

**Technische Universität Dresden**  
**Department of Computer Science**  
Institute for Systems Architecture  
Diploma Thesis in Computer Networks

# WebSnippets

Extracting and Ranking of entity-centric knowledge  
from the Web

**Author**

Christopher Friedrich

**Supervisors**

Dipl. Med. Inf. David Urbansky

Prof. Dr. rer. nat. habil. Dr. h.c. Alexander Schill

**Date of submission**

April 30, 2010

## Abstract

Today's Web comprises a vast and growing knowledge of information organized and presented in HTML pages. Major search engines like Google and Bing are continuously crawling and indexing the Web to provide a simple and convenient keyword based access to this information. Their focus primarily is to help users find relevant URLs that satisfy a particular information need. Besides Web search there are also popular directories (e.g., DMOZ, Yahoo) that support a browsing based exploration. Both of these paradigms are based on a page level type of aggregation and navigation.

We argue that there is an emerging interest to organize information around entities or concepts, which is not well served with the current information retrieval and browsing capabilities provided on the Web. As a solution we're introducing *WebSnippets*, relevant context surrounding entities or concepts, automatically discovered and extracted from the Web. Those *WebSnippets* comprise information, facts, and knowledge ("soft facts") that can then be consumed by users to learn more about those topics. We're presenting a large scale system architecture based on the *Web carnivore* approach that systematically leverages web search engines to find information sources as potential candidates, and then extracts, filters, and ranks snippets based on their overall relevancy and interestingness. We evaluated our approach and compared the quality of extracted snippets with Google, and other popular information sources that show page summaries and extracted information. Our results show that our *WebSnippets* are considered on average three times as relevant and four times as interesting compared to those sources. We also show a case study where we integrated *WebSnippets* into the *Live Like a German* travel web site to augment their editorial content with high quality, algorithmically extracted information snippets related to places and travel destinations in Germany.

We conclude that *WebSnippets* can be useful for a variety of web applications, knowledge discovery, and present a new way of allowing users to conveniently browse and explore the vast knowledge of the Web organized by their favorite entities and topics of interest.

# Statement of authorship

I hereby certify that this diploma thesis has been composed by myself, and describes my own work, unless otherwise acknowledged in the text. All references and verbatim extracts have been quoted, and all sources of information have been specifically acknowledged. It has not been accepted in any previous application for a degree.

Dresden, April 30, 2010

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Terminology . . . . .	1
1.2	Motivation . . . . .	3
1.3	Problem Statement . . . . .	8
1.4	Outline . . . . .	9
<b>2</b>	<b>Background</b>	<b>11</b>
2.1	Preliminaries . . . . .	11
2.1.1	Knowledge Representation . . . . .	12
2.1.2	Web Information Retrieval and Extraction . . . . .	13
2.1.3	Evaluation of Web Information Retrieval . . . . .	14
2.1.4	Natural Language Processing . . . . .	16
2.1.5	Machine Learning . . . . .	18
2.2	Related Work . . . . .	20
2.2.1	Web Carnivores . . . . .	20
2.2.2	Word Sense Disambiguation . . . . .	21
2.2.3	Text Summarization . . . . .	21
2.2.4	Web Information Extraction . . . . .	23
2.2.5	Near Duplicate Elimination . . . . .	25
<b>3</b>	<b>Design</b>	<b>26</b>
3.1	System Overview . . . . .	27
3.2	Source Aggregation Process . . . . .	28
3.2.1	Search Engine Selection . . . . .	29
3.2.2	Query Generation . . . . .	29
3.2.3	Ranking and Filtering . . . . .	30
3.3	Snippet Extraction Process . . . . .	31
3.3.1	Web Result Summary . . . . .	32
3.3.2	Document Sentences . . . . .	34
3.3.3	Document Snippets . . . . .	35

3.4	De-Duplication Process . . . . .	36
3.5	Snippet Ranking Process . . . . .	37
3.5.1	Feature Extraction . . . . .	38
3.5.2	Feature Learning and Ranking . . . . .	41
3.6	Persistence . . . . .	41
3.7	Summary . . . . .	42
<b>4</b>	<b>Implementation</b>	<b>43</b>
4.1	Technology . . . . .	43
4.1.1	Development Languages . . . . .	43
4.1.2	Frameworks and Libraries . . . . .	44
4.1.3	WebKnox . . . . .	46
4.2	System Architecture . . . . .	48
4.2.1	Persistence . . . . .	49
4.2.2	Lifecycle . . . . .	51
4.2.3	Source Aggregation Process . . . . .	52
4.2.4	Snippet Extraction Process . . . . .	53
4.2.5	De-Duplication Process . . . . .	54
4.2.6	Snippet Ranking Process . . . . .	54
4.3	Engineering Issues . . . . .	55
4.4	Summary . . . . .	57
<b>5</b>	<b>Evaluation</b>	<b>58</b>
5.1	Introduction . . . . .	58
5.2	Methodology . . . . .	59
5.2.1	Baseline . . . . .	60
5.2.2	Test Set . . . . .	60
5.2.3	Evaluation Guideline . . . . .	61
5.2.4	Error Rate . . . . .	63
5.3	Problem Validation . . . . .	63
5.3.1	Evaluation of the Baseline . . . . .	63
5.3.2	Problem Validation Summary . . . . .	64
5.4	Snippet Extraction Performance . . . . .	64
5.4.1	Evaluation of Snippet Candidate Set . . . . .	64
5.4.2	Evaluation of Snippet Ranking . . . . .	65
5.4.3	Overall Snippet Extraction Performance Summary . . . . .	67
5.5	Discussion of Results . . . . .	67

<b>6 Case Study: WebSnippets in Live Like a German Travel Web Site</b>	<b>71</b>
6.1 Overview of Live Like a German . . . . .	73
6.2 Web-Snippets Integration . . . . .	73
<b>7 Conclusions and future work</b>	<b>77</b>
7.1 Achievements . . . . .	78
7.2 Future Work . . . . .	78
7.2.1 Automatic Content Construction . . . . .	78
7.2.2 Enhanced Ranking . . . . .	79
7.2.3 Personalized Recommendations . . . . .	79
7.3 Summary . . . . .	79
<b>A Snippets</b>	<b>80</b>
A.1 Preliminary Snippets . . . . .	80
<b>B Error Rates</b>	<b>88</b>
<b>C Prediction Models</b>	<b>90</b>

# List of Figures

1.1	Screenshot of the DMOZ web directory . . . . .	3
1.2	Screenshot of a web result that contains insufficient information about an entity . . . . .	4
1.3	Screenshot of a web result that contains “good” information about an entity . . . . .	4
1.4	Screenshot of a web page that contains an entity . . . . .	5
1.5	An example of an entity-centric search query . . . . .	6
1.6	Screenshot of the experiment evaluating snippets . . . . .	7
2.1	Overview over the related research areas . . . . .	11
2.2	The basic supervised machine-learning process . . . . .	19
3.1	Flow-diagram system-design . . . . .	27
3.2	Flow-diagram of the source aggregation process . . . . .	29
3.3	Flow-diagram snippet-extraction-process . . . . .	31
3.4	A best-case sample of a web result summary . . . . .	34
3.5	A worst-case sample of a web result summary . . . . .	34
3.6	Flow-diagram of the De-Duplication Process . . . . .	36
3.7	Sample of near-duplicate snippets for “Frankfurt” . . . . .	37
3.8	Flow-diagram of the Snippet Ranking Process . . . . .	38
3.9	Entity Relationship Diagram of the database design . . . . .	42
4.1	WebKnox high-level overview . . . . .	46
4.2	Class-diagram of WebKnox packages . . . . .	47
4.3	Sequence-diagram websnippets . . . . .	51
5.1	Screenshot of the Evaluation Tool . . . . .	59
5.2	Results of the WebSnippets Performance . . . . .	65
5.3	Overview of the snippet ranking prediction queue . . . . .	65
5.4	Comparison of the Baseline, WebSnippets and the Gold Standard . . . . .	68
5.5	Overview over the judgements given by the editorial team . . . . .	70

6.1	Screenshot of Live Like a German Website . . . . .	71
6.2	Traditional editorial approach of Live Like A German content aggregation . . . . .	72
6.3	Screenshot of Live Like a German Travel Snippets . . . . .	74
6.4	Flow-diagram of Live Like a German system-integration . . . . .	75



# List of Tables

2.1	Partitioning of IR documents, with respect to a given query . . .	15
2.2	Partitioning of IR documents, with respect to a given query . . .	16
4.1	Database schema for snippets . . . . .	49
4.2	Search Engine IDs used in WebSnippets . . . . .	50
5.1	Test set used for evaluation . . . . .	60
5.2	Performance Evaluation of the Baseline . . . . .	63
5.3	Evaluation of the prediction model for relevancy, interestingness, and curiosity after automatic feature selection . . . . .	66
5.4	Evaluation of the prediction model for relevancy, interestingness, and curiosity after automatic feature selection . . . . .	66
5.5	Performance comparison of the <i>Baseline</i> , <i>WebSnippets</i> , and our <i>gold standard</i> . . . . .	67
B.1	Error rate for Google Web . . . . .	88
B.2	Error rate for Twitter . . . . .	88
B.3	Error rate for Google Blogs . . . . .	89
B.4	Error rate for Editorial Picks . . . . .	89
B.5	Error rate for WebSnippets . . . . .	89

# Acknowledgments

I appreciate the many contributions that people have made for this thesis. First of all, I am grateful to my advisor, *David Urbansky*, for his guidance and patience with me during these past nine months. I also wish to thank the faculty and staff of the computer science department at TU Dresden for their help and support during the last months.

Second, I would like to thank my friend and mentor *Reiner Kraft*, who helped and guided me during the many stages of my career and this thesis. Without his continuous support, insightful comments, thorough proofreading and innovative ideas this thesis wouldn't be what it is today.

Furthermore, I would like to thank the editorial team, and everyone else who helped developing, building and evaluating the *WebSnippets* system.

Especially, I would like to thank my girlfriend *Anna-Sophie Klotz* for her kindness, support, and understanding in so many situations. Therefore, I am looking forward to our trip to the Silicon Valley, to show her *WebSnippets* competitors as well as *Live Like a German* headquarters.

Last not least, I would like to thank my family for their unending and generous support throughout the last several years, and for their continuous encouragement to further pursue my Diploma degree.

# Chapter 1

## Introduction

This thesis explores research questions that aim to automatically extract relevant contextual information (and so called “soft-facts”) for entities and concepts from the Web.

We first introduce the terms and terminology used throughout this thesis, followed by an introduction that motivates the research on automated entity-centric snippet extraction. Then we state the problems that we explore in this thesis. The chapter ends with an overview and outline of the subsequent chapters of the thesis.

### 1.1 Terminology

This section briefly defines the terminology we are using in this thesis paper.

**concept** We refer in general to everything that exists as an idea or abstract concept as a *concept*. For example, “car insurance” is a concept. There is no concrete or specific instance to it, but it still exists as an abstract idea or thought. Topics (e.g., “gardening”) are also considered concepts, and entities itself are also concepts.

**entity** With *entities* we refer to something which is perceived, known or inferred to have its own distinct existence (living or nonliving). For example, a person “Brad Pitt” is an entity. It is a specific instance of a person. Within the research area of NER [36] common entity types are person, places, or organizations. Entities are therefore a specific subset of concepts.

**snippet** With *snippet* we refer to a piece of text of varying length. This can either be a full sentence or an excerpt, that can be surrounded by ellipses.

Throughout this thesis we refer to a snippet in the context of an entity-centric snippet, which is a piece of text surrounding an entity or concept. In other literature a snippet is also referred to as *context* [51], *information nugget* or simply *nugget*. We also might refer to a snippet as a *soft-fact*, as opposed to a “hard-fact” or *fact*, which is a single value, that has one truth, such as a birthday.

**entity-centric** With *entity-centric* we refer to information or content that is about a specific entity.

**query** With *query* we refer to a string of text, which is send to a search engine or IR system as input to express an *information need*.

**search engine** With *search engine* we refer to a general purpose (web) search engine like Google<sup>1</sup>, Yahoo<sup>2</sup>, Bing<sup>3</sup>, or an IR system which accepts a query as input and returns an ordered list of web results.

**web result** With *web result* we refer to an object, which is returned by a search engine. Throughout this thesis, we assume that a web result is composed of a *URL* or reference to an external document, the *title* and/or optionally the *description* of the document. Furthermore, we assume that a web result contains additional meta-data and information about the *search engine* it was retrieved from, including its *rank* position assigned by the search engine and the *query* used.

**information overload** With *information overload* we refer to the issue of making decisions based on too much information provided. Especially on the Web, without knowing the validity of the content and the risk of misinformation, it is hard for the user to make a decision.

**information need** With *information need* we refer to the user’s desire to obtain information to satisfy need [48]. An information need can be expressed using one or multiple queries.

**user-centric** As opposed to *machine-centric* systems, where the primary consumer of output is a machine, with *user-centric* we refer to systems, where the primary consumer of output is a human person.

---

<sup>1</sup><http://www.google.com>, accessed 31/01/2010

<sup>2</sup><http://www.yahoo.com>, accessed 31/01/2010

<sup>3</sup><http://www.bing.com>, accessed 31/01/2010

## 1.2 Motivation

Terrabytes of information are nowadays freely accessible, which leads to information overload to users. In 1983 Gerald Salton, father of IR, predicted a time when people would be forced to use information that was easily available and ignore the rest. [44]

Search engines like Google aim to organize the world's information. But they have limitations: Their focus is primarily to help users find relevant URLs that satisfy their information need using keyword search queries. This type of retrieval works extremely well for navigational queries, where a user for example is looking for a particular homepage. Besides navigational queries there are also informational type of queries [12]. Search engines are also very good at locating documents for this type of queries, but it is up to the user then to scan through the documents and retrieve the desired information somewhere buried within its text. This manual scanning process can be very labor intensive. The overall retrieval paradigm for Web search is based on locating URLs (representing web pages) given a query.

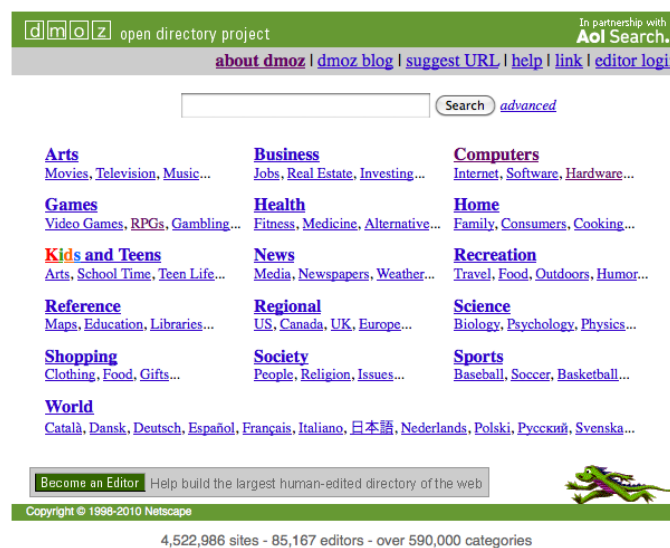


Figure 1.1: Screenshot of the DMOZ web directory

As an alternative to web search many major web directories (e.g., DMOZ<sup>4</sup>, Yahoo) exist to support browsing to allow users more in an exploration mode to locate useful web resources. In this model a user typically navigates through

<sup>4</sup><http://www.dmoz.org>, accessed 31/01/2010

a taxonomy of topics, organized by the creator of the directory, to eventually find links to URLs that seem promising. Similarly the navigation is based on following URLs representing web pages.

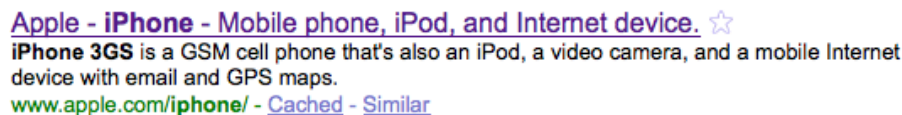
Search engines help users by returning summaries or a “snippet” of a document that try to summarize a web page and provide guidance on whether the content of the web page is worth exploring further [26]. Often those summaries are dynamically generated to include the user’s query as an anchor into the document. One problem with this approach is that those snippets are optimized for page summarization, not necessarily for information consumption of a particular topic. For example, a page may comprise a large subset of topics or entities. A good page summarizer needs to provide an overview to the user that a page is about these topics, but given the space constraints cannot then go further and provide more detailed information for each topic or entity that is listed. Furthermore, the same topic can occur multiple times across the document. Suppose that all occurrences would reveal interesting information, a snippet has to be short, therefore would not be able to incorporate all this data and needs to focus in typically on one prominent occurrence.



[Apple iPhone: iPhone 3G reviews, news, photos and videos - CNET.com](#) ☆  
Check out CNET's iPhone coverage, including up-to-the-minute news about the latest iPhone 3G S, in-depth reviews of the different iPhone models, ...  
[www.cnet.com/apple-iphone.html](#) - 8 minutes ago - [Cached](#) - [Similar](#)

Figure 1.2: Screenshot of a web result that contains insufficient information about an entity

We are interested in a use case where a user wants to explore a concept, or entity, and would like to consume the most interesting facts or news very quickly. The reason is that there are many of this type of queries. Guo et al. argue [21], that 71 percent of search queries contain named entities.



[Apple - iPhone - Mobile phone, iPod, and Internet device.](#) ☆  
iPhone 3GS is a GSM cell phone that's also an iPod, a video camera, and a mobile Internet device with email and GPS maps.  
[www.apple.com/iphone/](#) - [Cached](#) - [Similar](#)

Figure 1.3: Screenshot of a web result that contains “good” information about an entity

For more static type of facts users would typically go to Wikipedia<sup>5</sup> and enter the topic as a query. Then scan through the Wikipedia page to find pieces of interest. Wikipedia articles can be very long, and a lot of interesting facts are sometimes difficult to locate if the article is not well organized. Besides Wikipedia there are a plethora of web documents available that provide more information for a given entity that is very relevant and can highly augment the information that has been provided by Wikipedia. However, it is very tedious to manually search for these - given the limitations of the current retrieval systems. Scanning through hundreds of search results, then reviewing and reading each of the pages to find a particular interesting fact is a not scalable for human consumption. In addition, many pages may contain interesting facts, but a user already had known about this one from a previous page, so the provided information, although considered relevant from a search engine's perspective is not all that useful to the user.



Figure 1.4: Screenshot of a web page that contains an entity

For more dynamic type of sources, like News or blogs, a user can use the same approach outlined in the previous paragraph, but the same limitations apply. Particularly the problem of finding the same news over and over again is more apparent here, since when a news article spreads it gets typically copied and minor modifications are gradually added (e.g., the same article, minor change in

<sup>5</sup><http://en.wikipedia.org>, accessed 31/01/2010

the headline), which is a problem area major news aggregators face. Removing duplicative news articles is a challenging task itself.

We therefore argue that there is a lack of entity or concept centric retrieval capabilities to support this use case, and none of the major search engines are optimized for this usage scenario.

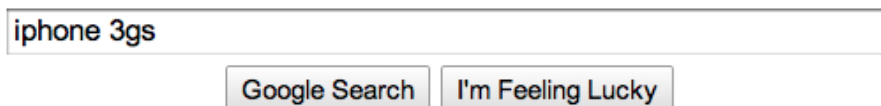


Figure 1.5: An example of an entity-centric search query

In an “entity-centric” retrieval system the underlying information units are entities or concepts, not simple text tokens. For example, the actor “Tom Cruise” represents an instance of such an entity of type “actor”. While the research area related to entity-centric retrieval [5] has seen some increased popularity in the past years [15], there is still a need to bring more focus towards the discovery and consumption aspects of what we call “soft-facts” for a given entity or concept. It would be desirable to discover new facts and information about “Tom Cruise” and organize this data intuitively to make it convenient for in-page consumption.

An “entity-centric” search system takes as an input an entity or concept, then constantly scours the Web for new “information nuggets” (e.g., soft-facts, news), and adds those to a knowledge base. There are many research challenges in such a system. For example, entities might be ambiguous (*Washington* the place vs. *Washington* the person). Even within one category (e.g., a place) there can be ambiguity (e.g., *Washinton* the state vs. *Washington* the city.) Facts need to be filtered out that are not relevant for a particular meaning of an entity. In addition, there is plenty of redundant information available for a given entity (e.g., “Frankfurt is a town in Hesse” vs. “Frankfurt am Main is a town in Hesse”). Both of these are very similar and therefore redundant from a user’s perspective. An “entity-centric” search system should therefore identify this semantic similarity and only add information to the knowledge base that is not already there. Furthermore, ranking information “nuggets” or snippets (which we use as the terminology in this thesis) is critical to quickly identify the most relevant snippets based on a user’s interest. Such ranking can work on different dimensions, such as “relevancy” of snippets in regard to the entity,



“interestingness”, or “buzzyness” to name a few. Last not least, such a system should be personalized, and know what a user has already learned about, and only present over time new knowledge that a user has not seen before.

To validate our hypothesis we conducted an experiment where we collected a small random set of entities (e.g., places, products, organizations, persons). We then issued these as queries to various search engines (e.g., Google, Yahoo, Bing, ...) and obtained the top  $k$  ranked snippets and title of each result item. We would then merge them in random order into a list, sorted by entity, and then present that list to a user. The user then would be ask to judge the quality of each snippet along different dimensions (e.g., overall relevancy, interestingness, likelihood to click on it to learn more.)

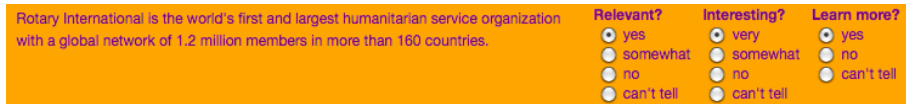


Figure 1.6: Screenshot of the experiment evaluating snippets

The result of this study suggests that the existing search engines are not ideal for supporting this entity-centric exploration use case, based on their overall low relevancy and interestingness scores for their snippets that were judged in this experiment. This validates our initial observation that search engine snippets are primarily optimized for page summarization, not in-place information consumption.

Besides end users there are also publishers and web site owners who’s goal is to keep users/traffic on their site. By providing contextual relevant facts and information for each entity that appears on a page a user can conveniently consume that information in-context, therefore staying longer on a the page instead of leaving and searching for that information elsewhere [27, 28, 51]

A key problem that remains for a publisher is to find such relevant contextual information. There are feed provider that sell this type of information, and those feeds may not be cheap to license. In general editorial resources are expensive, and traditionally many of those commercial feeds are built and organized by editors to maintain high quality.

We were therefore motivated to find an algorithmic and scalable solution for finding high quality web content and associate it with corresponding entities.

### 1.3 Problem Statement

Our experiments have shown that there are interesting information “snippets” that are buried in deep web results, and then further on large pages with lots of non-relevant data around it (e.g., sidebar navigation, ads, long articles, footer, ...) - these comprise valuable entity-centric information for consumption, but are difficult to discover using traditional information retrieval methods.

We argue that there is an emerging interest to organize information around entities or concepts, which is not well served with the current information retrieval and browsing capabilities provided on the Web. We would like to enable a user therefore to conveniently locate relevant and useful information for their topics of interest that can be consumed effectively without requiring the user to scan through long documents or read relevant news articles. We envision interfaces that can be accessed via Search or Browse based paradigm, or applications that will provide direct access to a user’s favorite entities and concepts (e.g., bookmarking) and then allow to consume relevant soft-facts or news for those entities/concepts in place.

To address this we need to solve the following problems:

1. Constantly and efficiently identify documents on the Web that contain relevant entities or concepts, along with associated information for a given context.
2. Recognize entities in context to ensure the correct meaning of an entity is used when extracting surrounding context. For example, for the entity “Washington” the person we do not want to extract context related to “Washington” the place.
3. Extract relevant context located in a window around entities (or located on the same page), select appropriate boundaries to avoid under- or over-selection of the surrounding context.
4. Collect all contexts for a given entity (recall) and remove redundant information and duplicates.
5. Keep up with real-time and fresh information as it becomes available for a given entity.
6. Rank remaining contexts for a given entity based on their overall quality and interestingness. We do not consider personal relevance, but focus on general relevancy and interestingness to a broader audience. Ultimately personal relevance seems an important direction to consider in future work.

7. Build a platform that provides APIs with convenient access to these contexts given an entity as an input

The research questions are addressed by designing, implementing and evaluating a system that automatically finds and extracts knowledge from websites and stores it so that a user can easily access it. The requirements for such a system are as follows:

- The system needs prior knowledge as input to find contextual-relevant information. It therefore takes as input a list of entities / concepts, along with possible meta-data for those entities.
- Editorial interfaces need to be provided to manage meta-data.
- The system design needs to be scalable, large-scale, and possess good performance to be able to handle a large set of entities, and generate snippets on an ongoing basis.
- The system design needs to be domain independent. For example, should work well for places, persons, organizations, and general topics.
- Support for the multiple sources of information from the web. There are different web search engines that are accessed with possible different APIs and protocols. In addition, document sources can have different file formats.
- Support for context (provenance) of information: entity descriptions are given in the context of a website or dataset;
- The system needs to provide APIs for retrieving or subscribing to snippets for a given entity or list of entities.

## 1.4 Outline

The remainder of this thesis is organized as follows:

Chapter 2 provides more background and an overview of related work.

Chapter 3 presents the system design and reviews its main components.

Chapter 4 discusses implementation and engineering aspects, and presents an overview of technologies used, which design features have been implemented, and how.

In chapter 5 we evaluate and study the system's performance and overall quality of the snippet extraction.

Chapter 6 shows a concrete use case on how the WebSnippet system can be integrated into a travel web site, helping with the automatic generation of entity centric content for consumption.

Chapter 7 concludes with a summary of our results, achievements, and lessons learned while working on our WebSnippet system. An outlook is provided on future work and open problems.

## Chapter 2

# Background

This chapter comprises two parts. The first part reviews the preliminaries in the area of knowledge representation, web information retrieval and extraction, natural language processing, and data mining. The second part reviews state-of-the-art related work that has been proposed in different research areas relevant to our work.

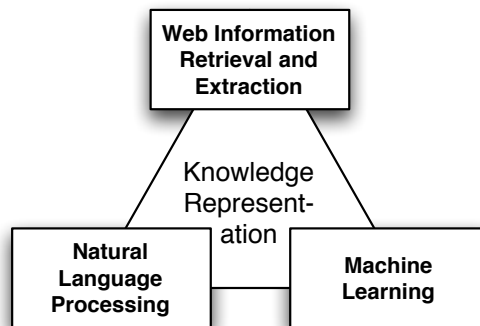


Figure 2.1: Overview over the related research areas

### 2.1 Preliminaries

This section explains briefly common approaches in the field of knowledge representation, information retrieval, natural language processing and machine learning, which serve as foundation for our system.

### 2.1.1 Knowledge Representation

*Knowledge Representation* determines the form in which data is stored and processed. For this thesis, that representation is important in two ways: (1) as input, that is, how the source information is represented and to some degree (2) as output, that is, how to store extracted data. Although there are many ways to represent knowledge [3, 39, 43], this section focuses on representations that are relevant for this thesis.

#### Web Information Sources

In the field of *Web Information Retrieval* (Section 2.1.2), sources of information are mainly categorized by their degree of structured-ness and semantics, into three classes of information sources.

**Unstructured Information Sources** A source, that provides no machine-readable structure to gain any semantic information, is considered unstructured. The only way to extract that information is *Natural Language Processing* (Section 2.1.4). Due to the huge number of plain text documents or documents, which structure can't be inferred, this is the most important source of information to deal with in our thesis.

**Semi-Structured Information Sources** A semi-structured source contains much unstructured content, which is wrapped in some extractable, structure-providing facility, such as HTML tags. Semi-structured documents may also contain meta-information, such as the date of creation or the author. Most documents on the Web that are considered semi-structured are either feeds or HTML pages.

**Structured Information Sources** A source is considered structured, if it can be fully parsed and understood by a machine. Therefore structured data requires a schema (meta-information) that describes the data. Common sources for structured data are XML<sup>1</sup> (eXtensible Markup Language) files with a defined schema or databases.

#### Knowledge Base

To find and extract information from the Web, some extraction systems use previous domain knowledge about the concepts and attributes defined in an ontology and entities, facts and relationships stored in a knowledge base.

---

<sup>1</sup><http://www.w3.org/XML>, accessed 31/01/2010

*Ontologies* represent concepts within a certain knowledge domain and make the relationships between those concepts explicit. Ontologies enable to separate instance data from the schema. The data structure and the actual data or knowledge are stored apart from each other.

The schema of an ontology and instance data is called a *Knowledge Base*. Since it is not the focus of this thesis to deal with ontologies in great detail, we refer to further literature [38].

### 2.1.2 Web Information Retrieval and Extraction

This section surveys the field of *Information Retrieval and Extraction* in general and the retrieval and extraction tasks for the Web in particular. The following paragraph presents the different terms and research fields, in the area of automated information aggregation on a set of sources.

**Information Retrieval** The term *Information Retrieval* (IR) refers to the the task of helping users to find information that matches their information needs [30].

A typical IR system operates on a collection of documents, where the input is a *query* and the output is a ranked list of documents. The documents in the collection are organized through an index, which is used to efficiently route queries to the correct documents.

The typical components of an IR system are: (1) the document index, (2) the query processing, (3) the document retrieval, (4) the document ranking. As a representative for a well known IR system, we mention Lucene<sup>2</sup> here.

**Web Information Retrieval** *Web Information Retrieval* (WIR) is a special form of IR, where the document collection is the Web and the documents are web pages. Furthermore, a WIR system uses a crawler component, to find documents on the Web based on their hyperlink structure. In literature, a WIR systems is also referred to as a *search engine* or *web search engine*.

The world's probably best known WIR systems are Google, Yahoo, and Bing. But also an open source system, called Nutch<sup>3</sup> should be mentioned here.

---

<sup>2</sup><http://lucene.apache.org>, accessed on 20/03/2010

<sup>3</sup><http://lucene.apache.org/nutch>, accessed on 20/03/2010

**Information Extraction** In contrast to information retrieval, *Information Extraction* (IE) is the process of extracting information to a given target structure such as a template or an ontology. The IE tasks are defined by an input (such as an HTML page) and an output (such as a populated database).

**Web Information Extraction** Similar to the analogy of WIR, a *Web Information Extraction* (WIE) system is an adapted IE system for the special requirements of the Web, to deal with semi-structured content [13].

**Open Information Extraction** *Open Information Extraction* (OIE) is a special type of IE system, which was first proposed in 2007 by Banko et al. [6]. The main idea of OIE is to extract unstructured, textual statements in the form of <SUBJECT, PREDICATE, OBJECT> from unstructured text.

State-of-the-art IE/IR systems will be discussed in Section 2.2.4.

### 2.1.3 Evaluation of Web Information Retrieval

This section presents common evaluation measures for the effectiveness of Web Information Retrieval and Extraction systems.

The standard evaluation measures of a information retrieval system revolve around the notion of *relevant* and *non-relevant* documents. The documents in the test collection are classified either relevant or non-relevant, which is referred to as *gold standard* judgement. Relevance is assessed relative to an *information need*, not a query.

#### Evaluation of Unranked Retrieval Results

The standard measures for comparing unranked result sets, which are used in IR and IE systems, are precision (Equation 2.4), recall (Equation 2.2) and their combination which can be expressed in the F-Score (Equation 2.3).

**Precision** *Precision* is the fraction of retrieved results that are relevant.

$$Precision = \frac{\text{relevant retrieved}}{\text{retrieved}} = \frac{TP}{TP + FP} \quad (2.1)$$

**Recall** *Recall* is the fraction of relevant results that are retrieved.

$$Recall = \frac{\text{relevant retrieved}}{\text{relevant}} = \frac{TP}{TP + FN} \quad (2.2)$$



**F-Score** The combination of precision ( $p$ ) and recall ( $r$ ) leads to a weighted *F-Score* [50]. The parameter  $\beta$  can be used to weight the importance of either precision or recall. The higher  $\beta$  the more importance is given to precision instead of recall.

$$F_{\beta} = (1 + \beta^2) \frac{p r}{\beta^2 p + r} = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta^2 FN + FP} \quad (2.3)$$

With respect to a given query, the documents can be partitioned into four sets, as shown in Table 2.1.

	<b>Relevant</b>	<b>Non-relevant</b>
<b>Retrieved</b>	true positives (TP)	false positives (FP)
<b>Not retrieved</b>	true negatives (TN)	false negatives (FN)

Table 2.1: Partitioning of IR documents, with respect to a given query

### Evaluation of Ranked Retrieval Results

Precision and recall are measures of sets. In a ranked list, which is used in a WIR or web search system, we can measure the precision at each *recall point*. Recall increases when a relevant document is retrieved.

**Precision Recall Curve** There is a tradeoff between precision and recall. Increasing recall to retrieve more results decreases precision. The *Precision-Recall-Curve* shows the retrieval performance at each point in the ranking.

**Precision at  $k$**  *Precision at  $k$*  ( $P@k$ ) shows the precision at  $k$  documents retrieved. This is useful, for example if you look at first search result page. Table 2.2 shows an example including precision at each recall point.

$$P@k = \frac{\sum_{i=1}^k \text{relevant?}(i)}{k} \quad (2.4)$$

**Average Precision** The *Average-Precision* (AP) is the average of the precision values for the set of top  $k$  documents existing after each relevant document is retrieved. If document at rank  $k$  is not relevant, its skipped. This value is calculated per query ( $q$ ).

$$AP_q = \frac{\sum_{i=1}^k P@i * \text{relevant?}(i)}{|\text{relevant}|} \quad (2.5)$$

Rank	Relevant	P@ <i>k</i>	AP
1	NO	0	0
2	YES	0.5	0.5 = 0.5 / 1
3	NO	0.333	0.5 = 0.5 / 1
4	YES	0.5	0.5 = (0.5 + 0.5) / 2
5	NO	0.4	0.5 = (0.5 + 0.5) / 2
6	NO	0.333	0.5 = (0.5 + 0.5) / 2
7	YES	0.429	0.476 = (0.5 + 0.5 + 0.429) / 3
8	YES	0.5	0.483 = (0.5 + 0.5 + 0.429 + 0.5) / 4
9	YES	0.556	0.497 = (0.5 + 0.5 + 0.429 + 0.5 + 0.556) / 5
10	NO	0.5	0.497 = (0.5 + 0.5 + 0.429 + 0.5 + 0.556) / 5

Table 2.2: Partitioning of IR documents, with respect to a given query

**Mean Average Precision** The *Mean-Average-Precision* (MAP) provides the average precision for multiple queries.

$$MAP = \frac{\sum_q AP_q}{|Q|} \quad (2.6)$$

**(Root) Mean Squared Error** The *Mean Squared Error* (MSE) measures the *mean difference* between actual (a) and predicted (p) values of data instances (i). Also often used is the *Root Mean Squared Error* (RMSE).

$$MSE = \frac{1}{n} \sum_{i=1}^n (a_i - p_i)^2 \quad (2.7)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (a_i - p_i)^2} \quad (2.8)$$

### 2.1.4 Natural Language Processing

The task of *Natural Language Processing* (NLP) is a collective term for a bunch of technologies that deal with processing natural language. A typical NLP scenario consists of a chain of NLP subtasks. This section gives a short overview of the relevant technologies as well as metrics of readability of text.

**Tokenization** NLP deals with unstructured (raw) text. A series of characters without semantic. For most NLP tasks the next larger unit are words. Tokenization transforms a series of characters into a series of tokens, which in most cases, are words.

**Sentencing** The next larger unit are sentences. Sentencing splits a series of tokens into a series of larger tokens or sentences by detecting sentence boundaries.

**Part Of Speech Tagging** In natural language each sentence is made up of a series of words. In each language there exists a set of rules for constructing a sentence, the so called grammar. Part Of Speech Tagging assigns each token its grammatical role in the sentence.

**Phrase Extraction** Phrase Extraction refers to the automatic selection of important words that best describe the contents of a document. Phrase Extraction is often used in page summarization techniques.

**Named Entity Recognition** Named Entity Recognition or Entity Extraction refers to the task of identification of known entities or concept-tokens in the text.

**Co-Reference Resolution** Co-Reference Resolution refers to the task of finding indirect mentionings of named entities in the text. For example, “Frankfurt is a city in Hesse, *it* is the financial center of the country.”

### Readability Tests

Readability tests are formulas for evaluating the readability of text by counting syllables, words, and sentences. This section presents common readability indices, which will be used later in this thesis.

The *Flesch-Kincaid Grade Level*, *Gunning Fog Index*, *SMOG Index*, and *Coleman-Liau Index* produce an approximate representation of the US grade level needed to comprehend the text.

**Flesch-Kincaid Reading Ease** The *Flesch-Kincaid Reading Ease* (Flesch) test rates text on a 100 point scale. The higher the score, the easier it is to understand the text. A score of 60 to 70 is considered to be optimal, as it is easily understandable by 13- to 15-year-old students.

$$Flesch = 206.835 - 1.015 * \frac{wordcount}{sentencecount} - 84.6 * \frac{syllablecount}{wordcount} \quad (2.9)$$

**Gunning-Fog Score** The *Gunning-Fog Score* (FOG), developed by Robert Gunning, indicates the number of years of formal education a reader would need to understand the text on the first reading.

$$FOG = \left( \frac{wordcount}{sentencecount} + 100 * \frac{longwordcount}{wordcount} \right) * 0.4 \quad (2.10)$$

**Flesch-Kincaid Grade Level** The *Flesch-Kincaid Grade Level* (Kincaid) translates the 100 point scale to a U.S. grade school level. So a score of 8.0 means that the document can be understood by an eighth grader. A score of 7.0 to 8.0 is considered to be optimal.

$$Kincaid = \left( 0.39 * \frac{wordcount}{sentencecount} \right) + 11.8 * \frac{syllablecount}{wordcount} - 15.59 \quad (2.11)$$

**Automated Readability Index** The *Automated Readability Index* (ARI) is a readability test designed to gauge the understandability of a text.

$$ARI = \left( 4.71 * \frac{lettercount}{wordcount} \right) + 0.5 * \frac{wordcount}{sentencecount} - 21.43 \quad (2.12)$$

**Coleman-Liau Index** The *Coleman-Liau Index* (Coleman) is a readability test designed by Meri Coleman and T. L. Liau to gauge the understandability of a text.

$$Coleman = \left( 5.89 * \frac{lettercount}{wordcount} \right) - 0.3 * \frac{sentencecount}{wordcount} - 15.8 \quad (2.13)$$

**SMOG Index** The *Simple Measure of Gobbledygook Index* (SMOG) is a readability formula that estimates the years of education needed to completely understand a piece of writing.

$$SMOG = 1.043 * \sqrt{longwordcount * \frac{30}{sentencecount}} + 3.1291 \quad (2.14)$$

All these indices have the same issue. Since they do not directly take syntactic or semantic complexity into account, they are not considered definitive measures of readability.

### 2.1.5 Machine Learning

This section briefly introduces the main concepts of machine learning, and highlights certain aspects that are relevant to this thesis. This is not a complete overview, therefore we refer to the literature dedicated to machine learning [14].

Machine learning algorithms are organized into a taxonomy, based on the desired outcome of the algorithm. One machine learning paradigm is *supervised learning*, which is explained as follows.

*Supervised Learning* refers the task of “learning” a target function and predict an outcome, based on a previously trained prediction model. Figure 2.2 illustrates the basic learning process, where (1) training data is used by (2) a learning algorithm to generate (3) a classification or prediction model. Given (4) a set of test data, (5) the accuracy of the trained model can be evaluated.

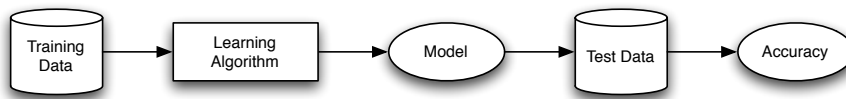


Figure 2.2: The basic supervised machine-learning process

To evaluate the performance or accuracy of a trained model, typically the data are split into three disjoint subsets: (1) training data (2) validation data, and (3) testing data. Each of these subsets play a different role in the design process. All three data sets have to be representative samples of the data that the model will be applied to.

The *training set* (seen data) is used to build the model (determine its parameters). The *test set* (unseen data) is used to measure its performance (holding the parameters constant). The *validation set* is used to tune the model (e.g., for pruning a decision tree). The validation set can’t be used for testing (as it’s not unseen).

The division of the data into three subsets could be done in many different ways. For our purposes we use *cross-validation*, where the idea is (1) to split the whole data set in  $k$  folds (subsets) of equal size, (2) train the model on  $k - 1$  folds, use one fold for testing, (3) repeat this process  $k$  times so that all folds are used for testing and (4) compute the average performance on the  $k$  test sets. Cross-validation maximizes the use of the data.

While a classification model uses a set of *features* to predict which *class* label shall be assigned to a data instance, a similar method called *regression* calculates a continuous output variable, so that a certain approximation error is minimized. This is especially useful for our ranking component, where we get a fine grained regression score rather than a true/false prediction.

## 2.2 Related Work

This section reviews related work, state-of-the-art technologies, and systems in the field of entity-centric context selection and information extraction.

There is actually to the best of our knowledge no such system that provides a solution for our research and problem statement in this thesis.

However, certain technologies and research areas are related to various aspects of our proposed system. We describe that related work in this section and explain how a certain approach differs from ours.

### 2.2.1 Web Carnivores

Etzioni coined this colorful phrase in [19]. In this analogy, Web pages are at the bottom of the Web information food chain. Search engines are the herbivores of the food chain, grazing on Web pages and regurgitating them as searchable indices. Carnivores sit at the top of the food chain, intelligently hunting and feasting on the herbivores.

The carnivore approach leverages the significant and continuous effort required to maintain a world-class search engine (crawling, scrubbing, de-spamming, parsing, indexing, and ranking). In a search context, the carnivore approach is applicable when standard web search engines are known to contain the documents of interest but do not return them in response to naive queries. In literature, this pattern is also referred to as *meta-search* or *federated-search*.

In the past years this architecture style became more and more popular. For example, Yahoo recently outsourced its search engine to Bing and builds its search products as carnivores on top of Bing. Even for real time search this is interesting, because Google and Bing, both started to real-time index Twitter. Some of those carnivore implementation are relatively simple, since they just forward queries without leveraging an extensive query rewriting strategy, or try to leveraging multiple information in parallel. Nevertheless, we see that is going to happen gradually. For example, one can envision the usage of Twitter for related tweets given an input document. In that case a carnivores have to extract key concepts from that articles and devise smart query rewriting rules to harvest the most relevant tweets using the Twitter API, since no direct access to an index is available.

### Iterative, Filtering Meta-Search (IFM)

*IFM* is a concrete embodiment of the carnivore architecture style [28]. IFM generates multiple subqueries based on the user query and appropriate terms from the search context, uses a standard search engine to answer these subqueries, and re-ranks the results of the subqueries using *rank aggregation* methods.

*Rank Aggregation* is the problem of combining several ranked lists in the best way possible into a single ranked list. A survey of rank aggregation algorithms is presented in [16].

Our approach of source aggregation is based on the idea of *Iterative, Filtering Meta-Search* (IFM). The difference is, that our query generation phase is quite different to that described in IFM. Furthermore, we use multiple search engines as carnivores. Our goal is to leverage IFM to come up with great candidate source documents that comprise the given entity in context. Instead of crawling the Web ourselves and do all the scrubbing and pre-processing to find, locate, and use those document sources, the existing web search engines will do this laborious and tedious work for us.

### 2.2.2 Word Sense Disambiguation

*Word Sense Disambiguation* (WSD) is the NLP task of identifying which sense of a word is used in a given text, when the word has multiple distinct senses. It tries to solve the ambiguity of words.

WSD is considered an AI-complete problem, that is, a task whose solution is at least as hard as the most difficult problems in artificial intelligence. Current approaches can be classified into supervised, unsupervised, and knowledge-based approaches, which are survey in [37].

Our system performs word sense disambiguation to increase relevancy during its *Source Aggregation Process*, to only retrieve on-topic documents as basis for snippet extraction. This presents a different approach, since we're using meta-search and rank aggregation for disambiguation. The details of this approach are discussed in Section 3.2.

### 2.2.3 Text Summarization

The research area our system probably is most related with, is the field of *Text Summarization*, a NLP technique to solve the problem of *information overload*.

The research on text summarization dates back to the 1950s, when Luhn [31] and [17] proposed the first automated summarization techniques. Later on, the first linguistic approaches were proposed between 1961 and 1979 and the first artificial intelligence approaches were presented in the 1980's.

There are two major forms of text summarization techniques, (1) *extraction* of representative paragraphs, sentences, and phrases and (2) *abstraction*, “a concise summary of the central subject matter of a document” [41]. Furthermore there two dimensions (Single-document vs. multi-document) and two contexts (Query-specific vs. query-independent) for text summarization.

Related work in the fields of Traditional approaches [1, 9, 10, 24, 45], Multi-document summarization: Summarizing differences and similarities across documents [8, 34, 42], Knowledge-rich techniques [24, 47], Recent approaches [2, 7, 22, 23, 25, 29, 46] Further readings on the topic are presented in [32, 33]

The remainder of this section describes two state-of-the-art systems, that are to some degree related to our system.

### SenseBot

*SenseBot*<sup>4</sup> is a semantic search engine that generates a text summary of multiple web pages on the topic of the users search query and presents those along with a cloud of concepts above the summaries.

SenseBot parses top results from the Web and prepares a text summary of them. The summary serves as a digest on the topic of the query, blending together the most significant and relevant aspects of the search results. SenseBot uses text mining to parse Web pages and identify their key semantic concepts and multi-document summarization to extract the “sense” of the web pages and present it to the user.

This approach is similar to our system, since it delivers a summary in response to a users search query instead of a collection of links to Web pages. This way the summary itself becomes the main result of each search. The major differences are, that they operate in real-time and therefore only provide summaries for only the top 10 search results of only one search engine. Furthermore, their system doesn't provide any de-duplication or quality ranking

---

<sup>4</sup><http://www.sensebot.net>, accessed on 20/04/2010



component.

### Cpedia

*Cpedia*<sup>5</sup> is an automated encyclopedia from and based on the *Cuil*<sup>6</sup> search engine. The main idea is to automatically generate a report or summary of the searched topic instead of a list of ranked web results.

Cpedia algorithmically summarizes and clusters the ideas on the web for each search query and uses this to generate a report. Furthermore, they remove repetition and duplicated content. Similar to Wikipedia, they try to combine all the documents written about an idea on the web to generate one article.

This approach is similar to our system, because it algorithmically discovers web documents and extracts the most interesting information snippets for each topic of interest. The difference is, that it then combines the extracted snippets to one report, which is presented to the user.

## 2.2.4 Web Information Extraction

The field of *Web Information Extraction* was already discussed earlier in this chapter. This section presents state-of-the-art WIE systems and shows how those systems are related to our system and what the differences are.

### DBpedia

*DBpedia* [4] is knowledge extraction framework, which goal is, to publish structured data extracted from Wikipedia. DBpedia allows to ask sophisticated queries against Wikipedia data, and to link other data sets on the Web to Wikipedia data.

Wikipedia is one of the biggest, freely-available entity-centric sources of information worldwide. The main difference between DBpedia and our system is, that they use a single source of information (Wikipedia) and publish their data in RDF. Furthermore, this is a machine-centric approach, which does not try to analyze the quality of the data.

---

<sup>5</sup><http://www.cpedia.com>, accessed on 20/04/2010

<sup>6</sup><http://www.cuil.com>, accessed on 20/04/2010

## WebKnox

*WebKnox* [49] is a domain independent, self-supervised system that automatically extracts entities and facts from the Web at high precision and recall.

The entity extraction component extracts entities using concept names from their ontology. They describe three extraction techniques, which use the semi-structure of web documents. (1) *Phrase extraction*, which queries search engines and extracts patterns such as `CONCEPT such as`. (2) *Focused crawl extraction*, which queries search engines with patterns such as `list of CONCEPT` and extracts entities from list structures and tables. (3) *Seed extraction*, which queries search engines with patterns such as `ENTITY1 ENTITY2` and extracts entities from list structures.

The fact extraction component extracts facts using single- and multi-attribute queries such as `ENTITY + facts` or `the ATTRIBUTE of ENTITY is` and also explodes the semi-structure of web documents. They use (1) tables, (2) colon patterns, (3) phrases and (4) free text for extraction.

For both, entities and facts, they describe algorithms to calculate a trust value for each extraction. In the case of entities they use an entity trust voting algorithm that makes use of extraction graph, which is similar to *PageRank* [40] and *AprioriRank* [20]. They take the entity trust votes it back to its source pages. For facts they use cross-validation across multiple occurrences on different web sites.

The main difference to our system is, that *WebKnox* focuses on extracting machine-readable entities and “hard”-facts, which are facts where one true value exists. Our system focuses on user-centric, “soft-facts”, which are represented as snippets of text. The goal of our system is, to find snippets which are *interesting* to the user, even if there is no say about this statement is either true or false.

## TextRunner

*TextRunner* [6] is an OIE system, which performs statement extraction with a precision of over 80%. For example, from the sentence “The iPhone 3GS even comes with a 3MP camera and has video capability as well.”, *TextRunner* can extract the entity-relation tuple (`iPhone 3GS, comes with, 3MP camera`).

It takes a corpus of documents as input and extracts the information in a single pass. (1) The noun phrases of the sentence are tagged, (2) nouns that

are close to each other are put into a candidate tuple set, and (3) the tuples are analyzed and classified as true or false using a self-supervised classifier [52].

*TextRunner* stores the extracted information in natural language without any semantic annotation or knowledge. It does not “know” what it extracts. Its not able to answer questions such as “List all features of the iPhone 3GS”.

With its statement extraction approach, *TextRunner* is the IE system that is closest related to our approach. The difference is, that our system extracts full, user-centric sentences that are interesting to the human reader. *TextRunner* on the other hand focuses on extracting as many relations as possible at high precision, but with no regard to interestingness.

### 2.2.5 Near Duplicate Elimination

The Web contains multiple copies of the same content. By some estimates, as many as 40% of the pages on the Web are duplicates of other pages. Many of these are legitimate copies; for instance, news sites, that don’t have exclusive material, present content provided by news agencies. Many websites use the same agencies and therefore present the same content.

The simplest approach to detecting duplicates is to compute, for each web page, a fingerprint that is a succinct digest of the characters on that page. Then, whenever the fingerprints of two web pages are equal, its tested whether the pages themselves are equal and if so declare one of them to be a duplicate copy of the other. This simplistic approach fails to capture a crucial and widespread phenomenon on the Web: near duplication.

The task of *Near Duplicate Elimination* is to identify these near-duplicates. There are various methods to do so. Elmagarmid et al. [18] present an overview of common duplicate elimination techniques. The most prominent ones are: (1) edit distance, which is good for very short texts, (2) shingling [11], which is good for long texts, and (3) semantic distance, which uses previous- or domain-knowledge, to determine the “semantic distance” between two texts.

Our system deals with a special form of de-duplication, since we are interested to determine the new information gain for our snippets. Another specialty is, that our snippets are composed of short text segments. Metzler et al. [35] present a technique, that especially focuses on “Similarity Measures for Short Segments of Text”.

# Chapter 3

## Design

This chapter describes the design of the *WebSnippets* system for entity-centric snippet extraction from the Web.

First, an overview of the system is given. We then describe the source aggregation process followed by a description on how the retrieval of the web pages works. Then we present the snippet extraction process in more detail, and go over the de-duplication process. Finally, we are reviewing how the snippet ranking process works and show how and which snippets are stored persistently.

The *WebSnippets* system uses techniques that have been earlier described in the research literature and introduces new techniques in snippet extraction processes. The goals are to show how different techniques can work together and complement each other and to introduce novel techniques that have not been described in the literature yet. Existing techniques are embedded in the design as they were described in literature and only small changes are made to fit the design of *WebSnippets*.

The main contributions of the *WebSnippets* system are pointed out in the particular sections and a thorough evaluation is shown in a later chapter. It is not the aim of this chapter to use all possible aggregation, extraction and ranking mechanisms that are possible in every combination, but to show one embodiment that solves the problems depicted in the problem statement. The design is guided by the requirements and research questions that motivate this thesis paper.

### 3.1 System Overview

The *WebSnippets* system comprises four components. Figure 3.1 shows an overview of the data and processes of WebSnippets.

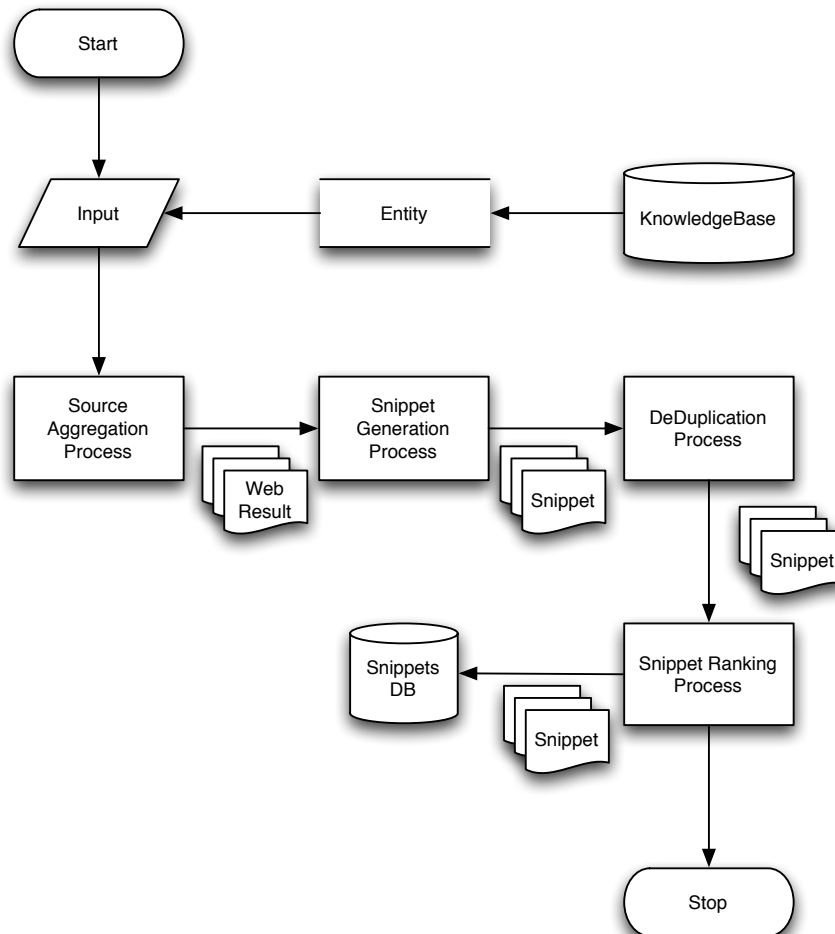


Figure 3.1: Flow-diagram system-design

The *source aggregation process* receives entities from the knowledge base as input, retrieves a ranked list of web results from search engines, and passes the list to the snippet extraction process.

The *snippet extraction process* takes the ranked list of web results from the source aggregation process as input, retrieves the related web pages (possibly need to fetch or obtain a cached version), and extracts a set of snippet can-

didates. The candidate set of snippets is then passed to the de-duplication process, including their related web results and entities. The overall snippet extraction process is an off-line process, so no particular requirements are given in terms of latency and throughput.

The *de-duplication process* takes the snippet candidates from the snippet extraction process as input, eliminates duplicates (possibly also near duplicates), which are already stored in the database. The snippets are then passed to the snippet ranking process.

The *snippet ranking process* takes the set of extracted snippet candidates from the de-duplication process as input, extracts certain quality parameters, and uses these parameters for regression-based ranking on a trained model. In a final step it saves the remaining scored snippets that are not duplicates in the database.

The source aggregation process, the snippet extraction process, the de-duplication process and the snippet ranking process are explained in more detail in the following sections. All processes may request additional data from external sources.

## 3.2 Source Aggregation Process

The *Source Aggregation Process* takes an entity as input, selects a set of search engines to query, retrieves domain knowledge from the knowledge base and generates queries, sends the queries to the search engines, re-ranks the results according to rank aggregation algorithms and passes an ordered list of web results to the *De-Duplication Process*.

The *Source Aggregation Process* is implemented as a web carnivore and the design of this process is guided by the idea of IFM and Meta-search discussed in Section 2.2.1.

Figure 3.2 shows the components in greater detail.

The design of the Source Aggregation Process is discussed in the following subsections in greater detail starting with the selection of the appropriate search engines, followed by the query generation component, finally describing the rank aggregation component.

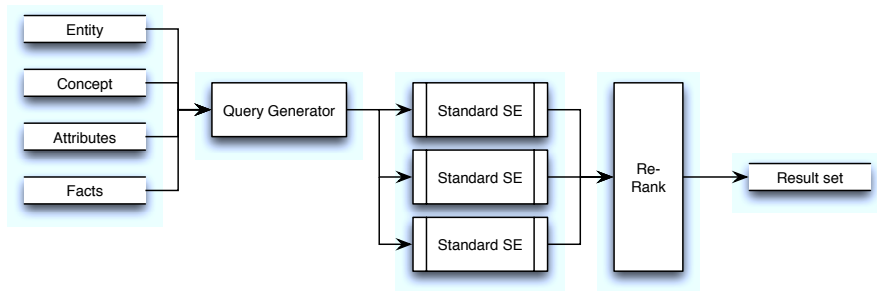


Figure 3.2: Flow-diagram of the source aggregation process

### 3.2.1 Search Engine Selection

The first subtask of the Source Aggregation Process is the selection of the search engines for the current run.

In our analogy the first task of our “carnivore” is to choose his “herbivores”, where we choose the herbivores which offer the most “meat”.

Each search engine or information provider will have a common wrapper and API so that there is a standardized way of accessing those. Initially we’re hand-selecting a set of search engines where we expect good recall (e.g., Google, Google Blog Search, Bing, ...).

Future work will be to have the carnivore to intelligently choose the indices to query (e.g., depending on the change frequency of results lists, quality of and quantity of extracted snippets for a given source.)

### 3.2.2 Query Generation

The Query Generator component implements the template approach of IFM to build a set of queries that can be send to the search engines.

A query typically comprises the entity name plus a list of modifiers. The modifiers are related concepts, names, or synonyms of domain knowledge that is stored in the knowledge base.

Modifiers are primarily used to further focus the queries and help with the disambiguation task. For instance, an entity “Washington” alone is ambiguous. However, if there is a modifier like “place” then both combined as a simple

*AND* query will result in search results more biased towards the desired entity.

There are different approaches on how context can be added to these entities and transformed into search queries (see Kraft et al. [28] for a more detailed study on this topic.) The modifiers are constructed or collected editorially. For example, an editor could build a list of all travel locations in Germany and associate a few modifiers for this list. List building effort is not done by us, but by the users of the system. The extra editorial effort needed for our system is to provide modifiers either per list or per entity.

In our experiments the former works well and requires very minimal labeling work. We suggest adding modifiers per entity only in very ambiguous cases where that extra effort would help to increase the document selection process significantly. This typically can be achieved by having editors spot check a large list of entities and looking at problematic terms, or combined with algorithmic tooling that can scan entities for ambiguity. We think this is an interesting area for future work.

In general the idea of using co-occurrences of an entity and a concept modifier and use this as an input for search engines to bias results and therefore help with the disambiguation task itself is not new. Standard search users on the Web doing this manually every day when they google for their information needs. Using a query plus context itself also has been explored in the aforementioned paper in depth.

The novelty of our approach here is to leverage contextual search in a different setup to avoid having to do complex word-sense disambiguation downstream in our system in the filtering part. We therefore elegantly leveraging the search engine's algorithms to return the most relevant results for a given entity plus modifier, and also leveraging all the work of thousands of engineers and scientists who are tuning these major search engine on a daily basis.

### 3.2.3 Ranking and Filtering

We utilize rank aggregation technologies [16] that take as input a set of lists from different search engine providers and produce one aggregated ranked list of results. Ranking at this level is mostly useful for prioritization for the snippet extraction process to ensure the most important documents with promising snippets are extracted first. The actual ranking of snippets itself is described later as part of the snippet ranking process.



### 3.3 Snippet Extraction Process

The *Snippet Extraction Process* takes as input a ranked list of web results from the source aggregation process, retrieves the URLs via HTTP or from a cache and pre-processes those as a preparation step. Finally snippet candidates are being extracted, which surround a given entity. The snippet extraction process from Figure 3.1 is shown in more detail in Figure 3.3.

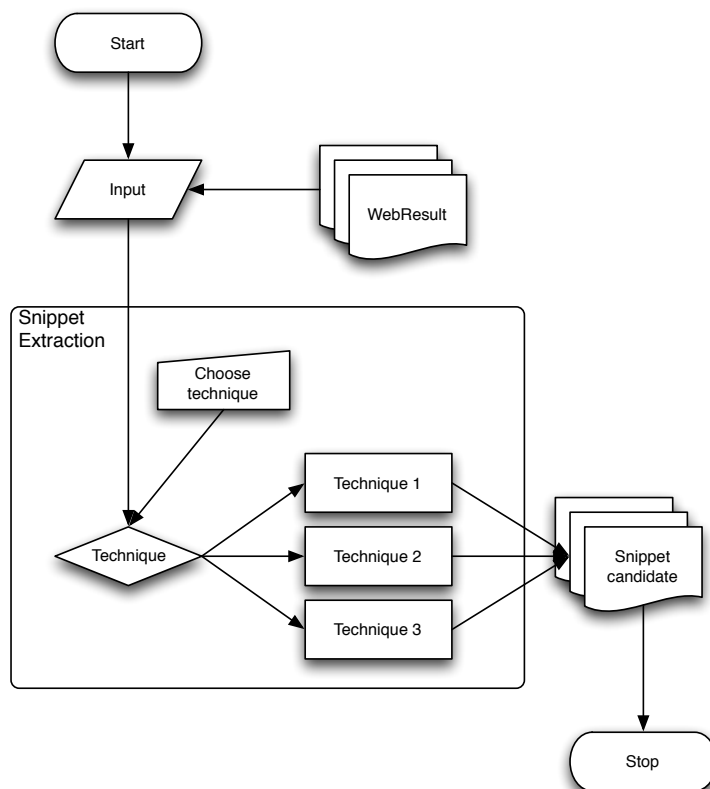


Figure 3.3: Flow-diagram snippet-extraction-process

The snippet extraction process can use different techniques to extract or generate the snippet candidates. These techniques are *Webresult Summary (WRS)*, *Document Sentences (DS1)* and *Document Snippets (DS2)*.

All these techniques have in common that they get the web results from the source aggregation process as an input and that they extract snippet candidates from the content related to the web result. What that related content is, depends on the search engine it has been queried from and the extraction tech-

nique used. This will be explained in more detail in the remainder of this section.

Each extraction technique consists of three steps. First, the *related content retrieval* step, which retrieves the raw-content related with the search result. Second, the *pre-processing* step, which extracts the main content block and removes unwanted characters and strings. And third, the *snippet extraction* step, which finally extracts the snippet candidates. The extracted snippet candidates are then passed to the feature extraction process.

The extraction techniques are explained in detail in the following sections.

### 3.3.1 Web Result Summary

The *Web Result Summary* (WRS) technique is a very simple one. It uses the summary field retrieved by the search engine. This represents the impressions Google would offer, if you would just enter the search term and read the provided summaries. This technique served as baseline for our problem validation experiment, which will be discussed in Section 5.3.

#### Related Content Retrieval

This is a simple technique, which takes the summary attribute of the results retrieved from each search engine.

The summary varies, depending on the type of search engine that was queried. The following list describes in more detail, which data is used as the summary attribute.

**web search engine** In case of a standard web search engine such as Google or Yahoo! the retrieved results contain a *title* attribute, which is the title of the linked web page, a *summary* attribute, which is a query-biased short summary of the linked web page and the *URL* of the linked web page.

**real-time search engine** In case of a standard real-time search engine such as Twitter or Facebook, which provide more like a feed of data, the retrieved results contain possibly a *title* attribute, a *summary* attribute, which is the tweet or feed post and a possibly *URL* to a related webpage.

**statement search engine** In case of a statement search engine such as TextRunner, the retrieved results contain no *title* attribute, a *summary* attribute, which is the extracted statement and a *URL*, which points to the originating document, the statement was extracted from.

### Pre-Processing

Depending on the search engine itself, which was queried, the raw-content of the summary attribute may contain a strong BIAS. For example in Twitter the typical tweet contain a lot of sequences such as RT @user:, #tag or upper-case text. To compare, we remove this BIAS from the summaries in a homogeneous way, to obfuscate the source of the snippet.

The following BIAS is removed from the summary:

BIAS-removal action	Example
remove tabs, line breaks and multiple whitespaces	the iphone
convert to lowercase	THE VERY COOL IPHONE
remove any XML/HTML tags and entities	<tag> or &entity;
remove any re-tweets and twitter tags	RT @user: or #tag
remove any URLs	http://tinyurl.org/nacs89d

### Snippet Extraction

In this simple technique, we don't apply any intelligence on the snippet candidate generation. The summary attribute is removed of any BIAS as shown above and the resulting string is added to the set of extracted snippet candidates, that are passed to the next processing step.

```
String summary = removeBIAS(webresult.getSummary());
Snippet snippet = new Snippet(entity, webresult, summary);
results.add(snippet);
```

Search engines such as Google constantly improve the quality of their web result summaries. In the early years only the description meta tag from the linked web page was shown, if one existed. If the description was missing, the web result summary was left blank.

Later on Google started to apply more intelligence to display a good summary, for example by showing the DMOZ in certain cases was better than the description meta tag provide by the page author. Currently Google shows very precise query-biased summaries of the page (dynamic summaries), focusing on the best passage the query is mentioned.

But all these snippets are still a summary of the content of the web page, and the focus is not to display entity-centric information in-place, such as *WebSnippet* does. To demonstrate the differences we show a best-case and a worst-case

scenario of summary snippets retrieved by a web search engine for the use case, which guides this thesis.

Figure 3.5 shows an example of a good entity-centric snippet, which contains relevant and interesting information about an entity, retrieved by a web search engine.

```
iPhone 3GS is a GSM cell phone that's also an iPod, a video camera,
and a mobile Internet device with email and GPS maps.
```

Figure 3.4: A best-case sample of a web result summary

In contrast, Figure 3.5 shows an example of a bad example of an entity-centric snippet, which contains neither relevant nor interesting information about the entity.

```
iPhone; iPod; Previous iPods. iPhone 3GS iPhone 3GS; iPhone 3G
iPhone 3G; iPhone iPhone; iPod classic 3G iPod classic 3G;
iPod nano 5G iPod nano 5G ...
```

Figure 3.5: A worst-case sample of a web result summary

This technique produces exactly one snippet candidate per web result.

### 3.3.2 Document Sentences

The *Document Sentences* (DS1) technique is more sophisticated than *WRS*.

Based on our assumption that the most relevant and interesting content is buried deep inside the web results (and their content pages), DS1 uses the URL field of each web result to retrieve a related web page or document and extracts all sentences containing the entity from the main content section of the document. These sentences are the snippet candidates.

#### Related Content Retrieval

The related web pages or documents mentioned in the *URL* attribute of each web result are downloaded from the web. This can be optimized for performance reasons to add a cache store.

### **Pre-Processing**

The main component are is extracted from the retrieved document. The main content extraction varies depending on the document retrieved. The main idea is to remove navigation, ads, etc.

### **Snippet Extraction**

The snippet extraction makes intensive use of the NLP processing chain discussed in Section 2.1.4. First, the main content block is segmented into sentence chunks. Then if the sentence contains at least one mention of the entity, it is added to the set of snippet candidates.

This technique produces an uncertain number of snippet candidates per web result, depending on the number of sentences with entity occurrences in the linked document.

### **3.3.3 Document Snippets**

The *Document Snippets* (DS2) technique is a more sophisticated version of *DS1*. DS2 generates less snippet candidates by applying certain filters while snippet extraction. Furthermore the pre-processing step combines the pre-processing steps of WRS and DS1.

### **Related Content Retrieval**

The raw-content retrieval strategy for the related-documents is the same as in DS1. The related web pages are downloaded from the Web.

### **Pre-Processing**

The pre-processing step of DS2 combines the pre-processing steps of WRS and DS1 by first applying WRSs pre-processing on each snippet candidate plus extracting the main content block as described in DS1.

### **Snippet Extraction**

The snippet extraction part of DS2 extends the capabilities of DS1. The improvement is that it applies additional filters on the snippet candidate set by removing snippet candidates of certain patterns. Those patterns were determined by our observations of candidates generated by DS1.

**Trim Sentence** Sentence with leading and trailing whitespaces omitted.

**Sentence Length** Sentence length is greater than 0 characters.

**Regular Sentence** A sentence must contain two nouns and one verb including “to be”, “to have” or “to do”.

**Upper Case Letter** Sentence starts with an uppercase letter.

### 3.4 De-Duplication Process

The *De-Duplication Process* takes as input a set of snippet candidates from the Snippet Extraction Process, applies several de-duplication strategies as filters to reduce the amount of duplicate or similar snippets and passes the rest to the Snippet Ranking Process.

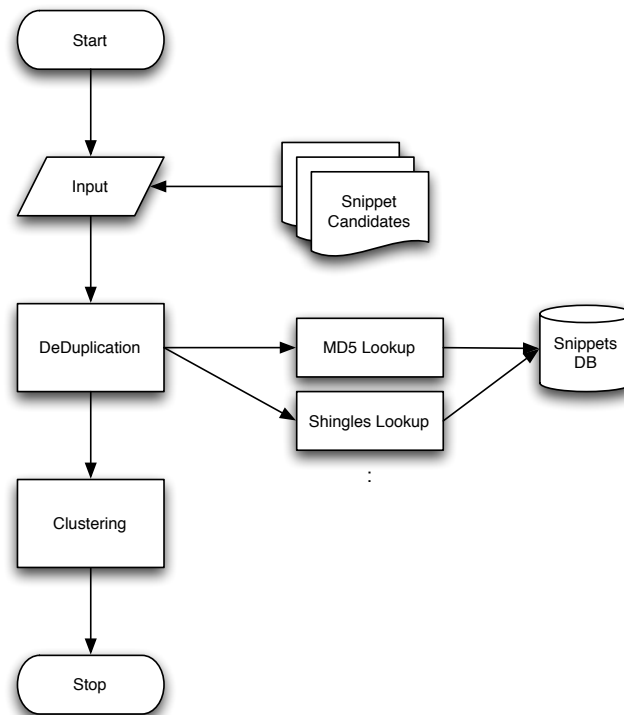


Figure 3.6: Flow-diagram of the De-Duplication Process

Our experiments show, that throughout the Web and especially in news feeds or on news sites, there is much information duplication and re-occurrence. For example, many news sites show the same ticker message from the same news

agency. This resulted in lots of duplicate or similar snippets.

```
Frankfurt is a city in Hesse  
Frankfurt is a big city in Hesse
```

Figure 3.7: Sample of near-duplicate snippets for “Frankfurt”

To avoid showing up or even having the same content stored in the snippet database, we added the de-duplication process. The task of the de-duplication process is to add snippets only to entities that comprise new knowledge. In section 2.2.5 we already discussed various methods to determine new information gain. Figure 3.7 shows an example of near-duplicate snippets for the entity “Frankfurt”.

As a simple strategy for the lookup of exact duplicates, we propose to calculate the MD5 hash of the snippet text and store it in the database together with the snippet. This achieves good performance for looking up the existence of a snippet. To detect near-duplicates we propose a shingling approach, which were already described in the background chapter.

A second strategy we propose is to cluster the snippets by various aspects. Snippets can be clustered by topic based on their similarity-measures or based on the relation, based on our ontology knowledge of the entity and the snippets.

Another aspect is, that by reducing the amount of duplicate and similar snippets, we increase the performance of our system, because we need to score and store less snippets and therefore can lookup snippets more efficient.

The de-duplication process from Figure 3.1 is shown in more detail in Figure 3.6.

## 3.5 Snippet Ranking Process

The *Snippet Ranking Process* takes the set of snippet candidates from the de-duplication process as input and extracts certain features used for determining their quality. There has been lots of research into whats good and bad in terms of text quality. Given the extracted features, we use a machine learning approach to learn, which snippets are the “best”.

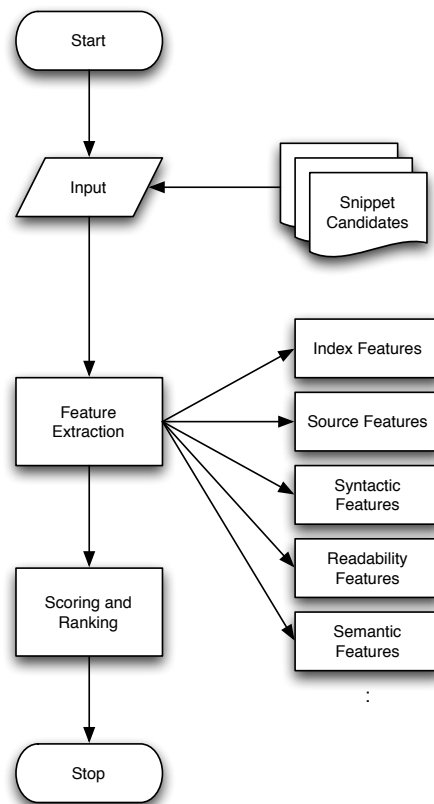


Figure 3.8: Flow-diagram of the Snippet Ranking Process

The snippet ranking process from Figure 3.1 is shown in more detail in Figure 3.8. The ranking process consists of two parts. In the first part a feature vector for regression analysis is constructed. The second part describes the ranking/scoring process in greater detail.

### 3.5.1 Feature Extraction

The feature extraction takes the snippet as input including the aggregated web result and the search engine the web result was retrieved from. Then in a series of extraction steps, it builds up a feature vector based on features extracted, collected and generated from the snippet, its source document and source search engine the web result was retrieved from.

We experimented with several extracted features that we use as indicators to determine the snippet quality. The following subsections describe the features our systems extracts, aggregates or retrieves for each snippet. Furthermore we



explain, why we choose each feature.

### Search Engine Features

First we extract features about the search engine we used to retrieve the web results from.

**Aggregated Rank** The source aggregation process retrieves web results from different search engines and combines them into one ranked list of aggregated results. We use the rank of each aggregated result as feature for each snippet to determine how important the document is in average.

**Search Engines** This is a list of search engines the related document was contained in. Some search engines might return better or worse results. Therefore we use the mentioning in a search engine as indicator.

### Source Document Features

Then we extract features about the source document we used to extract the snippet candidate from.

**Google Page Rank** The Google Page Rank provides information about the source documents prominence in the Web. Very popular sites such as Wikipedia might contain more interesting and valuable information than smaller or spam sites.

**Top Level Domain** The Top Level Domain, such as .com, .edu, .travel might contain relevant information about the class of document, which is provided. For example if it is from a .travel or an educational source (.edu).

**Main Content Character Percentage** The main content character percentage provides information about how many characters of the source document were actually in the main content block.

### Readability Features

Readability features are used to determine the quality of how good a snippet is readable by the user. They were chosen to show the “best” readable snippets to the user. These indices are discussed in section 2.1.4 in detail. Readability features are extracted from the snippet text.

We use the following readability features: The *Flesch-Kincaid Reading Ease*, the *Gunning-Fog Score*, the *Flesch-Kincaid Grade Level*, the *Automated Readability Index*, the *Coleman-Liau Index*, and the *SMOG Index*.

### Syntactic Features

We extract syntactic features of each snippet as indicator for quality. These are mostly text statistics values.

**Character Count** The number of characters of the snippets.

**Letter Number Percentage** The percentage of alpha-numeric characters.

**Capitalized word count** The number of capitalized words in the snippet text.

**Syllable Per Word Count** The average number of syllables per word in the snippet text.

**Word Count** The number of words in the snippet text.

**Unique Word Count** The number of unique words in the snippet text.

**Complex Word Percentage** The percentage of complex words in the snippet text. A word consisting of three or more syllables is considered “complex”.

**Sentence Count** The number of sentences in the snippet text.

**Words Per Sentence Count** The average number of words per sentence in the snippet text.

**Contains proper noun** Whether the snippet text contains a proper noun.

### Semantic Features

To complete the feature vector, we also extract semantic features from each snippet.

**Entity concept** This feature look for the occurrence of the concept name or one of its synonyms of the current entity. For example, “the city of Frankfurt” would count as an occurrence, if the entity was Frankfurt, but “the soccer team Eintracht Frankfurt” would not. The idea is to get more clues for word sense disambiguation and therefore relevancy.

**Starts with entity** This features is 1 if the snippet text starts with the mentioning of the entity. For example, “Frankfurt is the financial center of Germany” would count as 1, while “We like to visit cities such as Frankfurt and Munich”.

**Related entity count** This counts the number of occurrences of entities of the same concept in the snippet. For example, “Munich, Frankfurt, Berlin, Hamburg and Cologne are the biggest cities in Germany.” We use our concept network to lookup the number of entity occurrences in this snippet.

### 3.5.2 Feature Learning and Ranking

The extracted features are used to calculate a regression score per snippet based on a trained regression model. The foundations of machine learning and regression analysis were discussed in section 2.1.5 in detail.

For our system we used a simple linear regression algorithm and trained a model based on a collection of 200 snippets, which have been editorially scored. The model is trained on the overall features of all snippets.

With more samples it is also possible to train multiple models on a concept or on an entity-level. But for our system we trained only one model on a random distribution of snippets from different concepts. This is described in further detail in section 5.2.2 in the evaluation chapter.

After the regression score has been calculated, the snippets are stored persistent in the snippet database and can be retrieved by the end-system ordered descending by their regression score. The persistence is described in more detail in the next section.

## 3.6 Persistence

Extracted snippets as well as their sources are stored in a database. The database is designed as shown in the E/R diagram in Figure 3.9.

The Knowledge Model is the database equivalent for the knowledge ontology. A concept can have many attributes and those can have many facts. A concept can also have many entities and those can have many facts and many snippets.

Entities, facts and snippets can occur on many sources and a variety of entities, facts and snippets can occur on one single source, therefore the N to M relation between entity, fact, snippet and source.

For entities and facts the trust values are stored, which makes it possible to rank the tuples by their trust later on. Similarly, for snippets the regression score is stored which is used to rank the snippet by their quality.

For the training of the regression model described in section , several evaluations are stored in the database for each snippet. These evaluations are provided by human experts.

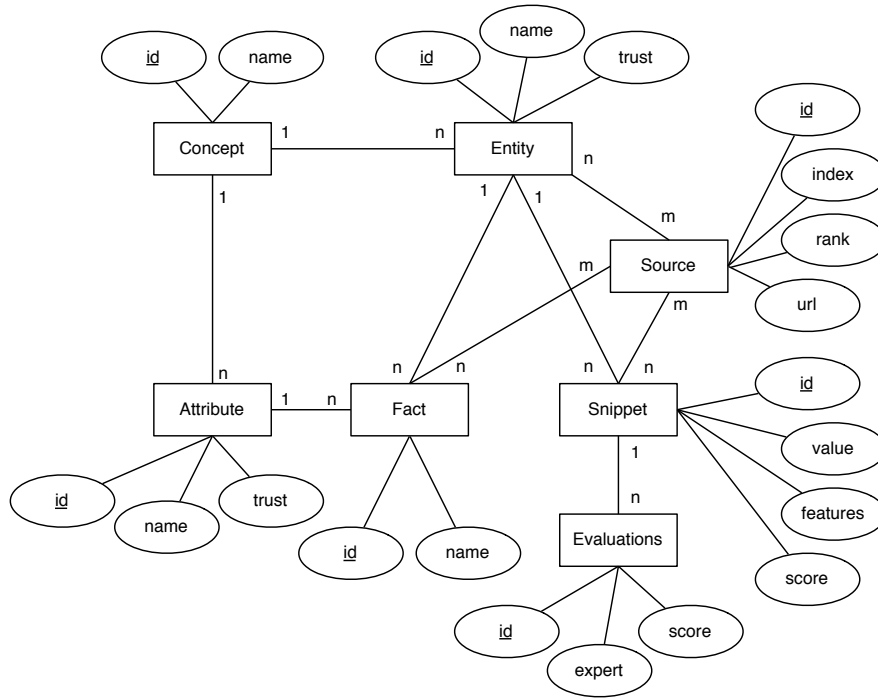


Figure 3.9: Entity Relationship Diagram of the database design

### 3.7 Summary

In this chapter, the design of *WebSnippets* has been presented. The system comprises four components, the source aggregation process, the snippet extraction process, the de-duplication process, and the snippet ranking process, which have been discussed in detail.

The *WebSnippets* system is proposed as a web carnivore. Following this concept we increase relevance by facilitating meta-search and rank-aggregation for disambiguation. Furthermore, we propose three new snippet candidate building techniques. For snippet quality scoring we use a regression-based approach.

Before the designed system is evaluated, details about the implementation are provided in the next chapter.

## Chapter 4

# Implementation

The previous chapter described the algorithms to extract and rank entity-centric snippets. We show now how this can be implemented for a particular embodiment to point out practical and system aspects, as well as performance and scalability issues.

### 4.1 Technology

This section depicts the technology used by *WebSnippets* starting with engineering aspects. In the first part the programming language is discussed, in which the prototype was implemented. The second part introduces the libraries and web services used within the prototype. The third part introduces *WebKnox*, the framework which has been used to implement our prototype.

#### 4.1.1 Development Languages

The prototype of *WebSnippets* is implemented within *WebKnox* (see Section 4.1.3), which has been developed in Java 1.6.

Java is an object oriented programming (OOP) language, which provides a large set of standard data structures, classes and methods, which can easily be extended as further functionality is required. Whenever a functionality is not supported by the standard Java API, a wide variety of free, well-documented packages are available to enhance the standard API of Java.

Java is available under the free GPL open-source license and has a wide platform support. For example, it is available under the major operating system such as Windows, Linux, Mac OSX, and Unix. Java also has a strong developer

community, which publishes a lot of useful libraries to enhance the core functionality of Java. The support regarding tools and documentation is superior to most other OOP languages.

Java seems to be the most often used programming language in current computer science research as many major research frameworks are available in Java first. All these reasons led to the choice of Java over other alternatives.

### 4.1.2 Frameworks and Libraries

The core functionality of Java can easily be extended by including packages from third party developers. *WebSnippets* makes extensive use of those extensions and includes a variety of frameworks and APIs. The most relevant of them are briefly discussed in this section.

**AlchemyAPI** AlchemyAPI is a web service provider<sup>1</sup>, which offers services for performing content analysis and meta-data annotation on web pages and posted HTML pages or plain texts. The service is accessed through a tiny Java library.

For HTML pre-processing and main content extraction of webpages, *WebSnippets* uses a web service called “Text Extraction / Web Page Cleaning” which normalizes HTML content by removing ads, navigation links, other undesirable content, and only extracts the key article or web page text.

**Fathom** Fathom is a Java package<sup>2</sup>, that measures the readability of English text. The package allows to generate common readability indices from English text.

Supported features are Fog Index, Flesch reading ease level, Flesch-Kincaid grade level score as well as common text statistics such as complex words count, syllable count and sentence count, which are discussed in detail in Section 2.1.4. These scores are used in the *WebSnippets* snippet ranking component, which is explained in Section 3.5.

---

<sup>1</sup><http://www.apchemyapi.com>, accessed on 22/11/2009

<sup>2</sup><http://www.representqueens.com/fathom/>, accessed on 22/11/2009

**Google PageRank API** Google PageRank API is a Java package<sup>3</sup>, which retrieves Google Page Rank for any domain from Google Toolbar<sup>4</sup>. Google Page Rank is a numeric value that represents how important a page is on the web [40].

**LingPipe** LingPipe is a Java package<sup>5</sup> for natural language processing. LingPipe provides classes for tokenization, sentence detection, named entity detection (NER), coreference resolution (co-ref), classification, clustering, part-of-speech tagging (POS), general chunking and fuzzy dictionary matching.

*WebSnippets* makes extensive use of chunking, sentence detection, POS, NER and fuzzy dictionary matching.

**Twitter4J** Twitter4J is a Java library<sup>6</sup> for accessing the Twitter web services. It provides a full feature set, which includes retrieving tweets for users, topics and trends. *WebSnippets* uses Twitter4J to retrieve tweets that contain mentions of entities.

**WEKA** WEKA<sup>7</sup> is a collection of machine learning algorithms for data mining tasks. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

WEKA is one of the most relevant and best supported machine learning frameworks in Java. *WebSnippets* uses WEKA in the snippet ranking process, which is explained in Section 3.5, to rank the snippet based on the extracted snippet features and a trained regression model.

**WebKnox** WebKnox<sup>8</sup> is a web information extraction toolkit implemented in Java. *WebKnox* features are described in detail in Section 4.1.3.

---

<sup>3</sup><http://www.temesoft.com/google-pagerank-api.jsp>, accessed on 22/11/2009

<sup>4</sup><http://toolbarqueries.google.com>, accessed on 30/03/2010

<sup>5</sup><http://alias-i.com/lingpipe/>, accessed on 22/11/2009

<sup>6</sup><http://twitter4j.org/>, accessed on 31/03/2010

<sup>7</sup><http://www.cs.waikato.ac.nz/~ml/weka/index.html>, accessed on 22/11/2009

<sup>8</sup>[http://www.inf.tu-dresden.de/index.php?node\\_id=578&refer\\_id=579&refer\\_sID=5&ID=117&ln=de](http://www.inf.tu-dresden.de/index.php?node_id=578&refer_id=579&refer_sID=5&ID=117&ln=de), accessed on 22/11/2009

### 4.1.3 WebKnox

One implementational requirement of the thesis was, that the prototype has to be implemented using *WebKnox* as framework, which is described in [49] in greater detail.

*WebKnox* is an extraction framework of the computer networks chair of the Technische Universität Dresden and part of the Aletheia<sup>9</sup> research project. Figure 4.1 gives an high-level overview of the *WebKnox* research project.

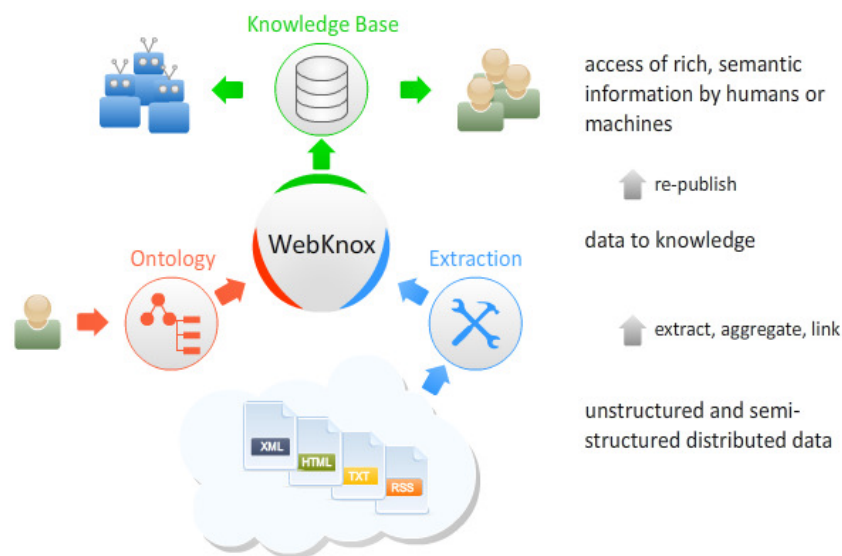


Figure 4.1: WebKnox high-level overview

The *WebKnox* system consists of several core packages that are explained in more detail in this section. Figure 4.2 gives an overview of these packages. The bold packages are extended and the italic packages are used by *WebSnippets* as described in 4.2.

**Control-Package** The control package consists of a controller class that instantiates the graphical user interface and is the entry point for the *WebKnox* core application.

**Graphical User Interface-Package** The Graphical User Interface (GUI) package consists of a GUI manager class that manages the complete layout of the *WebKnox* core application. The interface offers functionality to launch the

<sup>9</sup><http://www.aletheia-projekt.de/>, accessed on 31/03/2010



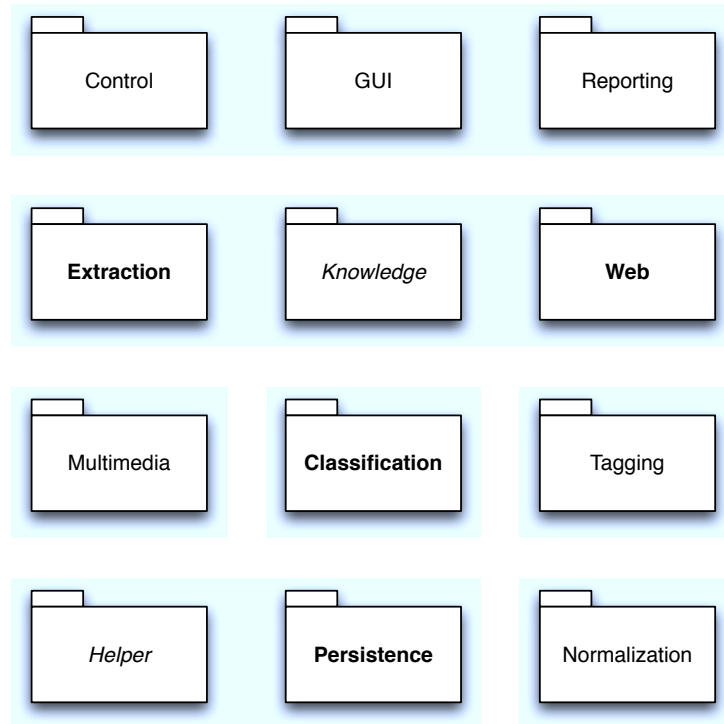


Figure 4.2: Class-diagram of WebKnox packages

extraction processes, performs cleansing and emptying operations on data and knowledge base, creates reports about the extracted data, and shows real-time logging data.

**Reporting-Package** The reporting package consists of classes that provide functionality to generate reports for entities and facts as well as a chart creator class. A report is a list of measures and calculations such as total number of entities and facts, precision and the F1 score.

**Extraction-Package** The extraction package consists of all classes that are part of the extraction process. This includes *Focused Crawl Extraction* for entity extraction, *Page Analysis* and *Extraction Sequence* for fact extraction.

**Knowledge-Package** The knowledge package consists of container classes that keep extracted knowledge in memory, such as *Concept*, *Attribute*, *Entity*, *Fact*, *Snippet* and *WebResult*.

**Web-Package** The web package consists of classes that are the interface between *WebKnox* and the web. These classes are the Crawler, which downloads web pages, the SourceRetriever which is able to retrieve URLs from Google, Haika, Microsoft Live Search and Yahoo!.

**Multimedia-Package** The multimedia package consists of classes that deal with the extraction and handling of images.

**Classification-Package** The classification package consists of classes that deal with machine learning tasks. The abstract *Classifier* class provides mean to operate on data stored in the MySQL database and perform algorithms provided by WEKA on the data.

**Tagging-Package** The tagging package consists of classes that deal with natural language tasks, such as *Named Entity Recognition* (NER).

**Helper-Package** The helper package consists of helper classes that extend the capabilities of Java such as *CollectionHelper*, *ThreadHelper*, and *StringHelper*, and provide useful functions that can be used throughout *WebKnox*. The *Logger* class is also part of this package.

**Persistence-Package** The persistence package consists of classes that read from and write to disk, which includes config files, OWL files, the Lucene index and the MySQL database.

**Normalization-Package** The normalization package consists of the *DateNormalizer*, the *StringNormalizer* and the *UnitNormalizer* classes. These classes are responsible for transforming different fact representations into a common format.

## 4.2 System Architecture

The *WebSnippets* design proposed in the previous chapter has been implemented as a prototype within *WebKnox*, as described in Section 4.1.3. Therefore the package structure of *WebKnox* has been used, and additional classes have been added to fit the needs for the *WebSnippets* system.

Figure 4.2 shows the core packages of *WebKnox*. All packages that are affected of the *WebSnippets* system are highlighted in bold. Other packages used by *WebSnippets* are highlighted in italic. The classes that have been added or

affected by *WebSnippets* are described in more detail in the following subsections.

To streamline the development process, the *WebSnippets* prototype makes extensive use of 3rd party libraries and web services. All full list had been presented in Section 4.1.2.

Figure 4.2 gives an overview of these packages, packages that are not relevant for the implementation of *WebSnippets* have been left out.

### 4.2.1 Persistence

To fit the requirements of *WebSnippets*, the *WebKnox* database schema has been altered, to store additionally required data. This section discusses the tables that have been added in detail.

Each table has a corresponding class which is part of the Knowledge package. It serves as object representation of the data stored in the table.

#### Snippet

First the snippets have to be stored. Therefore we added a snippets table and a corresponding `Snippet` class. The entity class provides a method `classify()`, which calculates the quality score based on the `SnippetClassifier` class, which will be discussed later in this Section.

Snippets		
ID	INTEGER	Primary key of the Snippet
ENTITY_ID	INTEGER	Reference to the related Entity
SOURCE_ID	INTEGER	Reference to the related Source
TEXT	TEXT	Content of the Snippet
EXTRACTED_AT	DATETIME	Date/time of creation
F_*	FLOAT	Features extracted from the Snippet
SCORE_*	FLOAT	The regression scores for the Snippet

Table 4.1: Database schema for snippets

Table 4.1 shows the database schema used for the storage of snippets. When the extraction process extracts a set of snippets for an entity, this results in an entries of the relation `Snippet`.

### Source

The `Source` class of *WebKnox* has been extended to store additional values and provide additional functionality. First the capability to retrieve the main content block from the URL was added. Then the capability of retrieving Google Page Rank for the URL was also added.

### WebResult

The `WebResult` class is an in memory container class for the retrieved `WebResults`. It is used to generate the `AggregatedResult` instances.

### AggregatedResult

The `AggregatedResults` generated by the `SourceAggregator` contain references to the `WebResults` they result from, as well as an aggregated rank value. They are also not stored in the database directly, but keep a reference to the `Source` to which they refer.

### Search Engine IDs

The *Search Engine IDs* are defined in the `SourceRetrieverManager` class of the web package. The following Search Engines have been implemented including their IDs.

Search Engine IDs		
1	Web Search	Yahoo!
2	Web Search	Google
3	Web Search	Microsoft Live
4	Semantic Search	Hakia
5	Custom Search	Yahoo! Boss
6	Web Search	Bing
7	Real-Time Search	Twitter
8	Blog Search	Google Blogs Search
9	Statement Search	Texrunner
99	Editorial Content	-

Table 4.2: Search Engine IDs used in WebSnippets

### DatabaseManager

To store data in and retrieve data from the database, *WebKnox* provides the `DatabaseManager`, described in section 4.1. The `DatabaseManager` is part of

*WebKnox* Persistence-Package.

The **DatabaseManager** has been implemented to provide the additional capabilities to store new snippets, update the snippets regression rank and check whether a snippet is already stored in the database.

Furthermore the **DatabaseManager** has been extended to also persist the extracted snippets for a given KnowledgeManager.

## 4.2.2 Lifecycle

This section describes a full snippet extraction cycle as described in chapter 3. Figure 4.3 depicts the lifecycle of the extraction cycle and describes the threading-model in detail.

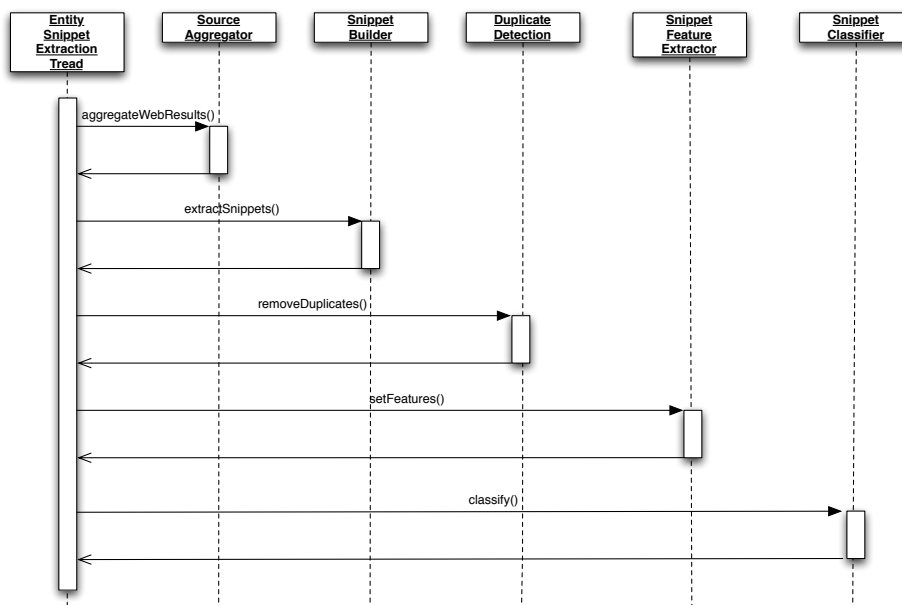


Figure 4.3: Sequence-diagram websnippets

The *WebKnox* prototype has been implemented to run in batch-mode, which means for every run a full extraction cycle for all entities in the knowledge-base is performed.

One requirement of the system design was, that the *WebKnox* system provides a scalable infrastructure. Therefore, the prototype make extensive use of

threading. The threading model was implemented through the `SnippetExtractor` and `EntitySnippetExtractionThread` classes. Those classes will be discussed in detail in the following subsections.

### **SnippetExtractor**

The `SnippetExtractor` is a subclass of the abstract `Extractor` class provided by *WebKnox*. *WebSnippets* extends this singleton class with the `SnippetExtractor`, which retrieves all entities from the knowledge base and schedules  $k$  thread runs in parallel, where  $k$  is the number of entities.

For each entity a separate thread is started. Each thread is a subclass of `EntitySnippetExtractionThread`, which will be explained in the next subsection. To avoid overloading the system, we implemented a threading queue to only run  $i$  threads in parallel, where  $i$  was 3 in our experiments.

### **EntitySnippetExtractionThread**

The `EntitySnippetExtractionThread` is instantiated by the `SnippetExtractor` on a per-entity basis. Each instance is responsible for handling and controlling the execution of the source aggregation process, the snippet extraction process, the de-duplication process and the snippet ranking process as described in section 3.2. The implementation details of each process are explained in detail in the remainder of this section.

## **4.2.3 Source Aggregation Process**

The *Source Aggregation Process* is implemented through several classes, which are located in the web package and will be discussed in detail in the remainder of this subsection. The design of the *Source Aggregation Process* has already been discussed in section 3.2.

### **SourceRetriever**

The `SourceRetriever` class is provided by *WebKnox*. The *WebKnox* version provides functionality to query four search engines (Google, Haika, Microsoft Live, and Yahoo) and retrieve a list of URLs.

We extended the class to return the full web results as described in section 1.1 including title, summary, rank and the URL. Furthermore, we added additional search engines as described in subsection 4.2.1.

### SourceAggregator

The `SourceAggregator` class has been added to implement a web carnivore, which is described in section ???. The class acts as an interface for a collection of source aggregation algorithms.

We implemented the IFM [28] algorithm as described in section 3.2. It aggregates a set of `WebResults` given an entity-centric `SnippetQuery` using the `SourceRetriever` class and combines the web results from all search engines that have been queried into one ranked list of `AggregatedResults` using the `RankAggregator` class.

### RankAggregator

The `RankAggregator` class acts as interface to different rank aggregation algorithms it provides. In the prototype implementation we implemented *RankAverage* as described in section 3.2.3.

### SnippetQuery

The `SnippetQuery` class acts as query template builder factory. Given an entity, it generates a set of queries, which are sent to the search engines. Its design has been discussed in section 3.2.2.

## 4.2.4 Snippet Extraction Process

The *Snippet Extraction Process* is implemented through a single `SnippetBuilder` class, which is located in the extraction package. The design of the *Snippet Extraction Process* has already been discussed in section 3.3.

### SnippetBuilder

The `SnippetBuilder` class serves as interface to different snippet extraction techniques. We implemented *WRS*, *DS1* and *DS2* as described in section 3.3 in detail. All these techniques have in common that they receive the `Entity` and an `AggregatedResult` as input and return a set of `Snippets`.

**WRS** WRS takes the summary of the first web result from the aggregated result object and removes the BIAS.

**DS1** DS1 retrieves the main content section from the linked document using the `AlchemyAPI`, which has been described earlier. As the next step it tokenizes the main content part using an `IndoEuropeanTokenizer` from the

LingPipe framework into sentence chunks.

Next we check each sentence chunk if it contains our entity using an approximate-based chunker also from LingPipe. The chunker receives a dictionary of spellings and synonyms of our entity as input plus a maximum edit distance, which is two in our case. If the sentence contains a mentioning of our entity, its added to the set of snippet candidates.

**DS2** DS2 works very much the same as DS1, but it makes use of a part-of-speech tagger, which has been trained on the Brown corpus using a Hidden-Markov-Model.

Based on the part-of-speech tags we filter out sentences that do not contain at least two nouns and one verb. Additionally each sentence has to start with an uppercase letter. If so, the sentence is added to the set of snippet candidates.

#### 4.2.5 De-Duplication Process

The *De-Duplication Process* is implement through a single `SnippetDuplicateDetection` class, which is located in the extraction package. The design of the *De-Duplication Process* has already been discussed in section 3.4.

##### **SnippetDuplicateDetection**

The `SnippetDuplicateDetection` class serves as interface to different de-duplication techniques and algorithms. We implemented a simple lookup mechanism to eliminate exact duplicates within the list of snippet candidates.

The algorithm checks whether the exact snippet is either extracted multiple times during the current run or whether it is already contained in the database. If so, the snippet is removed from the set of candidates.

Other de-duplication techniques were already discussed in section 2.2.5 and the SimMetrics plugin discussed in section 4.1.2 also provides various different algorithms for that problem.

#### 4.2.6 Snippet Ranking Process

The Snippet Ranking Process is implemented using a machine learning approach that is described in Section 3.5 in greater detail.



### **SnippetFeatureExtractor**

The first part of the process is to extract the relevant features from the snippet candidates. This is implemented in the `SnippetFeatureExtractor` class, which is part of the classification package.

The feature extraction class implements the `extractFeatures()` method, which gets the snippet as input and extracts, generates and retrieves the feature vector as described in section 3.5.1.

### **SnippetClassifier**

The second part is to apply the regression analysis on each snippet. This is done through the `SnippetClassifier` class, which is an extension of `WebKnoxClassifier` class and located in the *Classification-Package*.

We use the `SimpleRegression` regression learner of the WEKA framework.

## **4.3 Engineering Issues**

This section explains the *engineering problems* that occurred during the development of the prototype system and how we solved it.

**Disambiguation** The first challenge was to disambiguate the web results and related documents that were retrieved by the source aggregation process. This was solved using intelligent query rewriting and rank aggregation as described in the design chapter.

**Main Content Block** The second challenge was to retrieve the main content block from the related documents. First we just removed the HTML tags, but that resulted in very bad snippet candidates. Next we tried using *WebKnox* XPath, which was better but still gave us lots of chunk content.

Finally we solved the issue by using *AlchemyAPI* web service, which uses sophisticated machine learning algorithms and trained models. This is still not perfect, but gave a reasonable main content block, which we used for snippet candidate extraction.

**Sentence Chunking** The third challenge was to separate the main content block into sentence chunks. First we took a simple approach by just separating by . (dot) ; (semicolon) ! (exclamationmark) and ? (questionmark).

This didn't result in good sentence boundaries. Next, we tried to use *WebKnox*'s sentence chunker, which gave reasonably good results.

Finally we tried LingPipe's Indo-European sentence chunker, which provided most precise results.

**Named Entity Recognition** The next challenge was to recognize sentences that contained the searched entity. First we tried simple string matching, but that missed most of the entities because of different spelling and presentation. Next we lowercased the entities, which caught more entities, but still very few.

Finally we solved the problem by using a dictionary- and proximity-based NER component from LingPipe.

**Memory Consumption** The first extraction run failed completely and stopped the entire application with an out of memory error. We found out that we needed to increase the Java heap memory (to 1 gigabyte in our case), which is 128 megabytes per default.

After further investigation we found out, that the real problem was to keep all extractions in memory until all snippets for all entities were extracted. We were able to reduce the memory consumption considerably by writing results to the database after each entity extraction loop and destroyed the object references afterwards.

**Performance Bottlenecks** The next extraction loop worked but took a very long time to complete. The reason for this was, that each extraction loop was run in serial. We were able to reduce the extraction for all entities by using *WebKnox*'s threading model.

The first major optimization was to encapsulate each per-entity extraction loop into its own process. Our experiment showed that three threads working in parallel produced a good performance gain for our system.

The next optimization was, to parallelize the search engine querying. Since we send several queries to a number of search engines, we were able to parallelize those queries to one thread for each query/search engine combination. After all threads complete, we perform rank aggregation of the retrieved lists of web results.

The last optimization was to parallelize the document retrieval and snippet candidate extraction for each aggregated web result in the case of DS1 and DS2.

## 4.4 Summary

In this chapter we explained the architecture of *WebSnippets* in more detail and the most important packages and classes were described. The chapter also enumerated which programming languages and which frameworks / APIs have been used to implement and enhance the system.

The next chapter evaluates *WebSnippets* in a series of experiments starting with a problem validation.

# Chapter 5

## Evaluation

In this chapter we are evaluating the overall performance and quality of the proposed algorithms. For this purpose we conducted a thorough user study with human judges that were shown a set of guidelines, and then were asked to judge snippets and applying the criteria in the guidelines. Those judgments then were aggregated. The final results are presented below followed by a result discussion.

### 5.1 Introduction

In a typical information extraction system (see Section 2.1.2), the implicit consumer of the results was again a machine, e.g. a knowledge base or semantic content store. Thus precision and recall were the natural measures in such systems, with maximum precision and maximum recall being the ideal. We refer to these works as *machine-centric* extraction systems.

When it comes to *user-centric* extraction systems, we claim that the simple goals of maximum precision and maximum recall are not optimal since they measure only the quantity but not the quality of the displayed results.

For our scenario quality matters, because users may not necessarily want to see every possible snippet for an entity because not every single snippet is necessarily relevant to the entity, nor is it necessarily interesting. Displaying such irrelevant or uninteresting snippets is a nuisance and a distraction.

Given the above issues we propose to evaluate the quality of user-centric extraction systems along three core dimensions:

**Relevance** Is the presented information relevant to the entity in question?

**Interestingness** Is the presented information interesting in general?

**Curiosity** Is the presented information interesting enough to the reader to warrant further action or investigation?

We distinguish between user-centric and machine-centric information extraction systems, and argue that for the former the simple precision/recall scores fail to accurately reflect the quality of extracted information. Therefore we propose to measure the quality of extracted information within user-centric extraction systems in the three core dimensions mentioned above. This evaluation methodology is adopted from [51].

## 5.2 Methodology

We describe the methodology used to evaluate the WebSnippets system along the three aforementioned criteria – relevancy, interestingness, and curiosity.

Our standard evaluation methodology consists of a team of human judges rating the snippets extracted and presented by WebSnippets. Using an interface specifically designed for this task (see Figure 5.1), the extracted set of snippets is grouped by entity and presented to the judges. A judge is asked to read the snippet carefully prior to issuing any judgments. The judge is then asked to rate each snippet per entity in terms of its relevancy, interestingness, and curiosity.

**AT&T Inc. (Company)**

**Evaluation Criteria**  
Please vote each summary by the following criteria:

**Relevant?**  
Does the summary provide on-topic information for the specified entity.

**Interesting?**  
Do you find the information provided interesting.

**Learn more?**  
Would you like to learn more about the information provided.  
See [Evaluation Guideline](#).

AT&T Inc. (T) is scheduled to report the third quarter results on Thursday, October 22, 2009. In the last four quarters, the company's actual earnings exceeded the market's consensus significantly. ... In the third quarter last year, AT&T activated 2.4 million iPhones, of which 40% were for subscribers who jumped over from other carriers.	<b>Relevant?</b> <input type="radio"/> yes <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Interesting?</b> <input type="radio"/> very <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Learn more?</b> <input type="radio"/> yes <input type="radio"/> no <input type="radio"/> can't tell
The U.S. Federal Communications Commission late Thursday gave final approval to ATT Inc's \$944 million bid to buy Centennial Communications Corp. ATT Inc. (NYSE:T, \$25.84, +\$0.41, 1.61%) fell 0.93 percent to \$25.70 on Friday morning ...	<b>Relevant?</b> <input type="radio"/> yes <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Interesting?</b> <input type="radio"/> very <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Learn more?</b> <input type="radio"/> yes <input type="radio"/> no <input type="radio"/> can't tell
Choose the best high speed Internet, home phone, wireless, advanced TV, and bundled services for you.	<b>Relevant?</b> <input type="radio"/> yes <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Interesting?</b> <input type="radio"/> very <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Learn more?</b> <input type="radio"/> yes <input type="radio"/> no <input type="radio"/> can't tell
Get detailed information on ATT INC. (T) including quote performance, Real-Time ECN, technical chart analysis, key stats, insider transactions, ...	<b>Relevant?</b> <input type="radio"/> yes <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Interesting?</b> <input type="radio"/> very <input type="radio"/> somewhat <input type="radio"/> no <input type="radio"/> can't tell	<b>Learn more?</b> <input type="radio"/> yes <input type="radio"/> no <input type="radio"/> can't tell

Figure 5.1: Screenshot of the Evaluation Tool

### 5.2.1 Baseline

To evaluate our system and prove our hypothesis, we compared our system to state-of-the-art search systems as well as editorially picked results. We used results from the following search engines as baseline.

- **Google Web Search**, as state of the art in web search
- **Google Blogs Search**, as state of the art in blog search
- **Twitter Search**, as state of the art in real-time search
- **Editorial Picks**, as *gold standard* for such a scenario (see Appendix A).

The presented baseline snippets are the summaries of the top-3 web results returned by the above mentioned providers. We proposed, implemented, and described this technique as *WRS* in Section 3.3.1.

### 5.2.2 Test Set

The *Test Set* was thoroughly chosen containing 20 different entities out of four different categories. The categories were products, places, people and organizations, which is a default for IE tasks.

The entities for each concept were chosen manually by applying the following criteria to gain a more representative sample for each concept. Each category consists of four well known and one less known entities.

Products		Places	
Palm Pre	<i>Cell Phone</i>	San Francisco	<i>City</i>
MacBook Pro	<i>Laptop</i>	Frankfurt	<i>City</i>
Bugatti Veyron 16.4	<i>Car</i>	Riesa	<i>Town</i>
Volvo C30 BEV	<i>Car</i>	Gilroy	<i>Town</i>
The Lost Symbol	<i>Book</i>	Luxor	<i>Village</i>
People		Organizations	
Barack Obama	<i>Politician</i>	Yahoo Inc.	<i>Company</i>
Amy MacDonald	<i>Musician</i>	AT&T Inc.	<i>Company</i>
Robin Williams	<i>Actor</i>	Rotary International	<i>NGO</i>
Bill Gates	<i>Founder</i>	IKEA	<i>Company</i>
Reiner Kraft	<i>Researcher</i>	Live Like a German	<i>Company</i>

Table 5.1: Test set used for evaluation

Table 5.1 shows the entities of the test set used for evaluation. The first row presents the entities name, the second column shows the entities concept name.

### 5.2.3 Evaluation Guideline

The *Evaluation Guideline* is presented to the editors before they start the survey. The guideline provides the editors with precise instructions on how to answer each question and gives examples for each possible choice.

#### Question 1: Evaluation of Relevancy

The *Relevancy* is evaluated to determine whether the summary provides on-topic information for the specified entity. The following guideline for evaluation of relevancy is provided to the editors:

- Does the summary provide on-topic information for the specified object?
- Is the summary provided in any way relevant to the object named above (in the title).
- Relevant are facts and opinions of any kind, as long as they are on-topic.
- Summaries must refer to or must provide information about the object directly.
- Summaries of websites that are talking about an object are considered somewhat relevant.
- The object in question must be the subject of the sentence or paragraph to be relevant.

The judges can select from the following choices:

**Relevant** – An example: “iPhone 3GS is a GSM cell phone that’s also an iPod, a video camera, and a mobile Internet device with email and GPS maps.”

**Somewhat Relevant** – An example: “AT&T, Inc. has seen incredible gain on the back of the Apple iPhone in the last two years.”

**Not Relevant** – An example: “iPhone Halloween Costumes Make Us Jealous, Cost \$2,000 To Build: Looking for the best halloween costume ever?”

#### Question 2: Evaluation of Interestingness

The *Interestingness* is evaluated to determine whether the provided information is interesting. The following guideline for evaluation of interestingness is provided to the editors:

- Do you find the information provided interesting?

- Does the provided summary contain any interesting information.
- Interesting are facts, experiences, opinions, etc. about the object in question.
- A summary is interesting if it is easily understood by the reader, potentially useful, novel, or validates some hypothesis that a user seeks to confirm.

The judges can select from the following choices:

**Very Interesting** – An example: “The *iPhone 3GS* finally adds common cell phone features like multimedia messaging, video recording, and voice dialing.”

**Somewhat Interesting** – An example: “The *iPhone 3GS* doesn’t make the same grand leap that the iPhone 3G made from the first-generation model, but the latest Apple handset is still a compelling upgrade for some users.”

**Not Interesting** – An example: “If I seem cooler, it’s because I’m no longer using a 1st gen *iPhone*. *3gs* upgrade”

### Question 3: Evaluation of Curiosity

The *Curiosity* is evaluated to determine whether the editors would like to learn more about the information provided. The following guideline for evaluation of curiosity is provided to the editors:

- Would you like to learn more about the information provided?
- Based on the summary provided, would you be interested to learn more about that content or the topic (the named object).
- A message, or part of a message, designed to arouse curiosity and interest and cause the reader to explore further, but without revealing too much.
- If the object is underlined and clickable as a link, based on the summary around it would you actually click on it to learn more about it?

The judges can select from the following choices:

**Would Like To Learn More** – An example: “Did you know that the *iphone* is used twice as much as any other phone in the US?”

**Would Not Like To Learn More** – An example: “The *iphone 3GS* is manufactured by Apple.”



### 5.2.4 Error Rate

In statistics there are several ways to show that the number of samples is of sufficient size. For our experiments we calculated a 95% confidence interval for each sample, which is common for such a scenario. What that means is that for a given sample, an interval is calculated, that contains the mean of the sample with a confidence of 95%.

Appendix B shows detailed statistics about the samples we used throughout our experiments. All samples show small confidence intervals with low margins of error (less than 3.5%), which means that we have very stable data.

## 5.3 Problem Validation

In a preliminary experiment at the beginning of the thesis we evaluated our hypothesis stated in the introduction to validate that the problem stated in the problem statement actually exists.

### 5.3.1 Evaluation of the Baseline

For each entity out of the test set we presented the top-3 snippets for each provider from the baseline to the editors. *1785 judgements* were given by *18 distinct judges* during the duration of this experiment.

	<b>R</b>	<b>SR</b>	<b>RSR</b>	<b>I</b>	<b>SI</b>	<b>ISI</b>	<b>C</b>
<b>Precision of unranked baseline sets</b>							
<b>Google Web</b>	29.6%	34.2%	63.8%	18.5%	29.0%	47.4%	27.8%
<b>Twitter</b>	5.1%	26.8%	31.9%	2.4%	15.9%	18.3%	6.5%
<b>Google Blogs</b>	15.6%	34.9%	50.4%	10.5%	29.0%	39.5%	17.3%
<b>Editorial Picks</b>	68.3%	27.1%	95.4%	55.8%	36.0%	91.8%	67.4%
<b>Mean-Average-Precision of ranked baseline lists</b>							
<b>Google Web</b>	27.4%	32.8%	60.2%	16.4%	27.2%	43.6%	25.4%
<b>Twitter</b>	5.8%	30.1%	35.9%	3.0%	16.9%	20.0%	7.5%
<b>Google Blogs</b>	15.9%	34.1%	50.0%	10.6%	29.3%	39.8%	16.0%
<b>Editorial Picks</b>	71.6%	23.7%	95.3%	57.6%	34.4%	92.1%	67.5%

\* *Relevant (R), Somewhat Relevant (SR), Relevant + Somewhat Relevant (RSR), Interesting (I), Somewhat Interesting (SI), Interesting + Somewhat Interesting (ISI), Curiosity (C)*

Table 5.2: Performance Evaluation of the Baseline

Table 5.2 shows the Precision (which *not* considers the ranking) and the

Mean-Average-Precision (which considers the ranking) for the baseline snippets. The Baseline ranks are equivalent to the originating search engine ranks. The used evaluation measures have been discussed in Section 2.1.3 in detail.

### 5.3.2 Problem Validation Summary

The results from the baseline evaluation show that there is a clear gap between what state-of-the-art search systems are able to provide today and the *gold standard*, what a human editor is able to archive.

With this initial experiment we have shown that there is actually a problem, which then motivated us to investigate possible solutions on whether and how it is possible to further reduce the qualitative gap to the *gold standard* (upper bound).

## 5.4 Snippet Extraction Performance

Having developed and tuned the *WebSnippets* system with respect to the optimal algorithm configuration, we now put all the pieces together and evaluate the final system with respect to the overall metrics of relevance, interestingness, and curiosity. These results, shown in Table 5.5, are based on a set of *1,276 snippets*, extracted from a corpus of *top-100 aggregated documents*.

This evaluation consists of two parts. First, we evaluated the overall snippet extraction performance of the unranked snippet candidates. Second, we evaluated the snippet ranking process to see whether the quality of the top-*k* snippets could further be increased by ranking them according to the dimensions of *relevancy*, *interestingness*, and *curiosity*.

### 5.4.1 Evaluation of Snippet Candidate Set

The *Snippet Candidate Set* presents the unranked set of snippet candidates, extracted by our system. In a first step we evaluated their overall performance to see, how well the extraction algorithm performs.

Our evaluated snippet candidates were extracted using our *DS2* algorithm, using the top-100 aggregated documents as source for the snippet extraction process. We retrieved search results from the same search engines we used for our baseline, namely *Google Web*, *Google Blogs* and *Twitter*. For rank aggregation we used the *rank average* algorithm, which we discussed earlier. *1961*

judgements were given by 15 distinct judges during this experiment.

The results presented in Figure 5.2 show, the combination of source aggregation, snippet extraction, and de-duplication, combine to yield an relevancy score of 81.3% (1936 judgements), an interestingness score of 74.2% (1930 judgements), and a curiosity score of 40.7% (1864 judgements).

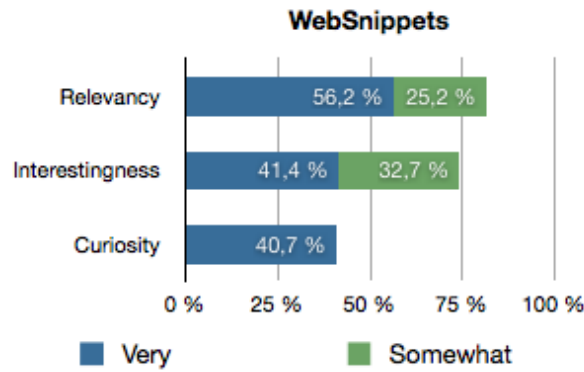


Figure 5.2: Results of the WebSnippets Performance

### 5.4.2 Evaluation of Snippet Ranking

The second part of this experiment evaluated the snippet ranking component as described in Section 3.5. With this experiment we evaluated how well our feature-based ranking approach increases the performance of WebSnippets.

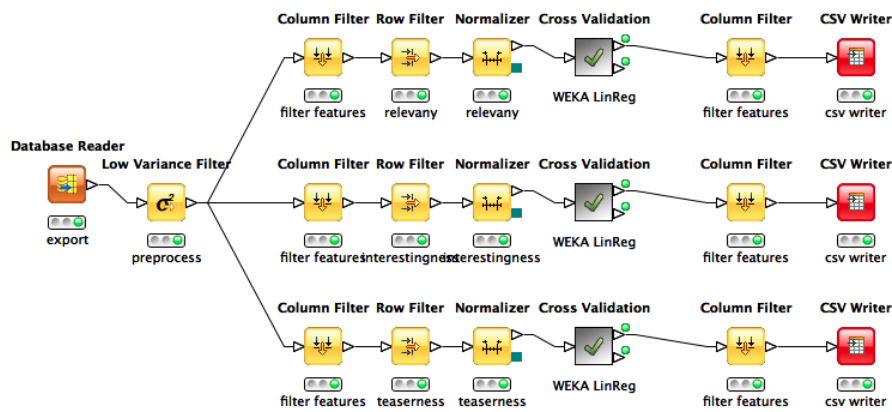


Figure 5.3: Overview of the snippet ranking prediction queue

Figure 5.3 shows the experimental setup, where we took the judgements as

input to train three distinct prediction models for each dimension, *relevancy*, *interestingness*, and *curiosity* on the snippets feature vector.

After preprocessing, we used *10-fold cross-validation* with *random sampling*, automatic feature selection using the *M5 attribute selection method* and the `LinearRegression` algorithm from the WEKA framework, to determine the “best” prediction function to calculate the continuous snippet ranking scores. The generated prediction models are shown in Appendix C.

As target variable of the training set, we used the judgements provided by our editorial team. Judgements voted as “yes” or “very” are scored with *1.0*, judgements voted as “somewhat” are scored with *0.5*, and judgements voted as “no” are scored with *0*.

	<b>Relevancy</b>	<b>Interestingness</b>	<b>Curiosity</b>
<b>Precision</b>	68.643%	57.511%	40.719%
<b>Average Precision</b>	78.762%	67.191%	49.013%
<b>Improvement</b>	11.742%	16.832%	20.368%

Table 5.3: Evaluation of the prediction model for relevancy, interestingness, and curiosity after automatic feature selection

We notice here that this is not representative for the overall performance of WebSnippets, but rather a way to show, that the overall performance can be improved using our feature-based ranking approach. We evaluated over all snippets in the same model, rather than on a per entity or concept basis. Therefore, we don’t have multiple queries and therefore no MAP or the MAP is the AP.

<b>Error Rate</b>	<b>Relevancy</b>	<b>Interestingness</b>	<b>Curiosity</b>
<i>Correlation coefficient</i>	0.2886	0.2379	0.0861
<i>Mean absolute error</i>	0.6284	0.6572	0.9357
<i>Root mean squared error</i>	0.738	0.77	0.9835
<i>Relative absolute error</i>	90.1919 %	97.2286 %	97.5604 %
<i>Root relative squared error</i>	95.7449 %	97.6638 %	100.8512 %

Table 5.4: Evaluation of the prediction model for relevancy, interestingness, and curiosity after automatic feature selection

Table 5.4 shows that even we can improve the snippet quality significantly by ranking them, the prediction of the snippet performance score shows a high margin of error. Therefore, this model is not good for prediction the true value somewhat would vote for an individual snippet, but rather to get just a “better”

ranked list of snippets.

### 5.4.3 Overall Snippet Extraction Performance Summary

With this experiment we have shown two things. First, that the gap between what state-of-the-art search systems are able to provide today and the gold standard can significantly be reduced through *WebSnippets*. Second, we were able to further improve *WebSnippets* performance by ranking the snippets according to features influencing the overall quality.

Relevancy	Tw	GB	GW	WS	EP
Relevant	5.1%	15.6%	29.6%	56.2%	68.3%
Somewhat Relevant	26.8%	34.9%	34.2%	25.2%	27.1%
Not Relevant	68.1%	49.6%	36.2%	18.7%	4.6%
<b>Interestingness</b>					
Interesting	2.4%	10.5%	18.5%	41.4%	55.8%
Somewhat Interesting	15.9%	29.0%	29.0%	32.7%	36.0%
Not Interesting	81.7%	60.5%	52.6%	25.8%	8.2%
<b>Curiosity</b>					
Would Like To Learn More	6.5%	17.3%	27.8%	40.7%	67.4%
Would Not Like To Learn More	93.5%	82.7%	72.2%	59.3%	32.6%

\* *The Baseline: Twitter (Tw), Google Blogs (GB), Google Web (GW); Our System: WebSnippets (WS); The Gold Standard: Editorial Picks (EP)*

Table 5.5: Performance comparison of the *Baseline*, *WebSnippets*, and our *gold standard*

As Table 5.5 shows, the overall combination of source aggregation, snippet extraction, snippet de-duplication, and snippet ranking combine to yield an relevance score of 81.3%, an interestingness score of 74.2%, and a curiosity score of 40.7%.

Furthermore, as shown in Table 5.3, we were able to boost these values by 11.7% in relevancy, 16.8% in interestingness, and 20.4% in the curiosity metrics by our ranking algorithm.

## 5.5 Discussion of Results

In an overall comparison *WebSnippets* shows an increase in *relevancy* of about 3.3 times from 16.8% of the baseline to now. Including somewhat relevant results, there is still an improvement of 1.67 times over the average of the baseline.

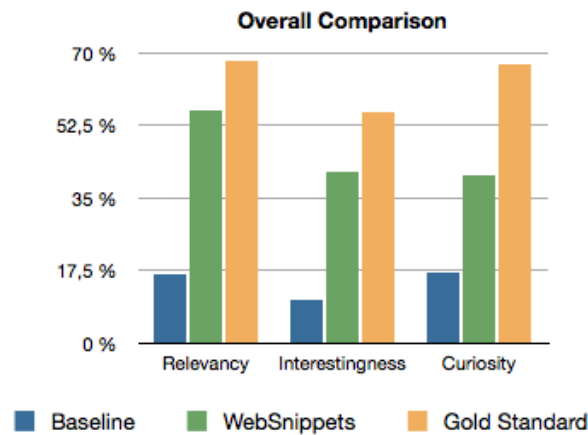


Figure 5.4: Comparison of the Baseline, WebSnippets and the Gold Standard

Furthermore, we were able to reduce the “not relevant” results by 2.75 times.

In addition, *WebSnippets* shows an increase in *interestingness* of about 4 times from 10.5% of the baseline to now. Including somewhat interesting results, there is still an improvement of 1.67 times over the average of the baseline. Furthermore, we were able to reduce the not interesting results by 2.5 times.

*WebSnippets* shows an increase in *curiosity* of about 2.4 times from 17.2% of the baseline to now.

Even so the precision of our snippet candidates yield significantly better results than those of the baseline. Our experiment on the *ranking* component showed an additional increase of relevancy by 11.7%, interestingness by 16.8%, and curiosity by 20.4%. The ranking model isn’t a good prediction model due to its high error rate.

The upper bound or *gold standard* for such a scenario is considered a human editor, who manually performs the task of finding and extracting the best possible snippets from the Web. That is the ultimate goal to achieve by an automated system such as *WebSnippets*. We have shown that we were clearly able to significantly reduce that gap between the baseline and the gold standard, even so there is still room for improvement. More efforts in future work is required to further reduce this gap in relevancy, interestingness, and curiosity.

Finally, the goal of this thesis to design and build a system that automates the process of extracting entity-centric knowledge from the Web and

improve over the baseline with respect to user-centric information need has been achieved. We were able to improve relevancy by 335%, interestingness by 396%, and curiosity by 237%.

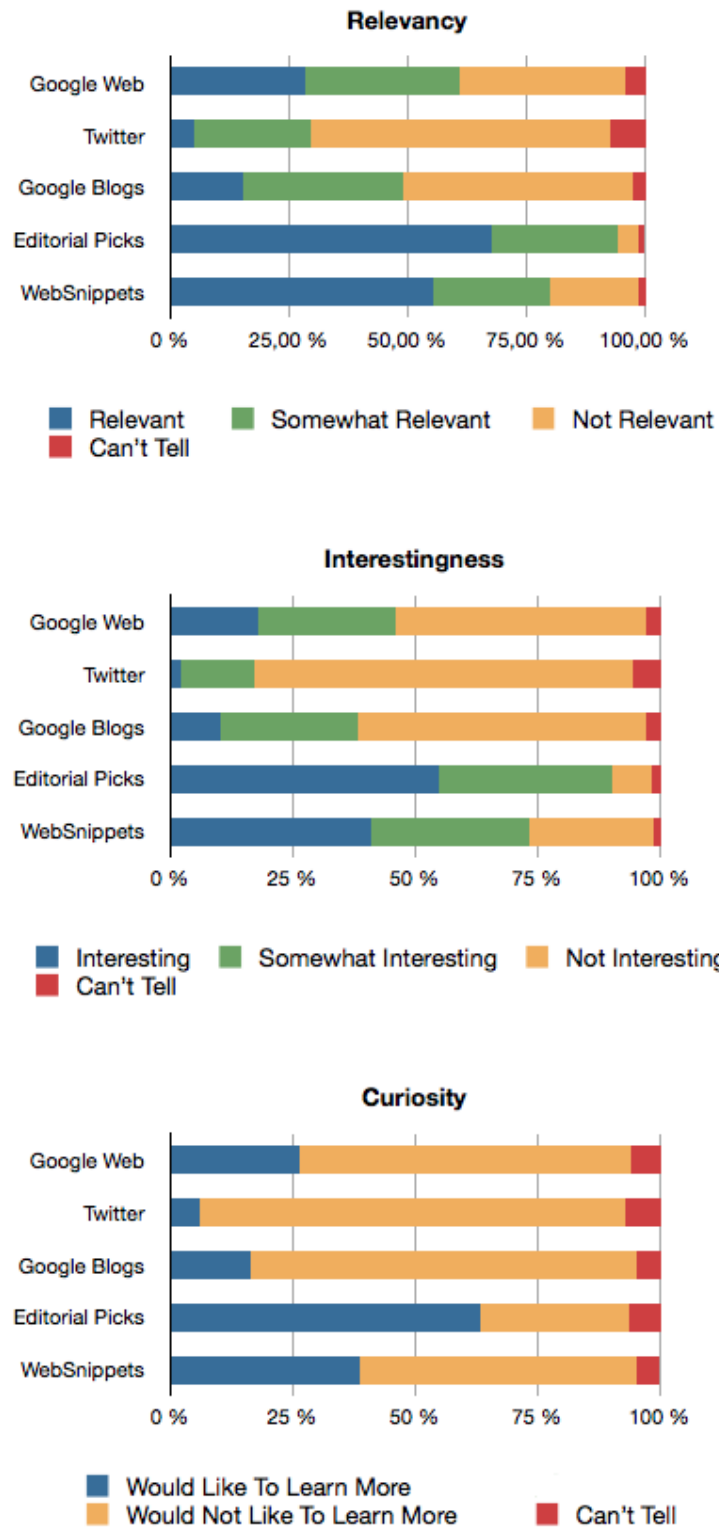


Figure 5.5: Overview over the judgements given by the editorial team



## Chapter 6

# Case Study: WebSnippets in Live Like a German Travel Web Site

The previous chapter described the algorithms to extract and rank entity-centric snippets. We show now how this can be implemented for a particular embodiment to point out practical and system aspects, as well as performance and scalability issues.

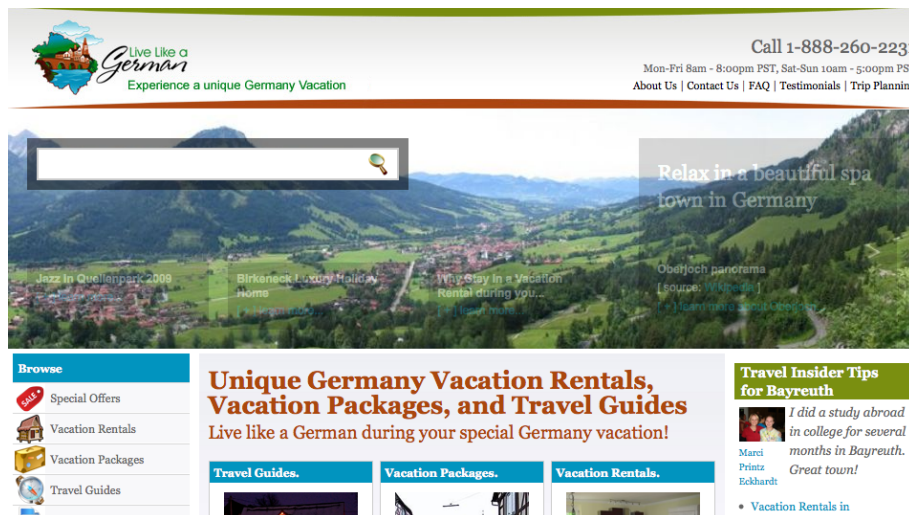


Figure 6.1: Screenshot of Live Like a German Website

Our case study focuses on a small travel web site, *Live Like a German*<sup>1</sup>, which is a portal for vacation rentals, vacation packages, and provides travel guides for travelers to Germany. One of the problem areas for this web site is overall high editorial cost for compiling travel guides and customized vacation packages. Relevant travel content is very important for this site, and encourages travelers to visit the site often and check back for new articles.

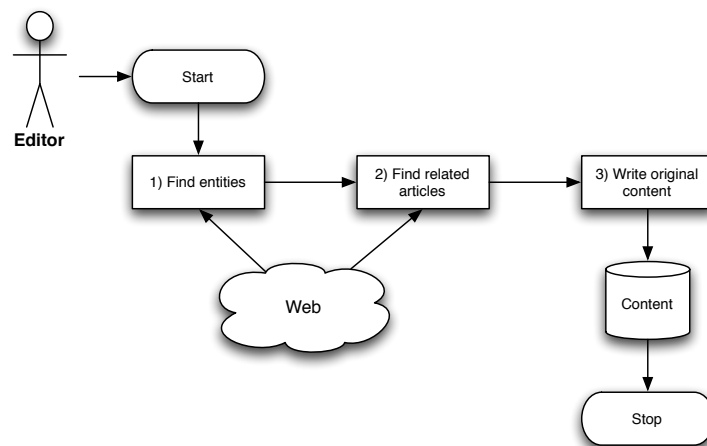


Figure 6.2: Traditional editorial approach of Live Like A German content aggregation

The traditional editorial approach as shown in Figure 6.2 can be summarized as 1) find popular Germany travel related entities (e.g., places, points of interest), 2) have an editor manually find and edit related articles (e.g., from known resources as Wikipedia, Wikitravel<sup>2</sup>), or 3) have an editor write original content.

In this chapter we provide an overview of Live Like a German and illustrate how WebSnippets are used to decrease their editorial costs while still providing a source of high quality travel related content. In addition, WebSnippets content can be further monetized, and helps with their overall SEO efforts to generate more organic search engine traffic.

<sup>1</sup><http://www.live-like-a-german.com>, accessed on 01/04/2010

<sup>2</sup><http://www.wikitravel.org>, accessed on 01/04/2010

## 6.1 Overview of Live Like a German

Live Like a German is a vacation rental portal focused on vacation apartments in Germany. The site differentiates itself from its competitors with its tight integration of travel content that allows travelers to learn more about the country, and make informed decisions on where to go and where to stay.

Good quality travel content is critical to the overall success of the site and its strategy. First, it helps to differentiate against other travel portals. Second, great content attracts new users and helps the site to rank high in search engines. Most of its content currently is being produced by editors, who are writing original articles, travel guides, and trip plans, based on input of property owners in Germany.

For example, a property owner in Munich puts together some ideas for a trip plan (e.g., a few itinerary items for a 7 day trip). The content is in raw form, mostly in German. An editor then translates this raw / original content and polishes it into a high quality trip plan that then can be consumed on the web site. Or, editors are compiling interesting points of interest (POI), along with travel insider tips related to this POI.

Especially this latter use case motivated for an integration with WebSnippets. An algorithmic approach for generating travel content related to POIs and travel destinations would decrease the dependency on editors, and therefore overall decreasing overall editorial costs. Furthermore, it would help to better scale and find content for lesser known destinations. Third, it would allow the web site to stay up to date since new content is being generated continuously.

## 6.2 Web-Snippets Integration

Live Like a German organizes its content around these entity types:

- Destination (e.g., Munich)
- Region (e.g., Bavaria)
- POI (e.g., Castle Neuschwanstein)

Those entities are linked together based on their geographical proximity and containment. This natural concept network then allows users to conveniently navigate related or nearby destinations and travel attractions.

The integration with WebSnippets therefore is organized based on these core entities. As a seed set we obtained from Live Like a German a list of all their current travel entities, sorted by their types. We then used these as a seed set to generate WebSnippets.

The original integration (based on the WebSnippets design and implementation described in the earlier chapter) was done by having an off-line process running once a day to obtain the latest candidate documents and extracted snippets for these travel entities. Those entities would then be saved and imported into their database on a daily basis.

Live Like a German then has a serving layer that reads these snippets and shows them along those entities in certain areas of their web site. For example, within the travel guide for Munich there is a small module “Travel Snippets” that lists the latest extracted snippets.



Figure 6.3: Screenshot of Live Like a German Travel Snippets

Since the focus of this initial integration was mostly to validate the usefulness of WebSnippets content the overall integration has been kept simple. Duplication is handled by generating a MD5 content checksum. WebSnippets are just generated once a day, and ranking of them has been kept simple too. Focus was mostly on recall to find content particularly for the lesser known destinations (e.g., smaller villages).

In particular, the following modifications and simplifications have been made regards to the full-blown production system as described earlier:

**Herbivore Selection** we used a fixed sets of herbivores for every run

**Extraction Techniques** WRS and DS1 are not used, using DS2

**Classification** no classification in de-dup is done

**Regression Ranking** we do not use regression score for ranking but extractedAt

The full system and application flow is shown here:

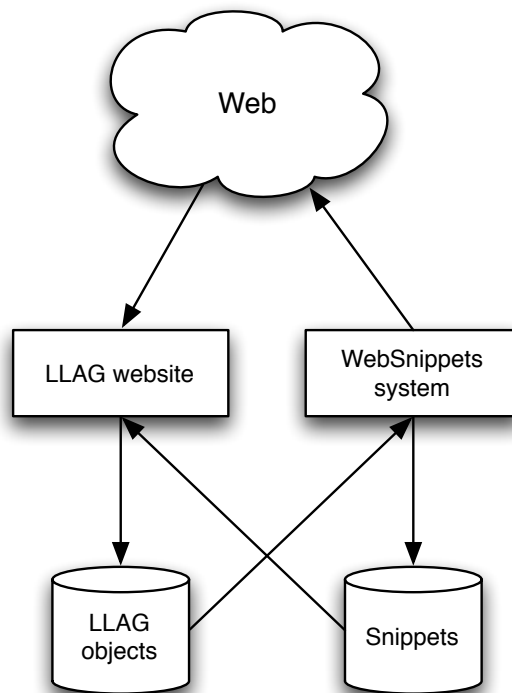


Figure 6.4: Flow-diagram of Live Like a German system-integration

For the query formulation strategy the imported data already contained lots of extra meta-data. That helped to basically come up with different query templates that proved to be very effective.

For example, for destinations the query would be “DESTINATION Germany”. This one would already disambiguate sufficiently with other places (e.g., Berlin in USA vs. Berlin in Germany). The destinations already had in most of the cases disambiguation information in their name (e.g., Rothenburg an der Fulda vs. Rothenburg ob der Tauber). This further helped to decrease overall ambiguity in the queries and led to good quality and relevant source documents.

Destination:

DESTINATION-NAME, STATE-NAME, COUNTRY-NAME  
DESTINATION-NAME travel guide, COUNTRY-NAME  
DESTINATION-NAME travel tips, COUNTRY-NAME

Region:

REGION-NAME, COUNTRY-NAME  
REGION-NAME region, COUNTRY-NAME  
REGION-NAME tourist guide, COUNTRY-NAME

Business:

BUSINESS-NAME, DESTINATION-NAME, COUNTRY-NAME  
BUSINESS-NAME, STREET-NAME + NUMBER, DESTINATION-NAME  
BUSINESS-NAME, BUSINESS-TYPE, DESTINATION-NAME

For the document sources we selected the following search providers:

- Google Web Search
- Google Blog Search
- Bing Web Search
- Twitter Search

## Chapter 7

# Conclusions and future work

Chapter 1 provided an introduction and motivation for our work, and highlighted the research problems that this thesis is addressing. We also pointed out directions of future research related to user-centric information extraction.

Chapter 2 provided more background to our research and an overview of related work.

Chapter 3 presented the system design and reviewed its main components.

Chapter 4 discussed implementation and engineering aspects, and presented an overview of technologies used, which design features had been implemented. We also provided detailed implementation details.

In chapter 5 we evaluated and studied the system's performance and overall quality of the snippet extraction.

Chapter 6 showed a concrete use case on how the *WebSnippet* system can be integrated into a travel web site, helping with the automatic generation of entity centric content for consumption.

This chapter 7 concludes with a summary of our results, achievements, and lessons learned while working on our *WebSnippet* system. An outlook is provided on future work and remaining open problems.

## 7.1 Achievements

This thesis studied the problem of extracting entity centric context *WebSnippets* automatically from the Web. To validate that the problem exists we conducted an preliminary experiment that showed evidence that current search engines and web information retrieval tools are not well designed for that task.

We introduced *WebSnippets*, a system designed for cost-effectively extracting and ranking entity-centric knowledge from the Web with respect to user-centric information needs. We investigated disambiguation using a web carnivore approach. We proposed three snippet extraction algorithms and showed how to rank snippets according to features extracted from the snippets.

We implemented the proposed system design as a prototype using the *Web-Knox* framework. During the implementation we able to continuously refine and tweak the system to yield optimal results.

We evaluated the results of the system and showed that it performed on average three times as good in relevancy, four times as good in interestingness and twice as good in curiosity evaluation measures than its state-of-the-art competitors based on the human judgement.

We demonstrated the real world use of the system and its relevancy to practical application by integrating a prototype into the *Live Like a German* travel site.

## 7.2 Future Work

There are still many open questions and issues that are worth exploring in future work. This section depicts those questions.

### 7.2.1 Automatic Content Construction

As shown in our user study, the *WebSnippets* system presents a way to reduce the cost of licensing or creating editorial content by automatically extracting knowledge from the Web. To extend this approach further it would be very helpful if *WebSnippets* were able to not only provide small snippets of text, but could also be used to construct aggregate documents, such as travel guides. This would even enhance the value, but requires more ideas on how to combine web snippets intelligently to form a useful and comprehensive aggregation document.



As shown in the background section, Cpedia tries to address this issue, but still lacks the ability to provide good and interesting homogenous texts that are easily consumable by the reader.

### 7.2.2 Enhanced Ranking

The *WebSnippet* system extracts and ranks knowledge according criteria that focus on human consumption. Therefore, *WebSnippets* tries to show the best possible content to the user, by taking quality features into account. This approach enables to explore many more possible features that might indicate the quality for the users.

One promising approach is to take semantic knowledge about the entity into account, to increase the overall interestingness further to the user.

### 7.2.3 Personalized Recommendations

In addition, each user has potentially different interests and experience. By showing more personalized ranked web snippets to the user, and remembering which snippets the user liked and not liked (or already knew) we can adapt the system further to provide a very personalized experience and ranking.

## 7.3 Summary

This thesis has demonstrated various approaches to automatically extract and rank entity-centric knowledge from the Web with respect to the users interest. Therefore, we proposed, implemented, and evaluated the *WebSnippets* system and showed that we have reached this goal.

# Appendix A

## Snippets

### A.1 Preliminary Snippets

For the preliminary experiment, the following ranked snippets were picked by editors from the top-100 Google Web search results and used as editorial picks:

#### **Palm Pre (Product, Cellular Phone)**

1. Despite some missing features and performance issues that make it less than ideal for on-the-go professionals, the Palm Pre offers gadget lovers and consumers well-integrated features and unparalleled multitasking capabilities. The hardware could be better...
2. The phone was launched on June 6, 2009, and is the first to use Palm's new Linux-based operating system, webOS. The Pre functions as a camera phone, a portable media player, a GPS navigator, and an Internet client.
3. According to Debian developer Joey Hess, Palm Pre phone is periodically sending users' information to Palm. Palm is gathering users' GPS information, along with data on every application used, and for how long it was used. This information gets uploaded to Palm on a daily basis, Hess claims.

#### **MacBook Pro (Product, Laptop)**

1. There are still a few items on our 13-inch wish list—matte screens, mobile broadband options, Blu-ray—but Apple has done an admirable job filling in some of the major missing pieces. By offering more features for less money, the 13-inch MacBook Pro remains one of the most universally useful laptops available.

2. The good: Adds SD card slot and reacquires FireWire; lower starting price; same solid unibody construction and giant multitouch trackpad. The bad: Nonremovable battery; no matte screen or discrete graphics options. Previously known as the MacBook, Apple's basic 13-inch aluminum unibody laptop has been promoted to the "Pro" series with added features and cutting the base price.
3. Radically overhauled last year, Apple's MacBook line of laptops moved to aluminum construction, edge-to-edge glass over LED displays, and oversize multitouch trackpads

#### **Bugatti Veyron 16.4 (Product, Car)**

1. The Bugatti Veyron is, by every measure, the world's fastest production road car.
2. The \$1.3 million Veyron will reach a top speed of 253 mph - a speed it can maintain for 12 minutes before all the fuel is gone.
3. A model drove by superstars like Tom Cruise, couldn't name itself cheap, rising the Bugatti Veyron at least of \$1,700,000, a price that measures it's quality.

#### **Volvo C30 BEV (Product, Car)**

1. The Volvo C30 BEV is the first model we will try out with electric power. This car's excellent properties in city traffic and its relatively low weight make it particularly suitable, since electric cars are primarily expected to be used in and around cities and for daily commuting.
2. Several C30 BEV prototypes have been produced by Volvo and it is powered by a lithium-ion battery pack. As the C30 BEV runs purely on electricity, its battery has a larger capacity of 24kwh compared to the 12kwh battery found in the plug-in hybrid.
3. The C30 BEV doesn't have the forceful energy recuperation when lifting off the accelerator that we've experienced in some electric cars, though any time you're freewheeling or braking the Volvo is scavenging back energy that would otherwise be lost.

#### **The Lost Symbol (Product, Book)**

1. Nevertheless, I enjoyed "The Lost Symbol" as mindless entertainment. It definitely peaked my interest to get back to Washington D.C. and see the

Capitol Building, the Washington Monument and the Masonic Temple. It is not as good as "Angels & Demons" or even "The Da Vinci Code". Although I usually don't rate books this way, I would give it three of five stars.

2. There's not much tension-relieving humor in "The Lost Symbol", but there is one spot in which Brown seems to be poking fun at himself and his delay in finishing the manuscript for this book.
3. Anyway, I bring this up because it looks like Dan Brown's latest novel, "The Lost Symbol", is the first book on record that is selling better on the Amazon Kindle than its hardcover counterpart.

### **San Francisco (Place, City)**

1. Today, San Francisco is a popular international tourist destination renowned for its chilly summer fog, steep rolling hills, eclectic mix of Victorian and modern architecture and its famous landmarks, including the Golden Gate Bridge, the cable cars, and Chinatown.
2. According to the 2005 American Community Survey, San Francisco has the highest percentage of gay and lesbian individuals of any of the 50 largest U.S. cities, at 15.4
3. San Francisco Chinatown is the largest Chinatown outside of Asia as well as the oldest Chinatown in North America. It is one of the top tourist attractions in San Francisco.

### **Frankfurt (Place, City)**

1. Located on the river Main, Frankfurt is the financial capital of Continental Europe and the transportation centre of Germany. Frankfurt is home of the European Central Bank and the German Stock Exchange. Furthermore, it hosts some of the world's most important trade shows, such as the Frankfurt Auto Show and the Frankfurt Book Fair. It is also birthplace of Johann Wolfgang von Goethe.
2. Frankfurt is a city of contrasts. Wealthy bankers, students and granola drop-outs coexist in a city that has some of the highest, most avant-garde skyscrapers of Europe next to well maintained old buildings.
3. The downtown area, especially Römer square and the museums at the River Main, draw millions of tourists every year. On the other hand, many

off the beaten track neighborhoods, such as Bockenheim, Bornheim, Nordend and Sachsenhausen, with their intact beautiful 19th century streets and parks, are mostly neglected by tourism and lesser visited by tourists.

**Riesa (Place, Town)**

1. Riesa is a town in the district of Meissen in the Free State of Saxony, Germany. It is located at the river Elbe, approx. 40 km northwest of Dresden.
2. Riesa is well-known for its pasta, which is produced in the Teigwaren Fabrik Riesa
3. One town boosts east German image: Riesa exemplifies the region's new formula for growth: flexible labor and stars like Muhammad Ali.

**Gilroy (Place, Town)**

1. Gilroy is the southernmost city in Santa Clara County, California, United States, and in the San Francisco Bay Area. The population was 41,464 at the 2000 census.
2. Gilroy is well known for its garlic crop; the Gilroy Garlic Festival which occurs annually, featuring various garlicky foods including garlic ice cream.
3. Gilroy's nickname is "Garlic Capital of the World," although Gilroy does not lead the world in garlic production.

**Luxora (Place, Village)**

1. Luxora is a town in Mississippi County, Arkansas, United States. The population was 1,317 at the 2000 census.
2. The unemployment rate in Luxora is 14.20 percent(U.S. avg. is 8.50%). Recent job growth is Negative. Luxora jobs have Decreased by 0.40 percent.
3. The median home cost in Luxora is \$53,450. Home appreciation the last year has been -3.80 percent. Compared to the rest of the country, Luxora's cost of living is 32.62% Lower than the U.S. average.

**Barack Obama (Person, Politician)**

1. In December 2007, Money magazine estimated the Obama family's net worth at \$1.3 million. Their 2007 tax return showed a household income of \$4.2 million-up from about \$1 million in 2006 and \$1.6 million in 2005-mostly from sales of his books.
2. Barack Hussein Obama II (born August 4, 1961) is the 44th and current President of the United States. He is the first African American to hold the office, as well as the first president born in Hawaii.
3. Obama is the fourth U.S. president to be awarded the Nobel Peace Prize. He is the third to become a Nobel laureate during his term in office, and the first to be recognized in the first year of his presidency.

**Amy MacDonald (Person, Musician)**

1. Amy MacDonald is a self-taught musician, playing her father's guitar after being inspired by Travis at the T in the Park Festival in 2000, where she heard Travis' song "Turn" and wanted to play it herself.
2. The thrill of topping the U.K. album charts has waned for Scottish singer Amy MacDonald, because she hasn't made a penny from her number one record.
3. Macdonald started playing in pubs and coffee houses around Glasgow at 15...

**Robin Williams (Person, Actor)**

1. Robin Williams has been very public about his own ADD. His creative comic explosions are a great way to visualize what it's like. But not all his thoughts are funny.
2. Robin Williams needs heart surgery and must cancel the remainder of his one-man comedy show, "Weapons of Self-Destruction," his publicist said Thursday.
3. During the late 1970s and early 1980s, Williams had an addiction to cocaine

**Bill Gates (Person, Founder)**

1. Bill Gates, the chairman of Microsoft and the world's wealthiest man, is heading toward his new role as a full-time philanthropist.

2. Gates has pursued a number of philanthropic endeavors, donating large amounts of money to various charitable organizations and scientific research programs through the Bill & Melinda Gates Foundation, established in 2000.
3. Time magazine named Gates one of the 100 people who most influenced the 20th century, as well as one of the 100 most influential people of 2004, 2005, and 2006.

**Reiner Kraft (Person, Researcher)**

1. Reiner is a technologist and innovator, working at Yahoo! in Sunnyvale, CA. During the past three years he managed engineering teams working on cutting edge search technology. His most recent project "Yahoo! Glue
2. One of Reiner's key strength is innovation and idea generation: He filed over 115 patent applications and continuously contributed many valuable ideas to IBM's and Yahoo!'s patent portfolio. In 2003 he has been recognized as a Master Inventor within IBM's research division – the highest achievable honor and recognition for individual contributors.
3. Furthermore, Reiner has been recognized with the TR100 award from MIT Technology Review in 2002, a prestigious award for recognizing individuals for extraordinary innovation and impact to our society.

**Yahoo Inc. (Organization, Company)**

1. Not wanting to be left completely behind, Yahoo will soon launch their own real time search engine too. But unlike Microsoft and Google, they won't be partnering with Twitter and Facebook directly for the data...
2. Consumers can now personalise their Yahoo! experience with the best of the Web; customisable apps stay with you when you're mobile.
3. For Yahoo, the move furthers the strategy under chief executive Carol Bartz to focus the company on its strengths as a producer of Web media sites, as a marketer and a leader in on-line display advertising that accompanies published Web sites and to grow its audience.

**AT&T Inc. (Organization, Company)**

1. AT&T Inc. (T) is scheduled to report the third quarter results on Thursday, October 22, 2009. In the last four quarters, the company's actual earnings exceeded the market's consensus significantly. ... In the third quarter last year, AT&T activated 2.4 million iPhones, of which 40

2. AT&T has faced a fairly steady stream of complaints from customers—the majority of those complaints coming from the dedicated legions of iPhone users. There have been complaints that the battery can't be replaced, complaints that the data service is slow, complaints that 3G access is dysfunctional, complaints that the device couldn't do MMS messaging, and more.
3. AT&T is throwing iPhone users a bone by offering MMS as of September 25, but is that enough to quiet a growing chorus of angry customers upset by what they see as shoddy service? Even though many people are pouring on ...

### **Rotary International (Organization, NGO)**

1. The Gates Foundation has awarded The Rotary Foundation a total of \$350 million in two challenge grants to help end polio. In response, Rotary has committed to raising a total of \$200 million by 30 June 2012. This effort is called Rotary's US\$200 Million Challenge.
2. Rotary has no secret handshake, no secret policy, no official creed, no secret meeting or rituals. It is an open society of men and women who simply believe in helping others.
3. Youth exchange has been one of Rotary's most successful and popular programs. Today, Rotary annually places over 8,000 students aged 15-19 with Rotarian families in another country.

### **IKEA (Organization, Company)**

1. IKEA is a privately-held, international home products retailer that sells flat pack furniture, accessories, and bathroom and kitchen items in their retail stores around the world. The company, which pioneered flat-pack design furniture at affordable prices, is now the world's largest furniture retailer.
2. Penguino is a citizen of Ikea World, a state of mind that revolves around contemporary design, low prices, wacky promotions, and an enthusiasm that few institutions in or out of business can muster. Perhaps more than any other company in the world, Ikea has become a curator of people's lifestyles, if not their lives.
3. IKEA products are identified by single word names. Most of the names are Swedish in origin. Although there are some notable exceptions, most



product names are based on a special naming system developed by IKEA in conjunction with Colin Edwards.

### **Live Like a German (Organization, Company)**

1. The Live Like a German travel guide to Germany provides you with special trips, unique activities and travel ideas, which you can't find in a typical travel guide. Sometimes these are small, but hidden gems, that all the local people know about, but as a regular tourist in Germany you may not be able to discover it easily...
2. On the Live Like a German web site a user can find hundreds of articles about specific destinations, fun trips and activities as well as fantastic restaurants and nighttime spots. Written by Germans, who know from first-hand experience what they are talking about, the texts are short and sweet and include pictures as well as links for further reading.
3. The best feature about Live Like A German probably is, it is very user-oriented and intuitive to use. Additionally to being a helpful travel guide, where a user can find information about beautiful locations and interesting sights, it can work as a personalized travel agent for their users.

## Appendix B

### Error Rates

Google Web	Relevancy	Interestingness	Curiosity
Count of observations	442	449	435
Sample mean	2.934	2.659	2.556
Sample variance	0.656	0.596	0.805
Estimate of standard error	0.039	0.036	0.043
Margin of error	0.076	0.072	0.085
Margin of error %	2.579%	2.692%	3.307%
Lower limit	2.859	2.588	2.472
Upper limit	3.010	2.731	2.641

Table B.1: Error rate for Google Web

Twitter	Relevancy	Interestingness	Curiosity
Count of observations	370	377	371
Sample mean	2.370	2.207	2.129
Sample variance	0.337	0.212	0.243
Estimate of standard error	0.030	0.024	0.026
Margin of error	0.059	0.047	0.050
Margin of error %	2.503%	2.115%	2.362%
Lower limit	2.311	2.160	2.079
Upper limit	2.430	2.254	2.180

Table B.2: Error rate for Twitter

Google Blogs	Relevancy	Interestingness	Curiosity
Count of observations	450	448	439
Sample mean	2.660	2.500	2.346
Sample variance	0.537	0.461	0.574
Estimate of standard error	0.035	0.032	0.036
Margin of error	0.068	0.063	0.071
Margin of error %	2.552%	2.521%	3.029%
Lower limit	2.592	2.437	2.275
Upper limit	2.728	2.563	2.417

Table B.3: Error rate for Google Blogs

Editorial Picks	Relevancy	Interestingness	Curiosity
Count of observations	457	453	433
Sample mean	3.637	3.477	3.349
Sample variance	0.324	0.414	0.880
Estimate of standard error	0.027	0.030	0.045
Margin of error	0.052	0.059	0.089
Margin of error %	1.439%	1.708%	2.647%
Lower limit	3.584	3.417	3.260
Upper limit	3.689	3.536	3.437

Table B.4: Error rate for Editorial Picks

WebSnippets	Relevancy	Interestingness	Curiosity
Count of observations	1930	1936	1864
Sample mean	3.375	3.156	2.814
Sample variance	0.608	0.649	0.966
Estimate of standard error	0.018	0.018	0.023
Margin of error	0.035	0.036	0.045
Margin of error %	1.031%	1.137%	1.586%
Lower limit	3.340	3.120	2.770
Upper limit	3.410	3.192	2.859

Table B.5: Error rate for WebSnippets

# Appendix C

## Prediction Models

Linear Regression Model: relevancy =

```
0.3229 * f_SearchEngine1 +
0.114 * f_SearchEngine8 +
0.0631 * f_TopLevelDomain +
0 * f_MainContentCharCount +
0.0004 * f_CharacterCount +
-0.0055 * f_WordCount +
0.0066 * f_UniqueWordCount +
-0.0051 * f_ComplexWordPercentage +
0.203 * f_SentenceCount +
0.0053 * f_WordsPerSentenceCount +
-0.0253 * f_AutomatedReadabilityIndex +
0.0256 * f_ColemanLiauIndex +
0.0424 * f_SmogIndex +
-0.0441 * f_CapitalizedWordCount +
-0.064 * f_RelatedEntityCount +
0.3991
```

Linear Regression Model: interestingness =

```
0.0011 * f_AggregatedRank +
0.26 * f_SearchEngine1 +
0.047 * f_TopLevelDomain +
0 * f_MainContentCharCount +
0.0003 * f_CharacterCount +
-0.0108 * f_WordCount +
0.016 * f_UniqueWordCount +
-0.0127 * f_ComplexWordPercentage +
```

```
0.1393 * f_SentenceCount +
0.0039 * f_WordsPerSentenceCount +
-0.0237 * f_AutomatedReadabilityIndex +
0.0354 * f_ColemanLiauIndex +
0.0514 * f_SmogIndex +
-0.0538 * f_CapitalizedWordCount +
-0.0627 * f_RelatedEntityCount +
0.1683
```

Linear Regression Model: curiosity =

```
0.002 * f_AggregatedRank +
0.2118 * f_SearchEngine1 +
0.0284 * f_PageRank +
0.0451 * f_TopLevelDomain +
0 * f_MainContentCharCount +
0.0013 * f_CharacterCount +
0.29 * f_SyllablesPerWordCount +
-0.0169 * f_WordCount +
0.0215 * f_UniqueWordCount +
-0.0155 * f_ComplexWordPercentage +
0.0985 * f_SentenceCount +
0.0038 * f_WordsPerSentenceCount +
-0.0325 * f_AutomatedReadabilityIndex +
0.0323 * f_ColemanLiauIndex +
0.0459 * f_SmogIndex +
-0.0922 * f_CapitalizedWordCount +
-0.0649 * f_RelatedEntityCount +
-0.5227
```

# Bibliography

- [1] *The automatic construction of large-scale corpora for summarization research* (1999), ACM Press.
- [2] *Statistics-Based Summarization - Step One: Sentence Compression* (2000).
- [3] ACKOFF, R. L. From data to wisdom. *Journal of Applied Systems Analysis* 16 (1989), 3–9.
- [4] AUER, S., BIZER, C., KOBILAROV, G., LEHMANN, J., CYGANIAK, R., AND IVES, Z. Dbpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, Eds., vol. 4825. Springer Berlin Heidelberg, Berlin, Heidelberg, November 2007, ch. 52, pp. 722–735.
- [5] BADI, A., KOLOMIYETS, O., LALLAH, C., KHAN, A., ZHU, M., AND GARDINER-JONES, B. State-of-the-art for entity-centric repository and authoring environment for multimedia. In *MIV'07: Proceedings of the 7th Conference on 7th WSEAS International Conference on Multimedia, Internet & Video Technologies* (Stevens Point, Wisconsin, USA, 2007), World Scientific and Engineering Academy and Society (WSEAS), pp. 226–232.
- [6] BANKO, M., CAFARELLA, M. J., SODERLAND, S., BROADHEAD, M., AND ETZIONI, O. Open information extraction from the web. In *IJCAI'07: Proceedings of the 20th international joint conference on Artificial intelligence* (San Francisco, CA, USA, 2007), Morgan Kaufmann Publishers Inc., pp. 2670–2676.
- [7] BARZILAY, R., AND MCKEOWN, K. R. Sentence fusion for multidocument news summarization. *Comput. Linguist.* 31, 3 (2005), 297–328.
- [8] BARZILAY, R., MCKEOWN, K. R., AND ELHADAD, M. Information fusion in the context of multi-document summarization. In *Proceedings of the*

- 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics* (Morristown, NJ, USA, 1999), Association for Computational Linguistics, pp. 550–557.
- [9] BORKO, H., AND BERNIER, C. L. *Abstracting Concepts and Methods*. Academic Press, Inc., 1975.
- [10] BRANDOW, R., MITZE, K., AND RAU, L. F. Automatic condensation of electronic publications by sentence selection. *Inf. Process. Manage.* 31, 5 (1995), 675–685.
- [11] BRODER, A. Identifying and filtering near-duplicate documents. 2000, pp. 1–10.
- [12] BRODER, A. Z. A taxonomy of web search. *SIGIR Forum* 36, 2 (2002), 3–10.
- [13] CHANG, C.-H., KAYED, M., GIRGIS, M. R., AND SHAALAN, K. A survey of web information extraction systems. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* (2006).
- [14] CIOS, K., SWINIARSKI, R., PEDRYCZ, W., AND KURGAN, L. Concepts of learning, classification, and regression. In *Data Mining*. Springer US, 2007, ch. 2, pp. 49–67.
- [15] DEMARTINI, G., FIRAN, C. S., GEORGESCU, M., IOFCIU, T., KRESTEL, R., AND NEJDL, W. An architecture for finding entities on the web. In *LA-WEB '09: Proceedings of the 2009 Latin American Web Congress (la-web 2009)* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 230–237.
- [16] DWORK, C., KUMAR, R., NAOR, M., AND SIVAKUMAR, D. Rank aggregation methods for the web. In *WWW '01: Proceedings of the 10th international conference on World Wide Web* (New York, NY, USA, 2001), ACM, pp. 613–622.
- [17] EDMUNDSON, H. P. New methods in automatic extracting. *J. ACM* 16, 2 (1969), 264–285.
- [18] ELMAGARMID, A. K., IPEIROTIS, P. G., AND VERYKIOS, V. S. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng.* 19, 1 (2007), 1–16.
- [19] ETZIONI, O. Moving up the information food chain: Deploying softbots on the world wide web. *AI Magazine* 18, 2 (1997), 11–18.

- [20] FRIEDRICH, C. Link analysis using apriori information. Master's thesis, Technische Universität Dresden, March 2009.
- [21] GUO, J., XU, G., CHENG, X., AND LI, H. Named entity recognition in query. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2009), ACM, pp. 267–274.
- [22] HAHN, U., AND MANI, I. The challenges of automatic summarization. *Computer* 33, 11 (2000), 29–36.
- [23] HOVY, E. Building semantic/ontological knowledge by text mining. In *COLING-02 on SEMANET* (Morristown, NJ, USA, 2002), Association for Computational Linguistics, pp. 1–1.
- [24] HOVY, E., AND LIN, C.-Y. Automated text summarization and the summarist system. In *Proceedings of a workshop on held at Baltimore, Maryland* (Morristown, NJ, USA, 1996), Association for Computational Linguistics, pp. 197–214.
- [25] JING, H., AND MCKEOWN, K. R. The decomposition of human-written summary sentences. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1999), ACM, pp. 129–136.
- [26] KOBAYASHI, M., AND TAKEDA, K. Information retrieval on the web. *ACM Comput. Surv.* 32, 2 (2000), 144–173.
- [27] KRAFT, R., CHANG, C. C., AND MAGHOUL, F. Y!Q: Contextual search at the point of inspiration. In *Proceedings of the 14th Conference on Information and Knowledge Management (CIKM)* (2005).
- [28] KRAFT, R., CHANG, C. C., MAGHOUL, F., AND KUMAR, R. Searching with context. In *WWW '06: Proceedings of the 15th international conference on World Wide Web* (New York, NY, USA, 2006), ACM, pp. 477–486.
- [29] LIN, C.-Y. Training a selection function for extraction. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management* (New York, NY, USA, 1999), ACM, pp. 55–62.
- [30] LIU, B. *Web Data Mining*. Springer, 2007.
- [31] LUHN, H. P. The automatic creation of literature abstracts. *IBM J. Res. Dev.* 2, 2 (1958), 159–165.



- [32] MANI, I. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, MA, USA, 1999.
- [33] MANI, I., AND BENJAMINS, J., Eds. *Automated Text Summarization*. MIT Press, 2002.
- [34] MANI, I., AND BLOEDORN, E. Summarizing similarities and differences among related documents. *Inf. Retr.* 1, 1-2 (1999), 35–67.
- [35] METZLER, D., DUMAIS, S., AND MEEK, C. Similarity measures for short segments of text. In *Advances in Information Retrieval*, Lecture Notes in Computer Science. 2007, ch. 5, pp. 16–27.
- [36] NADEAU, D., AND SEKINE, S. A survey of named entity recognition and classification. In *Linguisticae Investigationes* (2007), vol. 30, pp. 3–26.
- [37] NAVIGLI, R. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41, 2 (2009), 1–69.
- [38] NOY, N. F., AND MCGUINNESS, D. L. Ontology development 101: A guide to creating your first ontology. Tech. rep., Stanford University, Stanford, CA, 94305, 2001.
- [39] NUNAMAKER, J. F., JR., N. C. R., AND BRIGGS, R. O. A framework for collaboration and knowledge management. In *HICSS* (2001).
- [40] PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The pagerank citation ranking: Bringing order to the web.
- [41] PAICE, C. D. Constructing literature abstracts by computer: techniques and prospects. *Inf. Process. Manage.* 26, 1 (1990), 171–186.
- [42] RADEV, D. R., JING, H., AND BUDZIKOWSKA, M. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop on Automatic summarization* (Morristown, NJ, USA, 2000), Association for Computational Linguistics, pp. 21–30.
- [43] ROWLEY, J. The wisdom hierarchy: representations of the dikw hierarchy. *J. Information Science* 33, 2 (2007), 163–180.
- [44] SALTON, G., AND MCGILL, M. J. *Introduction to modern information retrieval*. McGraw-Hill, New York, 1983.
- [45] SALTON, G., SINGHAL, A., MITRA, M., AND BUCKLEY, C. Automatic text structuring and summarization. *Inf. Process. Manage.* 33, 2 (March 1997), 193–207.

- [46] SAUPER, C., AND BARZILAY, R. Automatically generating wikipedia articles: a structure-aware approach. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1* (Morristown, NJ, USA, 2009), Association for Computational Linguistics, pp. 208–216.
- [47] SCHANK, R., AND ABELSON, R. *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Earlbaum Assoc, 1977.
- [48] TAYLOR, R. S. The process of asking questions. *American Documentation* 13, 4 (1962), 391–396.
- [49] URBANSKY, D. Webknox: Web knowledge extraction. Master’s thesis, Technische Universität Dresden, 2009.
- [50] VAN RIJSBERGEN, C. J. *Information Retrieval*. Butterworth, 1979.
- [51] VON BRZESKI, V., IRMAK, U., AND KRAFT, R. Leveraging context in user-centric entity detection systems. In *CIKM (2007)*, ACM, pp. 691–700.
- [52] YATES, A., CAFARELLA, M., BANKO, M., ETZIONI, O., BROADHEAD, M., AND SODERLAND, S. Textrunner: open information extraction on the web. In *NAACL '07: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations on XX* (Morristown, NJ, USA, 2007), Association for Computational Linguistics, pp. 25–26.