fully unsupervised system. It comes with benefits that it does not need any training data or corpora to make it works and it relies directly on the given text as its information source. It makes TextRank works well for various topics from several domains without any problem of a performance change. This model is close to what humans do when they get involved with producing a summary or abstract for a given document they read.

As a fully unsupervised system TextRank is also pretty successful. Its performance is very competitive opposing supervised systems which need training data to make them work. It is reported from the previous evaluation that TextRank even outperformed the supervised system by a wide margin for small and medium sized documents which have about 100 to 1000 tokens per document. Unfortunately, for larger documents, the supervised method Kea outperforms TextRank. However, by using noun phrases instead of noun and adjective tokens as vertices, the performance for large documents can be increased by 80% (34).

This algorithm is suitable for extracting keyphrases from wiki sites since the wiki articles are not domain specific, and mostly, they are not a large document because of its “split-up” strategy.

Furthermore, although TextRank is derived from google PageRank, it can be easily integrated with other graph-based ranking algorithm, e.g. *HITS algorithm* developed by Jon Kleinberg, or it can even be combined with positional function to rank the vertices with additional positional information.



### 2.4.2. Summarization

*Automatic text summarization* is a process of automatically creating a compressed version of a given text that provides useful information for the user. According to its type, summary can be classified as *topic-oriented summary* or *generic summary*. A topic-oriented summary focuses on the interest of user's topic, and it only extracts the information in the text which is related to the topic. By contrast, a generic summary tries to cover as much as possible information from the contents. It preserves the general topical organization of the text (35).

According to the way how a summary is produced, automatic text summarization can be classified as *extractive summarization* or *abstractive summarization*. An extractive summarization produces summary by choosing a subset of sentences in the document(s). It means that the algorithm uses a kind of "cut and pasted" method to produce the summary or abstract of a text. By contrast, an abstractive summarization is a summarization in which the text can be rephrased. It is actually similar to what humans usually do when they deal with summarization. In fact, truly abstractive summarization has not reached to a mature stage today in natural language processing. Thus, we will focus only on automatic extractive summarization. Since we want to summarize general wiki articles and do not want to orientate to a specific topic, the outputs should be generic summaries.

The following subsections are the several approaches for automatic, extractive, generic summarization.

#### *Centroid-Based Summarization*

To choose the subset of a text, extractive summarization tries to identify the most *central* sentences in a cluster of the document(s) that give sufficient amount of information to the main topic of the text. One of the famous algorithms for automatic extraction is the *centroid-based summarization*. *Centroid* is generally a text unit which statistically represents a cluster of the document. In this case, a centroid consists of words that have TF-IDF scores above a given threshold. Sentences that contain more words from the centroid of the cluster are considered as central. They are ranked together, and the most central sentences are chosen to build the summary (36).

#### *Summarization Using Similarity Graph*

The next approach of extractive summarization uses the similarity graph to identify the most central sentences in the document (or multi-documents). It includes *degree-based methods* proposed by Erkan and Radev (35), as well as *TextRank* which was proposed by Mihalcea and Tarau (32).

#### **Degree-Based Methods**

There are three types of degree-based methods which were proposed by Erkan and Radev, i.e. *degree-centrality*, *LexRank*, and *continuous LexRank*.

### Degree Centrality

Degree-based methods use *cosine similarity* to define the relations between two sentences. In this method, each sentence represents N-dimensional vector, where N is the number of all possible words in the target language. The similarity between sentence  $x$  and sentence  $y$  is defined as follow:

$$IDF\text{-modified-cosine}(x, y) = \frac{\sum_{w \in x, y} (TF_{w,x} \cdot TF_{w,y} \cdot IDF_w^2)}{\sqrt{\sum_{x_i \in x} (TF_{x_i,x} \cdot IDF_{x_i})^2} \cdot \sqrt{\sum_{y_i \in y} (TF_{y_i,y} \cdot IDF_{y_i})^2}}$$

$TF_{w,s}$  is the number of occurrences of the word  $w$  in sentence  $s$ .  
 $IDF_i = \log\left(\frac{N}{n_i}\right)$ , where  $N$  is the number of documents in the collection, and  $n_i$  is the number of documents in which word  $i$  occurs.  
 $IDF\text{-modified-cosine}(x, x) = 1$ , since each sentence is trivially similar to itself.

Equation 5

To sort and rank the sentences, the concept of degree centrality is used. Degree centrality of sentence  $x$  is the number of  $(x, y)$ -relations which have *IDF-modified-cosine* score above the given threshold.

For example, given is a text with 11 sentences. The following tables are the table of cosine similarities and the table of degree centralities.

	1	2	3	4	5	6	7	8	9	10	11
1	<b>1.00</b>	<b>0.45</b>	0.02	<b>0.17</b>	0.03	<b>0.22</b>	0.03	<b>0.28</b>	0.06	0.06	0.00
2	<b>0.45</b>	<b>1.00</b>	<b>0.16</b>	<b>0.27</b>	0.03	<b>0.19</b>	0.03	<b>0.21</b>	0.03	<b>0.15</b>	0.00
3	0.02	<b>0.16</b>	<b>1.00</b>	0.03	0.00	0.01	0.03	0.04	0.00	0.01	0.00
4	<b>0.17</b>	<b>0.27</b>	0.03	<b>1.00</b>	0.01	<b>0.16</b>	<b>0.28</b>	<b>0.17</b>	0.00	0.09	0.01
5	0.03	0.03	0.00	0.01	<b>1.00</b>	<b>0.29</b>	0.05	<b>0.15</b>	<b>0.20</b>	0.04	<b>0.18</b>
6	<b>0.22</b>	<b>0.19</b>	0.01	<b>0.16</b>	<b>0.29</b>	<b>1.00</b>	0.05	<b>0.29</b>	0.04	<b>0.20</b>	0.03
7	0.03	0.03	0.03	<b>0.28</b>	0.05	0.05	<b>1.00</b>	0.06	0.00	0.00	0.01
8	<b>0.28</b>	<b>0.21</b>	0.04	<b>0.17</b>	<b>0.15</b>	<b>0.29</b>	0.06	<b>1.00</b>	<b>0.25</b>	<b>0.20</b>	<b>0.17</b>
9	0.06	0.03	0.00	0.00	<b>0.20</b>	0.04	0.00	<b>0.25</b>	<b>1.00</b>	<b>0.26</b>	<b>0.38</b>
10	0.06	<b>0.15</b>	0.01	0.09	0.04	<b>0.20</b>	0.00	<b>0.20</b>	<b>0.26</b>	<b>1.00</b>	<b>0.12</b>
11	0.00	0.00	0.00	0.01	<b>0.18</b>	0.03	0.01	<b>0.17</b>	<b>0.38</b>	<b>0.12</b>	<b>1.00</b>

Table 3: Cosine Similarity of each Sentence as a Matrix

ID	Degree (threshold = 0.1)	Degree (threshold = 0.2)	Degree (threshold = 0.3)
1	5	4	2
2	7	4	2
3	2	1	1
4	6	3	1
5	5	3	1
6	7	5	1
7	2	2	1
8	9	6	1
9	5	5	2
10	6	4	1
11	5	2	2

Table 4: Degree of each Sentence for several Threshold



For the given example, the degree centrality method (with threshold = 1) will return sentence 2, 6, and 8 as the most salience sentences if three sentences are needed to construct the summary. That is the way how degree centrality method works.

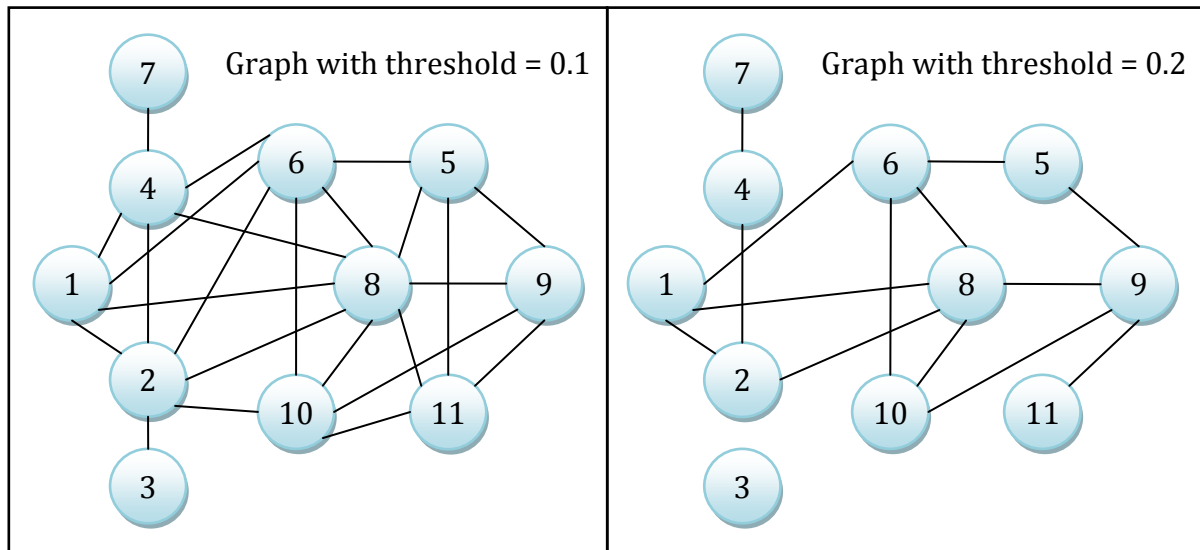


Figure 11: Similarity Graphs with Threshold 0.1 and 0.2

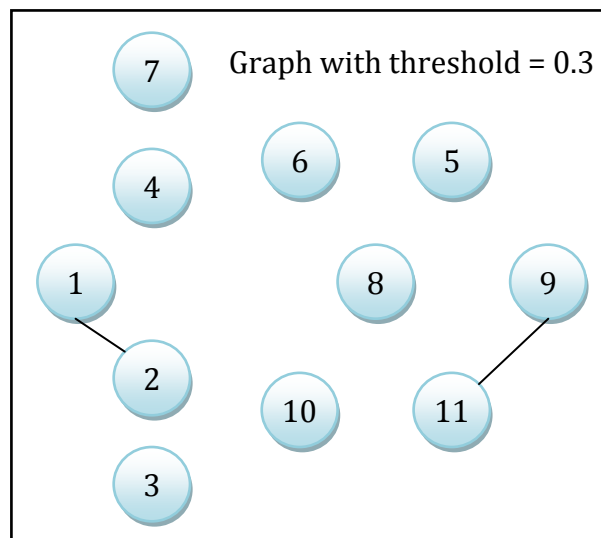


Figure 12: Similarity Graphs with Threshold 0.3

### *LexRank*

However, some sentences will get artificially high centrality scores if they get local votes from a specific set of sentences. To avoid it, we can additionally use *eigenvector centrality concept* to take the centrality of each vote into account by weighting each vote.

The formulation can be expressed by the following equation:

$p(u) = \sum_{v \in \text{adj}[u]} \frac{p(v)}{\text{deg}(v)}$
<p><math>p(u)</math> is centrality of node <math>u</math> by using eigenvector centrality concept.  <math>\text{deg}(v)</math> is degree centrality of node <math>v</math>  <math>\text{adj}[u]</math> is the set of nodes that adjacent to <math>u</math>. Each node is adjacent to itself.</p>

Equation 6

Equivalently, the equation can be formulated in the matrix notation as follow:

$p = B^T \cdot p \quad \text{or} \quad p^T \cdot B = p^T \quad \text{where} \quad B(i, j) = \frac{A(i, j)}{\sum_k A(i, k)} = \frac{A(i, j)}{\text{deg}(j)}$
<p><math>p</math> is the right eigenvector of matrix <math>B^T</math>; <math>p^T</math> is the left eigenvector of matrix <math>B</math>.  Matrix <math>B</math> is obtained from adjacency matrix <math>A</math> of similarity graph by dividing each element by the corresponding row sum.  <math>A(i, j)</math> is an element of matrix <math>A</math>; it is either 1 (<math>i</math> is adjacent to <math>j</math>) or 0 (<math>i</math> is not adjacent to <math>j</math>).  <math>A(i, i)</math> has value of 1.</p>

Equation 7

Row sum is never zero since each sentence similar to itself ( $A(i, i) = 1$ ). Matrix  $B$  can be assumed as a stochastic Matrix. A stochastic matrix  $M$  is the transition matrix of Markov chain, where each element  $M(i, j)$  is the probability of transition from state  $i$  to the state  $j$  in the corresponding Markov chain. Element  $M(i, j)$  is a value of 0 up to 1.

In order that the Equation 7 can be solved with guarantee, Matrix  $B$  must be *irreducible* and *aperiodic*. A Markov chain is irreducible if each state is reachable from any other state, i.e. for each  $i, j$  there is an  $n$  so that  $M^n(i, j) \neq 0$ , where  $M^n(i, j)$  is the probability to reach state  $j$  from state  $i$  in  $n$  transitions. A Markov chain is aperiodic if for all  $i$ ,  $\text{gcd}\{n: X^n > 0\} = 1$ .

To solve this problem, the equations are modified by assigning damping factor to each node as a uniform probability for jumping to any node in the graph. The following equation which is known as PageRank is the modified version of Equation 6 by assigning damping factor  $d$ .

$p(u) = \left(\frac{1-d}{N} + d\right) \cdot \sum_{v \in \text{adj}[u]} \frac{p(v)}{\text{deg}(v)}$
<p><math>d</math> is the damping factor (it is set to 0.85).  <math>N</math> is the total number of nodes in the graph</p>

Equation 8

In the matrix equation it is formulated as follow:

$p = [(1-d) \cdot U + d \cdot B]^T \cdot p$
<p><math>U</math> is a square matrix with all elements being equal to <math>\frac{1}{N}</math>.</p>

Equation 9

We use a simple iterative algorithm called *power method* to compute the stationary distribution of Markov chain by iterating  $p_t = M^T \cdot p_{t-1}$  of the Equation 9 until the threshold  $\epsilon$  is reached, i.e.  $\|p_t - p_{t-1}\| < \epsilon$ . Each element of eigenvector  $p$  is initialized

with value of 1 ( $p_0 = (1, 1, \dots, 1)$ ). Since Markov chain  $M$  is irreducible and aperiodic the algorithm is guaranteed to terminate.

At the end, the  $n$  sentences with highest  $p(i)$ -s in the eigenvector  $p$  are selected to compose the summary. This new measure of sentence similarity is called *Lexical PageRank* or *LexRank*.

#### *Continuous LexRank*

The similarity graphs which are used in degree centrality and LexRank are unweighted since the binary discretization by using threshold that we perform causes some information loss. To model the weighted graphs, one improvement over LexRank can be obtained by making use of the strength of similarity links:

$$p(u) = \frac{d}{N} + (1 - d) \cdot \sum_{v \in \text{adj}[u]} \frac{\text{IDF-modified-cosine}(u, v)}{\sum_{z \in \text{adj}[v]} \text{IDF-modified-cosine}(z, v)}$$

$$p = [d \cdot U + (1 - d) \cdot B]^T \cdot p \quad \text{where} \quad B(i, j) = \frac{\text{IDF-modified-cosine}(i, j)}{\sum_k \text{IDF-modified-cosine}(k, j)}$$

Equation 10

This new model is called *continuous LexRank*.

#### *Advantages of Degree-Based Methods*

Degree-based methods have several advantages over centroid-based method. It accounts the relations between sentences, and prefers to include sentence that contains more information in the summary. Moreover, it prevents unnaturally high IDF scores which come from the sentences which are unrelated to the topic.

According to Erkan and Radev (35), the results of these methods outperform both centroid-based method and other system participating in DUC in most of the cases. Even the degree centrality is good enough so that it still serves as a plausible alternative when a simple implementation is needed.

#### **TextRank**

As previously mentioned, TextRank can also be used for automatic summarization (32). Like LexRank, TextRank also choose sentences as the representations of vertices in the graphs. It also uses similarities between sentences to build the edges. The difference is that TextRank uses the number of words that two sentences have in common (normalized by the sentences' lengths) instead of IDF-cosine-similarity in order to define its similarity relations. How the algorithm is performed any further has been discussed in the previous section of TextRank.

Both TextRank and LexRank are algorithms which were derived from PageRank. These two methods were simply developed by different groups at the same time. As an important distinction, TextRank was used for *single-document summarization* while LexRank was used for *multi-document summarization*. In multi-document

summarization, the input can come from more than one document. Thus, the risk of selecting duplicate or highly redundant sentences is also greater. Although LexRank was initially focused on summarization, it can also be used for keyphrase extraction (37).

### 2.4.3. UIMA Framework

UIMA (Unstructured Information Management applications) are software systems that analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user. There are some important concepts in UIMA: *Analysis Engine (AE)*, *Annotator*, *Featured Structure*, *Annotation*, *Type System Descriptor*, *Common Analysis Structure (CAS)*, and *Subject of Analysis (SOFA)*. This section contains many information parts which are cited directly from UIMA Tutorial and Developers' Guides (38).

An *annotator* is a component that contains analysis logic. Annotators analyze an artifact, for example, a text document. They create additional data or metadata about that artifact and produce their analysis results in the form of typed *feature structures*. A feature structure is simply a data structure that has a type and a set of “(attribute, value)” pairs. An *annotation* is a particular type of feature structure that is attached to a region of artifact being analyzed, for example a span of text in a document. An annotator can also record information which is associated with the entire document rather than a particular span. These types of records are considered as feature structures but not annotations.

An *Analysis Engine (AE)* is a program that analyzes artifacts (e.g. documents) and infers information from them. They are constructed from building blocks called annotators. An Analysis Engine may contain a single annotator or may be a composition of multiple annotators. An Analysis Engine which contains a single annotator is called *Primitive Analysis Engine* whereas an Analysis Engine which contains multiple annotators is called *Aggregate Analysis Engine*. In UIMA, each Analysis Engine is described by using an XML data which contains one or more *Type System Descriptors* which are also described by using an XML data. In Type System Descriptor, the structures of annotations are defined as types. For example we have a type named “word”, and it contains two attributes: “text” and “partOfSpeechTag” as two String attributes of that type system.

All feature structures, including annotations, are represented in the UIMA *Common Analysis Structure (CAS)*. The CAS is the central data structure through which all UIMA components communicate. An easy-to-use, native Java interface of the CAS which is included with UIMA SDK is called the *JCas*. Each CAS represents one artifact to be analyzed.

Each representation of an Artifact is called a *Subject of Analysis*. It is abbreviated using the acronym “*Sofa*” which stands for Subject Of Analysis. An Analysis Engine can have multi-view CAS and one CAS can have several Sofas. Each Sofa is accessed by means of a named View.

## 2.5. Guideline for Designing User Interface

There are two goals we want to achieve when we build a user interface. They are *usability* and *utility*. *Usability* is a qualitative attribute which assesses how easy a user interface can be used by users. It also refers to user friendliness of a user interface. *Utility* on the other hand refers to the design functionality, whether a user interface does and provides what users need.

Those two aspects are equally important. It matters if the user interface is easy to use but it does not satisfy what users actually need, but it also matter if the user interface actually provides the functionality but it is too difficult to use.

Jakob Nielsen (39) defines five quality components of usability:

- **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the interface?
- **Efficiency:** How quickly can users accomplish their tasks after they learn how to use the interface?
- **Memorability:** After a period of non-use, how long does it take users to reestablish proficiency?
- **Errors:** How many errors do users make, how severe are these errors, and how easy is it for users to recover from these errors?
- **Satisfaction:** How pleasant or satisfying is it to use the interface?

To achieve these goals, researchers and practitioners in the field of Human-Computer Interaction have proposed dozens of sets of guidelines to build user interfaces successfully. In the following subsections we will have a look to some important points which are mentioned by Schneiderman et al. (40) and by Marti A. Hearst (10).

### 2.5.1. Keep the Interface Simple

The job of a user interface is to aid users in the expression of what they need to complete their tasks, e.g. to find any information. When a person tries to do any task by using a user interface, they are usually involved in some larger tasks. Therefore, we do not want to distract them by providing an intrusive user interface. We want to support them so that they can stay focused on their main task and do not want to give them some other problems with the user interface. The simplicity is important so that a user interface can be used by several users of all ages, culture, and background.

### 2.5.2. Offer Efficient and Informative Feedback

Informative feedback is important so that a user knows about the status or response of the system when they are doing some interactions with the system. People do not like to wait all day long on something without knowing anything. If a person talks to somebody, at least they need to know that the other one is listening to them. Therefore, it is important that a system at least informs the user that the system is working and that

the user should wait for a moment. The term that we use here is to support *rapid response* to users.

It is also important to give feedback to users when the system has finished with its task. Even if it has no relevant results, an informative feedback should be offered. Moreover, as previously mentioned in context dimension of browsing, the feedback should be efficient. Too much inefficient feedback can also be distracting, annoying, and slow down the finishing time.

### 2.5.3. Balance User Control with Automated Action

Greene et al. (41) write that users prefer comprehensible, predictable and controllable environments in which they can rapidly and safely explore and use information. It is a good design guideline in general but it also comes with tradeoff if a user has to take control for each detail of the entire system. Sometimes it is necessary to do some automation.

For example, many people like to use digital cameras where the automatic mode is well provided. Any people from different ages and knowledge can use them for taking pictures without any difficulty to set focus, shutter speed, aperture or lighting. However, sometimes they also want to override the default behavior. For example, some people occasionally want to adjust shutter speed so that they can take a picture of moving light on the street.

Similarly, automation is also necessary in the user interface design so that the tasks can be done faster and easier by anybody. However, manual user controls are also important to give users possibilities to accomplish their tasks in better precisions to the users' needs.

### 2.5.4. Reduce the User's Memory Load

The main idea of this guideline is to always provide relevant information to users, so that they do not have to remember everything about what they have done as well as about the user interface itself. There are many approaches can be done to follow this guideline, for example by giving simple history mechanism so that the user can trace back of what they have done. Another example is by giving some information about any item or user interface component by using tooltip, so that users do not have to remember each component but they can also have some clues if they read it.

### 2.5.5. Recognize the Importance of Small Details

A user interface can have some complex information within. Small details can make a difference between a successful and failed design. They improve the efficiency and effectiveness of learning phases. They also reduce the user's memory load and enhance the clarity of the user interface.

### 2.5.6. Strive for Consistency

Giving a good consistency makes the user interface easier to learn and provides a better structure to the interface. The consistency includes the structural way how the functionalities are presented to users, as well as the look and feel or the appearance of a user interface.

### 2.5.7. Provide Shortcuts for Skilled Users

Provide shortcut usually refers to providing alternative mechanism in the user interface so that users can do their tasks faster and more efficiently. The classic example is the keyboard shortcut. By using keyboard shortcut a user can save their time for executing a command without an effort to move their hand away to the mouse and click at some points on the display.

There are many other ways to provide shortcut mechanism to users, e.g. mouse gestures as provided in Opera browser for quick navigation, snap feature as provided in Windows 7 for positioning and resizing windows, providing a list of recent opened files so that users can open their recent files quickly, or navigation link that can teleport the users to certain location without the need to go through its original path.

### 2.5.8. Reduce User's Errors and Offer Simple System's Error Handling

Another important guideline is to reduce the user's error. This guideline tends to overlap with other guidelines. An example approach for this guideline is by avoiding empty results caused by misspelled query. Often, a user does not even recognize that they misspelled the query, and they simply think that the search engine does not have what they need. Hence, by avoiding empty results and by giving some result suggestions and spelling correction can reduce some errors.

Other errors are also often caused by the vocabulary that the user does not understand. Hence, it is important to address the vocabulary problem for labeling the items in a user interface.

Another important point related to system errors is to offer a simple error handling. Syntax error should be avoided where possible. All error messages should be specific, constructive and no more technical than necessary. A general user is often confused if they get an error message which is too technical and that they do not understand. As the result, they often get panic, and it makes the user interface uncomfortable to use.

### 2.5.9. Permit Easy Reversal of Actions

Sometimes, when a mistake is done by any action, it would be more comfortable if there is a possibility to undo the action. This guideline also helps us to reduce user's error as well as the errors caused by the system.

### 2.5.10. Recognize the Importance of Aesthetic in Design

Another important guideline is about the importance of aesthetic in design. Many user interface developers try to put much more functionalities and features to their system



but put aside this aspect of good user interface design. Many of them concentrate on having a high utility but disregard the aesthetic of design which is actually closely related to usability.

Parush et al (42) who performed a study of 16 different version of layout with 75 participants found that task accomplishing time for worst layout was twice that of the best. Moreover, the very well designed screen had higher subjective preferences as its result.

Aesthetic impressions also play an important role in user acceptance. It has been found that aesthetic impression is correlated with perceptions of an interface quality, user satisfaction, and overall impression of an interface (43) (44). Hans van der Heijden found that the visual appeal of a Web site affected participants' enjoyment and perception of ease of use, and to a small degree, the usability of the system (45). Ben-Bassat et al. were able to show that more aesthetic designs were perceived as more useful even if they were actually less useful than a comparable, less attractive design (46) (10).

The success of Google search engine is also influenced by the aesthetic of design. A Google VP (47) confirmed in an interview that their web page design is as a result of careful usability testing of small design elements, e.g. how they put textual advertisement, and also how they arrange the height and width proportions for icons. He suggested that even if the result hits for other search engines are equivalent in quality to Google's, they sometimes show advertisements which are not relevant at the top of the results list and those are degrading the user experience.

## 2.6. What is a Snippet

*Snippet* (pronounced ['snɪ.pət]) is an English word, means a tiny piece or part. In Wordnet (48), it is defined as a small piece of anything (especially a piece that has been snipped off). In Cambridge Advance Learner's Dictionary (49), it is defined as a small and often interesting piece of news, information or conversation. An example in a sentence is "I heard an interesting snippet on the radio this morning."

In our implementation, snippet means a small part of information which can be placed in a certain position on a web page. It can be implemented as a small overlaying pop-up box as well as a static or dynamic embedded text or content. "Static" means that the embedded content is fixed and not interactive. "Dynamic" means it provides some interactive functionality, e.g. the functionality to collapse or expand the content.



## 3. Patents and Related Works

### 3.1. Patents

#### 3.1.1. Expanded Snippets

It has been mentioned shortly that a snippet can be implemented as an overlaying pop-up box or as an embedded text. This patent section explains some appearance models about how snippet can be implemented. Besides, this patent also provides some structural models about how the data communications between client's browser and server can be implemented.

The following subsections are some appearance models and structural models of expanded snippet which are introduced by Paul Fontes et al. (51) to improve a list of search results.

#### *Appearance Models*

There are three appearance models of expanded snippet have been introduced:

- **Overlay Snippet**

Expanded snippet is presented within a snippet box as a small overlay over the list of search results.

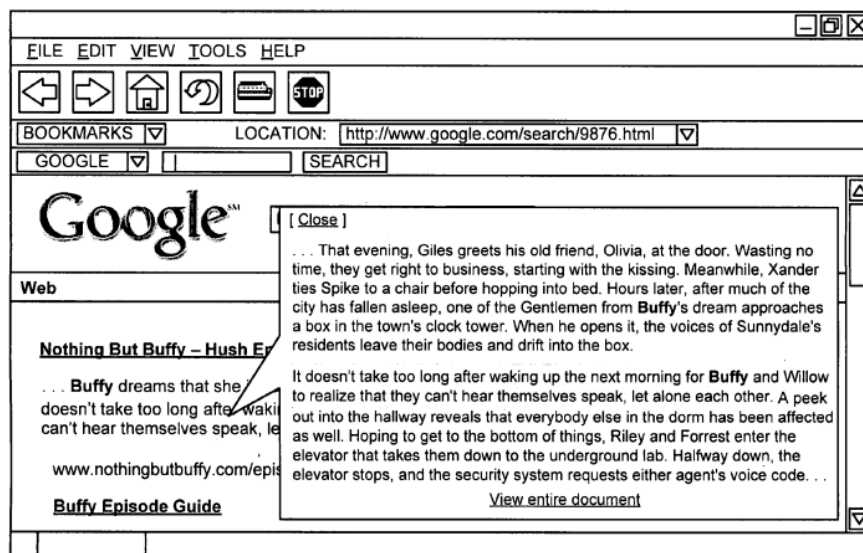


Figure 13: [Overlay Snippet] the expanded text is shown in the pop-up balloon

- **Frame Snippet**

Expanded snippet can also be presented as a frame snippet in a fixed position on the page. The text to expand can be visually distinguished from the original text (e.g. highlighted) to identify which text is being expanded.

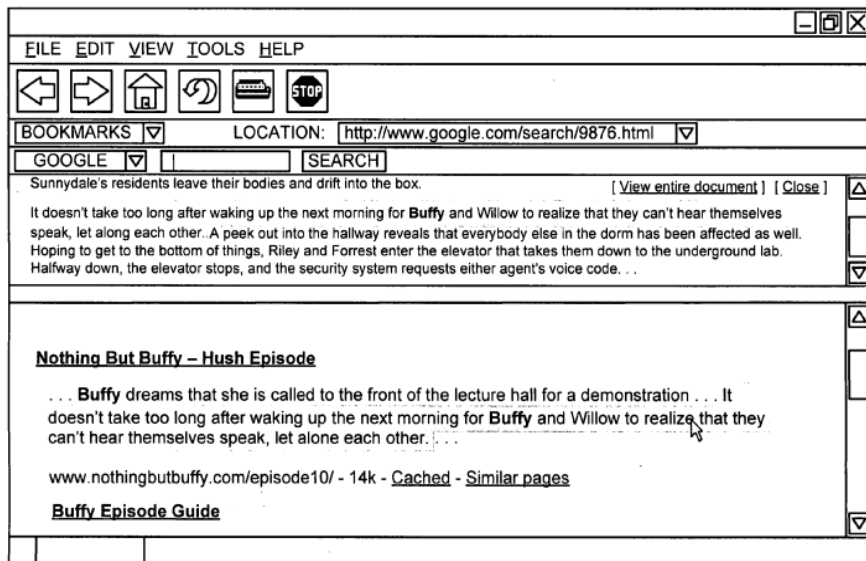


Figure 14: [Frame Snippet] the expanded text is shown on the top frame

- **Inline Snippet**

As another alternative, expanded snippet can be presented as an inline within the list of search result. The original text in the snippet might also be visually distinguished in some manner to identify which text is expanded.

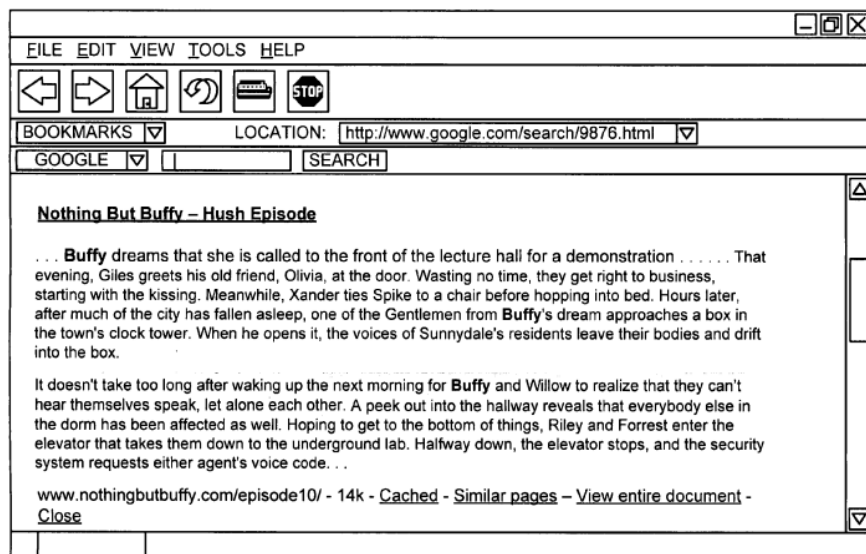


Figure 15: [Inline Snippet] the text is directly expanded as inline

There are some important components which are also mentioned by Paul Fontes et al. They are “view entire document” link and a button to permit the user to close the expanded snippet.

### *Structural Models*

There are three structural models which have been introduced in this patent:

- **Getting Snippet Data by User Request**  
Expanded snippet data is sent according to user requests via their browser. As soon as the browser detects that a user selects a snippet, the browser sends request to the server in order to get the data to expand the snippet. The server generates the snippet data based on the selection and sends it as a response to the browser.
- **Sending Snippet Data along with the Search Results**  
We assume that each item of the search results has expanded snippet data. The server sends this entire data along with the search results in a single blow.
- **Prefetching and Caching the Snippet Data**  
Browser may prefetch an expanded snippet for one or more items of search results. Prefetching can be done by automatically sending some requests for a set of search results or alternatively by detecting mouse movements or logical movements on the page to send more specific requests to server. The expanded snippet data is cached to anticipate that the user might actually request it.

The benefit of the first method is that it has the lowest bandwidth among others but on the other hand it exposes a disadvantage that it needs the longest time to present the expanded snippet because of the latency of request and response. The second method needs most bandwidth among others but it has a benefit that the snippet can be presented quickly after page load. The third one is the compromise solution of these two methods.

#### **3.1.2. Auto-Zoomable Snippets in Multiple Snippet Windows**

It is often that a snippet, or generally a window or a screen, should present multiple kinds of information simultaneously, e.g. when people work with multiple windows side by side in a screen, or when a snippet or a window itself contains multiple elements to present several kinds of information simultaneously. A problem appears when the place to present the information is limited. Moreover, the place for each element itself becomes smaller when multiple elements must be presented together in one place.

Udo Arend and Martin Hensel (52) try to solve this problem by introducing “Auto-zoomable snippets in multiple snippet windows”. Auto-zoomable snippet in multiple snippet windows is a method to zoom a certain snippet in which a user’s interest is detected. This patent also includes how the other snippets should be logically arranged in a window. Automatic zooming may adjust size, location, context, etc. to maintain the reasonable relationships between the components around the zoomed snippet.

The goal of this approach is to allow multiple snippets or information to be presented simultaneously but also to enhance the comfort when a user wants to focus on a snippet for certain information. It can be done by magnifying it and arranging the other snippets

logically. Adjustment is done to make the information in the zoomed snippet more readable or detailed and on the other hand to keep the other snippets accessible until some limits. Once a user interest to a certain snippet is detected by a logical mechanism, the snippet will be automatically zoomed in and the window rearrangement begins. When a snippet is zoomed in, we might need to reduce the size of other snippets. Reduction can be done by reducing the amount of displayed information or by reducing the font size as well as proportional size of the graphical elements. As a constraint, the fonts should be readable in side by side arrangement.

Although this approach is a powerful approach, it also comes with drawback. For example, if a user wants to copy some information from one window to another one, they may have to switch and focus to the other snippet to bring the information foreground.

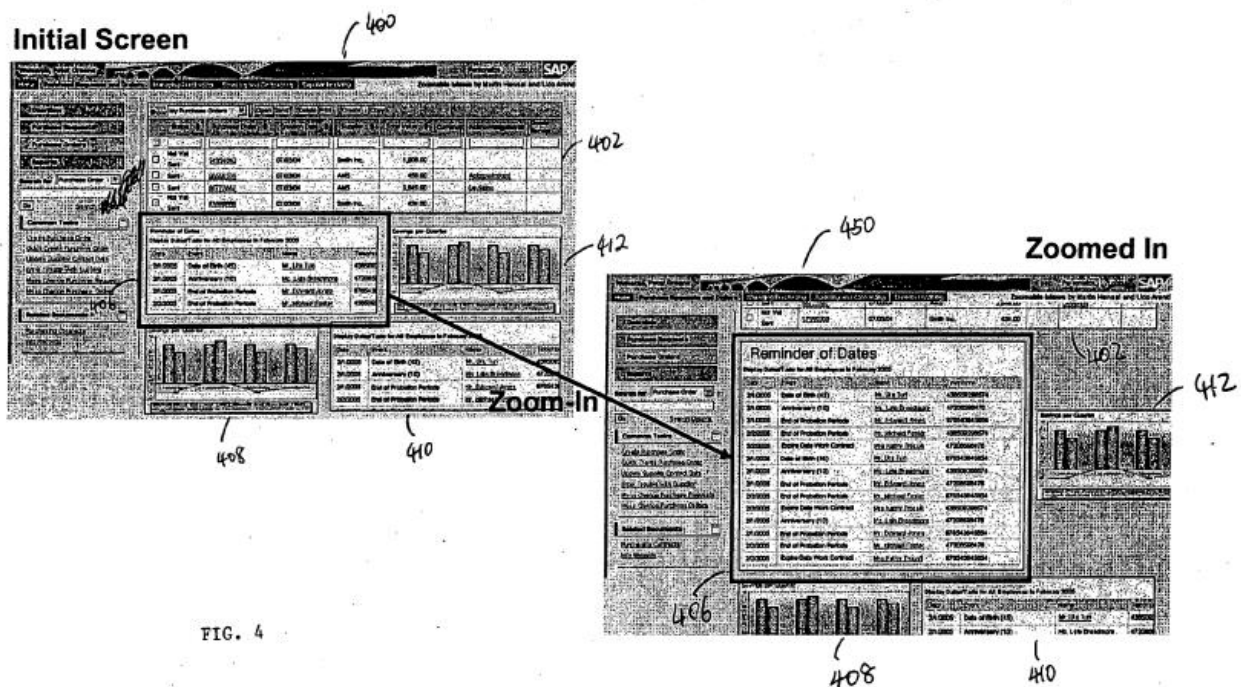


Figure 16: Auto-Zoomable Snippet

### 3.1.3. Collapsible Dialog Window

In a graphical user interface, it is often that information is displayed in one or more windows with one window as the main working window and some secondary windows that support the main window.

An example is shown in the Figure 17 where sometimes it is desirable to keep each window or dialog open but the problem is that they also consume some space and often distract the user's attention. Alternatively, the user can click the "minimize" button when they want to concentrate on the main window, and whenever they need the other window again, they can point by using mouse cursor to the "restore" button and click it to restore the dialog to its initial size. The problem is that it repeats again and again,

whenever the user wants to switch their view. We need a solution to make our work faster and more comfortable.

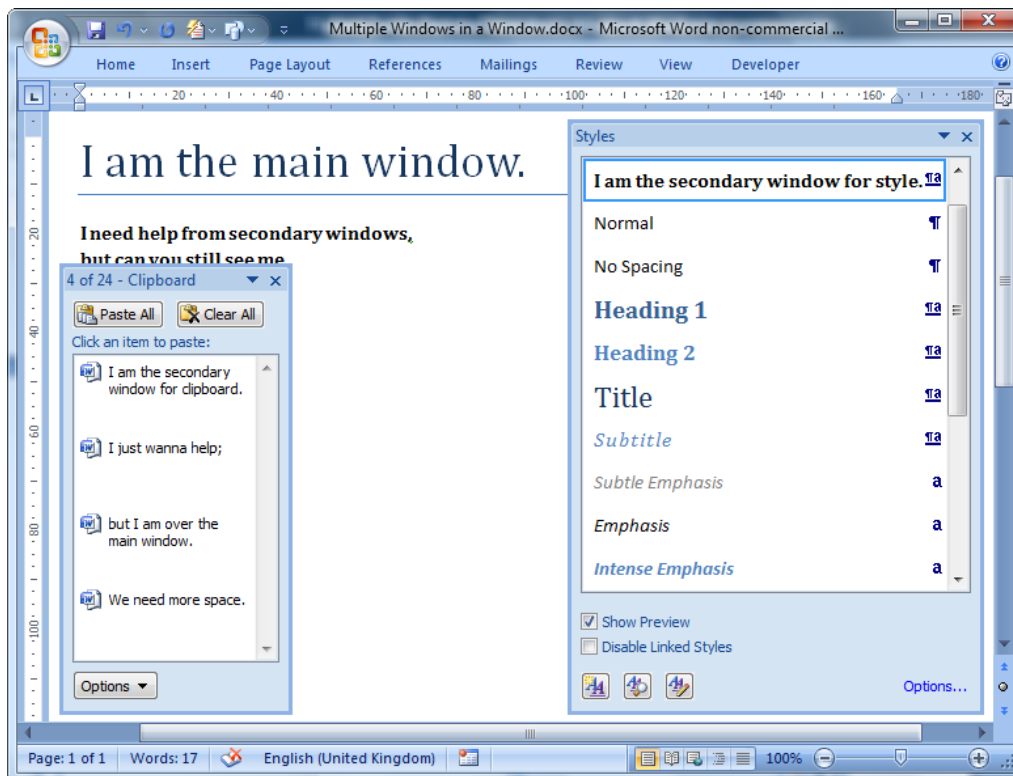


Figure 17: An Example of Multiple Secondary Windows in a Main Window

Mark Stephen Webb (53) offers a solution called collapsible dialog window. The idea is to configure the dialog to automatically collapse if the user leaves the dialog so that it consumes less workspace in the main application, and to automatically expand if a user interest is detected.

It can be done, for example by detecting the mouse position. If the mouse cursor moves away from the dialog window, the dialog window will be minimized to a small area, e.g. a title bar. In this manner it will be freeing up the space for the underlying application. If the mouse cursor is moved back, the dialog will be automatically restored up to its normal size. Its benefit is that no further complicated user action is necessary to perform this task.

To make the window works in this manner or to leave the window in normal behaviour, the user just need to activate or deactivate the automatic collapse mode by clicking an implemented system icon, e.g. pushpin icon.

Several additional features can also be implemented if necessary to expand or collapse the dialog, e.g. implementation of automatic collapsing and expanding by using time delay.



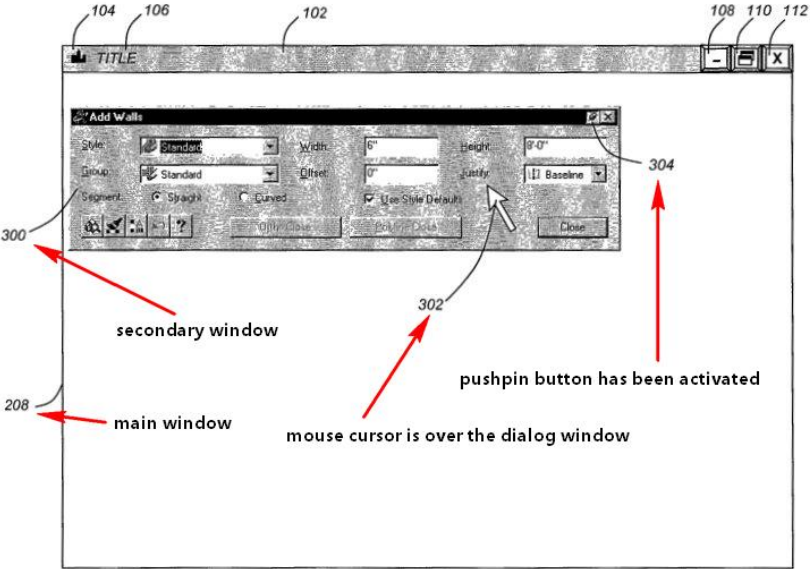


Figure 18: Collapsible dialog window is still open

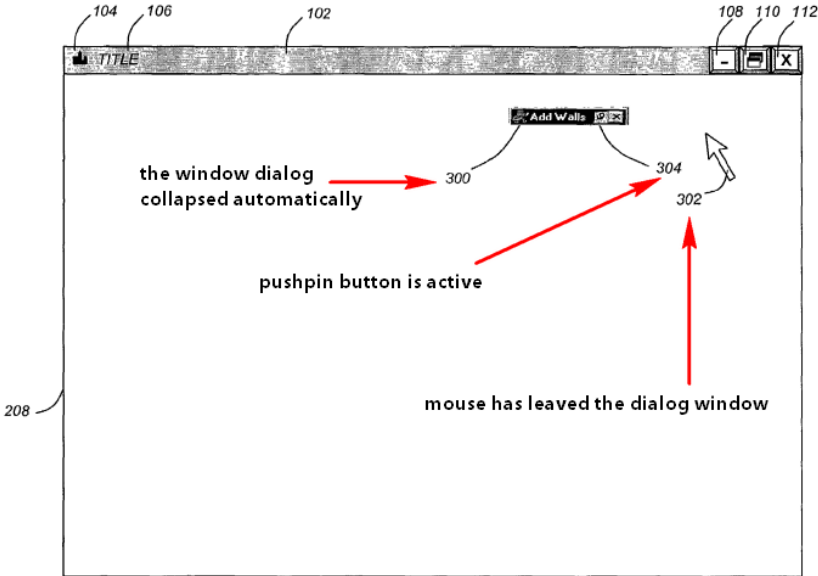


Figure 19: Collapsing dialog window

### 3.2. Related Works

#### 3.2.1. Google Snippet

We have discussed about the Google expanded snippet in the patent section. So, an example of snippet implementation is the *Google snippet* itself. *Google snippet* is a part of search results page. It works as an overview which represents a web page behind it. In this case, the snippet is implemented as a static embedded text on the search results page as the main content of the search results page itself.

Matt Cutts, the head of “search quality” at Google (50) breaks down the following components of the search result snippet to title link, description under link, plus expansion box, URL, Bolding keywords, “cache” link, “similar page” link, site links, and “more results” link.

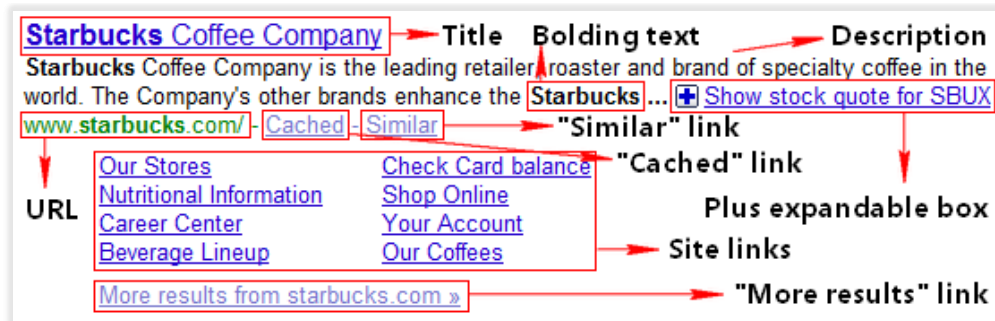


Figure 20: Google Snippet

This related work gives us some ideas how information can be arranged in a snippet.

### 3.2.2. Firefox CoolPreviews

*CoolPreviews* is a free browser add-on that lets users to preview links and rich media without leaving the current page (54). The stated goal of the software is to optimize browsing by allowing review of multiple links while preserving the context of the originating page. For example, one could preview pages from Google search results before deciding to go to the page to read in depth (55).

The content within the pop-up box is the page itself. It can be interactively browsed like a normal web page. The font size of the page can be adjusted by using “font-size” button. It provides also a small bookmark functionality to put some pages in a stack so that users can view them later in the pop-up box. Opening the page in browser after previewing can be done by clicking “open the current link in new tab” button.

CoolPreviews works as a single instance. It means that only one single preview window can be opened in a page.

Initializing the preview window is quite simple. By default, CoolPreviews icon will appear if a user hovers<sup>17</sup> a link, and by hovering this icon a pop-up box will appear in the current page. By leaving the pop-up box, the box will be closed automatically. However, users can also hold the lock button to prevent automatic closing or click close button to close the preview window manually.



Figure 21: CoolPreview Link Icon

<sup>17</sup> Hovering is a term for placing the mouse cursor over a hyperlink or icon without clicking.

It also provides extra preferences as a user driven customization functionality. In extra preferences, a user can specify how the CoolPreviews' preview window should be initialized. There are five methods that can be chosen by users: by hovering the CoolPreviews icon, by clicking the CoolPreviews icon, by hovering a link, by clicking a link, or by using a cooperation of two methods: CTRL+clicking or hovering the CoolPreviews icon.

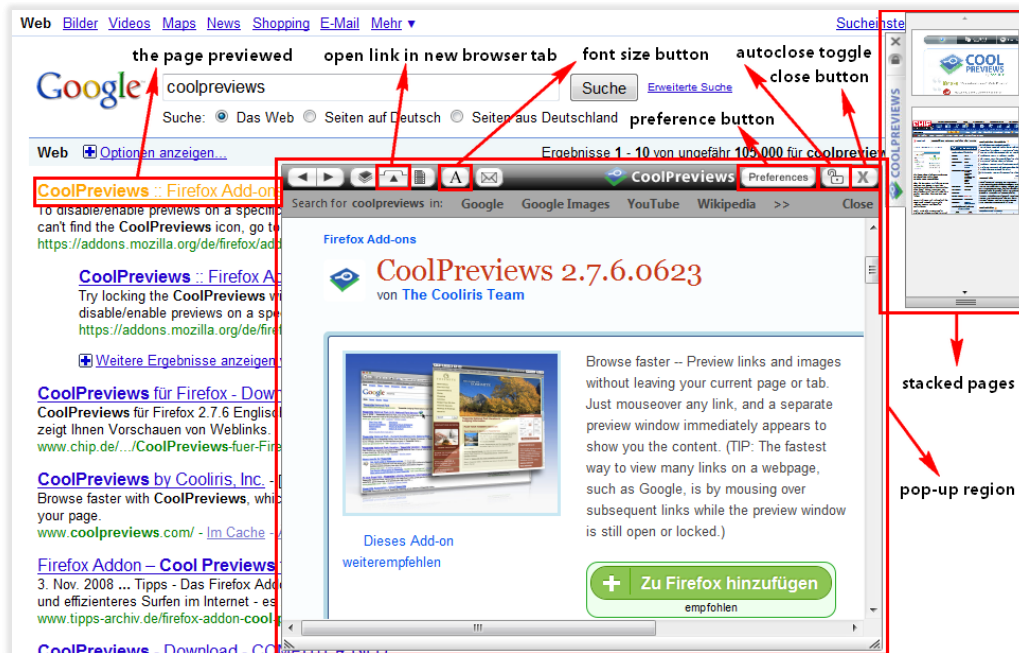


Figure 22: CoolPreviews is previewing a page with some pages already in stack

The delay time for showing the CoolPreviews icon for opening the preview window as well as for closing it can also be customized. The standard time delay for auto-opening the preview window is 500 ms or a half second and the standard time delay for auto-closing it is 1000 ms or 1 second.

Moreover, switching on or off CoolPreviews is quiet simple. The user just needs to click the CoolPreviews icon at the bottom of the browser and choose an item in the menu to disable or enable the CoolPreviews partly or completely.

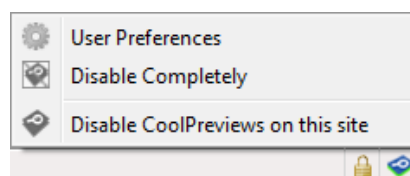


Figure 23: Switching On or Off in CoolPreviews

From this implementation we can get many ideas how we design our snippet. We can even use it to partly test some features that we want to implement later to know whether they are comfortable to use or whether we should find any other implementation.



## 4. Wiki Sniffer

### 4.1. Investigating Information Structure in the Wiki Pages

Each wiki application has a different structure of how the information is presented in their pages. In this thesis we want to investigate two different wiki applications: *MediaWiki* and *TWiki*.

#### 4.1.1. MediaWiki

*MediaWiki* is a web-based wiki software application used by, written by, and maintained by the Wikimedia Foundation. Originally it was developed to serve the needs of free content Wikipedia encyclopedia. It is used for all projects in the WikiMedia foundation, i.e. *Wikipedia*, *Wikibooks*, *Wikiversity*, *Wiktionary*, *Wikiquote*, *Wikispecies*, *Wikinews*, *Wikisource*, and *Commons*. It is also used today by companies for internal knowledge management and as a content management system (CMS) (54).



Figure 24: MediaWiki Logo

#### *Information Sources*

To get information, we need information sources. Because the information will be processed in the machine, we need sources which are also understandable by the machine. To find such sources, we have observed on how the information including the contents in the wiki pages is produced in the machine's way.

After several investigations we found that there are several information sources in MediaWiki environment which can be used in the wiki mining area:

- **HTML text**  
The first source that can be found directly by opening MediaWiki page is an HTML text. This source contains all information what people see whenever they open a page in the web browser.
- **MediaWiki API**  
Another source which can be used to obtain information from the MediaWiki engine is the *MediaWiki API*. An API (Application Programming Interface) is an interface which is implemented by a system in order to allow other systems to interact with it. MediaWiki API provides a set of procedure calls for communicating directly in a high level access to the data contained in the MediaWiki databases (54).

MediaWiki API can be accessed by requesting "[api.php](#)" together with a set of defined parameters via an HTTP request. The output can be in an XML format or in some other formats. Which output should be produced is defined in the HTTP request parameters.

This is an example of accessing English Wikipedia API by requesting maximal three external links of “Darmstadt University of Technology” article, where the output should be in an XML format.

The HTTP request is:

<http://en.wikipedia.org/w/api.php?action=query&prop=extlinks&titles=Darmstadt%20University%20of%20Technology&ellimit=3&eloffset=2&format=xml>”.

The API in English Wikipedia is located at <http://en.wikipedia.org/w/api.php>. The configured parameters are:

- action: what action should be performed; in this case querying; value: query
- titles: a list of titles to work on; in this case “Darmstadt University of Technology”; value: %20University%20of%20Technology
- prop: which properties to get for the titles; in this case external links; value: extlinks
- ellimit: how many links maximal to return; in this case three; value: 3
- eloffset: show existing external links after this offset; in this case two; value: 2
- format: The format of the output; in this case XML; value: xml

As an output, we get an XML text:

```
<api>
  <query>
    <pages>
      <page pageid="1664084" ns="0" title="Darmstadt University of
        Technology">
        <extlinks>
          <el xml:space="preserve">http://www.darmstadt.de</el>
          <el xml:space="preserve">http://www.darmstadt.ihk24.de</el>
          <el xml:space="preserve">http://www.etit.tu-darmstadt.de</el>
        </extlinks>
      </page>
    </pages>
  </query>
  <query-continue>
    <extlinks eloffset="5"/>
  </query-continue>
</api>
```

- **Wikitext in MediaWiki syntax**

Article’s content in MediaWiki is kept in the database in a form of Wikitext. Hence, the WikiText is also another important information source in MediaWiki. The format is in MediaWiki syntax. The Wikitext of an article can be obtained by requesting it via MediaWiki API.

The following command requests the wikitext of “Darmstadt University of Technology” article from the English Wikipedia in an XML format:

[“http://en.wikipedia.org/w/api.php?action=query&prop=revisions&rvlimit=1&rvprop=content&titles=Darmstadt%20University%20of%20Technology&format=xml”](http://en.wikipedia.org/w/api.php?action=query&prop=revisions&rvlimit=1&rvprop=content&titles=Darmstadt%20University%20of%20Technology&format=xml).

### *Information Sorts*

Several sorts of information can be found in MediaWiki pages. Most MediaWiki contents are focused on natural language texts. Images are also used to support the contents. Audio files in an Ogg Vorbis data format also can be found in some articles. Video is not yet fully supported in MediaWiki. Some plug-ins have been implemented but general uses are still rare. However, it has been reported that WikiMedia Foundation is close to launching an editable online video encyclopedia to enhance the current textual one (55).

Natural language text can be found as *flowing texts*, *lists*, as well as *tables*. Flowing text is needed as the input for generating summary and extracting keyphrases. Lists and tables are not the focus of this thesis since the summarization algorithm that we use is not intended to work on lists or tables but flowing texts.

The flowing texts of an article can be obtained from the HTML text as well as from the wikitext of the article. To process them by using our automatic summarizer and keyphrase extractor, they must be converted first to a plain text.

A non-natural language part that we also concern is *images*. Images can be presented in the snippet as a thumbnail<sup>18</sup> to support the summary with visual cognition. Listing images as thumbnails is also a good possibility to present them. This information parts can be obtained by extracting them from HTML text, wikitext, or by using MediaWiki API.

A structural part that can be obtained from the sources is *Headings*. This part can be used to construct table of contents. *Links* related to an article are also an important information sort that we can find in MediaWiki page. There are several types of links in MediaWiki:

- **Anchor links**

An *Anchor link* is a link which points to a certain position in the same page, for example, links in the table of content.

- **Internal links**

An *Internal link* is a link to another page in the same wiki. Pages on a MediaWiki are classified into collections called *namespaces*. Namespaces differentiate between the purposes of the pages at a high level. There are 18 namespaces provided by MediaWiki software. Articles with a certain namespace except *main namespace* are indicated by a prefix of its namespace ID which is put behind the colon “:”. The “Help:Special\_pages” link indicates that the page about some help of special pages is

---

<sup>18</sup> Thumbnail is a reduced-size version of a picture.

in the “Help” namespace. The “User:Fabian.l.tamin” link indicates that the page about a user named “Fabian L. Tamin” is in the “User” namespace. Articles without namespace prefix are standard content articles which are in the main namespace. We put our concern for two namespaces, i.e. the *main namespace* and *category namespace*.

The “Category:X” page collects information about article links which are grouped together into a category named “X”. For example, the “Category:2009\_software” page collects information about links of software articles which are published in 2009. Category links can be found at the bottom of MediaWiki pages. An article which is assigned to a specific category can be assumed that the article is tagged with the assigned category as its tag. Hence, categories in MediaWiki represent tags, and they also belong to a sort of suggested information which is assigned manually by human.

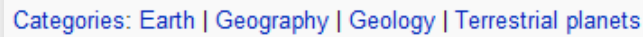


Figure 25: Categories assigned to “Earth” article in Wikipedia (56).  
This part can be found at the bottom of the page.

- **Interlanguage links**

An *Interlanguage link* is a link to another page which is registered as another language version of a wiki page. If another language version of a wiki article exists, it can be linked to the article manually by describing it in the wikitext. It can be done by creating a wikilink in the wikitext. The link’s name must be the article’s name in the other language which is prefixed by the language code in ISO 639<sup>19</sup> coding. For example, by typing “[[id:Bahasa]]” in the wikitext of the “Language” article, an interlanguage link to the Indonesian version of “Language”, i.e. “Bahasa”, will be provided in the sidebar of the “Language” article.

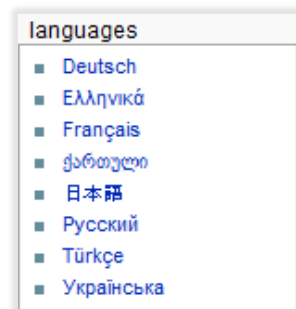


Figure 26:  
Interlanguage links

- **External links**

An *External link* is a link to other website. External links are also a useful information sort. They can be assumed as information which is suggested by the author of the article.

All links that can be found in a page are called *outgoing links* or *forward links* because if we click the link, we go out from that page to another page which is pointed by the link. However, there is also another interesting type of links, i.e. *incoming links* or *backlinks*. Backlinks of a page “X” are a set of pages which have outgoing links to page “X”. In other words, backlinks are a set of predecessors of a certain page, whereas outgoing links are

<sup>19</sup> ISO 639 is the set of international standards that lists short codes for language names (71).

a set of successors of a certain page. Both outgoing links and backlinks of an article are links which are related to the article according to the structure of their interconnectivity.

Information about backlinks of an article in MediaWiki can be accessed by using MediaWiki API or by parsing them from “what links here” page of an article. The “what links here” page of an article is accessible by prefixing the article name with “Special:WhatLinksHere/”.

For example requesting <http://en.wikipedia.org/wiki/Special:WhatLinksHere/Europe> will return a page which contains a list of 50 backlinks found in “Europe” article. By using some request parameters the number of maximal returned backlinks can be adjusted.

Another structural information sort that people can find in many MediaWiki articles is the so called *wiki infobox*. This part of information is very important for understanding about the content in the article. Ye Shiren et al. (8) reported that summarizing definition from Wikipedia can be enhanced by obtaining information from the article’s infobox.

<b><i>Streptomyces achromogenes</i></b>
<b>Scientific classification</b>
Kingdom: <a href="#">Bacteria</a>
Phylum: <a href="#">Actinobacteria</a>
Order: <a href="#">Actinomycetales</a>
Family: <a href="#">Streptomycetaceae</a>
Genus: <a href="#">Streptomyces</a>
Species: <b><i>S. achromogenes</i></b>
<b>Binomial name</b>
<b><i>Streptomyces achromogenes</i></b>
<small>Okami and Umezawa 1953</small>

Figure 27: Infobox (70)

### 4.1.2. TWiki

TWiki is a flexible and powerful wiki application which is normally used in enterprises for collaboration platform and web application platform. Among others, TWiki is mainly used in commercial environment, such as in NASA, Yahoo, SAP, Motorola, Disney, and British Telecommunications. It is often used internally to manage project planning, for documentation and for collaborative information exchange (57).



Figure 28: TWiki Logo

TWiki is a *structured wiki* (58). It means that it allows the user to add structure, such as database-like manipulation of fields stored on pages, *as needed*. “As needed” means that TWiki allows the user to add and editing articles in a simple free-form wiki-way, but additionally, it also allows them to create structured wiki application with complex forms, queries, and reports if they need some automation. Hence, the complexities of the articles in TWiki are very varied, from the simple ones to the complex ones.

### Sources

Similar to MediaWiki, various information sorts can be obtained from TWiki by using the *HTML text* and the *wikitext*. The HTML text can be accessed directly by requesting the page URL. The wikitext (in TWiki syntax) of a page can be extracted and parsed from the *raw view* page.

The raw view of a TWiki page can be accessed by using parameter “raw=on” in HTTP request. For example: By requesting “/wiki/bin/view/Teaching/FabianTamin?raw=on”, we will get the raw view of “FabianTamin” page (“/wiki/bin/view/Teaching/FabianTamin”).

TWiki API is not the major of our information source since it is not provided in TWiki by default. However, if it is needed, TWiki API can also be integrated on TWiki platform by installing plug-in named *TWiki Plugins*. By using this plug-in, one has possibilities to hook into the core of TWiki codes from the external Perl plugin module.

### Information Sorts

Compared to MediaWiki which puts more focus on textual contents, the contents in TWiki pages are more varied both for its natural language contents and for its non-natural language contents. Most of HTML tags can even be used in TWiki without problem while MediaWiki currently allows only 41 tags to be used in its platform. Overall, there are 91 tags in HTML 4.01 specification (59) (60). TWiki even allows its users to easily embed various HTML objects such as videos from YouTube by using normal HTML codes. Of course, it is worth of noting that the restriction of using a certain HTML tag does not always mean that the feature of that HTML tag is not provided by the platform. It can also be provided in a form of wiki syntax, and the restriction is done just to make sure that the wikitext is always written consistently. Moreover, as mentioned previously, TWiki can also contain many structural text as well as graphical components in its complex pages.

The following points are the information sorts of natural language contents that can be extracted from a TWiki page:

- **Flowing text**

Flowing text is an important information sort for the input of summarization and keyphrase extraction method. Flowing texts can be obtained from a TWiki page by extracting them from the HTML text or the wikitext.

- **Links**

Similar to MediaWiki Links, links in TWiki can also be classified as *internal links* (including links to other articles and anchor links) or *external links*. They can be extracted from the HTML text or the wikitext. However, by using the structure of the information sources, it is sometimes not as easy as in MediaWiki to distinguish whether a link is an external link or an internal wiki link because TWiki also allows that links can be created via HTML Tag “<a>” in addition to its wiki syntax. In

MediaWiki, links can be created only by using wiki syntax. It produces better structural information in the rendered HTML page.

TWiki also provides a list of backlinks of its pages. A list of backlinks of a page can be obtained by accessing its backlink HTML page. It can be done by replacing the keyword “view” in the article page URL with keyword “oops” and suffixing it a request parameter. The request parameter can be either “template=backlinksweb” to search backlinks in the *TWiki web* of the page or “template=backlinksallwebs” to search backlinks in all TWiki webs. For example:

“/wiki/bin/oops/Teaching/FabianTamin?template=backlinksweb” requests the list of backlinks of article “FabianTamin” from “/wiki/bin/view/Teaching/FabianTamin” in scope of web “Teaching”. TWiki webs are a structural part in TWiki hierarchy. The list of TWiki web is normally shown on the TWiki sidebar. It is indicated with a small box with several colors.

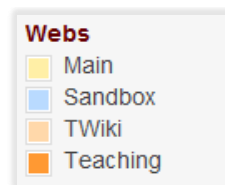


Figure 29: TWiki Webs – Hierarchical Part of TWiki

- **Tags**

TWiki provides tags functionality that allows its users to assign tags onto TWiki pages. In contrast to MediaWiki’s tags which can be found at the bottom of the page, TWiki’s tags can be found at the top of the page. Tag Feature in TWiki also has functionality for voting. Each time a user assigns a certain tag to a certain page, the number of votes that the tag is assigned on the page will be incremented.

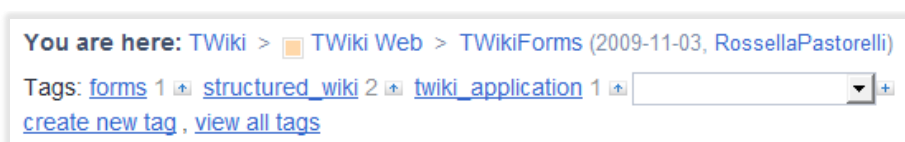


Figure 30: Hierarchical Path and Tags in TWiki  
(from <http://twiki.org/cgi-bin/view/TWiki/TWikiForms>)

The picture above shows us that “TWikiForms” page has three tags assigned: “forms”, “structured\_wiki” and “twiki\_application”. One person assigned “forms” tag and “twiki\_application” tag to the article. Two person assigned “structured\_wiki” tag to the article.

Tags can be obtained from a TWiki page by extracting them from the HTML text.



- **Hierarchical path to the page**

TWiki also provides information about the hierarchical path of a TWiki page. It can also be seen in Figure 30 (“You are here”). Similar to tags, it can be parsed from the HTML text.

#### 4.1.3. Information Sort Candidates to Present in Page Overview

From several information sorts provided by each wiki applications, we concentrate on the information sorts which are provided by the two applications, i.e. flowing text, headings, internal wiki links, external links, backlinks, and tags (categories) as natural language information sorts, as well as images as the only non-natural language information sort. From headings we planned to construct table of contents as a derived, constructed information sort. From flowing text, we planned to construct new information sorts, i.e. summary and keyphrases, by employing natural language processing (NLP) methods. How important each kind of information is for users were observed later in the first user study.

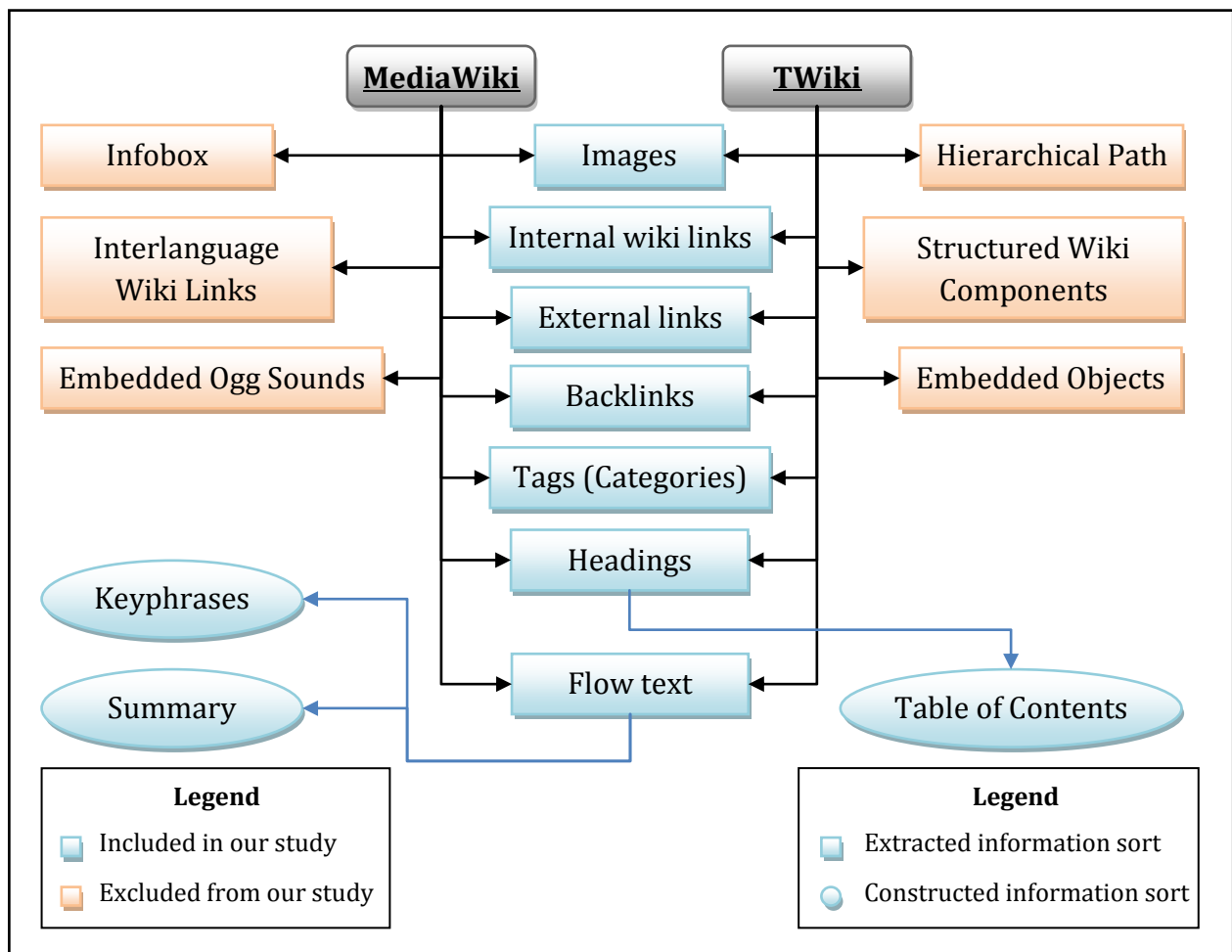


Figure 31: Information Sorts from MediaWiki and TWiki



## 4.2. Designing Mockups

After investigating which kinds of information that can be presented to users, the next step was to find the way how to present them to users. What we need here is a user interface. However, to reduce risks of unusable interface, we began this task with designing several *mockups* for our snippet. Designing several *mockups* is a part of rapid prototyping method to get useful feedback as fast as possible (9).

*Mockup* is a scale or full size model of a design or device which shows how the design will appear. In contrast to a real prototype, mockup does not provide high interactivity and cannot be used to test the system. Its interactivity is limited by the simulation on papers. A real prototype provides at least a part of functionalities of the system so that it can be used to partly test the design. Although a mockup is not a real prototype, sometimes mockup is also called paper prototype because it has a similar goal as a normal prototype.

### 4.2.1. Design Candidates

To produce the mockups, we used a graphic application named *Inkscape*. Inkscape is an open source<sup>20</sup> vector graphics editor, with capabilities similar to Adobe Illustrator, CorelDraw, or Xara X, using the W3C standard Scalable Vector Graphics (SVG) file format. Later in the first user study, the mockups were printed on papers and presented to the participants.

#### *Overlay Snippet as our Choice*

In the chapter 3, we have discussed about some appearance models of snippet. We found that the inline snippet is not suitable for wiki articles since we are reading a flowing text in the article. Inline snippet between words in a sentence will distract the user during reading. Considering the behavioral dimension of browsing, i.e. the eye movements, we also did not choose the frame snippet. The position of a frame snippet is fixed. By using this model, a user has to move their eyes much further in compare to the other models. As the result, the user can easily lose their position in the main article after they read the information in the snippet.

Thus, among the three models (overlay snippet, frame snippet and inline snippet), we chose overlay snippet as our model. Moreover, overlay snippet gives us a possibility to implement multiple instances feature. Considering the resource dimension of browsing, i.e. form aspect, this functionality is very important. Many people like to print out the documents on papers because they can compare them more easily side by side. By using multiple snippets, users can compare multiple articles side by side like when they are comparing something using sticky notes. For example, a user reads about Mac OS X in Wikipedia and encounters two articles about Mac OS X Leopard and Mac OS X Snow

---

<sup>20</sup> Open source is practices in production and development that promote access to the end product's source materials, typically their source code.



Each method has advantages and disadvantages. Hovering link is pretty easy. Users do not need to give their effort to click or do a complicated thing to activate it. Automation works here for us. However, initializing comes with delay. The users, who do everything very quickly and want everything to be quick, sometimes, are not patient enough to wait for 500 milliseconds, because they can do it more quickly by clicking the icon. Moreover, for some users, it is annoying if snippets appear whenever they do not intend to initialize them. For them, it is better to click the icon by themselves for initializing it than to let the snippet open automatically. However, clicking icon needs more effort. To click it, it needs both precision and time. For users who do not skillful in using mouse, and for them who have some patient to wait, hovering link is much more comfortable and efficient. Therefore, to find the design to implement later in our snippet, we created two mockups to compare them in the first user study.

In our design, the icon should be above the link instead of beside it because for some people who like to read with cursor, icons beside the links can be distracting and annoying.

### Information Sorts and Components

Layout represents how the information is structured, organized, and presented. Four layout mockups were created, and they were compared later in the first user study.

Some sorts of information are presented in the Mockups:

- **Summary**

Summary is one of the main information sorts to present. Its components are the *summary itself as a flowing text* [1], *thumbnail image* [2] for supporting the text with visual cognition about the article, and *detail-meter* [3] for increasing or decreasing the length of the summary. Summary part is presented equally in the four mockups.

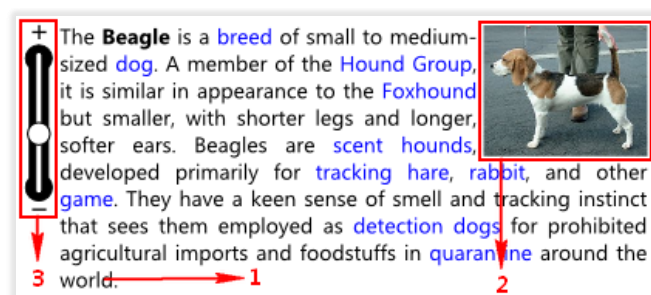


Figure 34: Summary Parts

- **Table of contents**

Table of contents is presented as a list of headings with indentation level, and it is also presented equally in the four mockups.

- **Keyphrases**

Keyphrases is presented as weighted phrases. The font size of the phrases with a higher weight is larger than the font size of the phrases with a lower weight. This

mechanism is implemented to induce the skimming behavior for quickly recognizing the information. Keyphrases is ordered alphabetically to support the scanning behavior.

- **Related articles**

Related articles part contains *outgoing internal links* and *backlinks*. They are also presented in alphabetic order for supporting scanning behavior. In one mockup, outgoing links and backlinks are mixed together as related article. In two mockups, they are grouped together but can be separated by using navigation. In one other mockup, they are totally separated.

- **External links**

This part provides a list of external links in an alphabetical order.

- **Tags**

Tags (categories) are also ordered alphabetically as a list.

- **List of Images**

Image feature is also provided in one mockup as a list of thumbnails.

- **Sniffed articles**

Sniffed articles are a history feature of the snippet. It contains a list of article links which have been sniffed previously on a page so that a user can reopen or resniff the articles more easily. This feature is an extra feature and not a mandatory feature to implement. We want to get some feedback from the participants about this feature.

The following points are the outline of main components in the snippet box as shown in Figure 35:

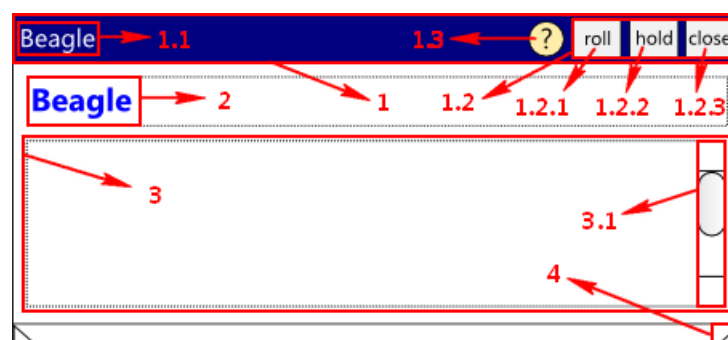


Figure 35: Snippet Box' Parts

- **Snippet's dragger [1]**

With this component, a user can drag the snippet and control the dialog's behavior. It contains *page title* [1.1], *snippet behavior control* [1.2], and *help button* [1.3]. Snippet behavior control contains 3 buttons: *roll button* [1.2.1], *hold button* [1.2.1], and *close button* [1.2.3]. Roll button is used for activating or deactivating auto-collapsible functionality. It is a similar feature as previously introduced in section

3.1.3 “Collapsible Dialog Window”. The hold button is a toggle for activating and deactivating auto-close. It is a similar feature as previously introduced in section 3.2.2 “Firefox CoolPreviews”. The close button is a button for manually closing the snippet. It gives users a possibility to close the snippet manually, for example, if hold or auto-collapsible mode is activated. The goal of the implementation of these three buttons is to enhance the usability and the utility of “multiple instances” feature. The help button is an additional button to get a little help or tips for using the snippet.

- **Clickable title [2]**

This part contains the title of the currently sniffed page. If it is clicked, the full page will be opened in the browser.

- **Content frame [3]**

This frame contains *displayed information*, e.g. summary. It can contain a *scrollbar* [3.1] as its navigation control.

- **Snippet’s resizer [4]**

A user can resize the snippet by dragging this part.

### Snippet Layout Mockups

In the following subsections, the four mockups are discussed in details.

#### 1<sup>st</sup> Candidate: Content Switcher



Figure 36: 1<sup>st</sup> Candidate  
(summary is selected)

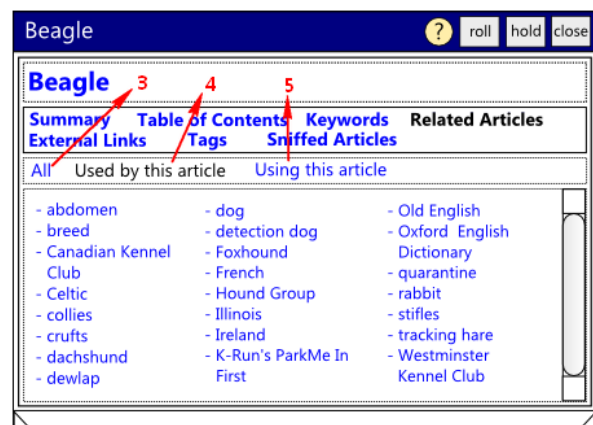


Figure 37: 1<sup>st</sup> Candidate  
(related article is selected)

As shown in Figure 36, by using the *navigation part* [1], users can choose the information sort which should be displayed in the *content frame* [2]. The *selected item* [1.1] in navigation is distinguished from the *unselected items* [1.2].

Figure 37 shows how internal links are displayed to users. They can be grouped together or sorted by using the sub-navigation (all related links [3], only outgoing links [4], only backlinks [5]).

## 2<sup>nd</sup> Candidate: Sidebar Extras and Main Frame



Figure 38: 2<sup>nd</sup> Candidate  
(sidebar is collapsed)

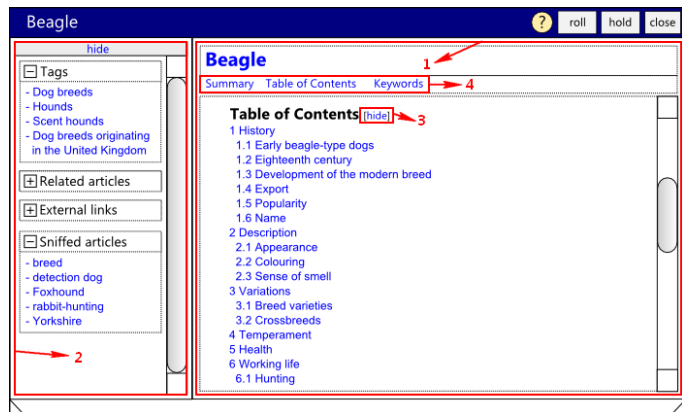


Figure 39: 2<sup>nd</sup> Candidate  
(after table of contents in navigation was clicked;  
sidebar is expanded)

The next candidate uses two frames to present the information. The main concept of this design is to present multiple sorts of information simultaneously. The main sorts of information such as *summary*, *table of contents*, and *keyphrases* are presented simultaneously in the *main frame* [1]. The other extra information such as *tags*, *related articles*, *external links* and *sniffed articles* are on the *sidebar* [2], and they are presented as vertical lists. In this design, outgoing wiki links and backlinks are grouped together as *related articles*.

Table of contents and keywords are collapsible via *show and hide button* [3]. By using *shortcuts* in the *navigation* [4], the clicked item's content will be automatically expanded and well positioned.

One advantage of this design is that one can see several kinds of information together. The sidebar can be collapsed to avoid information overload.



3<sup>rd</sup> Candidate: Sidebar Extras and Main Frame with Content Switcher

Figure 40: 3<sup>rd</sup> Candidate  
(viewing current sniffed article;  
sidebar is collapsed)

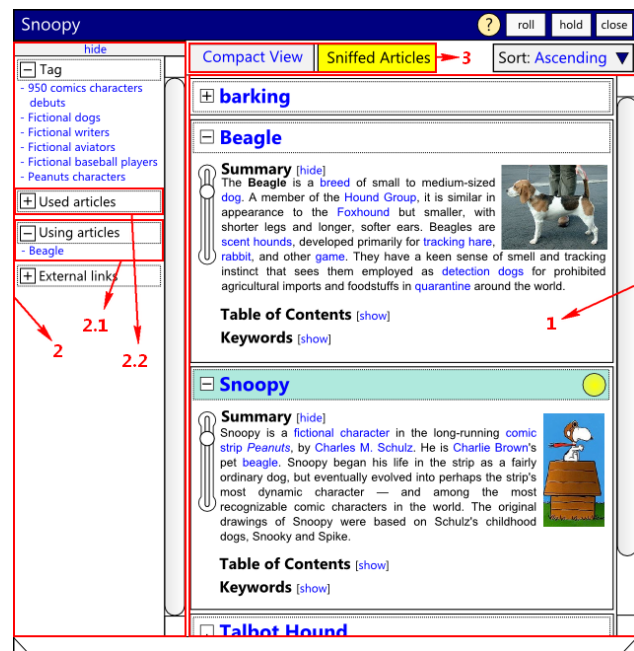


Figure 41: 3<sup>rd</sup> Candidate  
(viewing history or sniffed articles;  
sidebar is expanded)

This design candidate can be seen as the combination between the first and the second candidate. The *main information in main frame* [1] can be switched by using *main frame view switcher* [3] similar to the first candidate. The extra information is presented on *sidebar* [2] similar to second candidate. The difference is that the main information can be a set of several information sorts, for example, it can present summary, table of contents, and keywords together like in the second candidate.

In this design, *outgoing internal links* [2.1] and *backlinks* [2.2] are presented separately on sidebar. Sniffed articles are presented in the main frame and can be directly opened in the current snippet.

This design has a specialty to display multiple information sorts simultaneously and additionally, to implement other feature-rich views, e.g. sniffed articles or list of images.

## 4<sup>th</sup> Candidate: Main and Footer Frame

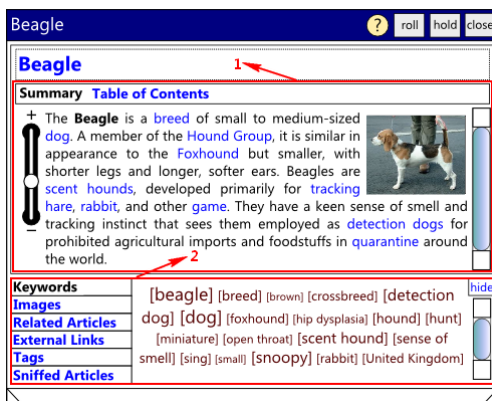


Figure 42: 4<sup>th</sup> Candidate  
(summary and keyphrases are selected)



Figure 43: 4<sup>th</sup> Candidates  
(summary and collapsed footer)

The *main frame* [1] of this candidate is similar to first candidate but this candidate also provides extra information in *footer frame* [2] so that two sorts of information can be presented simultaneously. Main frame contains main information which is switchable by users. The footer frame contains extra information which can support the information in the main frame. The footer frame can be collapsed to reduce the information overload.

Not like the other candidates that use *vertical list* to display most of their information sorts, this candidate uses *horizontal list* concept to display its sorts. Squared brackets are used as separator between keyphrases. The other candidates uses comma to separate them.

Keywords	All	Used by this article	Using this article	hide
Images	[abdomen] [breed] [Canadian Kennel Club] [Celtic] [collies]			<input type="checkbox"/>
Related Articles	[crufts] [dachshund] [dewlap] [dog] [detection dog]			<input type="checkbox"/>
External Links	[Foxhound] [French] [Hound Group] [Illinois] [Ireland] [K-Run's Park Me In First] [Old English] [Oxford English Dictionary]			<input type="checkbox"/>
Tags	[quarantine] [rabbit] [stifles] [tracking hare]			<input type="checkbox"/>
Sniffed Articles				<input type="checkbox"/>

Figure 44: Related articles presented in a horizontal list

## 4.3. User Study 1 – Process Data

### 4.3.1. Executive Summary

After creating mockups, our next step was to perform the first user study to obtain process data. As mentioned previously, this study consisted of informal data and qualitative data instead of quantitative or statistic data. This test ran individually for each participant to get better feedback from each individual. The goal of this study was to observe which parts of the design in the mockups work well and which parts do not work so that we could decide better what we had to do later in the real design implementation.



In this study, Wiki-Sniffer project was introduced to each participant so that they knew about the overview and the goal of this project. Afterwards, the plot and the tasks of this user study were described to the participant.

As the task, several mockups were compared by the participants. At the end of this study, the participants gave their opinion about the design of the mockups, and they suggested some improvements which could be done in the next iteration. In this study, user design aspects as well as information sorts were also observed.

#### 4.3.2. Task

After some introductions as well as some overviews about the project and the tasks, the participant compared the first two mockups about how the snippet should be initialized. They chose their preference and explained about their decision. The participants could also suggest their own solution.

Afterwards, the next task was to compare the four mockups for the snippet layout. Among these four mockups, they chose one mockup as their favorite design, and they grounded, what made them choose it. Several aspects, such as *simplicity*, *look and feel*, *clarity*, *comfort*, and *features* were investigated. Simplicity is about how simple the user interface is. Look and feel represents the aesthetic of the design. Clarity means whether each part of the interface is intuitive and understandable according to the context of the interface elements. Comfort is about how comfortable the design is. Features aspect is whether the design provides sufficient features, not too few and not too many. For each aspect, the participant rated their chosen mockup and explained how important each aspect is for them. The participant could suggest, if there were any other mockups that did better in each aspect.

The next task was to investigate each information sort in form of a feature as its representation. The features to investigate were summary, table of contents, keyphrases, related articles, external links, tags, list of images, and sniffed articles. For each feature, the participants rated their chosen mockup and informed how important each information sort is for them. Like in the previous task, the participants could suggest if there were any other mockup that did better for the investigated feature.

Afterwards, the participants were questioned about the size of the snippet and whether there were any distracting or confusing components in the mockups. They could also suggest if there were any modifications which could be done to improve the design. Whether the participants recognized the snippet control buttons by themselves was also observed in this study. At the end, they gave their opinions about how important each control button is.

#### 4.3.3. Participants

There were six persons who participated in this user study. They were between 20 to 30 years old. Four of them were (business) computer science students and doctoral

researchers. Two of them are electrical engineering students. All of them were Wikipedia daily users.

#### 4.3.4. Result

All participants preferred to initialize the snippet by hovering links with delay rather than clicking the icon. Some participants explained that they do not really like to click. For them, it is better with some automation in spite of delay because the delay is still tolerable.

Most of them like to see more than one information sort simultaneously. Three of them chose the second candidate (with sidebar) as their favorite mockup, and two of them chose the fourth candidate (with footer frame). A participant chose the first candidate (with switcher) because of the simplicity of design concept, and he preferred not to see too many information sorts simultaneously. Nevertheless, he liked to have such sidebar feature so that he can also see information simultaneously whenever he needs it. Some other participants commented that the first candidate was actually very simple but it was not really comfortable because they would need to switch very often to see several information sorts. The third candidate was too complicated and confusing to most of participants. According to the results, it is considered only for advance but not for the users in general.

For most participants, simplicity was one of the most important aspects. Nevertheless, a participant preferred feature-rich interface to a very simple interface. It means that it should be a balance between the simplicity and features. Although most people need simple user interfaces but there are also people who need some utilities. Aesthetic is for most of them important, but some commented that the simplicity should be the priority. Clarity and comfort were also important aspects for most participants.

Summary, list of images, and keyphrases were sorts of information which are favorable for most participants. Although some participants commented that table of contents was not really important, some of them informed that they would use it pretty often. Some participants thought that external links were also an important feature. One participant explained that he had often used external links in Wikipedia to search information suggested by the author of the articles. Related articles and tags did not get much support from the participants, but it is probably because of the unclear labeling of the design in the mockups. Furthermore, many participants did not even know about Wikipedia categories, although they were daily users of Wikipedia. Sniffed articles also did not get much support from participants.

Another interesting information sort which was suggested by one of the participants is links to other language in Wikipedia. From this suggestion, we got an idea of using inter-language wiki link to provide a new sort of information targeted to MediaWiki environment, i.e. a list of article translation links. Here, we can provide the translation names with their languages as a list. This sort of information is very useful for people who have mother languages other than English.

Most participants recognized the snippet control buttons very quickly. Some participants were even fascinated about them. Both the hold/auto-close button and the roll/auto-collapsible button got some supports from the participants. For some participants, the size of the snippet in the mockup tended to be too large.

#### 4.3.5. Feedback

From this study we got some feedback about the distracting component and some ideas to improve our design:

- The labels of the outgoing links and backlinks which were used in the mockups were confusing and unclear.
- The implementation with two titles is unnecessary. Combining them together can save some space so that the snippet can be presented in a smaller size.
- Detail-meter in the mockups is distracting and unclear. Some participants preferred a simple “more” button to detail-meter.
- A participant commented that there were too many frame borders in the mockup which were distracting.
- A participant who chose the first candidate as his favorite mockup commented that there were too many menu items in the navigation. From this feedback, we got an idea to implement a submenu to avoid overloading information.
- The outline between two items is not so clear when comma is used as their separator. The boundaries between two items are clearer by using squared bracket but the snippet seems somehow being overloaded by information. A better solution should be investigated.
- A participant suggested presenting some tags together in the main frame.
- A goal of using a horizontal flowing list is to save some space. However, it also comes with drawbacks. A participant mentioned that he could scan items in a vertical list faster than a horizontal list. This is because we read horizontally. In a horizontal list, scan and reading are sometimes mixed together more easily. As a trade-off, the scan becomes slower.

#### 4.3.6. Suggested Improvements

##### *Must do*

There are some improvements that must be done in the next iteration of our design implementation, i.e. finding suitable labels for the outgoing links and backlinks, simplifying menu items in the navigation by using submenu.

##### *Should do*

This category includes the problems that are annoying but tolerable. The size of snippet should be optimized by grouping together the title in the dragger frame and the title in the main frame. Designing layout with unnecessary frame borders should be avoided.

### *Could do*

This category includes some improvements that can be done, but it needs too much effort to implement. It is better to keep the ideas on the back burner for the next iteration. One of them is to implement auto-resizing snippet according to the content length instead of manual resizing. Another improvement that can be done is presenting inter-language wiki links as a new sort of information in a form of a list of article name's translations.

## 4.4. Creating Components

After the results have been obtained from the user study, the next task to do was to create the components and to implement the suitable design. Two main groups of the components are the back-end and the front-end. The back-end represents the internal components or the core of the entire system. It contains the algorithm and it is where the main processes should be done. The front-end represents the interface between users and the internal components or the back-end.

### 4.4.1. Creating Back-End

For the back-end we use java programming language and UIMA framework. In our implementation, several annotators and analysis engines have been created. Each type of annotation except internal link and external link has its own Analysis Engine. Eight annotators have been implemented in our programming design. They are PageMetadataAnnotator, HeadingAnnotator, LinkAnnotator, BacklinkAnnotator, TagAnnotator, ImageAnnotator, TextBlockAnnotator, and HTMLTextBlockAnnotator. There are two Sofas used for creating two views: "WikiText view" and "HTML view". Nine types are used for the annotators: PageMetadata, WikiTextBlock, HTMLTextBlock, Heading, InternalLink, ExternalLink, Backlink, Tag, and Image.

PageMetadataAnnotator is used as the container for page metadata information, such as page title, page URL and wiki syntax. HeadingAnnotator is used for annotating headings from the source. LinkAnnotator is used for annotating all outgoing links including internal wiki links and external links. BacklinkAnnotator is used for annotating backlinks. TagAnnotator is used for annotating TWiki tags and MediaWiki categories respectively. ImageAnnotators is used for annotating images. TextBlockAnnotator is used for annotating text blocks from wikitext. A text block represents a plain text part of a wikitext. HTMLTextBlock is used for filtering HTML source by annotating only relevant HTML parts. Each part is annotated as a block. Tables are excluded from HTML text by using this annotator.

Using separate annotators comes with benefit. The annotators can be used separately or combined together depending on our needs. To combine several annotators, aggregate analysis engines can be used. External link and internal link are combined together because of a performance issue and its semantic similarity.

For keyphrase extraction, we use TextRank keyphrase extractor from DKPro. For the summarization, we also used DKPro component, i.e. LexRank summarizer. However, while we were analyzing the results, we found that the summaries which are produced by LexRank are sometimes insufficient to understand the article. It is because it often does not include the sentences in the beginning of an article which are often very important to understand the text. Moreover, the sentence in the summary which is produced by LexRank often contains anaphora without mentioning the referenced word first. Examples for anaphora are this, that, he, she, it, therefore, and afterwards. As we know, those words are normally used to represent something which has been mentioned previously. Any approaches for filtering the sentences with anaphora or any other solutions are still needed to produce better summaries.

Hence, after some deliberations, we decided to include the first paragraph of an article into our summary. The text generated by LexRank is presented behind the first paragraph as the next sequel of the summary. Our decision was also based on Ye, Shiren et al. (8). They noted that Wikipedia provides a wealth of knowledge, where the first sentence, infobox (and relevant sentences), and even the entire document of a wiki article could be considered as diverse versions of summaries (definitions) of the target topic.

#### 4.4.2. Creating Front-End

For the front-end, JavaScript is used to communicate with user internet browser. The main library we use is *jQuery*. *jQuery* is a JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development (61). This library contains several features that are needed for the user interface implementation:

- **HTML DOM element selection, traversal and modification**  
By using this feature, we have possibilities to freely modify HTML DOM<sup>21</sup> element in the HTML page including the snippet part. Traversing and selecting internal wiki links in wiki pages as well as selecting snippet elements for further process can be done by using this feature.
- **CSS manipulation**  
The style of each HTML element can be manipulated freely by using this CSS<sup>22</sup> manipulation feature, for example to change the font sizes of an HTML text. This feature also gives us a possibility to hide or display any elements in the snippet.
- **Events**  
This feature is needed for detecting mouse event, for example, to detect hovering mouse event as well as clicking event.

---

<sup>21</sup> DOM (Document Object Model) is a specification of an interface for accessing or interacting with HTML and XML document.

<sup>22</sup> CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation semantics (that is, the look and formatting) of a document written in a markup language (72).

- **Ajax**  
Ajax (Aynchronous JavaScript and XML) is a concept of asynchronous data transfer between a server and the browser. This feature is used for the communication between the client browser and the server, in this case is Wikulu server.
- **Effects and animation**  
Last but not least, JQuery provides nice effects and animation which can be use to support the aesthetic of our user interface element, e.g. sliding and fading effect.

The further details how the design is implemented will be discussed in section 4.5.

#### 4.4.3. Integration of Back-End and Front-End

To integrate Java codes with JavaScript codes so that they can communicate each other, DWR library is used. DWR (Direct Web Remoting) is a Java library that enables Java on the server and JavaScript in a browser to interact and call each other as simply as possible. DWR is the Ajax library for Java in the server side, whereas JQuery is the Ajax library for JavaScript in the client side. DWR enables that a Java object can be converted to JavaScript object. We use this functionality to convert our *java facade controller class*<sup>23</sup> to a JavaScript class. By using DWR, the Java class methods and attributes of that class are accessible by using callback methods (asynchronous methods) in JavaScript.

Among the three structural models introduced in the section 3.1.1 about expanded snippets, we chose the first model, i.e. getting the snippet data by user request. In compare to the other models, this model is much simpler to implement, and on the other hand it can save much more bandwidth and memory. Sending data in a single blow as introduced in the second model is unsuitable for our implementation. It is because of the dense link structure of wiki articles. As previously mentioned, there are many Wikipedia articles which have 100 to 500 forward links in a page. Prefetching the snippet data as introduced in the third model also not always an optimal solution since the snippet data itself also consumes some memory in the client browser. Prefetching ineffective data can slow down the computers and cause discomfort.

---

<sup>23</sup> Façade controller class is a class which has a responsibility as an interface or a gate between internal components and the external components.



## 4.5. Wiki Sniffer's Design Implementation

### 4.5.1. Wiki-Sniffer Layout and Information Structure



Figure 45: Wiki-Sniffer Layout

After some deliberation based on the overall feeling from the first user study, we decided to choose the fourth candidate (main frame and footer frame) with some modifications as our model. Although the sidebar implementation got many supports from the participants, it tends to produce a huge snippet instead of the compact one. As previously mentioned, some participants also commented that the snippet size of the mockup was somehow too large. Listing related link in a small sidebar is also ineffective because many articles have pretty long names.

The design layout can be grouped to four main components: *dragger frame* [1], *main navigation* [2], *content frame* [3], *tag frame* [4], and *keyphrase frame* [5]. Three information can be viewed simultaneously, i.e. the information in content frame, tags/categories, and keyphrases. The information in the content frame can be either summary, table of contents, related articles, external links, backlinks (what links here), or a list of images. The content can be chosen in the main navigation or the submenu of “more action”. This implementation is similar to the design of the first mockup but the displayed menu items are reduced by using submenu to avoid information overload. The submenu is opened automatically by hovering “more action” menu item. The



selected item [2.1] font style is also distinguished from the unselected ones [2.2] to enhance the clarity.

A Summary contains a flowing text [3.1] as its main part. If the summary is longer than one paragraph, toggle more [3.2] is displayed. By clicking the toggle, the summary will be expanded. Additionally, a thumbnail [3.3] will be displayed if at least an image is found in the article.

Only maximal five categories [4.1] are shown at snippet initialization. To show more categories, toggle “more/less” [4.2] can be clicked. Keyphrases [5.3] can be hidden manually to avoid information overloading. As keyphrase's label we chose “keywords or keyphrases” instead of words “keyphrases” to enhance the context and information scent. It is done because, the word “keyphrase” is still rarely used and many people still do not recognize about its meaning. As the separator between two items, middle dot [5.4] is used instead of comma or squared bracket. It gives better clarity to show the boundary between two items. Additionally, it does not cause an impression of information overloading. When a keyphrase is clicked it will search the term in a previously defined user engine, for example, Wikipedia search. The search engine itself can be defined in the JavaScript data of Wiki-Sniffer.

Similar to the design in the mockups, the table of contents is presented as a list with indentation levels. If the item is clicked, it will lead the user to the pointed subsection of the article.

The snippet is freely draggable by using dragger frame. The title in the dragger frame has dual functionality: click or drag. It is a hybrid implementation of the two titles in the mockups. It can be clicked like a normal link to open the page but a user can also drag it to move the snippet.

To reduce user's memory load, a small details is provided by giving a tooltip to each component in the snippet.

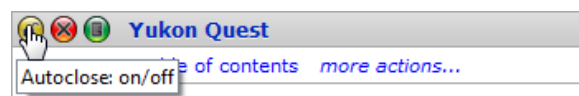
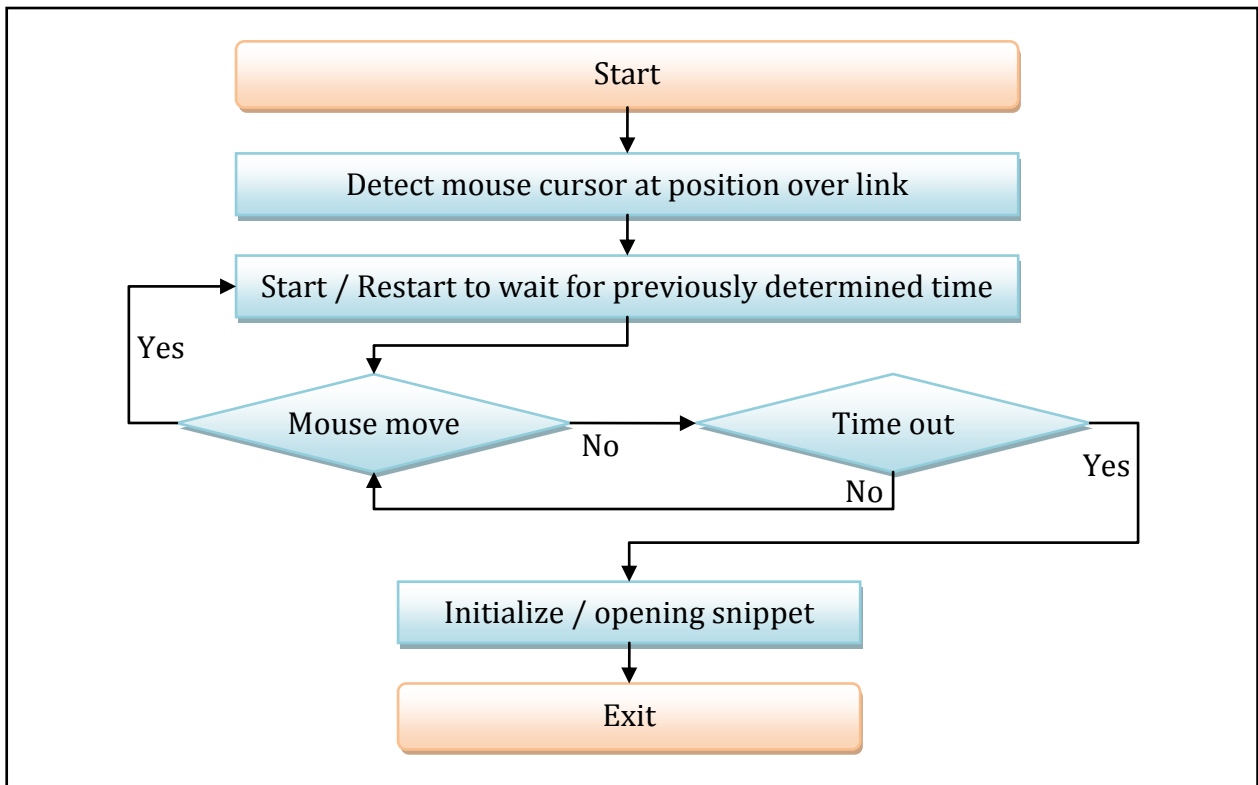


Figure 46: Implementation of Tooltip

#### 4.5.2. Opening and Closing the Snippet

Based on our first user study, we implemented the automatic mechanism for snippet initialization, i.e. by hovering a link with time delay. To enhance the comfort for the people who use mouse cursor for reading, intelligent mouse movement detector has been implemented. This mechanism detect whether a user really wants to open the snippet by hovering the link or just accidentally hovers it during reading.



**Figure 47: Flowchart for Opening the Snippet**

This mechanism prevents the snippet is initialized while the mouse is still moving. Mouse cursor which is moving continuously is assumed as a sign of cursor reading.

Another mechanism which has been implemented is an easy reversal of snippet initialization. If a snippet is initialized, and the mouse cursor has not entered the snippet, the snippet can be closed automatically without delay by moving the mouse cursor away from the snippet. For example, when a snippet is accidentally initialized but the user wants to abort their action. Considering the user interface design guidelines, this mechanism is important as a mechanism for reducing errors and easy reversal action. Moreover, this mechanism also facilitates the user for fast closing whenever a user just wants to quickly peek at the information in the snippet.

In the main scenario, the snippet is closed automatically after the mouse cursor has left the snippet box for a while. However, if the mouse cursor is moved back to the snippet before it automatically closed, the closing mechanism will be aborted. It is done to avoid the hypersensitivity of mouse precision. This mechanism is also done to reduce errors for enhancing the comfort. The details can be seen in the Figure 48.

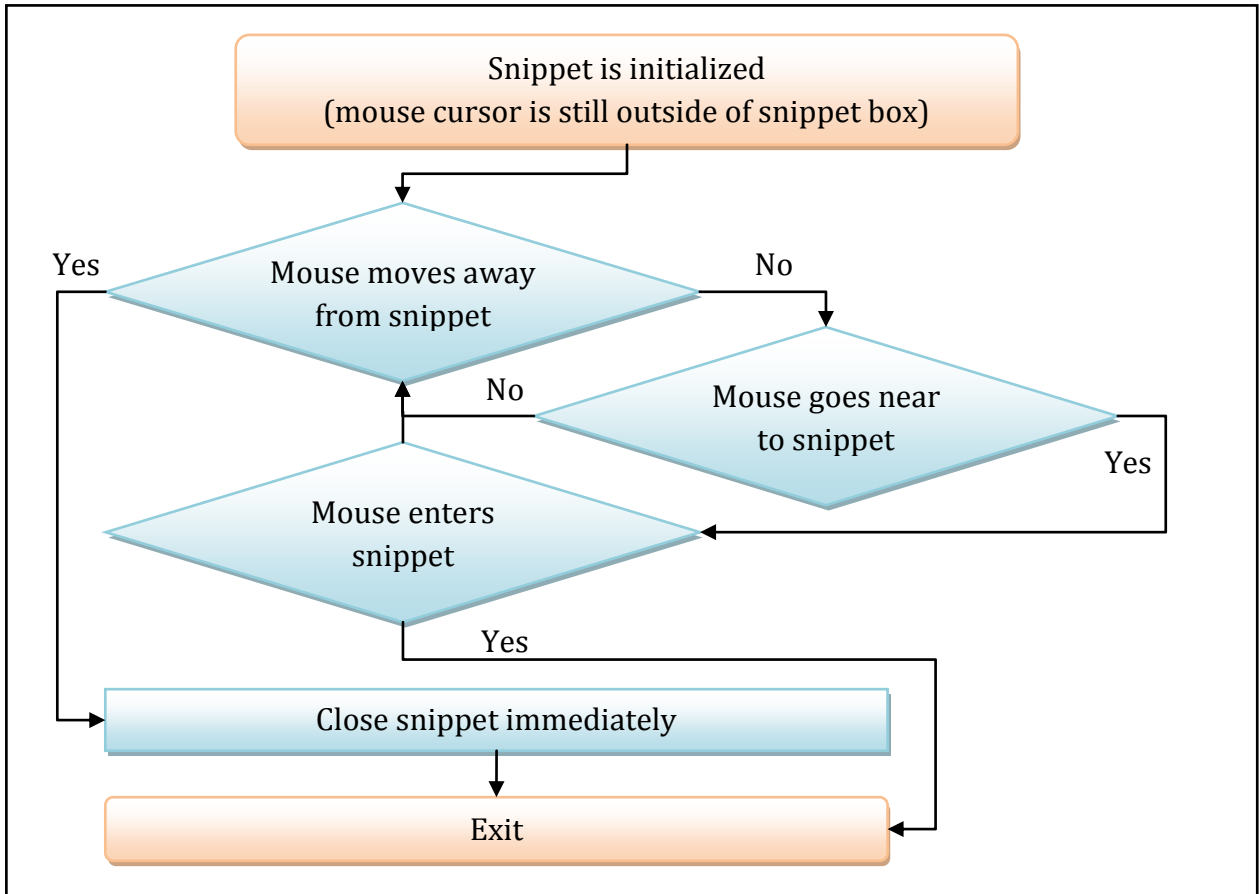


Figure 48: Flowchart of Easy Reversal of Snippet Initialization

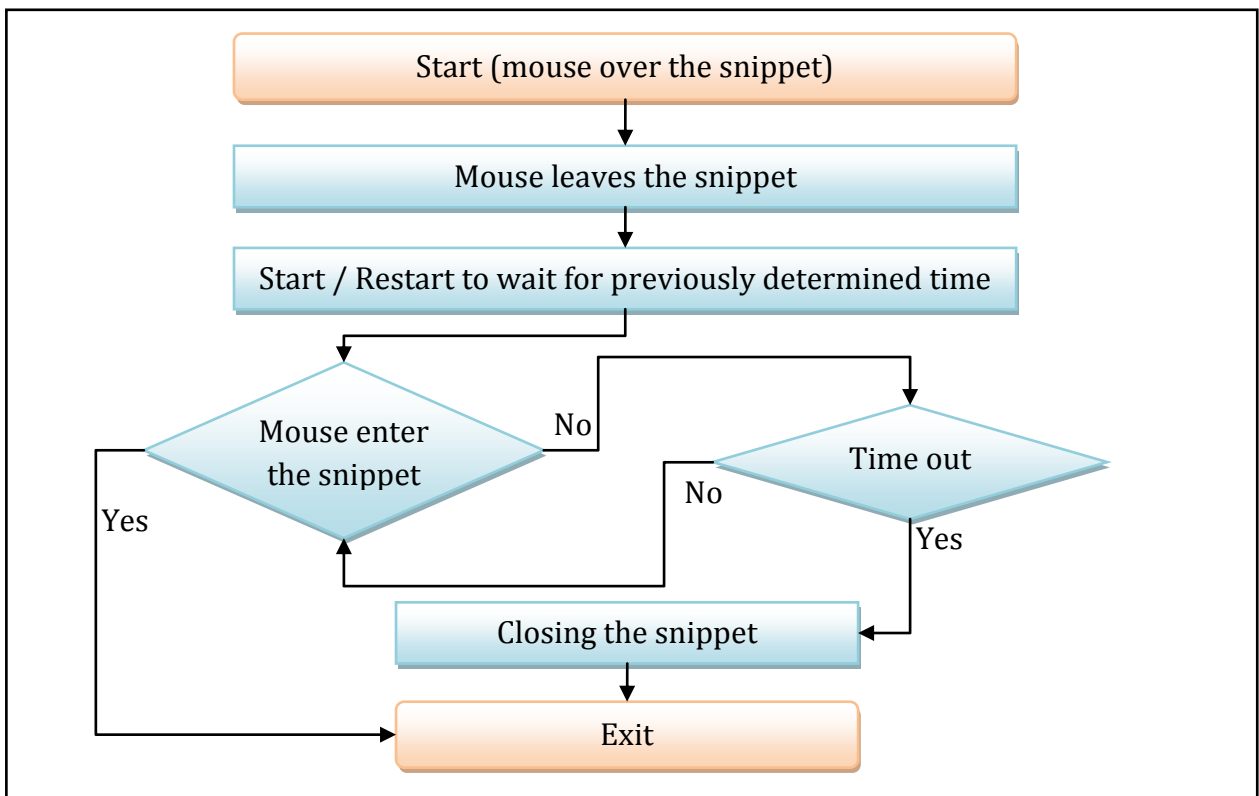


Figure 49: Flowchart of Auto-Closing the Snippet

Hold button (auto-close toggle button) and roll button (auto-collapsible toggle button) also have been integrated in the snippet. If hold or roll mode is active, auto-closing will be deactivated. By activating hold mode, a user can compare multiple snippets side by side. If the roll mode is active (auto-collapsible is enabled), the snippet will be automatically collapsed or rolled when the mouse cursor moves out from the snippet. It will be automatically expanded and moved up over any other opened snippets when the mouse cursor enters back the snippet. Roll mode is also a very advantageous feature. It can be used if a user wants to let the snippet minimized while they want to continue with reading. For example, if a user wants to read the information in some snippets later after they have finished with reading some paragraphs. They can arrange the snippets as a stack of rolled frames so that they do not forget to read them later, and on the other hand they have some space to continue with reading.

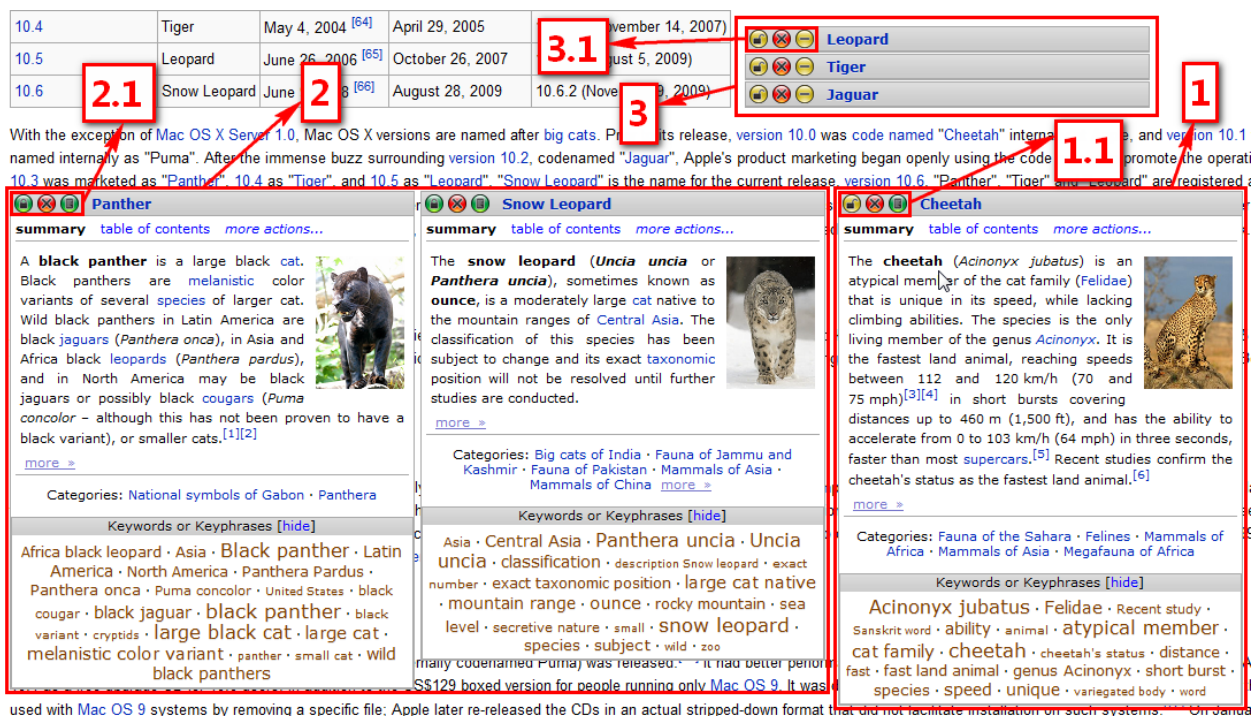


Figure 50: Multiple Snippets are Open

In the Figure 50, three snippets are expanded. One of them [1] is in the auto-close mode, and the two others [2] are in the hold mode. Three other snippets are collapsed and manually arranged as a stack. They [3] are in the auto-collapsible mode. To make the user interface simple, there are overall only three states of snippet control buttons instead of five: auto-close mode is identified by color yellow-red-green [1.1]; hold mode is identified by color green-red-green [2.1]; and auto-collapse mode is identified by color yellow-red-yellow [3.1].

Manual closing by clicking close button always can be done including while the hold mode or roll mode is active. After a snippet is closed, the auto-close mode will be reactivated.

Based on the investigation of mouse movement in the behavioral dimension of browsing, the snippet control buttons are placed on the left side instead of the right side. It is done in order to accelerate the activities, if a user wants to immediately activate hold or roll after snippet is initialized. After initialization, the mouse cursor is much nearer to the left side in compare to the right side.

### 4.5.3. Other Usability Features

Two other usability features which have been implemented are *auto-zoom* and *recursive sniffing*.

#### *Auto-zoom*

The snippet has a limited height in the initialization. If the information in the content frame exceeds this limit, a scrollbar will appear. The goal is to keep the snippet compact and not too huge. However, if a user interest to the main content is detected, the snippet will be automatically enlarged until the next height limit. The user interest is detected after they hover the content for a while. It is useful if a user wants to see images in the image list or during scanning related articles. The snippet size will be restored down whenever the user hovers the dragger with some delay. This implementation is similar with the patent which was introduced in section 3.1.2 about auto-zoomable snippet. The difference is that the zooming only adjusts the size of the snippet which is focused but not the other snippets around.

#### *Recursive sniffing*

The snippet can also contain some internal wiki links. It can be recursively sniffed by hovering them like a normal link in the main article. If a child snippet is open, the parent snippet will be currently locked until the child node has been closed or any control button in the parent snippet is clicked.

### 4.5.4. Look and Feel

Aesthetic is an important point in the design guidelines. Some approaches such as 3-D impression (by using beautiful icon, dragger background image, and embossed text), transparency during dragging, fading effect during initializing and closing, sliding animation during scrolling as well as resizing effect during auto-zooming have been implemented in the design. Unnecessary borders are also avoided. The color, contrast, padding and component size and position are also always in our consideration. The goal is to get best simplicity and clarity.

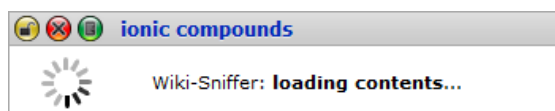


Figure 51: Rapid Response (Feedback)

Fast rotating busy icon is used to give a user some feedback that the snippet is working and is still processing some data.

## 5. Evaluation

### 5.1. User Study 2 – Bottom-line Data

#### 5.1.1. Executive Summary and Plot

After the user interface element had been implemented, the next thing to do was the evaluation. For the evaluation, a next user study for obtaining bottom-line data was performed. In contrast to the first user study, this study consists of formal and quantitative measurements of what happens. By performing this user study, our goal was to measure the improvements of using the new user interface element. To achieve this goal, the task's finishing times and the precision were measured. The performance of using the new tool and the performance of the baseline (without the tool) were also compared. It is worth for noting that a LexRank summarizer and the backlink feature are temporary disabled in this study because there was a time performance issue.

Firstly, the Wiki-Sniffer project was introduced to each participant. The goal of this study was also explained. Afterwards, we gave each participant a chance to try the new tools freely by browsing some Wikipedia articles. By using this chance, the participant can adapt to the new enhanced user interface.

After some orientation, the tasks were introduced to the participant step by step. While they were finishing the tasks, the times were measured. How they use the tool was also observed. After finishing all of the tasks, the participant filled a questionnaire to give some opinions and feedback about the new tool. The test took about 30 minutes for each participant.

Every person has own characteristics while browsing, such as speed, precision, and their previous knowledge. Each task also has own difficulty related to the person. As their results, some tasks can be done faster for some of them but maybe with worse precision of the correct answers. It is also a possibility that a person is really skillful to use computer programs and navigates faster than others. Hence, we performed some cross tests to each task. The goal is to get as much as possible an equal distribution of those characteristics. It means that the each participant should not do all of the tasks by using the same method, e.g. only by using the tool, but rather to let them do the tasks by using several methods. However, to avoid previous knowledge, each task is only done once by each participant, i.e. either by using the tool or without the tool (10). At the end, the number of tests took by using the tool and the number of tests took without the tool should be equal.

Furthermore, we also used the benefit of articles with some name ambiguities and the articles with foreign names. By using this method, the participant cannot decide directly for the answer by obtaining the context from the article names but rather to browse the

article for some overview. Thereby, the pure performance of using the new tool and the performance without using the new tool can be more effectively measured.

### 5.1.2. Task

For the tasks, we followed the guidelines that they should be realistic (9). There are four kinds of tasks which were given to each participant:

- **Task A: Determining the relatedness between articles**  
For each of the first two tasks, we gave an article to compare and a list of 15 Wikipedia articles. The participant should decide for each article on the list whether the article is related to the article, which was given to compare. The criteria, how two articles are related each other, had been explained before the participant began with these two tasks. Each participant did one of these tasks without using the new tool and the other task by using the new tool. To perform these two tasks, 32 Wikipedia articles were used.
- **Task B: “True or false” question answering**  
For each of the next two tasks, we gave a list containing four Wikipedia articles. For each of these articles, the participant should answer a “true or false” question. These tasks used eight articles from Wikipedia.
- **Task C: Finding overlapping categories**  
In this task, an article to compare and a list of 15 Wikipedia articles were given to the participant. The participant decided for each article in the list, whether it had at least an overlapping Wikipedia category with the categories of the article which had been given previously. This task used 16 Wikipedia articles.
- **Task D: Finding a matched category**  
In the last task, we gave a list of 14 Wikipedia articles and 2 defined Wikipedia categories to the participant. The participant decided for each article, which category the article belonged to.

Overall, 70 articles were used for performing the tasks. Twelve articles were featured articles.

### 5.1.3. Participants

There were 10 participants in this user study. They were between 20 to 31 years old from a high school student to an industrial employee. Most of them were from computer science faculty. The others are from electrical engineering, mechanical engineering, mechatronics, civil engineering, and hotel and restaurant management faculty.

### 5.1.4. Results

The results are evaluated for each type of the tasks. *Standard Mean* values are used to compare between the time performance of using Wiki-Sniffer and without it. However, to obtain a better confidence, *standard deviation* is also used for calculating the *standard*



error of the mean. Mean  $\bar{x}$  of sample  $x_1$  to  $x_n$ , standard deviation  $\delta$ , and standard error of the mean  $SE$  are defined as follow:

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^n x_i, \quad \sigma = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n (x_i - \bar{x})^2}, \quad SE_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

Equation 11

By using the standard error of the mean, we can say more precisely what we mean by plausible, i.e. by calculating the mean with 95% of confidence. The mean will be  $\bar{x}$  plus or minus  $SE_{\bar{x}}$  (9).

		Baseline (seconds)	Wiki-Sniffer (seconds)
<b>Task A</b>	Mean	14.240000	<b>11.653333</b>
	Standard Deviation	7.240527	<b>6.419336</b>
	Standard Error (150 samples)	0.591187	0.524137
	Mean with 95% confidence	13.648813 ~ 14.831187	<b>11.129197 ~ 12.177470</b>
	Precision	<b>0.986667</b>	0.960000
<b>Task B</b>	Mean	18.875000	<b>14.525000</b>
	Standard Deviation	14.742556	<b>11.383721</b>
	Standard Error (40 samples)	2.331003	1.799924
	Mean with 95% confidence	16.543997 ~ 21.206002	<b>12.725076 ~ 16.324924</b>
	Precision	<b>0.975000</b>	<b>0.975000</b>
<b>Task C</b>	Mean	17.040000	<b>15.786667</b>
	Standard Deviation	11.242343	<b>9.374049</b>
	Standard Error (75 samples)	1.298154	1.082422
	Mean with 95% confidence	15.741846 ~ 18.338154	14.704245 ~ 16.869089
	Precision	0.920000	<b>0.933333</b>
<b>Task D</b>	Mean	12.757143	<b>10.857143</b>
	Standard Deviation	<b>4.291669</b>	9.063841
	Standard Error (70 samples)	0.512953	1.083336
	Mean with 95% confidence	12.244190 ~ 13.270095	<b>9.773807 ~ 11.940479</b>
	Precision	<b>1.000000</b>	0.957143

Table 5: Results of User Study 2

From the standard mean values, it can be seen that by using the new interface element Wiki-Sniffer, the performance of task's finishing times are enhanced (task A: 2.6 seconds, task B: 4.4 seconds, task C: 1.23, and task D: 1.9 seconds better). By using Mean values with 95% of confidence, we also still have better performance for the task A, B, and D (1.5, 0.2, and 0.3 seconds better).

The standard deviation values of the tests by using the tool also tend to be lower. It shows that the finishing time's values by using the tool are also nearer to the mean if we compare with the finishing time's values of the baseline. In the task D the standard deviation value by using Wiki-Sniffer is higher because some participants did not recognize very quickly to find the categories which are hidden in the snippet. As

mentioned previously, only five categories are shown in the snippet at the initialization. The others can be shown by clicking “more” button.

Task A has a worse precision for Wiki-Sniffer than the baseline, because some participants tended to look the information only on the summary. They decided for the wrong answer if they did not find any clues in the summary. Some other participants who also used keyphrases or table of contents feature tended not to do these particular mistakes. Task D also has a worse precision because there was a participant who did not look to the hidden categories in the snippet and got some wrong answers.

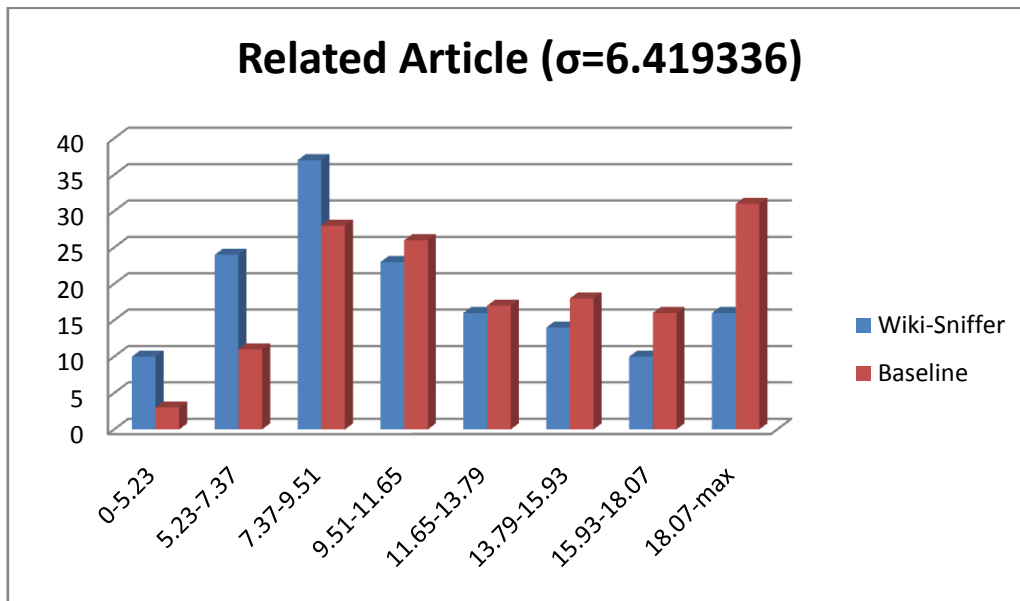


Figure 52: Related Article

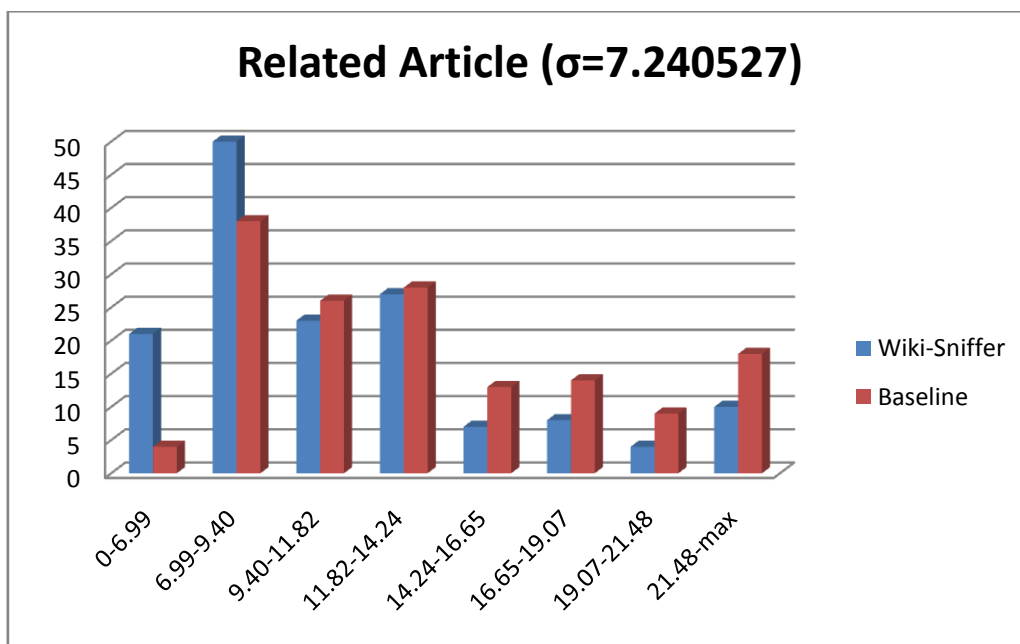


Figure 53: Related Article

From the graph, it can be seen that by using Wiki-Sniffer, 63% of the Task A were finished within 11.65 seconds, whereas in the baseline only 45%.

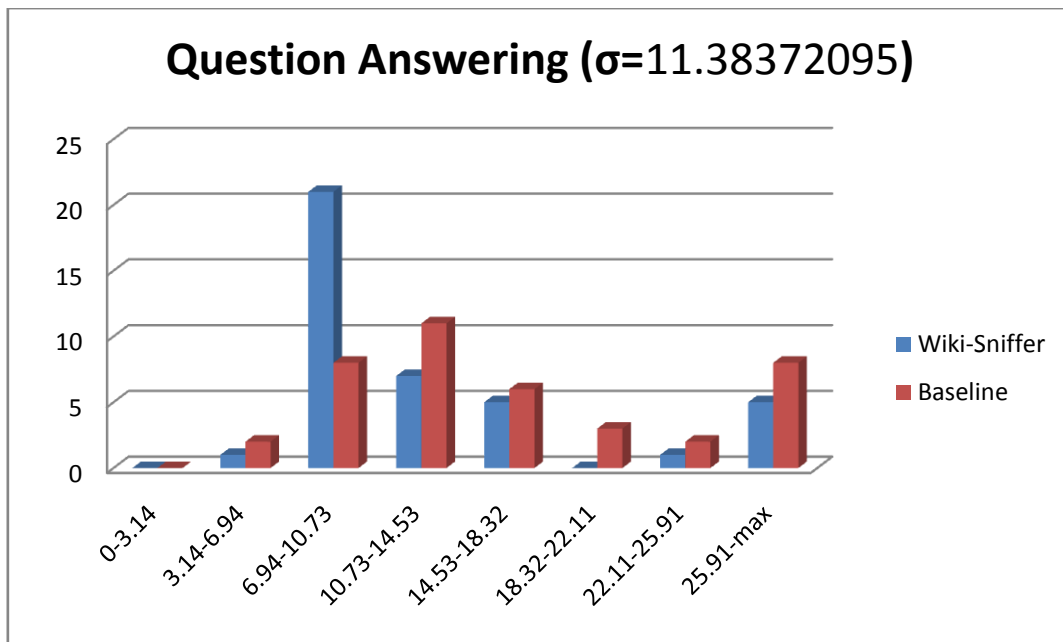


Figure 54: Question Answering

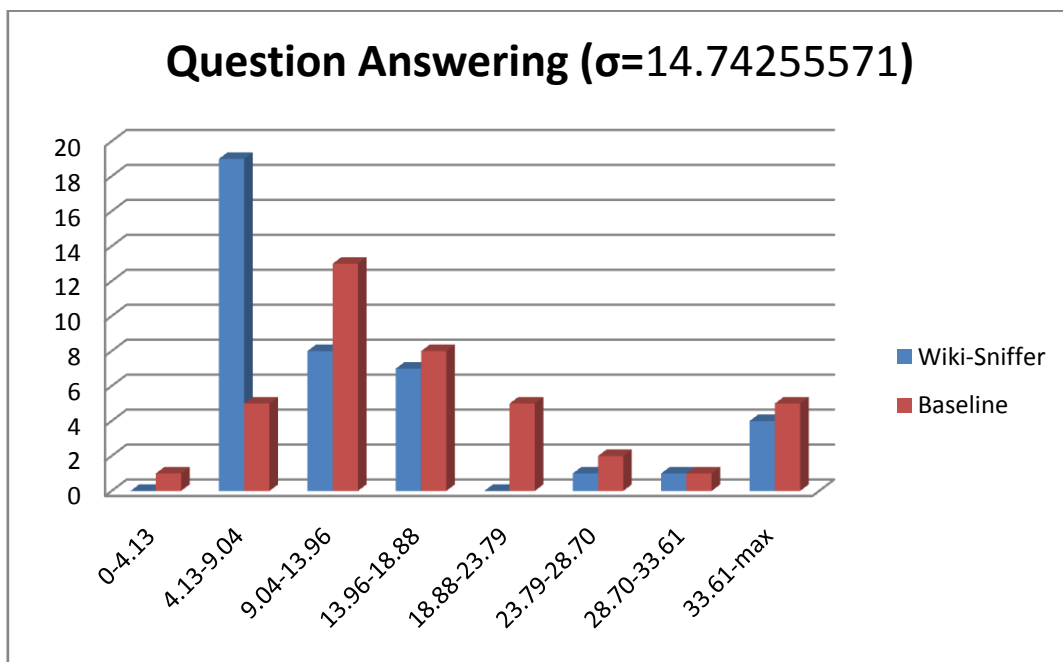


Figure 55: Question Answering

In the task B, 70% of the subtasks were finished within 10.73 by using the snippet, whereas only 25% were finished without the tool.

Another interesting phenomenon also occurred in Task A and B. Two of our participants used often the thumbnail in the summaries to quickly find out the answer without

reading the text. This phenomenon also shows that visual information like pictures also plays a very important role in the browsing process.

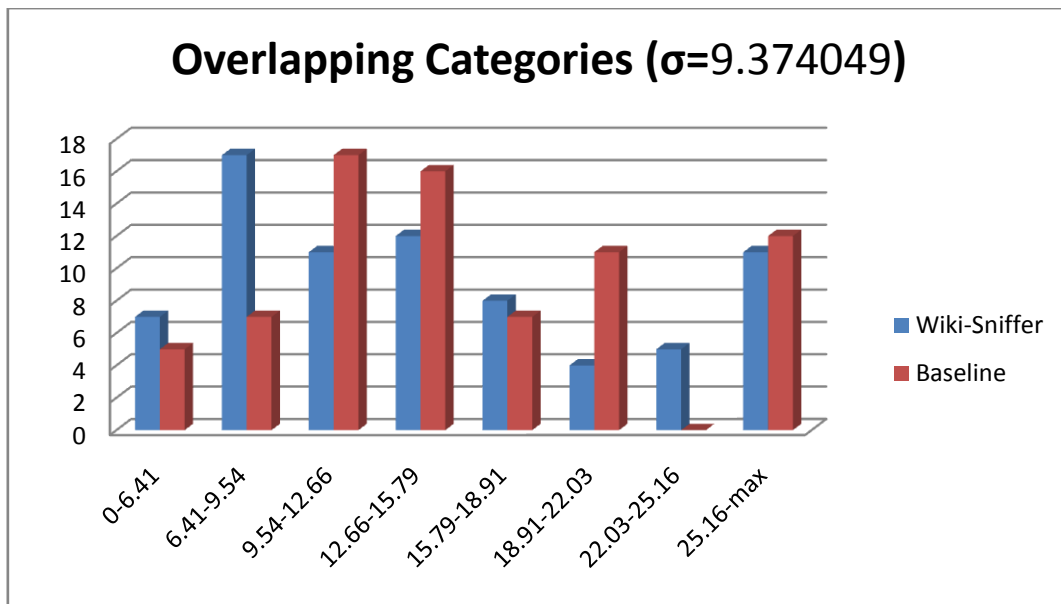


Figure 56: Overlapping Categories

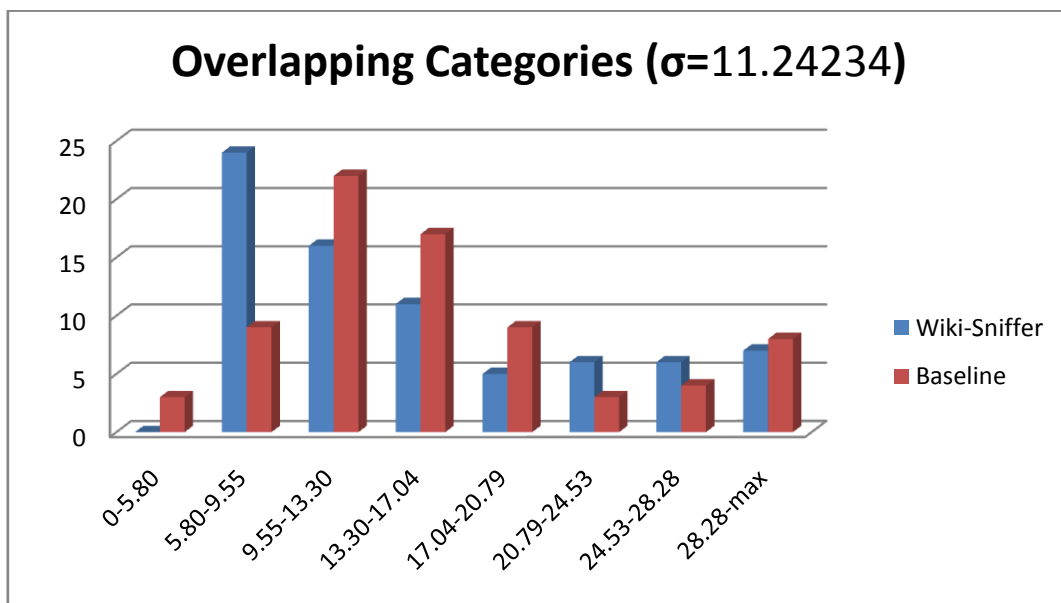


Figure 57: Overlapping Categories

Task C has the highest difficulty level. Most of the subtasks (about 60%) with or without the tool were finished between 5.8 to 20.8 seconds, but within 9.5 seconds, 32% of subtasks were finished with the tool, whereas only 16% of them were finished without the tool.

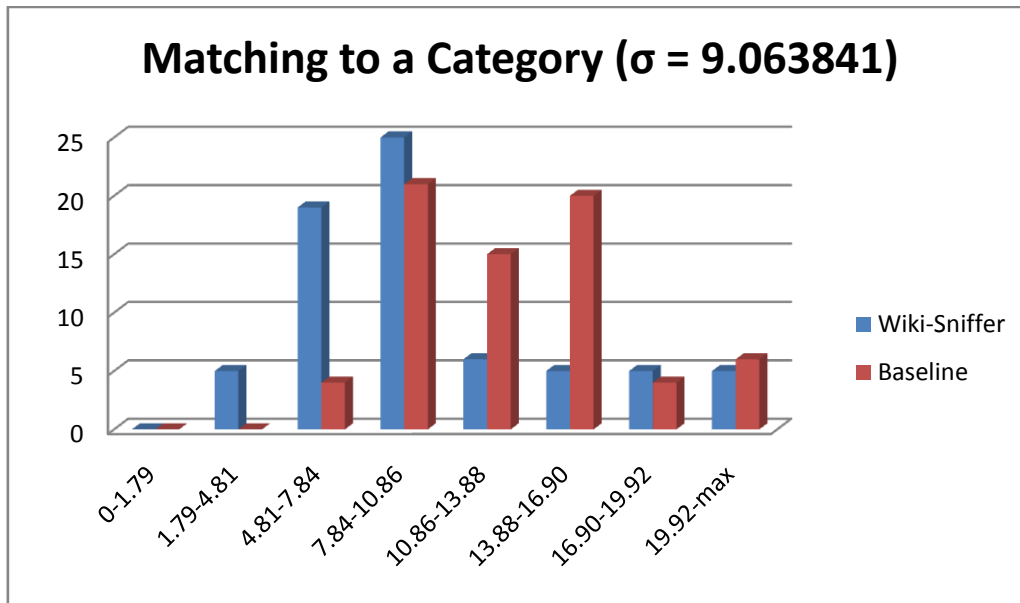


Figure 58: Matching to a Category

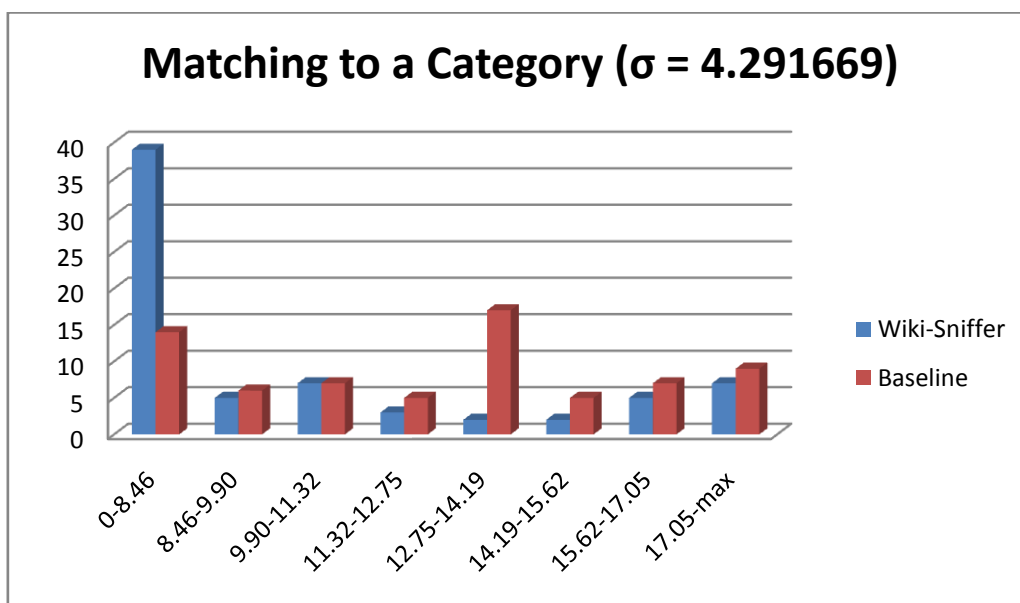


Figure 59: Matching to a Category

For the last task, 56% of the subtasks were finished within 8.5 seconds by using the tool, whereas only 20% of them were finished without the tool.

### 5.1.5. Feedback and Problems Found

Generally, we got much positive feedback about the tool. Some participants were even fascinated about what has been done in this project. We got only some negative feedback in the beginning of this study from the first two participants. There were problems with unclear “more” button and the unclear boundary between the main content frame and the category frame. The other problem was about some known bug

issues. However, those problems were fixed immediately during the study. The problem did not occur anymore in the next tests with other participants.

Here are some components and aspects that got positive feedback from the participants:

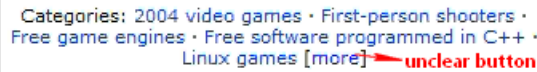
- The size of the snippet got some positive feedback from the participants.
- Some participants were fascinated with the hold button feature and also the possibility to open multiple snippets simultaneously. A participant also gave some positive feedback about the design of the control buttons by using red-yellow-green colors. Some participants were excited that the snippet is freely movable.
- Again, the summary part was the favorite sort of the information.
- The keyphrases also got some positive feedback. A participant commented that the feature was very nice because from that feature, a user can also get some other useful information from something which is not directly provided by the wiki.
- Categories which did not get much positive feedback were this time different. This sort also got some positive feedback from the participant, and some of them commented that this sort was also really useful
- The link behavior of the table of contents also got some positive feedback. A participant suggested having a possibility to sniff a link in a table of content to get an overview of a subsection of an article.
- Image also got some positive feedback, and as mentioned previously, it also plays an important role in the browsing process.
- A participant was really fascinated that the snippet also provided related articles feature. This shows that a black box encapsulation also plays an important role for the success of a user interface as previously mentioned by Marti Hearst (10).

All participants found that the new tool was useful, and they preferred to browse with the new tool. A participant even commented that we should not easily give up with the project even if the evaluation produced worse results of task finishing times by using the new tool, since on the other hand it made browsing much more comfortable.

### 5.1.6. Suggested Improvements

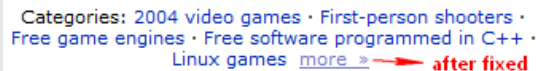
#### *Must do*

The design of “more” button was unclear. However, this problem was fixed immediately during the study.



Categories: [2004 video games](#) · [First-person shooters](#) · [Free game engines](#) · [Free software programmed in C++](#) · [Linux games](#) [more] → **unclear button**

Figure 60: “more” button – before fixed



Categories: [2004 video games](#) · [First-person shooters](#) · [Free game engines](#) · [Free software programmed in C++](#) · [Linux games](#) [more »](#) → **after fixed**

Figure 61: “more” button – after fixed

#### *Should do*

The boundary between main content and category frame was also unclear but this problem also has been fixed during the user study.

#### *Could do*

A participant suggested a feature to adjust the font size in the snippet. This feature is very useful so that the snippet can be used more comfortably for people in several ages.

Another improvement has been mentioned previously. It is also nice to have a feature to sniff the links in the table of contents to get an overview about the subsection of an article.



## 6. Conclusion

By using the opportunity of dense link structure in wiki articles, we have implemented a new user interface element to improve the seeking process. Two wiki environments, i.e. MediaWiki and TWiki, were firstly investigated to know which sorts of information can be presented to users. To present this information, a user interface is needed. A user interface presents the information in a structure. To design the user interface, we firstly learnt how people interact with the world. Since we want to enhance the navigation in the existing interface, we learnt to understand about the browsing process. Some patents and related works have been investigated to get some ideas for the design and to look what have been done so far. We followed the guidelines of user interface design to create a user interface element with high usability and utility.

To reduce risks of unusable interface, several mockups had been created and compared in the first user study before the real system was implemented. The first user study helped us to get the process data so that we had an overall feeling about what works and does not. The goal was to get feedback as fast as possible.

Among the several sorts of information which can be provided to users, some were extracted directly from the wiki page, and some were constructed by using NLP algorithms. We used NLP algorithms to construct keyphrases, i.e. by using TextRank algorithm, and to construct summary, i.e. by LexRank algorithm. Further preprocessing by filtering the sentences with anaphora are needed to make the text in the summary more understandable before the source text is summarized by using LexRank algorithm. This can be done in the next works to improve the performance of LexRank.

Speed performance to create a page overview snippet also plays an important role for the usability of the user interface element. If the process to generate the snippet takes too long, the snippet will become unusable even if the snippet itself has structured of how the information is presented.

From the evaluation in the second user study, we got much positive feedback. Although the new tool does not improve so much time performance according to the results of the study, the participants found that the new tools is useful and prefer to browse with the tool because of the comfort and the functionalities provided by the tool. Here, we see that the usability does not only depend on the task finishing time but also on the comfort and the utility. An encapsulation of an extracted information sort as a constructed information sort by labeling it to a term that is easier to understand also play an important role for the success of a user interface.

## Acknowledgment

First and foremost, I would like to thank God, for all of your countless blessings. You have always given me the power to work in passion to pursue my dreams. You are the beginning and the end; without you nothing is possible to happen.

I also want to thank to Germany and Technische Universität Darmstadt for giving me a beautiful place to study and live. Here I have been learning many things about life, cultures, and also about how to see the world from other perspectives and several point of views.

A sincere gratitude I want to express to my supervisor, Professor Iryna Gurevych for giving me the opportunity to work on such a wonderful project. It has been a long time since I was in high school when I was thinking that someday I would do something in natural language processing areas.

I wish to express my sincere thanks to my coordinator, Johannes Hoffart. Thank you for the encouragement, guidance, and patient you have given during my thesis. I am so thankful to have the opportunity to work with you. By working together with you, I have learnt many things not only for the hard skills but also for the soft skills.

I also want to thank to my coordinator, Joachim Caspar, who helped me to find some sources and for giving me some constructing feedback about my presentation.

Mostly, I want to deeply and warmly thank to my beloved family, to my father Surja Kurniawan Tamin, my mother Sara Farida Tamin, and my younger brother Odoric Pantera Tamin. Thank you for loving me so deeply and dearly. Thank you for always supporting me during my study. Although you are thousand miles away from here, you are always in my heart.

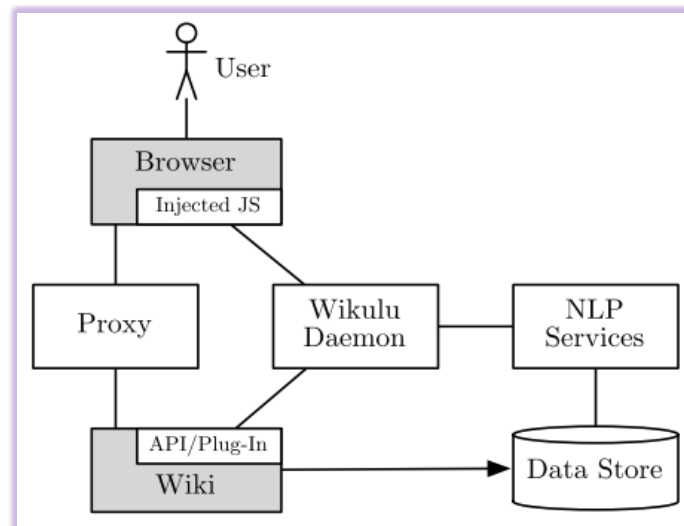
Another most deeply and sincerely thank, I want to express to my lovely and my precious one, Nicole. Thank you for always be there during my thesis and for always supporting me to finish this thesis. I can still remember that you were also there when I decided to take part in this project as my diploma thesis. For this moment we just physically thousand miles away apart but in another dimension of space we are close together.

Last but not least, I want also to thank all of my friends and all people who have supported me to finish this thesis. Thank you for your encouragement during my thesis and also thank you for your support in my user study.

# Appendixes

## Appendix A: Wiki-Sniffer Architecture

The communication between several components in our project can be seen in Figure 62.



**Figure 62: Communication Architecture**

In this scenario, proxy is configured in the user's internet browser, e.g. Firefox, Opera, Safari, Internet Explorer, and Google Chrome. The proxy is a Java servlet that intercepts the interaction between the client's browser and the wiki server. It adds additional JavaScript and CSS to the original HTML page rendered by wiki. Java servlet is a Java objects that dynamically process requests and construct responses. Wikulu daemon is also a Java servlet. It manages instances and also delegates calls to NLP services. NLP services offer the functionality of NLP algorithm as web services. The data store contains all data structures which are needed for the NLP algorithms, e.g. a search index for information retrieval or pages pre-processed by a part-of-speech tagger (62).

## Appendix B: User Study 1 Documents

### Introduction

---

# Creating Wiki Page Overview Snippets

---

## Introduction

First of all we want to thank you for your willingness to participate in our user study. In this study we want to understand about people's perceptions of usability and user friendliness of web sites in early stages of design.

To achieve this goal we have prepared some design models on the paper and some survey questions. We will call the models prototypes.

We would need you to compare these prototypes and later you could choose one of them as the best one. The goal is to find the best design to implement in later stages. We ask you to help us improve the prototypes. We want to stress that we are testing the prototypes and not you. So, if you have any troubles or problems with some tasks we ask you to perform, it is the prototype's fault and not yours.

Here is what we have planned for the next.

Firstly we will give a little overview about the project, what it is all about, what should be done in this project.

Then I will introduce four prototypes that we have. I would need you to find a prototype that you think as the best prototype or the one, which is most comfortable for you.

Next we would ask you some survey questions about your perceptions of the models.

Please remember that this test is totally voluntary. If you become uncomfortable or find this evaluation objectionable in any way, feel free to quit at any time.

## Project Overview

In the Wikulu project, we work on improving the usability of Wikis using natural language processing (NLP) methods. The goal is to find good ways of presenting the results of these methods to the user in a way that is meaningful and easy to understand.

One possible option is to improve the way how users interact with Wiki's links. Normally users have to clicks the link to get the information from that link. In the new scenario we plan to provide users with a compact and small pop-up window we call snippet. With this snippet users are able to get some parts of information without leaving the originating page.

In this study we like to know which information is important for users and how should we present it to them.

# Creating Wiki Page Overview Snippets

## User Study 1: Prototype Design

### Participant's Data

Gender :  male  female

Job :

How often are you using Wikipedia?

Daily  Weekly  Monthly  Less then monthly use

Which of our prototype design would be the best choice for you?

AT  HE  SA  SD

What do you like in particular in this prototype design?

For each of these aspects, please tell us the following:

<b>Simplicity</b>	How important is this aspect for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How would you rate the prototype for this aspect?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)
	Is there any prototype, which is better for this aspect?	<input type="checkbox"/> Yes <input type="checkbox"/> No <input type="checkbox"/> (no comment)
	Which prototype (if any)?	<input type="text"/>

<b>Look and feel</b>	How important is this aspect for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How would you rate the prototype for this aspect?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)
	Is there any prototype, which is better for this aspect?	<input type="checkbox"/> Yes <input type="checkbox"/> No <input type="checkbox"/> (no comment)
	Which prototype (if any)?	<input type="text"/>

<b>Clarity</b>	How important is this aspect for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How would you rate the prototype for this aspect?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)
	Is there any prototype, which is better for this aspect?	<input type="checkbox"/> Yes <input type="checkbox"/> No <input type="checkbox"/> (no comment)
	Which prototype (if any)?	<input type="text"/>

<b>Comfort</b>	How important is this aspect for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How would you rate the prototype for this aspect?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)
	Is there any prototype, which is better for this aspect?	<input type="checkbox"/> Yes <input type="checkbox"/> No <input type="checkbox"/> (no comment)
	Which prototype (if any)?	<input type="text"/>

<b>Features</b>	How important is this aspect for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How would you rate the prototype for this aspect?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)
	Is there any prototype, which is better for this aspect?	<input type="checkbox"/> Yes <input type="checkbox"/> No <input type="checkbox"/> (no comment)
	Which prototype (if any)?	<input type="text"/>



For each of the following features, please tell us the following:

<b>Summary</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

<b>Table of contents</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

<b>Keywords</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

<b>Images</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

<b>Related articles</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

<b>External links</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

<b>Tags</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

<b>Sniffed articles</b>	How important is this feature for you?	<input type="checkbox"/> Very important <input type="checkbox"/> Important <input type="checkbox"/> Okay <input type="checkbox"/> Less important <input type="checkbox"/> Not important <input type="checkbox"/> (no comment)
	How often would you use this feature?	<input type="checkbox"/> Always <input type="checkbox"/> Very often <input type="checkbox"/> Often <input type="checkbox"/> Sometimes <input type="checkbox"/> Rarely <input type="checkbox"/> Never <input type="checkbox"/> (no comment)
	How well does this feature work with the prototype?	<input type="checkbox"/> Very good <input type="checkbox"/> Good <input type="checkbox"/> Okay <input type="checkbox"/> Poor <input type="checkbox"/> Very poor <input type="checkbox"/> (no comment)

Which component of the prototype is distracting or confusing?

------------------

What modification(s) (if any) should we make to improve our prototype design?

------------------

How do you like to open or start the snippet?

- Hovering the link  
 Clicking the icon which appears when the link is hovered.

How do you think about the snippet size?

- Too big    Okay    Too small    (no comment)

Did you notice about the window control buttons before getting this question?

- Yes    No

For each of the following controls, please tell us the following

**Hold button (keeping snippet open)**

- Very important    Important    Neutral    Less important    Not important  
 (no comment)

**Close button (close the snippet)**

- Very important    Important    Neutral    Less important    Not important  
 (no comment)

**Roll button (switch to mini or normal snippet)**

- Very important    Important    Neutral    Less important    Not important  
 (no comment)


## 1st candidate: Prototype SA

Beagle ? roll hold close

**Beagle**

Summary Table of Contents Keywords Related Articles  
External Links Tags Sniffed Articles

+ The **Beagle** is a **breed** of small to medium-sized **dog**. A member of the **Hound Group**, it is similar in appearance to the **Foxhound** but smaller, with shorter legs and longer, softer ears. Beagles are **scent hounds**, developed primarily for **tracking hare**, **rabbit**, and other **game**. They have a keen sense of smell and tracking instinct that sees them employed as **detection dogs** for prohibited agricultural imports and foodstuffs in **quarantine** around the world.



Beagle ? roll hold close

**Beagle**

Summary Table of Contents Keywords Related Articles  
External Links Tags Sniffed Articles

beagle, breed, brown, crossbreed, detection dog, dog, foxhound, hip dysplasia, hound, hunt, miniature, open throat, scent hound, sense of smell, sing, small, snoopy, rabbit, talbot, United Kingdom

Beagle ? roll hold close

**Beagle**

Summary Table of Contents Keywords Related Articles  
External Links Tags Sniffed Articles

All Used by this article Using this article

- abdomen
- breed
- Canadian Kennel Club
- Celtic
- collies
- crufts
- dachshund
- dewlap
- dog
- detection dog
- Foxhound
- French
- Hound Group
- Illinois
- Ireland
- K-Run's ParkMe In First
- Old English
- Oxford English Dictionary
- quarantine
- rabbit
- stifles
- tracking hare
- Westminster Kennel Club

Beagle ? roll hold close

**Beagle**

Summary Table of Contents Keywords Related Articles  
External Links Tags Sniffed Articles

- Dog breeds
- Dog breeds originating in the United Kingdom
- Hounds
- Scent hounds

## 2nd candidate: Prototype SD


Beagle ? roll hold close

**Beagle**

Summary Table of Contents Keywords

m  
o  
r  
e

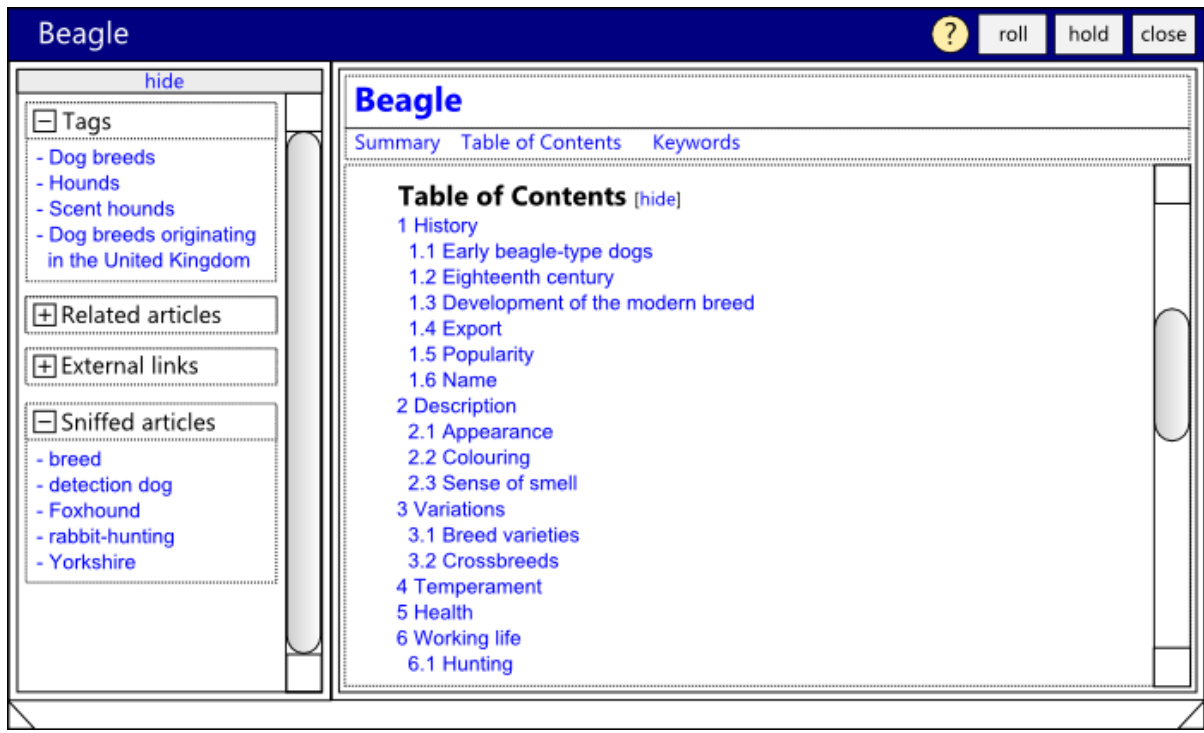
The **Beagle** is a **breed** of small to medium-sized **dog**. A member of the **Hound Group**, it is similar in appearance to the **Foxhound** but smaller, with shorter legs and longer, softer ears. Beagles are **scent hounds**, developed primarily for **tracking hare**, **rabbit**, and other **game**. They have a keen sense of smell and tracking instinct that sees them employed as **detection dogs** for prohibited agricultural imports and foodstuffs in **quarantine** around the world.



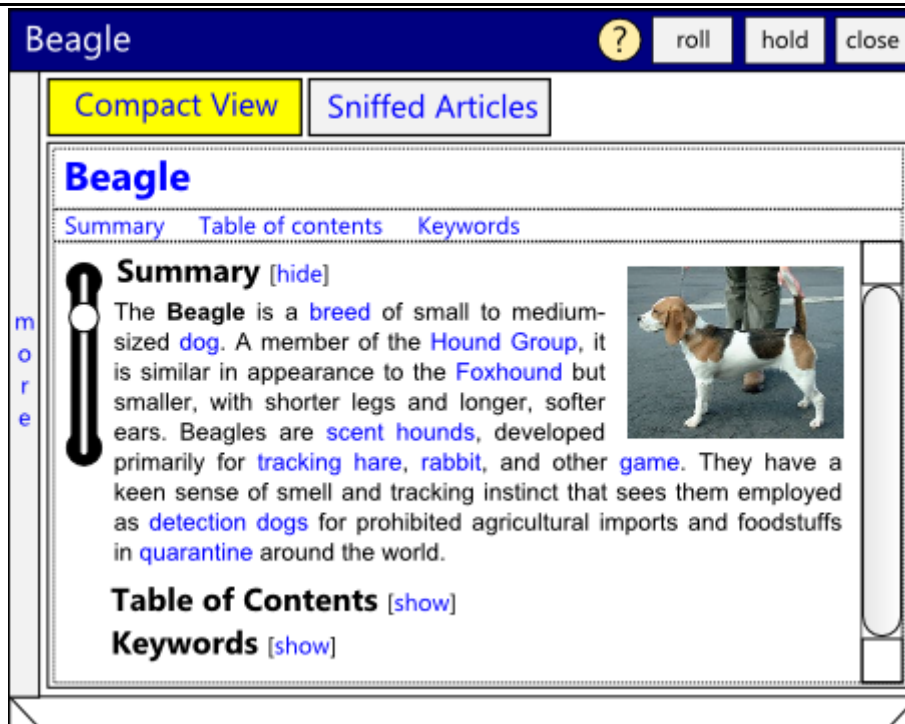
**Table of Contents** [show]

**Keywords** [hide]

beagle, breed, brown, crossbreed, detection dog, dog, foxhound, hip dysplasia, hound, hunt, miniature, open throat, scent hound, sense of smell, sing, small, snoopy, rabbit, talbot, United Kingdom



3<sup>rd</sup> candidate: Prototype AT





Beagle

?
roll
hold
close

hide

Compact View

Sniffed Articles

- Tags
 

- Dog breeds
- Hounds
- Scent hounds
- Dog breeds originating in the United Kingdom

+ Used articles

+ Using articles

+ External links

## Beagle


Summary
Table of contents
Keywords

**Summary** [hide]

The **Beagle** is a **breed** of small to medium-sized **dog**. A member of the **Hound Group**, it is similar in appearance to the **Foxhound** but smaller, with shorter legs and longer, softer ears. Beagles are **scent hounds**, developed primarily for **tracking hare**, **rabbit**, and other **game**. They have a keen sense of smell and tracking instinct that sees them employed as **detection dogs** for prohibited agricultural imports and foodstuffs in **quarantine** around the world.

**Table of Contents** [show]

**Keywords** [show]



Snoopy

?
roll
hold
close

hide

Compact View

Sniffed Articles

Sort: Ascending ▼

+ **barking**


- **Beagle**

**Summary** [hide]

The **Beagle** is a **breed** of small to medium-sized **dog**. A member of the **Hound Group**, it is similar in appearance to the **Foxhound** but smaller, with shorter legs and longer, softer ears. Beagles are **scent hounds**, developed primarily for **tracking hare**, **rabbit**, and other **game**. They have a keen sense of smell and tracking instinct that sees them employed as **detection dogs** for prohibited agricultural imports and foodstuffs in **quarantine** around the world.

**Table of Contents** [show]

**Keywords** [show]




- **Snoopy**

**Summary** [hide]

Snoopy is a **fictional character** in the long-running comic strip *Peanuts*, by **Charles M. Schulz**. He is **Charlie Brown's** pet **beagle**. Snoopy began his life in the strip as a fairly ordinary dog, but eventually evolved into perhaps the strip's most dynamic character — and among the most recognizable comic characters in the world. The original drawings of Snoopy were based on Schulz's childhood dogs, Snooky and Spike.

**Table of Contents** [show]

**Keywords** [show]



- **Talbot Hound**

## 4<sup>th</sup> candidate: Prototype HE

Beagle
roll hold close

---

### Beagle

**Summary** [Table of Contents](#)

+ The **Beagle** is a [breed](#) of small to medium-sized [dog](#). A member of the [Hound Group](#), it is similar in appearance to the [Foxhound](#) but smaller, with shorter legs and longer, softer ears. Beagles are [scent hounds](#), developed primarily for [tracking hare](#), [rabbit](#), and other [game](#). They have a keen sense of smell and tracking instinct that sees them employed as [detection dogs](#) for prohibited agricultural imports and foodstuffs in [quarantine](#) around the world.



[Keywords](#)
[Images](#)
[Related Articles](#)
[External Links](#)
[Tags](#)
[Sniffed Articles](#)


Beagle
roll hold close

---

### Beagle

**Summary** [Table of Contents](#)

+ The **Beagle** is a [breed](#) of small to medium-sized [dog](#). A member of the [Hound Group](#), it is similar in appearance to the [Foxhound](#) but smaller, with shorter legs and longer, softer ears. Beagles are [scent hounds](#), developed primarily for [tracking hare](#), [rabbit](#), and other [game](#). They have a keen sense of smell and tracking instinct that sees them employed as [detection dogs](#) for prohibited agricultural imports and foodstuffs in [quarantine](#) around the world.



<b>Keywords</b>	[beagle] [breed] [brown] [crossbreed] [detection	<a href="#">hide</a>
<b>Images</b>	dog] [dog] [foxhound] [hip dysplasia] [hound] [hunt]	
<b>Related Articles</b>	[miniature] [open throat] [scent hound] [sense of	
<b>External Links</b>	smell] [sing] [small] [snoopy] [rabbit] [United Kingdom]	
<b>Tags</b>		
<b>Sniffed Articles</b>		


Beagle
roll hold close

---

### Beagle

**Summary** [Table of Contents](#)

+ The **Beagle** is a [breed](#) of small to medium-sized [dog](#). A member of the [Hound Group](#), it is similar in appearance to the [Foxhound](#) but smaller, with shorter legs and longer, softer ears. Beagles are [scent hounds](#), developed primarily for [tracking hare](#), [rabbit](#), and other [game](#). They have a keen sense of smell and tracking instinct that sees them employed as [detection dogs](#) for prohibited agricultural imports and foodstuffs in [quarantine](#) around the world.



<b>Keywords</b>	All	Used by this article	Using this article	<a href="#">hide</a>
<b>Images</b>	[abdomen] [breed] [Canadian Kennel Club] [Celtic] [collies]			
<b>Related Articles</b>	[crufts] [dachshund] [dewlap] [dog] [detection dog]			
<b>External Links</b>	[Foxhound] [French] [Hound Group] [Illinois] [Ireland] [K-			
<b>Tags</b>	Run's Park Me In First] [Old English] [Oxford English			
<b>Sniffed Articles</b>	Dictionary] [quarantine] [rabbit] [stifles] [tracking hare]			

## Appendix C: User Study 2 Documents

### Introduction

---

# Wiki-Sniffer

## *Wiki Page Overview Snippet Tool*

---

### Introduction

First of all we want to thank you for your willingness to participate in our user study. In this study we want to evaluate our new Wiki tool that we have developed recently.

To achieve this goal we have prepared some tasks to be done by using our new tool as well as without our tool. Therefore we would like to ask you to help us to test our new tool. In this user study we want to stress that we are testing our tool and not you. So, if you have any troubles or problems with some tasks we ask you to perform, it is the problem with our tool and the setting of this user study, it is not yours.

Here is what we plan for the next. Firstly we will give a little overview about the project itself. Then we will introduce you about our new tool and we would give you time, to test it as you like by yourself. After that we would finish some tasks, and we would like to compare the performance of doing the task with and without the tool.

Please remember that this test is totally voluntary. If you become uncomfortable or find this evaluation objectionable in any way, feel free to quit at any time.

### Project Overview

In the Wikulu project, we work on improving the usability of Wikis using natural language processing (NLP) methods. The goal is to find good ways of presenting the results of these methods to the user in a way that is meaningful and easy to understand.

One possible option is to improve the way how users interact with Wiki's links. Normally users have to clicks the link to get the information from that link. In the new scenario we plan to provide users with a compact and small pop-up window we call snippet. With this snippet users are able to sniff the link to get some parts of information without leaving the originating page. That's why we call this tool Wiki-Sniffer.

In this study we like to evaluate our snippet and we like to know how much our new tool improves the previous Wiki's interface.

---

# Wiki-Sniffer

## *Creating Wiki Page Overview Snippets*

---

### *User Study 2: Evaluation*

#### **Participant's Data**

**Gender** :  male  female

**Job** :

**Age** :

---

For each of these aspects, please tell us how you would rate for the design implementation

	Very Poor	Poor	Okay	Good	Very Good	(no comment)
<b>Simplicity</b>						
<b>Look and feel</b>						
<b>Clarity</b>						
<b>Comfort</b>						
<b>Feature</b>						

Which components do you find very useful or nice?

What modification(s) (if any) should we perform to improve our design implementation?

Do you find the new tool is useful, or do you prefer to browse without the tool?

# Questions

---

Which kinds of information do you get from the snippets?

## Related Articles

### Transformers

Which articles related to Transformers?

<b>Arcee</b>	Yes	No
<b>Bumblebee</b>	Yes	No
<b>Cyathus</b>	Yes	No
<b>Delta Force</b>	Yes	No
<b>Eve</b>	Yes	No
<b>H. Weaving</b>	Yes	No
<b>Jazz</b>	Yes	No
<b>Knut</b>	Yes	No
<b>M. Fox</b>	Yes	No
<b>Optimus</b>	Yes	No
<b>P. Cullen</b>	Yes	No
<b>Revenge</b>	Yes	No
<b>S. Calvin</b>	Yes	No
<b>Soundwave</b>	Yes	No
<b>S. LaBeouf</b>	Yes	No

### Dinosaurs

Which dinosaurs are carnivores?

<b>Aerosteon</b>	Yes	No
<b>Bactrosaurus</b>	Yes	No
<b>Herrerasaurus</b>	Yes	No
<b>Pisanosaurus</b>	Yes	No

### X-Men

Which articles related with to X-Men?

<b>Beast</b>	Yes	No
<b>Chloe</b>	Yes	No
<b>Chris Claremont</b>	Yes	No
<b>Danger Room</b>	Yes	No
<b>H. Jackman</b>	Yes	No
<b>Iceman</b>	Yes	No
<b>Logan</b>	Yes	No
<b>Marvel Comics</b>	Yes	No
<b>Panic Room</b>	Yes	No
<b>Phoenix</b>	Yes	No
<b>Powerpuff Girls</b>	Yes	No
<b>Stan Lee</b>	Yes	No
<b>Storm</b>	Yes	No
<b>X3</b>	Yes	No
<b>Xavier</b>	Yes	No

Which dinosaurs are from America?

<b>Achillobator</b>	Yes	No
<b>Acrocanthosaurus</b>	Yes	No
<b>Dilong</b>	Yes	No
<b>Irritator</b>	Yes	No

## Articles with Same Categories

### Saffron

Which articles are at least in a same category as Saffron?

<b>Actaea racemosa</b>	Yes	No
<b>Amanita phalloides</b>	Yes	No
<b>Anise</b>	Yes	No
<b>Anthoxanthum odoratum</b>	Yes	No
<b>Carmine</b>	Yes	No
<b>Chromatophore</b>	Yes	No
<b>Chrysiridia rhipheus</b>	Yes	No
<b>Cumin</b>	Yes	No
<b>Durian</b>	Yes	No
<b>Euphorbia obesa</b>	Yes	No
<b>Gyromitra esculenta</b>	Yes	No
<b>Nelumbo nucifera</b>	Yes	No
<b>Safflower</b>	Yes	No
<b>Verbena</b>	Yes	No
<b>Vinyl bromide</b>	Yes	No

## Categorizing

### Multiplayer Online Game or Massively Multiplayer Online Game

What kinds of games are they?

	MOG	MMOG
<b>Accursed Lands</b>		
<b>Alteil</b>		
<b>Audition Online</b>		
<b>Avara</b>		
<b>Bomberman Live</b>		
<b>Cube 2: Sauerbraten</b>		
<b>Club Penguin</b>		
<b>Cosmic Rift</b>		
<b>Crysis Warhead</b>		
<b>Need for Speed: WO</b>		
<b>Silverfall</b>		
<b>Sudden Attack</b>		
<b>TrackMania</b>		
<b>Virtual Magic Kingdom</b>		

# Solutions

## Related Articles

### Transformers

Which articles related to Transformers\*?

<b>Arcee</b>	Yes
<b>Bumblebee</b>	No
<b>Cyathus*</b>	No
<b>Delta Force</b>	No
<b>Eve</b>	No
<b>H. Weaving</b>	Yes
<b>Jazz</b>	Yes
<b>Knut*</b>	No
<b>M. Fox</b>	Yes
<b>Optimus</b>	No
<b>P. Cullen</b>	Yes
<b>Revenge</b>	Yes
<b>S. Calvin</b>	No
<b>Soundwave</b>	Yes
<b>S. LaBeouf</b>	Yes

### Dinosaurs

Which dinosaurs are carnivores?

<b>Aerosteon</b>	Yes
<b>Bactrosaurus</b>	No
<b>Herrerasaurus*</b>	Yes
<b>Pisanosaurus</b>	No

### X-Men

Which articles related with to X-Men?

<b>Beast</b>	No
<b>Chloe</b>	No
<b>Chris Claremont</b>	Yes
<b>Danger Room</b>	Yes
<b>H. Jackman</b>	Yes
<b>Iceman</b>	Yes
<b>Loglan</b>	No
<b>Marvel Comics</b>	Yes
<b>Master Juba*</b>	No
<b>Phoenix</b>	Yes
<b>Powerpuff Girls</b>	No
<b>Stan Lee</b>	Yes
<b>Storm*</b>	No
<b>X3</b>	No
<b>Xavier</b>	No

Which dinosaurs are from America?

<b>Achillobator</b>	No
<b>Acrocanthosaurus*</b>	Yes
<b>Dilong</b>	No
<b>Irritator</b>	Yes



## Articles with Same Categories

### Saffron

Which articles are at least in a same category as Saffron\*?

<b>Actaea racemosa</b>	Yes
<b>Amanita phalloides*</b>	No
<b>Anise</b>	Yes
<b>Anthoxanthum odoratum</b>	Yes
<b>Carmin</b>	Yes
<b>Chromatophore*</b>	No
<b>Chrysiridia rhipheus*</b>	No
<b>Cumin</b>	Yes
<b>Durian*</b>	Yes
<b>Euphorbia obesa</b>	Yes
<b>Gyromitra esculenta*</b>	No
<b>Nelumbo nucifera</b>	Yes
<b>Safflower</b>	Yes
<b>Verbena</b>	Yes
<b>Vinyl bromide</b>	No

## Categorizing

### Multiplayer Online Game or Massively Multiplayer Online Game

What kinds of games are they?

<b>Accursed Lands</b>	MOG
<b>Alteil</b>	MOG
<b>Audition Online</b>	MMOG
<b>Avara</b>	MOG
<b>Bomberman Live</b>	MOG
<b>Cube 2: Sauerbraten</b>	MOG
<b>Club Penguin</b>	MMOG
<b>Cosmic Rift</b>	MMOG
<b>Crysis Warhead</b>	MOG
<b>Need for Speed: WO</b>	MMOG
<b>Silverfall</b>	MOG
<b>Sudden Attack</b>	MMOG
<b>TrackMania</b>	MOG
<b>Virtual Magic Kingdom</b>	MMOG

## Appendix D: Wiki-Sniffer Screenshots

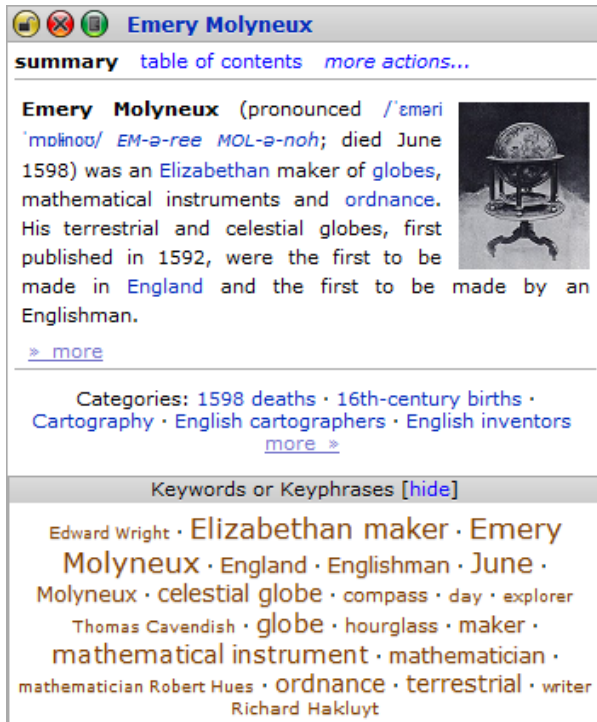


Figure 63: Wiki-Sniffer – Summary

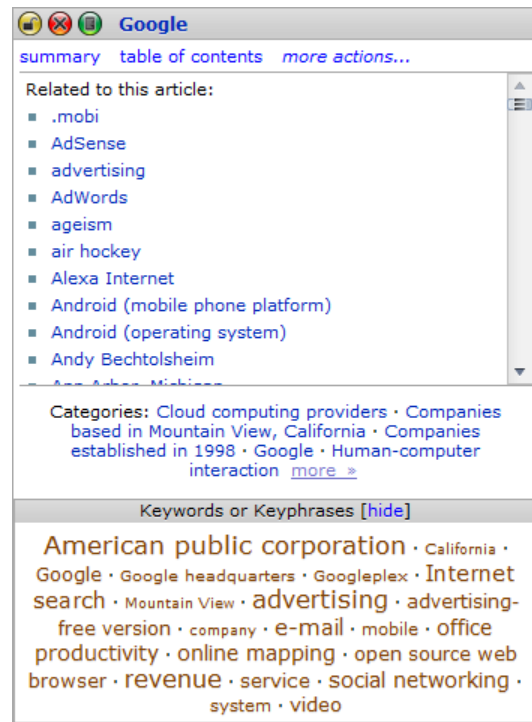


Figure 64: Wiki-Sniffer – Related Articles



Figure 65: Wiki-Sniffer – Table of Contents



Figure 66: Wiki-Sniffer – External Links



Figure 67: Wiki-Sniffer – List of Images



Figure 68: Wiki-Sniffer – Keyphrases are hidden

## Appendix E: Tools and Libraries

**Languages and Environments:** Java, HTML (Hypertext Markup Language), JavaScript, CSS (Cascading Style Sheets), XML (Extensible Markup Language), MediaWiki Wikitext, TWiki Wikitext.

### Libraries:

- **UIMA (Unstructured Information Management Applications):** framework for unstructured information management.
- **Mylyn/WikiText:** library for parsing WikiText (MediaWiki, TWiki, etc.).
- **HTML Parser:** library for parsing HTML.
- **jQuery:** JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.
- **DWR (Direct Web Remoting):** a Java library that enables Java on the server and JavaScript in a browser to interact and call each other as simply as possible. DWR is Easy Ajax for Java.

### Tools:

- **eclipse 3.4 (Ganymede):** Java IDE.  
Plug-ins used:
  - **Subclipse:** SVN tools for eclipse.
  - **Mylyn:** WikiText editor for eclipse.
  - **UIMA:** UIMA feature support for eclipse.
- **Komodo Edit 5.1:** JavaScript editor.

- **Maven:** project management and comprehension tool based on the concept of a project object model (POM). Maven can manage a project's build, reporting and documentation from a central piece of information.
- **Inkscape:** an open source vector graphics editor with capabilities similar to Illustrator, CorelDraw, or Xara X, using the W3C standard Scalable Vector Graphics (SVG) file format.

## Appendix F: Important References

### Links

1. Information about **MediaWiki links**:
  - <http://www.mediawiki.org/wiki/Help:Links>
  - <http://www.mediawiki.org/wiki/Help:Namespaces>
2. List of **wiki applications**: [http://en.wikipedia.org/wiki/List\\_of\\_wiki\\_software](http://en.wikipedia.org/wiki/List_of_wiki_software)
3. Official page of **CoolPreviews**: <http://www.coolpreviews.com/>
4. Official Page of **DWR**: <http://slurm.dojotoolkit.org/dwr/index.html>
5. Official Page of **eclipse**: <http://www.eclipse.org/>
6. Official Page of **HTML Parser**: <http://htmlparser.sourceforge.net/>
7. Official page of **Inkscape**: <http://www.inkscape.org/>
8. Official page of **jQuery**: <http://jquery.com/>
9. Official Page of **Komodo Edit**: [http://www.activestate.com/komodo\\_edit/](http://www.activestate.com/komodo_edit/)
10. Official Page of **Maven**: <http://maven.apache.org/>
11. Official page of **MediaWiki**: <http://www.mediawiki.org>
12. Official Page of **Mylyn/Wikitext**:  
<http://wiki.eclipse.org/Mylyn/Incubator/WikiText>
13. Official page of **TWiki**: <http://twiki.org>
14. Official page of **UIMA**: <http://incubator.apache.org/uima/>
15. **Ubiquitous Knowledge Processing (UKP) Lab**: <http://www.ukp.tu-darmstadt.de/>
16. **Wikulu** homepage: <http://www.ukp.tu-darmstadt.de/projects/wikulu/>

## Bibliography

1. Wiki History. *Cunningham & Cunningham, Inc.* [Online] March 17, 2009. [Cited: October 19, 2009.] <http://c2.com/cgi/wiki?WikiHistory>.
2. **Cunningham, Ward.** Wiki: What is Wiki. *Wiki.* [Online] June 27, 2002. [Cited: October 19, 2009.] <http://www.wiki.org/wiki.cgi?WhatIsWiki>.
3. Google Trends. *Google Trends: TWiki, MoinMoin, PmWiki, MediaWiki, DokuWiki.* [Online] Google, October 20, 2009. [Cited: October 20, 2009.] <http://www.google.com/trends?q=TWiki%2C+MoinMoin%2C+PmWiki%2C+MediaWiki%2C+DokuWiki&ctab=0&geo=all&date=all&sort=3>.
4. Alexa Top 500 Global Sites. *Alexa the Web Information Company.* [Online] October 20, 2009. [Cited: October 20, 2009.] <http://www.alexa.com/topsites>.
5. Statistic - Wikipedia, the free encyclopedia. *Wikipedia, The Free Encyclopedia.* [Online] Wikimedia Foundation, 10 20, 2009. [Cited: 10 20, 2009.] <http://en.wikipedia.org/wiki/Special:Statistics>.
6. **Buffa, Michel.** *Intranet wikis.* Edinburgh : Proceedings of the IntraWebs Workshop 2006 at the 15th International World Wide Web Conference, 2006.
7. Wikulu - Self-Organizing Wikis. *Ubiquitous Knowledge Processing (UKP) Lab.* [Online] Technische Universität Darmstadt. [Cited: November 26, 2009.] <http://www.ukp.tu-darmstadt.de/projects/wikulu/>.
8. **Ye, Shiren, Chua, Tat-Seng and Lu, Jie.** *Summarizing Definition from Wikipedia.* Singapore : National University of Singapore, 2009.
9. **van Duyne, Douglas K., Landay, James A. and Hong, Jason I.** *The Design of Sites.* Upper Saddle River : Prentice Hall, 2008. ISBN 0-13-134555-9.
10. **Hearst, Marti A.** *Search User Interfaces.* New York : Cambridge University Press, 2009. ISBN-13: 9780521113793.
11. **Bates, M. J.** The design of browsing and berrypicking techniques for on-line search interface. *Online Review.* 1989, Vol. 13, 5, pp. 407-431.
12. **O'Day, Vicki L. and Jeffries, Robin.** *Orienteering in an information landscape: how information seekers get from here to there.* Amsterdam : Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems, 1993. ISBN:0-89791-575-5.
13. **Jul, Susanne and Furnas, George W.** *Navigation in Electronic Worlds.* New York : ACM SIGCHI Bulletin, 1997. ISSN:0736-6906.

14. **Chang, S.-J. and Rice, R. E.** Browsing: a multidimensional framework. *Annual Review of Information Science and technology (ARIST)*. 1993, Vol. 28, pp. 231-76.
15. Scanning - Glossary Definition - UsingEnglish.com. *English Language (ESL) Learning Online - UsingEnglish.com*. [Online] August 6, 2009. [Cited: November 19, 2009.] <http://www.usingenglish.com/glossary/scanning.html>.
16. **Cove, J. F. and Walsh, B. C.** Online text retrieval via browsing. *Information Processing and Management: an International Journal*. 1988, Vol. 24, 1, pp. 31-37.
17. **Harper, Simon, et al.** *How Much is Too Much in a Hyperlink Link: Investigating Context and Preview – A Formative Evaluation*. Manchester UK : Information Management Group, Dept of Computer Science, University of Manchester, 2004.
18. **Morkes, John and Nielsen, Jakob.** Consise, SCANNABLE, and Objective: How to Write for the Web. *useit.com: Jakob Nielsen on Usability and Web Design*. [Online] 1997. [Cited: October 28, 2009.] <http://www.useit.com/papers/webwriting/writing.html>.
19. Scent hound. *Wikipedia, The Free Encyclopedia*. [Online] Wikimedia Foundation, Inc., November 3, 2009. [Cited: November 26, 2009.] [http://en.wikipedia.org/wiki/Scent\\_hound](http://en.wikipedia.org/wiki/Scent_hound).
20. **Browne, Glenda and Jermeij, Jonathan.** *Website indexing : enhancing access to information within websites*. Adelaide : Auslib Press, 2004. ISBN:1875145567.
21. **Marchionini, Gary and Levi, Michael.** *Digital government information services: the Bureau of Labor statistics case*. s.l. : interactions, 2003.
22. **Pirolì, Peter.** *Information Foraging Theory: Adaptive Interaction with Information*. New York : Oxford University Press, 2007.
23. **Nakayama, Kotaro, et al.** *Wikipedia Mining – Wikipedia as a Corpus for Knowledge Extraction*. Osaka : Dept. of Multimedia Eng., Graduate School of Information Science and Technology, 2008.
24. **Désilets, Alain, et al.** *Are Wikis Usable*. s.l. : Proceedings of the WikiSym, 2005.
25. **Chakrabarti, Soumen.** *mining the web – Discovering Knowledge from Hypertext Data*. San Francisco : Morgan Kaufmann Publisher, 2003. ISBN-13:978-1-55860-754-5.
26. **Turney, Peter D.** *Learning Algorithms for Keyphrase Extraction*. Ottawa, Ontario : National Research Council of Canada, 1999.
27. **Frank, Eibe, et al.** *Domain-Specific Keyphrase Extraction*. San Francisco : Morgan Kaufmann Publishers Inc., 1999.

28. **Hulth, Anette.** *Improved Automatic Keyword Extraction Given More Linguistic Knowledge.* Stockholm : Department of Computer and System Sciences Stockholm University Sweden, 2003.
29. **Medelyan, Olena and Witten, Ian H.** *Thesaurus Based Automatic Keyphrase Indexing.* Chapel Hill : Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries, 2006. ISBN:1-59593-354-9.
30. **Barker, Ken and Cornacchia, Nadia.** *Using Noun Phrase Heads to Extract Document Keyphrases.* s.l. : Canadian Conference on AI, 2000.
31. **Bracewell, D. B., Ren, F. and Kuriowa, S.** *Multilingual single document keyword extraction for information retrieval.* Japan : Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, 2005. pp. 517-522. ISBN:0-7803-9361-9.
32. **Mihalcea, Rada and Tarau, Paul.** *TextRank: Bringing Order into Texts.* Barcelona : In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), 2004.
33. **Brin, Sergey and Page, Lawrence.** *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* Brisbane : In Seventh International World-Wide Web Conference (WWW 1998), 1998.
34. **Zesch, Torsten and Gurevych, Iryna.** *Aproximate Matching for Evaluating Keyphrase Extraction.* Darmstadt : Ubiquitous Knowledge Processing Lab Computer Science Department - Technische Universität Darmstadt, 2009.
35. **Erkan, Güneş and Radev, Dragomir R.** LexRank: Graph-based Lexical Centrality as Saliency in Text Summarization. *Journal of Artificial Intelligence Research.* 2004, Vol. 22, pp. 457-479.
36. **Radev, Dragomir R., et al.** Centroid-based summarization of multiple documents. *Information Processing and Management.* 2004, Vol. 40, pp. 919-938.
37. **Goldberg, Andrew.** *CS838-1 Advanced NLP: Automatic Summarization.* Madison : University of Wisconsin-Madison, 2007.
38. Apache UIMA. *Apache Incubator.* [Online] The Apache Software Foundation, November 26, 2009. [Cited: November 30, 2009.] <http://incubator.apache.org/uima/>.
39. **Nielsen, Jakob.** Usability 101: Introduction to Usability. *useit.com: Jakob Nielsen's Website.* [Online] Nielsen Norman Group, 2003. [Cited: 11 24, 2009.] <http://www.useit.com/alertbox/20030825.html>.
40. **Shneiderman, Ben, Byrd, Don and Croft, W. Bruce.** *Clarifying search: A user-interface framework for text searches.* s.l. : DL Magazine, 1997. ISSN1082-9873.

41. **Greene, Stephan, et al.** Previews and overviews in digital libraries: Designing surrogates to support visual information seeking. *Journal of the American Society for Information Science*. 2000, Vol. 51, 4, pp. 380-393.
42. *Evaluating the Layout of Graphical User Interface Screens: Validation of a Numerical Computerized Model*. 4, Tel Aviv : International Journal of Human-Computer Interaction, 1998, Vol. 10, pp. 343 – 360.
43. **Hassenzahl, Marc.** The interplay of beauty, goodness, and usability in interactive. *Human-Computer Interaction*. 2004, Vol. 19, 4, pp. 319-349.
44. **Lindgaard, Gitte and Dudek, Cathy.** What is this evasive beast we call user satisfaction? *Interacting with Computers*. 2003, Vol. 15, 3, pp. 429-452.
45. **van der Heiden, Hans.** Factors influencing the usage of websites: the case of a generic portal in The Netherlands. *Information & Management*. 2002, Vol. 40, 6, pp. 541-549.
46. **Ben-Bassar, Tamar, Meyer, Joachim and Tractinsky, Noam.** Economic and subjective measures of the perceived value of aesthetics and usability. *ACM Transactions on Computer-Human Interaction (TOCHI)*. 2006, Vol. 13, 2, pp. 210-234.
47. **Hotchkiss, Gord.** Q&A With Marissa Mayer, Google VP, Search Products & User Experience. *Search Engine Land*. [Online] Third Door Media, Inc., Jan 26, 2007. [Cited: November 26, 2009.] <http://searchengineland.com/qa-with-marissa-mayer-google-vp-search-products-user-experience-10370>.
48. WordNet Search - 3.0. *WordNet Search - 3.0*. [Online] The Trustees of Princeton University. [Cited: November 27, 2009.] <http://wordnetweb.princeton.edu/perl/webwn?s=snippet&sub=Search+WordNet&o2=&o0=1&o7=&o5=&o1=1&o6=&o4=&o3=&h=>
49. Definition of snippet noun from Cambridge Dictionary Online. *Cambridge Dictionary Online: Free English Dictionary and Thesaurus*. [Online] Cambridge University Press, 2009. [Cited: November 27, 2009.] <http://dictionary.cambridge.org/define.asp?key=75173&dict=CALD>.
50. What is the Google Snippet? *Search Engine Roundtable*. [Online] RustyBrick ®, Inc. Web Development, November 27, 2007. [Cited: November 24, 2009.] <http://www.seroundtable.com/archives/015452.html>.
51. **Fontes, Paul, Battle, Alexis and Anderson, Corin.** *Expanded Snippets*. 11/394,192 United States, October 11, 2007.
52. **Arend, Udo and Hensel, Martin.** *AUTO-ZOOMABLE SNIPPETS IN MULTIPLE SNIPPET WINDOWS*. 11/320,690 United States, July 5, 2007.



53. **Webb, Mark Stephen.** *COLLAPSIBLE DIALOG WINDOW*. 09/905,298 Uniter States, Nov 17, 2009.
54. API - MediaWiki. *MediaWiki*. [Online] Wikimedia Foundation, May 10, 2009. [Cited: November 21, 2009.] <http://www.mediawiki.org/wiki/API>.
55. **Talbot, David.** Wikipedia Gets Ready for a Video Upgrade: The online encyclopedia is poised to let users find, edit, and embed clips. *Technology Review: The Authority on the Future of Technology*. [Online] Massachusetts Institute of Technology, June 19, 2009. [Cited: November 21, 2009.] <http://www.technologyreview.com/web/22900/page1/>.
56. Earth. *Wikipedia, The Free Encyclopedia*. [Online] Wikimedia Foundation, Inc., November 25, 2009. [Cited: November 26, 2009.] <http://en.wikipedia.org/wiki/Earth>.
57. What is TWiki? *TWiki.org*. [Online] TWiki, Inc., November 12, 2009. [Cited: November 27, 2009.] <http://twiki.org/cgi-bin/view/TWiki/WhatIsTWiki>.
58. Structured Wiki. *TWiki.org*. [Online] Twiki, Inc., July 21, 2008. [Cited: November 27, 2009.] <http://twiki.org/cgi-bin/view/Codev/StructuredWiki>.
59. Help:HTML in wikitext. *Wikipedia, The Free Encyclopedia*. [Online] Wikimedia Foundation, Inc., November 26, 2009. [Cited: November 27, 2009.] [http://en.wikipedia.org/wiki/Help:HTML\\_in\\_wikitext](http://en.wikipedia.org/wiki/Help:HTML_in_wikitext).
60. HTML Tags. *Quackit.com*. [Online] 2009. [Cited: November 27, 2009.] <http://www.quackit.com/html/tags/>.
61. JQuery – write less, do more. *JQuery – write less, do more*. [Online] John Resig and the jQuery Team, October 23, 2009. [Cited: December 1, 2009.] <http://jquery.com/>.
62. **Hoffart, Johannes, Zesch, Torsten and Gurevych, Iryna.** *An Architecture to Support Intelligent User Interfaces for Wikis by Means of Natural Language Processing*. Orlando : Proceedings of the 5th International Symposium on Wikis and Open Collaboration, 2009. ISBN:978-1-60558-730-1.
63. **Sauer, Christoph.** What you see is Wiki - Questioning WYSIWYG in the Internet Age. *i3G Institut Hochschule Heilbronn*. [Online] August 5, 2006. [Cited: October 27, 2009.] <http://www.i3g.hs-heilbronn.de/attach/Veroeffentlichungen/What+you+see+is+Wiki.pdf>.
64. **Nakayama, Kotaro.** *Wikipedia Mining for Triple Extraction Enhanced by Co-reference Resolution*. Tokyo : The Center for Knowledge Structuring, The University of Tokyo, 2008.
65. **Norman, D. A.** *The Psychology of Everyday Things*. New York : US: Basic Books, 1988.
66. **Schneiderman, B., Byrd, D. and Croft, WB.** *Clarifying Search: A User-Interface Framework for Text Searches*. s.l. : DL Magazine, 1997.

67. **Hertzum, Morten and Frøkjær, Erik.** *Browsing and querying in online documentation: a study of user interfaces and the interaction process.* New York : ACM Transactions on Computer-Human Interaction (ToCHI), 1996. ISSN:1073-0516.
68. Beagle. *Wikipedia, The Free Encyclopedia.* [Online] Wikimedia Foundation, Inc., November 23, 2009. [Cited: November 26, 2009.] <http://en.wikipedia.org/wiki/Beagle>.
69. World Wide Web. *Wikipedia, The Free Encyclopedia.* [Online] Wikimedia Foundation, Inc., November 23, 2009. [Cited: November 26, 2009.] [http://en.wikipedia.org/wiki/World\\_Wide\\_Web](http://en.wikipedia.org/wiki/World_Wide_Web).
70. Streptomyces achromogenes. *Wikipedia, The Free Encyclopedia.* [Online] Wikimedia Foundation, Inc., October 31, 2009. [Cited: November 26, 2009.] [http://en.wikipedia.org/wiki/Streptomyces\\_achromogenes](http://en.wikipedia.org/wiki/Streptomyces_achromogenes).
71. ISO 639. *Wikipedia, The Free Encyclopedia.* [Online] Wikimedia Foundation, inc., October 5, 2009. [Cited: November 27, 2009.] [http://en.wikipedia.org/wiki/ISO\\_639](http://en.wikipedia.org/wiki/ISO_639).
72. Cascading Style Sheets. *Wikipedia, The Free Encyclopedia.* [Online] Wikimedia Foundation, Inc., November 30, 2009. [Cited: December 1, 2009.] [http://en.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](http://en.wikipedia.org/wiki/Cascading_Style_Sheets).
73. Hypercard. *Wikipedia, The Free Encyclopedia.* [Online] Wikimedia Foundation, Inc., November 19, 2009. [Cited: November 26, 2009.] <http://en.wikipedia.org/wiki/Hypercard>.