

PHD THESIS

for obtaining the academic degree of
DOKTOR-INGENIEUR (DR.-ING.)

Community based Question Answer Detection

Klemens Muthmann
born 04/27/1984 in Löbau

First Reviewer: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

External Reviewer: Prof. Dr. Alexander Löser (BEUTH Hochschule für Technik Berlin)

Submitted on 10/09/2013

Acknowledgment

This thesis was only possible because of the support of many great persons.

My supervisor Professor Alexander Schill read many early drafts of the individual chapters and provided valuable feedback. He was very patient in times I did not make as much progress as I was supposed to and he always allowed me much room in between my other tasks to finish this thesis

I also got a lot of assistance from my colleagues in Dresden. Among them, David, Philipp, Miranda and Daniel are the ones who provided me with the most feedback. Learning to know these people was a great time. Thank you for your help and your support.

Finally, I would like to thank my family for the support I received over the years. My grandparents, even though they could not witness this project anymore, helped me acquire the skills I needed to finish it. My parents always kept me motivated, and tried to interfere not too much with the actual process of writing.

Lastly but most importantly, big thanks go to my great girlfriend Anja who stayed by my side all these years and provided me with enough room to finish the thesis. For her it meant spending many evenings or weekends without me, which she endured with little complaints.

There are many other people who supported me over the years. As there is not enough room to mention them all, I still want to thank them for all the backup I received.

Abstract

Each day, millions of people ask questions and search for answers on the World Wide Web. Due to this, the Internet has grown to a world wide database of questions and answers, accessible to almost everyone. Since this database is so huge, it is hard to find out whether a question has been answered or even asked before. As a consequence, users are asking the same questions again and again, producing a vicious circle of new content which hides the important information.

One platform for questions and answers are Web forums, also known as discussion boards. They present discussions as item streams where each item contains the contribution of one author. These contributions contain questions and answers in human readable form.

People use search engines to search for information on such platforms. However, current search engines are neither optimized to highlight individual questions and answers nor to show which questions are asked often and which ones are already answered.

In order to close this gap, this thesis introduces the *Effingo* system. The *Effingo* system is intended to extract forums from around the Web and find question and answer items. It also needs to link equal questions and aggregate associated answers. That way it is possible to find out whether a question has been asked before and whether it has already been answered. Based on these information it is possible to derive the most urgent questions from the system, to determine which ones are new and which ones are discussed and answered frequently. As a result, users are prevented from creating useless discussions, thus reducing the server load and information overload for further searches.

The first research area explored by this thesis is forum data extraction. The results from this area are intended be used to create a database of forum posts as large as possible. Furthermore, it uses question-answer detection in order to find out which forum items are questions and which ones are answers and, finally, topic detection to aggregate questions on the same topic as well as discover duplicate answers. These areas are either extended by *Effingo*, using forum specific features such as the user graph, forum item relations and forum link structure, or adapted as a means to cope with the specific problems created by user generated content. Such problems arise from poorly written and very short texts as well as from hidden or distributed information.

Contents

Acknowledgment	III
Abstract	V
1 Introduction	1
1.1 Types of forums	2
1.2 Definition of a Web Forum	2
1.3 Scenario and Use-case	4
1.3.1 The Effingo Vision	5
1.4 Research Theses	8
1.4.1 Post Aggregation	8
1.4.2 Question/Answer Identification	8
1.5 Research Questions and Outline	8
2 Basics	11
2.1 Evaluation	11
2.1.1 Measures	11
2.1.2 Cross-Validation	13
2.2 Information Retrieval	13
2.2.1 Bag of Words Model	14
2.2.2 The tf-idf Model	14
2.2.3 Kullback Leibler Divergence / Information Gain	15
2.2.4 Cosine Similarity	15
2.3 Classification Algorithms	16
2.3.1 Naïve Bayes Classification	18
2.3.2 Perceptron Algorithm	19
2.3.3 Support Vector Machine	19
2.3.4 Bagging	21
2.3.5 Decision Tree	22
2.3.6 Rule Based Classification - Ripper Algorithm	23
2.4 Summary	24

3	Forum Data Acquisition	25
3.1	Overview of the Focused Forum Crawler Process	26
3.2	Structure of an HTML page	29
3.3	Web Forum Retrieval	30
3.4	Features for Focused Forum Crawling	31
3.4.1	Page Preprocessing	33
3.4.2	HTML Structure	37
3.4.3	Uniform Resource Identifier	38
3.4.4	Metadata	38
3.4.5	Interactive Elements	39
3.4.6	Page text	40
3.5	Performance of Focused Forum Crawling	41
3.5.1	Dataset	42
3.5.2	Experiments	42
3.5.3	Result Discussion	43
3.6	Forum Page Clustering	45
3.6.1	Existing Approaches	45
3.7	Features for Forum Page Classification	46
3.7.1	Page URL	46
3.7.2	Link-to-Text Ratio	47
3.7.3	Height of List Elements	47
3.7.4	Page Flipping Links	48
3.8	Results of Forum Page Classification	49
3.8.1	Dataset	49
3.8.2	Experiments	50
3.9	Forum Data Extraction	51
3.9.1	Extracted Entities	51
3.9.2	Overview of Data Extraction Approaches	56
3.9.3	Unsupervised Data Extraction	56
3.9.4	Existing Forum Data Extraction Approaches	61
3.10	Forum Data Extraction Algorithm	61
3.10.1	Segmentation	63
3.10.2	Post Identification	65
3.10.3	Split/Merge/Refine - Algorithm	72
3.10.4	Template Detection - Algorithm	74
3.10.5	Tag Path Classification	76
3.10.6	Thread-Identifier Extraction	78
3.10.7	Summary	80
3.11	Evaluation of Forum Data Extraction	80
3.11.1	Post Identifier Extraction	81

3.11.2	Extraction Patterns	81
3.11.3	Entity Classification	82
3.11.4	Thread Identifier Extraction	83
3.11.5	Comparison to Existing Approaches	83
3.12	Summary	85
4	Finding Question and Answer Posts	87
4.1	Question/Answer Detection	87
4.1.1	General Question/Answer Detection	88
4.1.2	Speech Act Analysis	93
4.1.3	Question/Answer Detection in User Generated Content	96
4.1.4	Automatic Question/Answer Systems	101
4.1.5	Summary	101
4.2	Contribution Typology	102
4.2.1	Data Used to Identify the Forum Contribution Typology	103
4.2.2	Contribution Types	103
4.2.3	Preparing the Dataset	106
4.2.4	Calculation of Inter-Annotator Agreement	107
4.3	Feature Engineering for Question/Answer Detection	109
4.3.1	Token Level features	109
4.3.2	Sentence Level Features	111
4.3.3	Post Level Features	112
4.3.4	Thread Level Features	113
4.3.5	Forum Level Features	116
4.3.6	Summary	117
4.4	Experiments	117
4.4.1	Experimental Setup	117
4.4.2	Classifier Selection	119
4.4.3	Feature Selection	120
4.4.4	Results and Discussion	128
4.4.5	Comparison of Results from Cong, Wang, Lin, Song & Sun (2008) and Hong & Davison (2009) to Effingo	135
4.4.6	Discussion	141
4.5	Conclusion	142
5	Example Applications	145
5.1	Near-Duplicate Question Aggregation	145
5.1.1	Topic Detection	146
5.1.2	Question Aggregation	147
5.1.3	Evaluation	150

5.1.4	Summary	154
5.2	Other Applications	154
5.2.1	Stakeholders	155
5.2.2	Applications	156
5.3	Summary	159
6	Conclusion	161
6.1	Summary	161
6.1.1	Focused Forum Crawling and Extraction	163
6.1.2	Classification of Forum Contributions	165
6.1.3	Dataset for Forum Contribution Types	166
6.2	Future work	166
A	Dataset Details	173
B	Bibliography	175

1 Introduction

Since the time Berners-Lee & Mark (2000) proposed the Semantic Web and postulated that “[t]he ‘intelligent agents’ people have touted for ages will finally materialize” much work has been done on making the World Wide Web machine processable. The Internet, and especially the World Wide Web, is a huge collection of a large part of the knowledge available to humankind. However, in contrast to databases vast parts of this knowledge are still not intended to be used for automatic information querying. This is particularly true for content created by often technically inexperienced users in the form of free text. Even though this free text might be based on grammatical and lexical rules of natural language, it usually does not follow any consistent formal schema. It therefore is not machine processable and because there is so much of this unstructured user generated content available, it is not an easy task to find out whether a piece of information already exists or not.

Most user generated content is located in systems like Facebook¹, Twitter² or so called Web forums, bulletin boards or Question/Answer sites. Such systems are summarized as social media sites, because their content is produced by the community of all Web users. A large part of such content is located in Web forums. A forum is used to discuss current topics or to ask questions about specific problems. Especially Question/Answer sites like StackOverflow³ contain a huge number of high quality questions and their correct answers. Like the Web, a forum is a growing collection of information and although it is possible to answer many questions instantly using the information available, the information overload often hides relevant discussions from their audience. In order to address this problem, companies like Google or Microsoft created search engines which crawl the Web and try to point people to the information they seek. However, since search engines—either the forums’ or global ones—usually only match keywords to documents, they are not optimized to provide users with the answer they seek. So even though information might be available, users are not able to find it and thus repeatedly create new discussion threads about similar topics. Thereby, the forums grow even faster and get more and more confusing, leading to more and more duplicated information—a vicious circle.

Although there are several algorithms which are able to find similarity relations in databases or the Web, these algorithms either only capture very close text similarity or very broad definitions of topical similarity. In addition, most of them work on whole documents such

¹<http://facebook.com>

²<http://twitter.com>

³<http://stackoverflow.com/>

as Web pages rather than individual forum posts and none of them are focused on finding similar questions in online forums. Therefore, this thesis presents *Effingo*, a system which is able to find similar questions in Web forums and provide an aggregated view on the answers that are available for an aggregated question on the Web.

In order to introduce *Effingo*, this chapter starts with a definition of Web forums and similar systems constituting the research domain for the rest of the thesis. It continues with an application scenario and the vision of the system, and closes with a description of research tasks in the form of two theses and four research questions.

1.1 Types of forums

No forum is quite like any other. Indeed, there are several types of forums.

Very often one encounters a forum with short threads, opened by many different users, but answered by only a few. These are typical *Question/Answer forums*, usually containing one question and often a direct answer per thread. Note that such forums might be Question/Answer sites such as <http://stackoverflow.com> but there are also classical forums showing that user behaviour such as <https://forums.oracle.com/>. If there are more than two contributions in such a forum, they usually contain clarification requests or thank you contributions.

Other forums contain very long threads with discussions that are updated for months until interest in the topic fades. Such forums usually collect updates on some event or topic. They are using a forum engine, but they are more like a *collaborative Blog* or *Wiki*. One example for such forums is the German forum on www.schaesoft.de. On this forum, users discuss about plugins to “The Elder Scrolls” games. Each plugin has its own thread and all questions about that specific plugin are grouped into the same thread.

This work is going to concentrate on the type of Question/Answer forums.

1.2 Definition of a Web Forum

The following paragraph explains and defines online discussions and sets them within the context of the social Web and the Web 2.0. Figure 1.1 shows a model for streams of user generated Web items. The model does not only fit Web forums, but also news feed streams, Twitter streams, Facebook messages etc. Such streams are quite similar to Web forums and the techniques discussed in this work might be applicable to them as well. Without loss of generality this thesis is going to focus on Web forums as one source for many item streams on similar topics.

The model in Figure 1.1 shows that an online discussion—commonly referred to as *thread*—is an *item stream* consisting of several *items* also called *posts*, forming the discussion flow. Within each single thread, there is a specific start post which opens the discussion. The rest

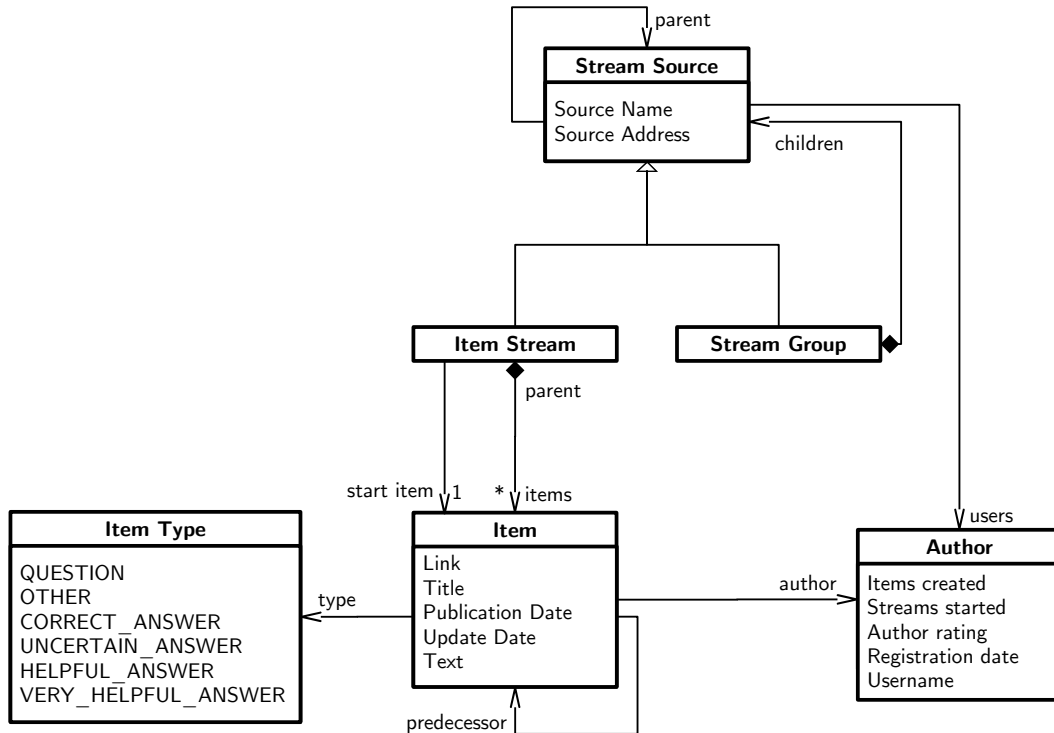


Figure 1.1: Typical structure of a Web forum.

is usually ordered by creation date. Each post, except for the first one, has a predecessor. Note that two items might have the same predecessor. That way, posts form a tree structure. This models the case that discussions may split and continue in different directions. Such a feature is not supported by all online discussion platforms and is mostly used in classical forums which present one contribution per Web page. Another type are bulletin boards, showing multiple posts on each page usually ordered by their time of creation.

Some online discussion pages support the rating of answers and the marking of the correct answer. Therefore, each item belongs to a specific category depending on whether it is a question, an answer of a certain type (i.e. correct answer, uncertain answer, helpful answer or very helpful answer) or some other contribution. This is one of the central features of Question/Answer sites. Hence, Question/Answer sites usually do not order answers by publication date but rather by answer quality with the highest ranking answer at the top. They also support the discussion of answers in a special comments section per answer. This is also mapped by the tree structure of posts in Figure 1.1.

Item streams (i.e. threads) are a special kind of *stream source*. Stream sources form a tree as well, usually starting at the forum level. Sometimes the tree only has a depth of one if the forum does contain the threads directly. However, forums may also be organized into an arbitrary hierarchy of subforums. Figure 1.2 shows an example for how subforums are presented on `forum.polar.fi`. Those subforums either contain other subforums, a list of item streams or both. Since forums and subforums group item streams concerning the same topic they are called stream groups in Figure 1.1.

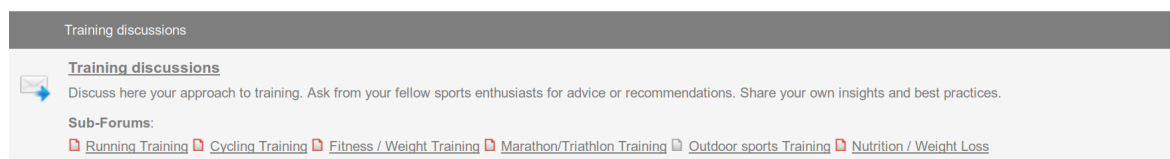


Figure 1.2: Example of a forum containing subforums on diverse topics.

Each forum has a number of *users* writing the contributions. One of those *users* is responsible as author for each item.

The following definitions describe the differences between contributions and posts as well as discussions and threads more formally. Definition 3 and Definition 4 are provided by Lin, Yang, Cai, Wang, Wang & Zhang (2009). Definition 1 and Definition 2 are created for the purpose of this work specifically.

Definition 1 Contribution *A contribution is the text a user contributes to an online community, with the intention on extending the content available to the community. Contributions are usually displayed as posts. Spam contributions are contributions as well, even though their intention is malignant.*

Definition 2 Discussion *A discussion is an exchange of user contributions via the Web. In Web forums a discussion is usually displayed in the form of a thread on a thread page.*

Definition 3 Post (Lin et al. 2009) *A post is a user-submitted content at a particular time stamp. Posts are contained in threads, where they appear as boxes one after another. If a post is not the first of the corresponding thread, it is referred to as a reply. Posts are physical entities shown on a forum thread page to present one forum contribution.*

Definition 4 Thread (Lin et al. 2009) *A discussion thread (without ambiguity we use thread hereafter) is a series of posts, usually displayed by default from the oldest to the latest.*

This means that a post is the visual representation in the form of rendered HTML code of a user's contribution. The contribution is only the content the user provided to the community and is independent of how it is presented on the forum. The same is true for the relation between a thread and a discussion.

1.3 Scenario and Use-case

Imagine the following scenario: A user is searching for a solution to a specific problem on the Internet. They enter a few keywords into a search engine and get a result list with some results from online discussions containing the keywords they entered.

If their problem is prominent or generic enough, they will get a huge amount of discussions from different forums, blogs, etc. In order to find a solution for the problem, they have an

intimidatingly extensive research task to do. They need to look through all the results, find the discussions that do not only share some keywords, but capture the queried problem and find one or several of them containing a helpful answer. If there are multiple answers they might be contradictory or complementary. The user needs the big picture to make an actual decision on how to solve the problem. The worst case scenario might be to find out that there is no solution to their problem, after having wasted several hours of reading.

If the problem is not prominent or generic enough those few people who already discussed it might have used very different sets of words. Thus, the user only finds a few pages, which contain a random subset of keywords from their query. The chance of finding at least the correct question could be increased if users formulated complete questions. However, this is not how search engines are used nowadays.

In order to solve their problem users need a system that collects all duplicates of a question and aggregates all answers. That way, if the system is able to present the user with at least one existing question equal to their own, it has the complete picture of what is going on about that question. The user therefore could skip reading all discussions and just press a big red button labeled: "just show me the answer". Since the system already knows everything about the question, users either get the one and only answer, they get several answers with similar ones grouped together and contradictory ones shown separately or they get the information that no answer to their problem was found in any discussion on the World Wide Web. In the last and worst case the system would ideally already indicate which forum has the highest probability of getting an answer to that question and would provide the user with the ability to directly post their question there. In addition, the user may participate in ongoing discussions on their questions provided by the system or they might try to solve the problem themselves, thus providing answers for future searchers. In the last case they could be sure to do meaningful work in solving the question since it has not been solved before, and thus have the good feeling of providing something to the community.

1.3.1 The Effingo Vision

In order to implement a system as described in the last section a multi-step processing of contributions should be applied as shown in Figure 1.3.

First of all the system needs to find every forum and every thread on the World Wide Web and put it into an index similar to a search engine. It also needs to keep those threads updated and continuously add new threads to the index. This is not an easy task since Effingo has no direct database access to all forums. Therefore, a focused crawler is required which crawls the Web but selects only pages from Web forums and chooses the ones containing the user generated content. From these pages an extractor needs to get data fields, thereby reconstructing the database of the forum operator.

The next step is to identify irrelevant contributions and select question and answer contributions. We define a question as follows:

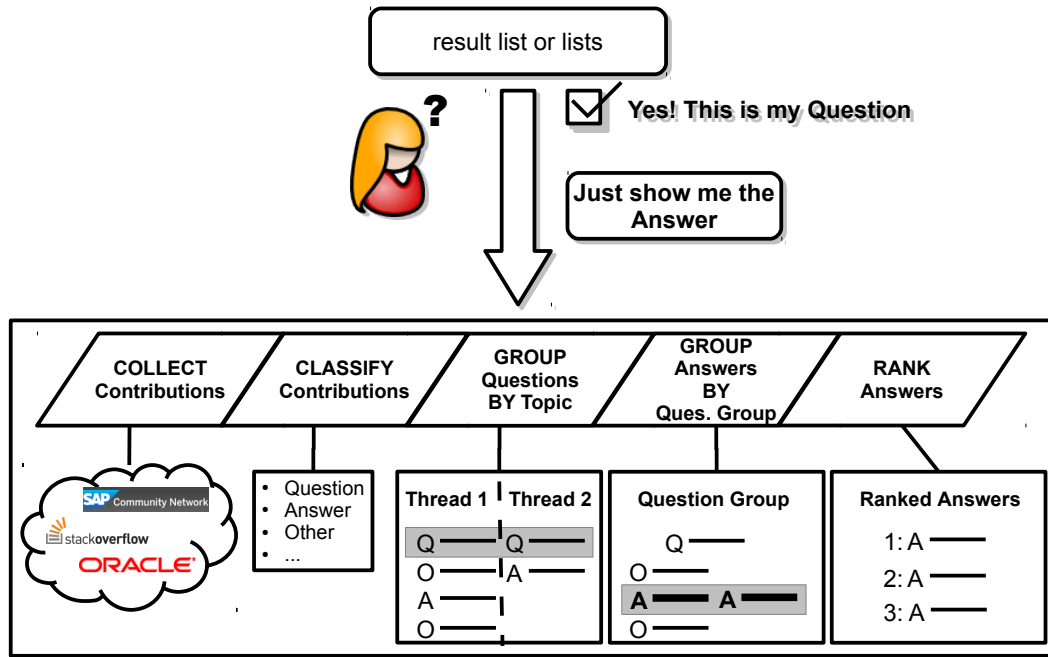


Figure 1.3: The Effingo process overview.

Definition 5 A question contribution is a contribution written by an author who is a forum user (questioning user) with an Information Need.

We also define an answer as:

Definition 6 A contribution with the intention to satisfy the information need provided by a question.

After we know which of the posts are questions and which of them are answers, we need to find equal questions. It is not easy to define what an equal question is. A first definition could be:

Definition 7 Topical Question Equality 1: Two questions are equal if there is an answer to both, satisfying both questioning users' information needs. The answer is not required to exist in the form of a forum post but it needs to be hypothetically possible.

However, it is possible to aggregate unrelated questions and still match this definition if an answer is so generic that it matches multiple questions. For example, restarting a computer is a valid solution for many problems and thus an answer such as "Turn your PC off and on again!" would group many unrelated questions. We thus need to reduce the set of answers that are valid for the definition of "Topical Question Equality". As already explained, very generic answers are not suitable even though they might be correct. However, even if a very specialized answer is available, it might still be correct for other questions, at least in some sense. In addition to an answer's precision we also require its correctness. Both span a two-dimensional space where we could point out the relation each answer has to a question.

In order to improve Definition 7 the most special and most correct answer is required. This leads to the following updated definition.

Definition 8 *Topical Question Equality 2: Two questions are equal if it is possible to find a most correct and most special answer, satisfying both questioning users' information needs. The answer is not required to exist in the form of a forum post but it needs to be hypothetically possible.*

Consider for example the question "How to improve my running time". "Eat a healthy diet!" is a correct answer for that question but also for "How do I loose weight effectively?". Since we do not want to group both questions but we might want to group the first with "How do I win the marathon?" another correct but more specialized answer might be "Do interval training while running!". This answer is very specialized and very correct for the first and the third question. It is in some sense also correct for loosing weight, but not as correct as for the other two questions.

This definition is very subjective. It is not intended to function as the basis for an algorithm, but rather as a means to provide a human evaluator with an idea of what the expected results from an algorithm should look like.

After questions are grouped, we have at least one or possibly several threads associated with each question. Hopefully, at least one thread per question contains an answer. If there are multiple answers, some of them might be topical duplicates as well although topical duplicates might be just as hard to define as for equal questions. The following definition will be used:

Definition 9 *Topical Answer Duplication: An answer A is a topical duplicate of another answer B if both are answers to the same question and if A provides no additional information over B for answering their shared question.*

Those duplicates are grouped during the fourth step from Figure 1.3.

Some of the answers might be wrong, some might be of low quality, others might be incomplete but hopefully some of them are correct and helpful to all users with the same question. In order to improve the visibility of correct high quality answers, a ranking is applied on all answers per question group in a final step.

The process as presented here would require to preprocess a complete corpus of forum contributions and store the grouped questions with their ranked answers statically. Adding new questions would mean reprocessing the whole corpus again. A better solution would be an online system, which carries out steps three to five for each user query. Such a system would continuously crawl the Web and extract forum content and classify contributions (i.e. steps one and two). For each user query it would try to find the most relevant questions, group them according to step three and wait for the user to choose the correct question. It then would prepare the answers from that question group, according to steps four and five. The focus of this work will be steps one "COLLECT Contributions" and two "CLASSIFY Contributions". Chapter 5 provides a brief overview and a simple approach to group questions and answers. There are algorithms for step five, which can provide substitutes for now.

For example Agichtein, Castillo, Donato, Gionis & Mishne (2008) provide a way to assess the quality of a forum contribution which may be used for ranking contributions.

1.4 Research Theses

The work is based on two theses presented in this section. Both are structured into a claim and a thesis. The claim is a prerequisite. Only if the claim is true we will be able to prove the thesis.

1.4.1 Post Aggregation

The first thesis is about extracting posts automatically from all over the Web.

Claim 1 Even though user generated questions and answers are distributed over different platforms on the World Wide Web, all of them follow a common basic schema.

Thesis 1 It is possible to extract the data, common to all user generated questions and answers from all over the Web automatically and with a low margin of error.

1.4.2 Question/Answer Identification

The second thesis deals with the automatic identification of questions and answers from a Web forum. It consists of the following claim and thesis:

Claim 2 Forums on the World Wide Web contain user posts with the intent to ask a question or to answer a question.

Thesis 2 It is possible to separate posts—with the intention of asking a question and answering a question—from each other and from posts with some different intention automatically and with high reliability.

The two theses cover step one and step two from Figure 1.3. There are three reasons for setting the focus on these two. First, the research areas of topic detection, question answer linking and answer ranking are well established. There are surely some research topics but our examinations of existing works showed that solutions for the first two steps are not as clear. Second, starting with the first steps removes the necessity of simulating the results from unknown steps. Starting with a later step would us to generate the results of prior steps using some naïve algorithm. Third, focusing on a small number of steps allows us to put more effort into each individual step, instead of doing a superficial analysis of all five.

1.5 Research Questions and Outline

This section provides an overview of the research questions this thesis is going to answer as well as a brief outline of the following chapters.

Following this introduction, Chapter 2 gives a brief summary of basic terms and algorithms used throughout this work. If you are well versed in machine learning you may safely skip this chapter.

After Chapter 2 the core chapters show validity of the theses discussed in the previous section. They are raising several research questions. For thesis 1 we need to find and extract Web forum posts automatically. Therefore the first research questions is:

1: How to identify a Web forum among other Web sites?

The second question asks for the data available on Web forum sites and how to access it:

2: What is the common schema for user generated questions and answers on the World Wide Web and how can we extract it from HTML source code?

Answers to both questions are discussed in Chapter 3. The chapter starts by describing a solution for question 1 from Section 3.3 to 3.5 and thereafter handles question 2 from Section 3.6 to 3.11.

For thesis 2 we claim that there are question and answer contributions in Web forums. In order to demonstrate this, it is necessary to identify the contribution types encountered in a Web forum and prove that questions and answers are among them. The third research question thus is:

3: Which typology is appropriate for user generated content in online forums?

It should be possible to show the answer to this question by examining a large slice of content from some big forum. The answer to this question is discussed in Section 4.2 of Chapter 4.

Having such a typology, questions and answers must be separated from other contribution types. So we need to find an answer to the question:

4: Is there an approach to map user contributions into a forum contribution typology automatically?

This answer is discussed in Chapter 4. A machine learning approach, as explained in Section 4.3, is used as a solution and evaluated on a large corpus of real world forum data in Section 4.4.

After answering those research questions, Chapter 5 provides a short overview of possible applications of the proposed algorithms and Chapter 6 concludes with a summary and an outlook on future work.

2 Basics

This chapter presents some basic techniques which are applied throughout this thesis to develop solutions for the research questions discussed in the last chapter. The research questions are concerned with how to find and extract forum posts on the Web and assigning types based on each posts content. Therefore, this chapter presents basics from the areas of information retrieval and extraction as well as classification with a focus on supervised classification methods. However, the chapter starts with some of the most common evaluation techniques used to measure the success of algorithms from those research areas. These measures are used in the following chapters to show how well the presented concepts work. The chapter continues with established information retrieval techniques. Such techniques are used in state of the art algorithms for forum data retrieval and question answering. Finally, the chapter briefly presents some of the most common classification algorithms. Those algorithms are used in Chapter 3 and 4.

2.1 Evaluation

The following sections present the principles and measures used for evaluating the degree of success of the theses from Chapter 1.4. Since most solutions presented in this document are based on classification, the sections concentrate on typical classification measures. These measures are frequently used in related work and thus increase comparability.

2.1.1 Measures

There are multiple measures available to assess the quality of an information retrieval or classification algorithm.. They are based on the idea that the goal of each algorithm is to identify a set of information entities—often called documents—from among a larger set of similar or less similar entities. Examples are the retrieval of a number of Web pages based on a user’s query from a larger set of Web pages or the classification of a text as either “spam” or “no spam”. There are two possible error cases as well as two possible indicators for success for such problems. An error is either an element in the result set which does not belong there or an element which should be included in the result set but is not. The first is usually called a false positive (FP) while the second is a false negative (FN). Correct results are either items that are correctly included in the result set also called true positives (TP) or items that are correctly excluded from the result set, called true negatives (TN). Figure 2.1 visualizes these

sets as subsets of the population the algorithm works on.

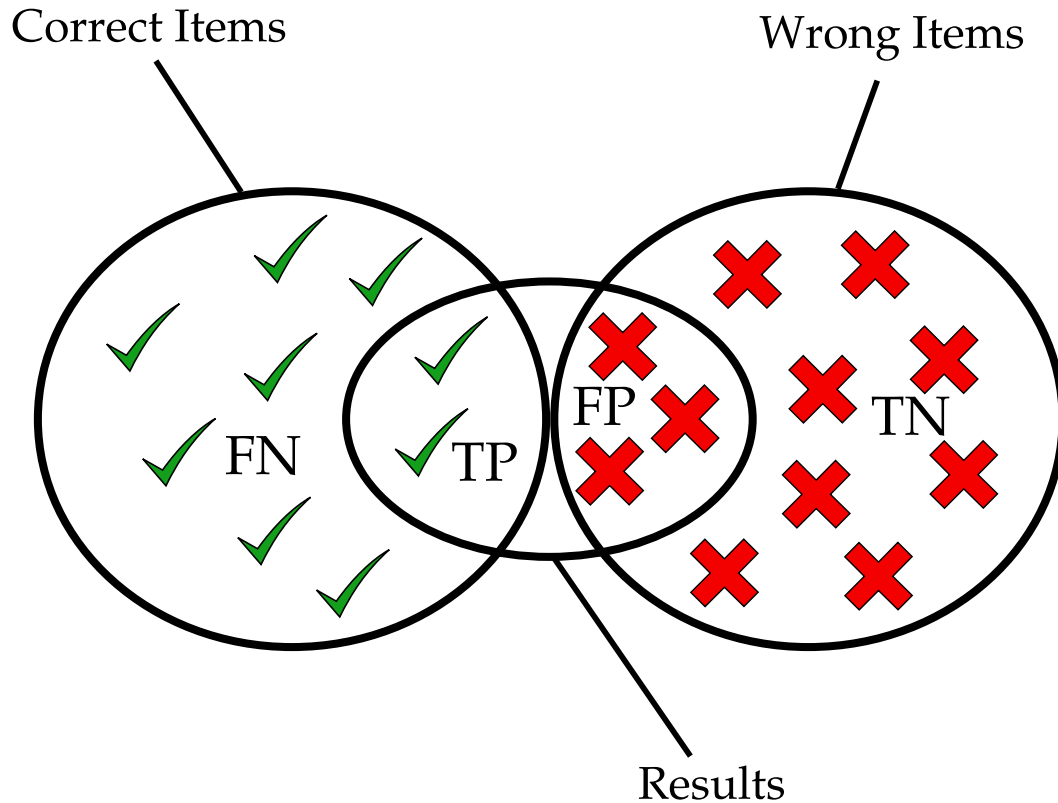


Figure 2.1: Result types used to evaluate information retrieval and classification algorithms.

Typical measures calculated from this layout are accuracy, recall, precision and F -Measure. Accuracy calculates the ratio of good results among all items. It therefore follows Equation 2.1.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

The biggest problem with accuracy is that it easily creates good values for populations containing few positive items. So if an algorithm searches for Web pages from the set of all pages called $pages$ and only five of all pages on the Web are correct results, that algorithm can achieve very high accuracy (close to 1.0) by simply discarding all pages as wrong. In this case, TP becomes 0 and TN is the number of pages minus 5. Thus, Equation 2.1 boils down to $\frac{|pages|-5}{|pages|}$. Assuming $|pages|$ is much larger than 5, equation 2.1 results in close to 1. For this reason, the measures of precision and recall are introduced. Precision results from the good entities among all resulting entities while recall is the proportion of good entities among all expected ones. Precision is calculated using Equation 2.2 and recall is calculated using Equation 2.3.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.3)$$

In addition, there is a measure to avoid having to look at two numbers for every algorithm result. This measure is called F -measure as in “Fallout”, because it describes the fallout of the algorithm. The F -measure is the harmonic mean of precision and recall, as shown in Equation 2.4.

$$F_1 = 1 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.4)$$

Actually, Equation 2.4 is a special case called the F_1 -measure, applying equal weight to recall and precision. The general form is presented in Equation 2.5.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \quad (2.5)$$

Using this equation it is possible to apply more weight to either precision or recall. Common weights are the $F_{0.5}$ -measure or the F_2 -measure with values of 0.5 or 2 for β respectively.

2.1.2 Cross-Validation

Some classifiers and some classification tasks show a high variance depending on the training set used to build the classifier. In order to calculate meaningful performance measures for such classifiers, multiple runs using different subsets of the gold standard are required. The typical approach is called cross-validation. Cross-validation is parameterized with a number of folds for the number of runs. So a 5-fold cross-validation does 5 runs over the gold standard set, while a 10-fold cross-validation does 10 runs. For a test on the whole dataset, each fold uses a different training and test set. This depends on the number of folds. The input gold standard data set is divided by the number of folds. Each part is used as a test set for one run, while the remaining input items are used for training purposes. So for a 10-fold cross-validation the input set would be split into 10 equally sized partitions and the final performance measures are calculated from the results of 10 runs using each of the 10 test sets exactly once as test set. A typical value for the number of folds according to the literature, for example McLachlan, Do & Ambroise (2005, p. 214), is 10. However, using only 5 folds produces reasonable results as well. Reducing the number of folds to 5 is for example necessary if single runs are too expensive to carry out 10 of them.

2.2 Information Retrieval

The discipline of information retrieval aims at retrieving documents from a larger collection of documents with arbitrary data thus satisfying some specific information need. The most common implementation of an information retrieval system is a search engine like Google¹. Such a search engine uses the World Wide Web as a document corpus. The user enters his information need, using a free text query to the search engine which retrieves matching

¹<https://www.google.com/>

documents, using an index of documents combined with some kind of similarity metric. More sophisticated information retrieval systems provide the particular piece of information the user is searching for instead of the document containing the information. Such systems are usually built around information extraction systems which collect information and try to understand it by applying semantic data structures like ontologies.

The Effingo system aims to be a second grade information retrieval system. It provides forum entries extracted from forum pages to satisfy user information needs. For this purpose some basics of information retrieval are briefly explained in the following.

2.2.1 Bag of Words Model

The Bag of Words Model is one way to index documents for an information retrieval system. It is also one of the most common approaches. The Bag of Words Model assumes each document to be a bag of words, while ignoring the word order. Each document is indexed under each of its words. In the pure Bag of Words Model the occurrence of each word is counted. Documents are retrieved based on the comparison between a user query and the documents' bag of words. Each word from the user's query is compared to the words in each document. Documents are ranked higher if they contain one of the words from the user's query with a higher frequency. This measure is called token frequency (tf).

A special case for the Bag of Words Model is the Set of Words Model. In this even simpler model only the presence of a word is important, not its frequency of occurrence. The advantage of this is that very common words are not highlighted that much and do not mask uncommon but specific words. As a disadvantage, documents containing a very specific word multiple times are not ranked higher than documents containing the same specific word only once.

2.2.2 The tf-idf Model

The token frequency/inverse document frequency (tf-idf) model is slightly more complex than the Bag of Words Model. However, it is the basis for the indices used by most major search engines. It combines the advantages of the Bag of Words and Set of Words Model while avoiding the disadvantages of both. It nevertheless features the big disadvantage of the need to know the whole vocabulary in advance. This is usually not possible for expanding document collections. It is however possible to update the vocabulary once in a while and still get reasonable values with a small error margin.

The tf-idf model is based on two measures. The token frequency (tf) is the number of occurrences of a token within a document as shown in Equation 2.6, where $f(t, d)$ is a function counting all occurrences of term t in document d . The inverse document frequency is the inverse of the number of documents a token occurs in, shown in Equation 2.7. It is common practice to apply a logarithmic scale as a dampening factor for very common or very

uncommon terms. So the tf-idf measure may be calculated according to Equation 2.8.

$$tf(t, d) = f(t, d) \quad (2.6)$$

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad (2.7)$$

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) \quad (2.8)$$

That way, the tf-idf model raises the importance of very specific terms frequently occurring in only a few documents and lowers the importance of unspecific terms occurring in most documents.

2.2.3 Kullback Leibler Divergence / Information Gain

The Kullback Leibler Divergence (Kullback & Leibler 1951) is a measure of the difference of two probability distributions. In information retrieval it is used to calculate the similarity of two documents based on language models calculated from the words occurring in both documents. Each language model defines a probability distribution over the occurrence probability of the words from both documents. So, assuming there are two language models defined by the probability distributions P and Q , the Kullback Leibler Divergence may be calculated using Equation 2.9

$$D_{KL}(P \parallel Q) = \sum_i \ln \left(\frac{P(i)}{Q(i)} \right) P(i). \quad (2.9)$$

In case of both models being equal, the divergence is equal to one. D_{KL} approaches zero as the difference between P and Q increases.

The Kullback Leibler Divergence is also known as Information Gain which is the basis for the Query Likelihood ranking. Query Likelihood ranking is the most simple ranking algorithm and enjoys wide coverage among researchers in Information Retrieval.

Information Gain is also used to evaluate the use of some attribute v with values $v_1 \dots v_n$ in distinguishing documents. In this sense, it is a ranking scheme arranging attributes as to how well they distinguish different document classes. The query language model is created from the possible values of attribute v in this case, while the document language model is created from the true distribution of the attribute values over a training set. This approach is for example used to build decision trees as explained in Section 2.3.5.

2.2.4 Cosine Similarity

Another similarity metric used to compare documents in information retrieval is the cosine similarity. It is based on the so called vector space model. The vector space model is based on the values calculated for each word using the bag of words or tf-idf model. Those values form a vector representing a document in the space of all words. This is a very high

dimensional space. Fortunately every document only contains a few of those words. The value for all other words is zero. Such a vector is therefore called a sparse vector. The cosine similarity is calculated as the cosine of the angle θ between two document vectors \vec{v} and \vec{u} using equation 2.10.

$$\cos(\theta) = \frac{\vec{v} \cdot \vec{u}}{\|\vec{v}\| \|\vec{u}\|} \quad (2.10)$$

There is a big difference between cosine similarity and the common euclidean distance with regard to the similarity calculation. It becomes apparent when you scale one of the two vectors that are compared. If you scale up one vector by a factor of two, the distance between the two documents doubles and so their similarity is halved using euclidean distance. The angle between both vectors stays the same and so, scaling has no influence on cosine similarity. This means that if you take a document and append it to itself and compare the appended version to the original version, both documents will still be identical using cosine similarity, whereas the distance will increase using euclidean distance. In practice this means if one document is very large and contains a term frequently it can still be very similar to a small document where the same term occurs only once. This is important for information retrieval, since often a very short user query is compared to a long result document.

2.3 Classification Algorithms

This section provides an overview of the classification algorithms used in this work. For additional information see for example the book by Manning, Raghavan & Schütze (2009, ch. 13, 14, 15) or Mitchell (1997).

This is not intended as a complete overview of classification methods but rather a selection of some of the more prominent approaches and a brief introduction to the algorithms used in this work and by authors of related publications. The explanations are focused on supervised algorithms. This means that all of the presented algorithms rely on human labeled training data to build a classifier for predicting unlabeled data.

Classification is about searching a function based on some training examples in some multi-dimensional space spanned by the features of the training examples. The training examples are also called instances. Every classification algorithm is based on the assumption that this function generalizes to unseen instances and thus allows an accurate prediction of their value. The function, called hypothesis or classifier, might be discrete valued or continuous valued. Discrete valued functions are most important for this work.

It is almost impossible to create a hypothesis for the training examples and further yet unseen examples from the same population of data items. As a consequence, there is always a tradeoff between either fitting the training examples and the quality of the classifier's prediction of unseen items. If a classifier tends to trend to one or the other direction, it is either called an underfitting or an overfitting classifier. Classification algorithms are arranged according to how much they tend to overfit or underfit training data, which is also called the

variance or the bias of the classifier. This means that some classification algorithms tend to produce overfitting classifiers, while others tend to create underfitting ones. Even though this thesis is focused on discrete valued classifiers, the following explains the concepts of bias and variance for a continuous valued example, since this is easier to visualize. So consider the three continuous valued hypotheses for a one-dimensional feature space (the second dimension is the predicted value) in Figure 2.2. The hypotheses represented by the

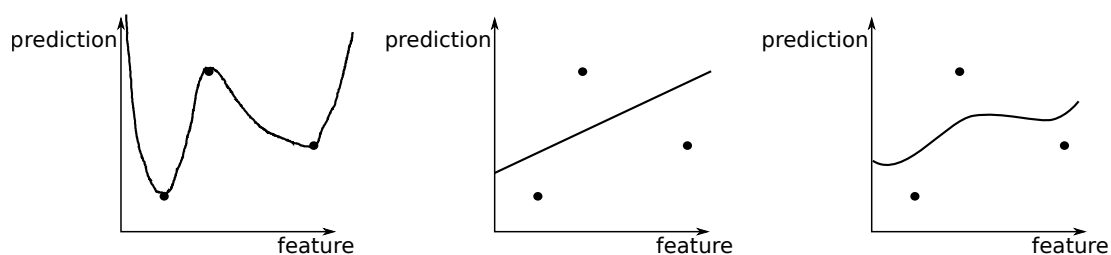


Figure 2.2: Example of a high variance classifier on the left, a high bias classifier in the middle and a balanced classifier on the right.

three functions drawn as black lines, one per picture. The black dots are the training examples used to create these classifiers. The figure shows that the hypotheses on the left and in the middle are extreme prospects of matching the three training instances. While the first will not generalize well to new examples because it matches all training instances exactly, the second will not generalize well since it is far away from all training instances. If adding new instances to the left training algorithm, the hypothesis will change extremely, while the training algorithm in the middle will hardly change at all if new instances are added. The left case is called overfitting or high variance, while the middle is called underfitting or high bias. Both cases should be avoided since otherwise, the assumption that the hypothesis will generalize to unseen test instances is not true. The classifier on the right is more desirable, since it captures the essence of the training set without fitting it exactly. Unfortunately, classifiers tend to either underfit or overfit training data under certain circumstances. However, if this tendency is known beforehand, it is possible to analyze and apply the correct classifier to the correct situation.

For evaluating classifiers the measures presented in Section 2.1.1 are used. However, in their pure form they are only applicable to evaluate a classifier's performance for a single class. In cases of multi class classifiers, it is necessary to calculate average precision, recall, accuracy and F-measure values for all classes the classifier is able to distinguish. That way, it is possible to report accumulated quality statistics for a complete classifier. There are two ways to build average measures. *Macro averaging* builds the common arithmetic mean for each value as overall performance of a classifier. That way, each class is weighted equally and if a small class produces many errors, the classifiers overall measure will decrease. If it is necessary to avoid this, *Micro averaging* weights the measures for each class with the size of that class. In contrast to macro averaging, it favors the results achieved on larger classes.

2.3.1 Naïve Bayes Classification

The Naïve Bayes classification algorithm is based on Bayes law (Bayes 1763) shown by Equation 2.11.

$$P(A|B) = P(A) \frac{P(B|A)}{P(B)} \quad (2.11)$$

It states that the probability of observing an Event A under the condition that event B was observed depends on the prior probability of observing A ($P(A)$), the probability of observing B under the condition that A was observed ($P(B|A)$) and on the probability of observing B at all ($P(B)$). Mapping this to the classification of forum entries or natural language texts A is the event that a document d_1 is assigned to a class c_1 and B is the event that a certain feature, e.g. the word “help”, occurs inside document d_1 . The prior probability that any document is assigned to c_1 is calculated from the distribution of c_1 in the training set. The probability $P(B)$ stands for the distribution of the word “help” in all the training documents (i.e., which fraction of the training set contains the term “help”). The probability $P(B|A)$ stands for the fraction of training documents containing the term “help” and belonging to class c_1 . In order to apply a Naïve Bayes classifier, these distributions are calculated for each feature using an adapted version of Equation 2.11 shown as Equation 2.12.

$$P(C = c_k | d_m) = P(C = c_k) \prod_i^n P(f_i | C = c_k) \quad (2.12)$$

With d_m being a set of features f_1, f_2, \dots, f_n representing any document; for example all the words from a document with n words.

It is important to note that the Naïve Bayes classifier assumes that all features are conditional- and position-independent which is usually not the case (Manning et al. 2009). This means Naïve Bayes cannot predict that you, for example, like movies starring only Matt Damon or Ben Affleck but not both. These assumptions result in high bias, which causes large deviations of classification probabilities. This however is negligible in practice. Experience shows that deviations usually point to the correct direction, meaning that documents belonging to a class almost always get a probability close to 100% while documents not belonging to a class get a probability close to 0%. In addition, a Naïve Bayes classifier converges quite quickly. The reason for that is that, as stated earlier, additional training examples don't change the hypothesis very much after some point for high bias classifiers. As a consequence Naïve Bayes can be used if the training set is small and the features are not tightly coupled.

An implementation of the Naive Bayes classifier has linear complexity in the time it takes to scan the training and the test data (Manning et al. 2009, p. 262) and thus is a very attractive text classification algorithm.

2.3.2 Perceptron Algorithm

The perceptron algorithm is a linear classifier, meaning it tries to find a plane separating the training data, which can be used to separate additional data points in the same space. It was originally presented by Rosenblatt (Rosenblatt 1958, Rosenblatt 1961). The input for the perceptron algorithm is a set of training instances in the form of points \mathbb{R}^n and associated labels from $-1,+1$. The perceptron algorithm iterates over this training set, starting with the prediction vector $\vec{v} = 0$. As long as the prediction vector produces correct predictions it stays alive. If a wrong prediction is encountered for a training position vector \vec{x} with label y , a new prediction value is calculated as $\vec{v} = \vec{v} + y\vec{x}$ and the iteration starts again. For the process to converge the data must be linearly separable which means that it needs to be possible to draw a straight line—or hyperplane in higher dimensional spaces—separating the positive from the negative examples. If true, predictions of the form $\Delta = \text{sign}(\vec{v} \cdot \vec{x})$ become possible.

The so called voted perceptron algorithm is one way to improve the classifier performance. This algorithm keeps all the calculated prediction vectors, even those which did not fit the training set. A weight is assigned to each prediction vector depending on how long it survived until making the first error. Predictions are made using all vectors and calculating the final result from the weighted sum of the predictions of the individual prediction vectors. This algorithm was developed by Freund & Schapire (1999). Although it shows improved performance as compared to the basic perceptron algorithm, its performance is still inferior compared to the support vector machines presented below. However, it is simpler and easier to implement.

The perceptron algorithm is the foundation for the more complex neuronal networks. Basic neuronal networks, called multilayer perceptrons, are built from many connected perceptron nodes.

2.3.3 Support Vector Machine

Support vector machines (SVM) are classifiers used to classify objects into two classes from the set of $\{-1, 1\}$. Similar to the previously presented perceptron algorithm, they try to create a separator between positive and negative examples of the training set. However, in contrast to the perceptron, an SVM finds an ideal (also called large margin) separator between instances of the training set. For this purpose, the discriminating features extracted from the objects are placed into a vector representation of these objects. The union of all features forms a vector space also called feature space. The training phase of an SVM tries to find a hyperplane inside the feature space with maximum distance from both classes according to the training set. This is shown in Figure 2.3 for a space with two features. An optimal separation is achieved by maximizing the margin between the separating hyperplane and the two feature vectors from both classes closest to each other. These two feature vectors are also called support vectors, hence the designation as support vector machines.

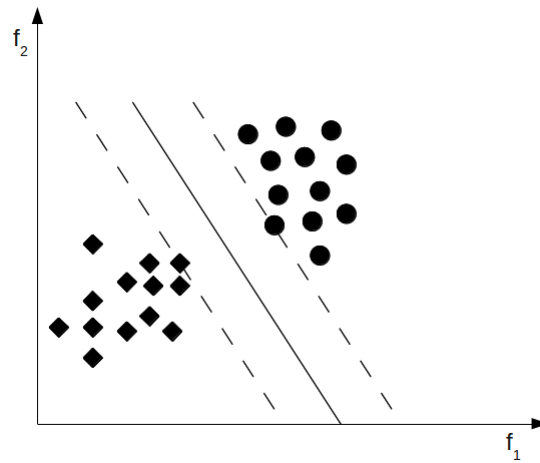


Figure 2.3: Example of SVM hyperplane for two-dimensional feature space.

Once the separating hyperplane is constructed, the classifier is able to assign new feature vectors to either class -1 or 1 by evaluating the result of Equation 2.13.

$$f(\vec{x}) = \text{sgn}(\vec{w}\vec{x} + b) \quad (2.13)$$

The question solved during the training phase is: which are the values of \vec{w} and b ? More information is provided for example by Manning et al. (2009, p. 319). An important fact to know when using support vector machines is that the optimization function used to train a support vector machine has a parameter, usually called C . This parameter is used to influence the support vector machine's tendency to either a higher bias or a higher variance. For large values of C , the trained SVM tends to overfit the training data set showing high variance, while for small values it tends to underfit showing high bias.

Support vector machines are widely used and perform exceptionally good with small sets of training data (Manning et al. 2009). On the other hand, they induce many rather arcane parameters which are required to be set correctly for the SVM to show its full potential.

The training data for a support vector machine need to be linearly separable as already explained for the perceptrons in the last section. This, of course, is not true for arbitrary datasets, at least not in the space spanned by all input features. Nevertheless, with SVMs it is possible to get a separable training set, using the so called "kernel trick". The principle is presented in Figure 2.4. The dataset with four training instances in the one-dimensional feature space represented in the left part of Figure 2.4 is not linearly separable. However, if the feature space is extended with an additional feature, e.g. the proximity to some point l in the original feature space (see middle of Figure 2.4), we get a linearly separable training set in the two-dimensional feature space shown on the right of Figure 2.4. This principle is applicable to arbitrary dimensional data. The function sim is called the kernel function. A linear support vector machine uses a linear kernel function. Other typical kernel functions are the polynomial kernel function and the gaussian kernel function (also called RBF for

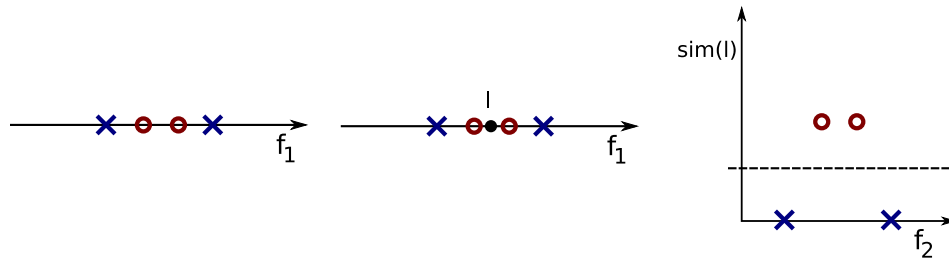


Figure 2.4: Example of the SVM kernel trick applied to a one-dimensional training set with four training instances, two positive and two negative ones.

Radial Basis Function). The gaussian kernel supports the usual gaussian parameter sigma σ , which influences the steepness of the gaussian function. Using this parameter, it is possible to influence bias and variance of the SVM classifier. A high value of sigma causes a higher bias and lower variance since it distributes the training instances more smoothly over the higher dimensional space. A low value has the exact opposite effect increasing variance and reducing bias.

2.3.4 Bagging

Bagging as *bootstrap aggregating* is a meta classifier, which aggregates the results produced by multiple classifiers. It was introduced by Breiman (1996) as a means to build an aggregated classifier from multiple versions of the same classifier. For this purpose, a number of classifiers of the form $\varphi(x, \mathcal{L}) = j$ are generated. In this formula, x is a feature vector, $j \in \{1 \dots J\}$ is the label the classifier φ assigns to x and \mathcal{L} are the training instances for the classifier φ . The training instances \mathcal{L} are randomly chosen with replacement from among the training set and are called bootstrap samples. That way, a number of bootstrap replicates $\mathcal{L}^{(B)}$ are created and used to train a set of classifiers $\varphi^{(B)}$. In order to aggregate the predictions performed by all $\varphi^{(B)}(x) = j^{(B)}$, the bagging classifier applies majority voting. This means that the prediction with the most agreement among all $\varphi^{(B)}$ is considered the correct prediction.

The goal of a bagging classifier is to improve the performance of classifiers which are very instable. This especially applies to classifiers which tend to easily overfit their training data, most notably decision tree classifiers like REPTree.

Breiman (1996) shows that a bagging classifier is at least as good as the best of its wrapped classifiers, if the $\mathcal{L}^{(B)}$ are chosen independently from the same distribution. Since we would require a new training set for each $\mathcal{L}^{(B)}$ this is usually not done. Instead, as already mentioned above, a set of instances from \mathcal{L} is chosen as training set for each individual classifier $\varphi^{(B)}$. Those sets may overlap or contain duplicates and are thus not optimal for building the subclassifiers. Still, they are shown to have a high potential for improving very instable classifiers like REPTree, which is explained in Section 2.3.5. On the other hand, such a bagging classifier might decrease the performance of already stable classifiers like k-nearest

neighbors (kNN).

The most important parameters for a bagging classifier are the size and amount of the bootstrap replicates $\mathcal{L}^{(B)}$. Breiman (1996) evaluated both parameters. The size of the bootstrap replicates should be chosen to be equal to $|\mathcal{L}|$. Breiman (1996) evaluated larger sizes but could not improve classification accuracy that way. The amount of bootstrap replicates depends on the number of classes $\{1 \dots J\}$. Breiman (1996) suggests to use 50 and shows that larger numbers only result in marginal improvements to classification error rate.

2.3.5 Decision Tree

A decision tree is one of the fundamental algorithms in machine learning. It is based on the idea of deriving a tree with a decision about one attribute at each node from the set of training examples. This is similar to a cascade of intermingled 'if - then - else' statements.

One of the most basic algorithms to derive a decision tree is the ID3 algorithm as proposed by Quinlan (1979). This algorithm builds the tree recursively over the set of attributes in the training set. On each step the algorithm decides which attribute to select for the next node based on the attribute's importance. Importance is calculated using the Information Gain measure as explained in Section 2.2.3. After a node is calculated, the partition of the training set corresponding to the node is split into new partitions with each partition corresponding to one value of the attribute at the current node. New nodes are calculated for each of these partitions. If a partition either contains no unseen attributes or only instances belonging to one class, a leaf node is added. In the first case, the leaf corresponds to the majority class of the training examples at that node. In the second case, the leaf corresponds to the one class common to all training examples of the partition at that node.

There are several limitations to the ID3 decision tree learner: it overfits the training data easily, it cannot handle continuous or missing attributes and in the space of all possible decision trees for a given problem it only finds a local optimum. For these reasons, extensions and different decision tree learning algorithms have been developed. Some of them are presented in the following paragraphs. With these extensions decision trees perform similar to support vector machines but do not have as many complicated parameters to set as the latter.

Reduced Error Pruning

One approach to avoid overfitting of decision trees is to prune the tree and thus to generalize it. A decision tree should be pruned if there is another more general decision tree that shows better performance on some validation data. Reduced Error Pruning (REP) as presented by Mingers (1987) is one of the simplest forms of pruning. It prunes the tree step by step starting at the leaves. At each step one node is replaced with its most popular class, but only if the prediction accuracy on some validation data is not reduced by the change. REP is naive but has the advantages of simplicity and speed.

Decision Stump

A decision stump is a very simple decision tree with only one level. It is often used within ensemble classifiers such as the bagging classifier presented in Section 2.3.4. Further explanations are available for example via Iba & Langley (1992).

C4.5

The C4.5 algorithm is an extended version of the ID3 algorithm, proposed in Quinlan (1993). The algorithm includes solutions to the problems mentioned for the ID3 algorithm. It reduces overfitting by pruning the created decision tree, allows missing and continuous attributes and uses an improved measure to decide which attribute should be used at which node of the tree. See Quinlan (1993) or Mitchell (1997) for further information.

Random Forest

Similar to the bagging classifier described in Section 2.3.4, random forest is a meta classifier for decision trees. It was developed by Breiman (2001), the author of the bagging classifier, which was extended by feature selection methods from Ho (1998) and Amit & Geman (1997). The random forest works on a set of possibly overfitted decision trees. Each is created from a subset of available training data whereas the remaining training data is used to calculate the error of that particular decision tree. New classification instances are pushed down all the trees. A mode vote based on the results from all the trees in the ensemble is used to calculate the final decision of the random forest.

A random forest usually reduces the overfitting problem encountered when using traditional decision trees. However, it has been observed that random forests still overfit in environments with noisy classification tasks.

2.3.6 Rule Based Classification - Ripper Algorithm

Rule based classification algorithms are based on decision trees and use notations like the one given by the programming language Prolog. There, for example, exists a version of the C4.5 algorithm called C4.5rules, which uses a flat representation of the C4.5 tree. This representation is generated from the tree by creating one rule per path through the tree. That way, each rule is expressed in conjunctive normal form.

According to Cohen (1995), rule sets are easy to understand and they perform better than decision trees on many datasets. However, rule based systems in general tend to be very complex to learn if huge amounts of training data, especially noisy data, are available and required. Cohen (1995) present, a good overview of the performance implications of using a rule based system in contrast to a tree based one. He also created the repeated incremental pruning to produce error reduction (RIPPER) algorithm, which is used in this and several related publications. Cohen (1995) shows that the RIPPER algorithm performs similar if

not better than C4.5rules and is faster on learning noisy datasets. This is due to a highly optimized pruning step in comparison to reduced error pruning or the incremental reduced error pruning (IREP) as invented by Fürnkranz & Widmer (1994).

2.4 Summary

This chapter started with an introduction to the most common evaluation measures in classification. The following chapters are going to report on those measures for various classification tasks.

The second part of the chapter introduced important terminology from the research area of information retrieval. Those terms are going to become important during retrieval of forum pages as well as during retrieval of questions and answers from corpora of Web forum posts. Finally, the chapter presented an overview of some of the most important classification algorithms. It is not always clear beforehand which of these algorithms is best used in which situation. If this is the case, our experiments will test different classifiers and report on their performance. Subsequent chapters will show that meta classifiers like bagging and random forest are constantly among the best performing algorithms for the problems in this thesis. However, it is important to note that in most cases the choice of the correct classifier is not as important as having good and plenty of data and the correct features for the classification task, as pointed out by Edwin Chen in his Blog².

²<http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/>

3 Forum Data Acquisition

The amount of online discussions is enormous. Even a single topic area is discussed by thousands of users distributed over several forum communities. A topic's scale depends on its popularity, of course.

As soon as manual steps are involved, even monitoring all questions from a single topic area such as Java programming, football, or travel becomes an impossible task. Even though it might be possible to gather all existing discussions, new ones appear constantly and old ones disappear on a similar rate. For this reason, the following chapter presents a fully automatic approach for acquisition of data from various different forums with low error margin. The idea is to have a black box system taking the World Wide Web as input and providing a database containing all discussions posted anywhere, anytime and keep them up to date. This is an idealized goal, of course, and can only be achieved partially. The following sections provide details for achieving this goal based on existing Web scale retrieval.

There are four possible source types for forum data: Web feed data, API access, a focused Web forum crawler, or, at best, direct access to the forum's database. Database access is the least error prone and fastest way to retrieve content. However, it is also the most limited, since most forum operators do not open their databases to public access, which is why it is only possible to gain access to the databases of a limited amount of self operated forums at best. Feed readers are well suited for retrieving most recent forum data, but they are not good at collecting a large amount of forum data at once. This is due to the fact that feeds usually show the most recent updates only. Everything that occurred a fixed time span or a fixed number of updates ago is deleted from the feed. In order to provide a good user experience and conduct meaningful research, historical data is required though.

Nowadays many platform operators provide API access to their data. While these provide some interesting data, for example via Twitter¹, they are usually not available for Web forums. Thus, the only viable solution is crawling the Web pages a forum consists of with a focused forum crawler maybe extended by a feed crawler to keep data up to date and extract the crawled pages with an approach that detects typical structures of Web forum pages.

The concepts and evaluations presented in this chapter result from the supervision of two diploma theses and three study theses by Drescher (2010), Rudolph (2011), Beyer (2011), Pretzsch (2011) and Pretzsch (2012). Further details are available from each individual work.

¹<https://twitter.com/>

3.1 Overview of the Focused Forum Crawler Process

The focused forum crawler requires three parts as shown in Figure 3.1 All three are pre-

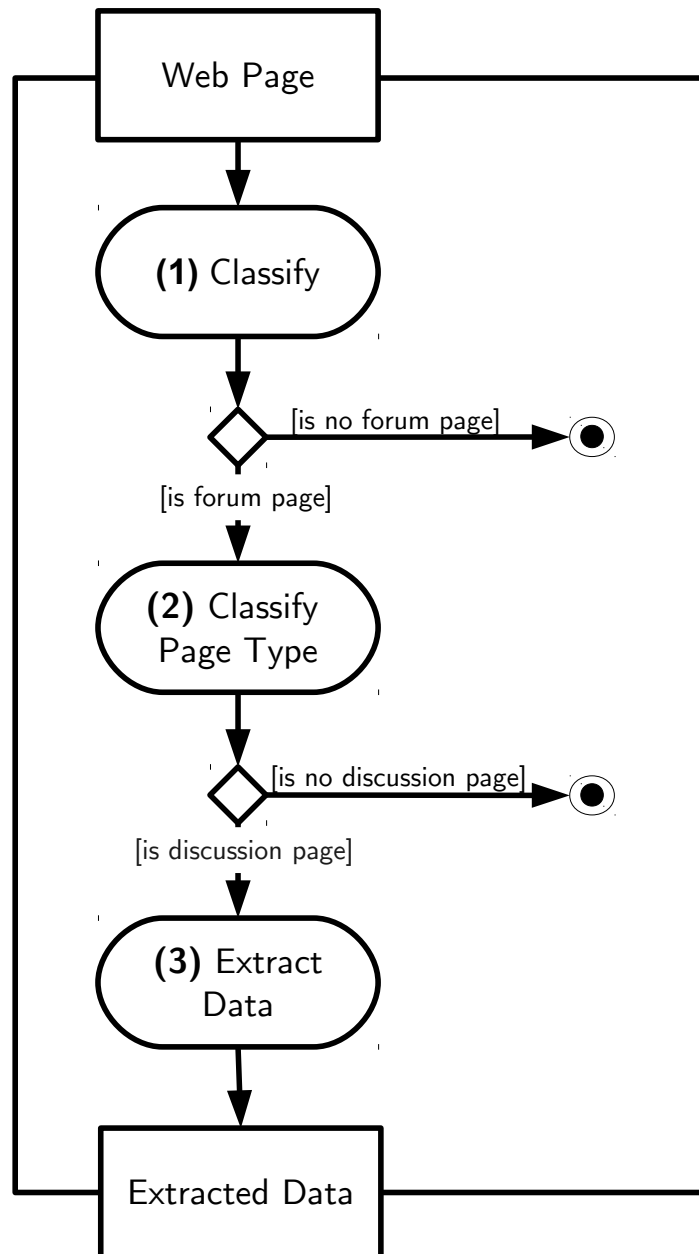


Figure 3.1: Sequence of activities required for a fully automatic forum data extractor.

sented in detail in the following sections. In short, the crawler needs to retrieve Web pages and select those belonging to a Web forum from among the millions of pages it receives during a crawling run. This is explained in Sections 3.3, 3.4, and 3.5.

If the selected pages truly are from a Web forum they belong to one of five classes based on the classes already identified by Cai, Yang, Lai, Wang & Zhang (2008). An example for each class is shown in Figure 3.2.

3.1 Overview of the Focused Forum Crawler Process

(a) list-of-board page

(b) list-of-thread page

(c) post-of-thread page

(d) user-profile page

Figure 3.2: Examples for forum page types

list-of-board Every forum usually has one page which lists all the sub forums also called channels or boards. This page is called list-of-board page. Figure 3.2(a) shows an example.

list-of-thread Each sub forum displays a page showing the threads ordered by most recent activity. Such pages are called list-of-thread and form the entry point for small forums with no list-of-board page. Figure 3.2(b) shows an example from the Oracle Developer Network². Older threads are usually displayed on list-of-thread pages that are accessible via so called page flipping links from the initial list-of-thread page.

post-of-thread The post-of-thread pages show the actual user generated content in the form of forum posts, such as shown in Figure 3.2(c). One post-of-thread page shows only posts from one thread, but one long thread can be distributed over multiple post-of-thread pages. Those pages are accessible via page flipping links similar to list-of-thread pages. For classical forums, posts are ordered by their creation date. For modern question/answer sites it is common to show one question as the first page and to either show the accepted answer below or to organize all answers by their user provided ranking.

user-profile Forums usually also make some information about each user accessible to other users. Pages presenting such statistics about a user's created threads, posts, time since registration, user name etc. are called user-profile pages. Some forums provide them only to registered users, which makes them inaccessible to robots and Web crawlers. Other forums such as the SAP Developer Network³ example in Figure 3.2(d) provide only a small part of the information to unregistered users.

Miscellaneous Everything that does not belong to one of the previous categories falls into this miscellaneous category. For Effingo it is used as a garbage category for all pages requiring no further processing.

Since only post-of-thread and user-profile pages contain relevant data, a second step separates those from the remaining forum pages. Sections 3.6, 3.7, and 3.8 show how to do that.

After having a set of content pages in raw HTML, the source is parsed to extract actual forum data and save it to a database. This database follows an aggregated schema describing all types of forums and discussion pages as already shown in the introductory chapter, Section 1.2 of this thesis. Sections 3.9, 3.10, and 3.11 provide detailed information about the extraction approach applied by Effingo.

²<http://forums.oracle.com>

³<http://forums.sdn.sap.com>

3.2 Structure of an HTML page

Crawlers and parsers make use of the fact that HTML pages consist of HTML tags to access the page content. Each tag is a node in an HTML tag tree. Each sub tree is called a segment. The process of splitting an HTML page into segments is called page segmentation. Thus, in order to find the page content, we need to find the segment spanning the complete content area. A segment is defined by a path of tags from the page's root tag to the segment's root tag. One segment can also contain other subsegments.

A linearization of the tree structure reshapes the tree into a list of so called tag paths. Tag paths are all paths leading to all segments on the page in order of appearance. A tag path consists, similar to XPath expressions, of HTML elements also called path elements. Each path element is one step in the sequence of HTML elements from the HTML root node to the block-level element surrounding a segment. That way, path elements represent a decision as to which node to choose from the sub nodes at the current path position until the addressed node is reached. Individual elements of a tag path are:

$$\underbrace{\text{html}[1]}_a > \underbrace{\text{body}[1]}_b > \underbrace{\text{div}[2]}_c > \underbrace{\text{div}[\text{@class}=\text{"post"}][1]}_d$$

a: *Root Element*; the root path element.

b: *Path Element*; the name of the HTML element

c: *Index*; Unambiguously addresses the element in a list of siblings, creating an absolute path element in contrast to a relative path element without such an index.

d: *Predicate*; refinement or selection predicate

Tag paths are either a representation of exactly one HTML element or of a whole set of similar HTML elements, depending on whether index information is provided or not. Therefore, we distinguish between the following types of tag path representations:

Relative Tag Path Representation (RTR): Addresses multiple segments if the selection predicate at one of the segments is ambiguous. The set of segments addressed by an RTR α is henceforth called S^α (segments under α). The quantity of segments addressed by α is denoted by $|RTR|$. The position at which the tag path starts to get relative has a special semantics. It is the first position inside the tag path where the HTML tree splits and the tag path follows multiple branches. The parent node of this split is called *template container*. Its direct child path elements are called *template nodes*.

Absolute Tag Path Representation (ATR): Addresses a single segment unambiguously. Either all path elements are identified by their index or the last path element has a unique identifier in form of an HTML id attribute selection predicate. An absolute

path's template container is the document's root node. Its only template node is the `<html>` node.

3.3 Web Forum Retrieval

This and the following two sections cover the box marked (1) in Figure 3.1. While crawling the whole Web, we encounter arbitrary Web pages. If the page belongs to a forum, it might contain important information for community based question answering. However, initially, we know nothing about those pages, so we need a way to tell whether any page we encounter either belongs to a forum or not.

As already shown for Blogs (Fetterly & Chien 2007, *NITLE Weblog Census* 2003) a Web crawler focused on a certain type of Web page can either use features from the page's HTML source to find the correct type or apply certain analyses on the page's Web address URL. The *NITLE Weblog Census* (2003)⁴ project and the Microsoft patent for a focused Blog crawler (Duong, Hall, Mayfield, McNamee & Piatko 2002), for example, identify Blogs based on features such as:

- If a page's URL is registered in some Web log update side or Web log database, such as `http://www.dmoz.org`
- If a page's URL belongs to some well-known Blog hosting provider, e.g. `http://wordpress.com/` or `http://www.blogger.com/`
- The page's URL contains certain substrings, such as "blog"
- The Web site contains META tag attributes created by well-known Blog generators
- The Web site contains outgoing links—especially a button, logo or a "Powered by" link—connected to a blogging engine
- The Web site's content contains code specific to blogging tools
- The Web site contains a certain keyword with a certain frequency. Commonly used keywords are: "blog", "blogroll", "metaphilter", "powered by", "permalink", "trackback", "comment", "comments", "blogad" or "posted at"
- The Web site provides a feed

Such an approach can be applied on forums as well, as already pointed out by Chen & Choi (2008) for a more generic case. They propose a hierarchical categorization of Web pages—not only forums—and provide an approach to classify new pages into these categories. In addition to "Discussing Forum" they also propose Web page types such as "Online shopping", "Information and resource page", "Information search page", and "Homepage". On a small handcrafted dataset of 1,000 Web pages they achieved an F_1 -Score of 91% using a Naïve Bayes like classification approach with features such as:

⁴The page is not available anymore as of 04/09/2013.

- Number of occurrences of telephone numbers, prices, email links, copyright notes, dates, times, and question sentences
- Special script functions such as JavaScript alerts and confirms and their VBScript pendants
- Type of top level domain (".edu" identifies academical sites etc.)
- Occurrence of the terms "forum", "forums", "FAQ", or "faq" in the URL's directory part
- Number of certain HTML tags from the following list: input box, message box, select and radio button, submit button, frame, hyperlink, link to ftp, list, password, reset, style sheet, table, textarea, textinput, image link, image

Unfortunately, their approach requires much adaptation. Every feature needs to be weighted and thresholds have to be set manually. Such an approach is prone to overfitting, expensive, and not easily adaptable to changing input data. Supervised classifiers, such as the ones presented in Section 2.3, can do all this automatically and perform on the same level or even better, as we are going to show at least for Web forum pages.

Our approach differs from the existing works by setting a clear focus on Web forums. Other focused crawling strategies are either specialized on a different domain or handle forums as one of many categories. In addition, we are going to show an approach using a supervised classification approach instead of handcrafted parameters and rules.

We intend to build upon feature sets identified by previous work and extend them with features found during an extensive analysis of an extensive sample of real world forums. For example, in contrast to the separation of links into hyperlinks and e-mail links as proposed by Chen & Choi (2008), we are going to show a more detailed separation of different links presented in Section 3.4.5. Moreover, we do not choose among the form elements present on a page, but rather take all of them, letting the classifier decide which ones are actually a good feature for a forum. The extended feature set is presented in the next section.

3.4 Features for Focused Forum Crawling

The previous section shows many features of Web and forum pages. In order to build a classifier which is able to separate forum pages from other Web pages, the following section takes a detailed look into features specific to forum pages. The classifier which was created based on these features is evaluated in the following section.

Feature engineering usually starts with a look on the actual data. For that reason a subset of forums was examined in detail. Those forums are examined with regard to different features and with the aim of illustrating to clarify those we already know from the related work discussed in Section 3.3 and to find new ones. The forums are chosen from the results returned by standard search engines using terms such as "forum", "forums", and "popular

forums". In addition, the list was complemented using hand-crafted lists of different forums provided by the following sites:

<http://www.mister-wong.de/tags/forum/> Forum directory with a list of several thousand forums.

<http://www.dmoz.org/> Web page directory providing several subcategories for "Chats and Forums".

Table 3.1 provides an excerpt from the list of investigated forums. The investigation of these

Table 3.1: An excerpt of the forums used for feature engineering for the focused forum web crawler.

Forum - Name	Forum - URL
Sun forum	http://forums.sun.com
Forum for the news page spiegel.de.	http://forum.spiegel.de/
Forum for the World of Warcraft game.	http://forums.worldofwarcraft.com/
Microsoft Developer Network	http://social.msdn.microsoft.com/Forums/de-DE
Forum for Ubuntu operating system	http://ubuntuforums.org/
ASP Forum	http://forums.asp.net/
A forum for travelers	http://www.tripadvisor.de/ForumHome
Oracle support forum	http://forums.oracle.com/
Yahoo question answer forum	http://de.answers.yahoo.com/
An SAP forum	http://www.dv-treff.de/sap-community/
Forum BMW cars	http://www.bmw-forums.net/

sites provided a list of commonalities, which might be helpful to identify forums among other sites on the World Wide Web. The goal is to find a feature set which is not prone to overfitting and which matches forums with high precision. Such features are explained in detail in the following sections.

For a human observer the obvious way to distinguish a forum from another site is the way its content is laid out in lists of posts, threads, and sub forums. The analysis of forum features is therefore focused on those HTML nodes surrounding a page's content. Content only occurs in the form of multimedia objects. Possible multimedia objects are text, images, videos, sounds, music, animations, interactive applets etc. Text is the most common way to show content in a Web forum. Therefore, segments containing text are most important for detecting Web forums. A quick analysis via Google shows that images are approximately twice as frequent as other multimedia objects. Since those other multimedia objects are not so easy to process, image objects are the only other type of multimedia content considered. Neither an image nor a text node can wrap other HTML nodes. Therefore, they are always leaves in the HTML tree.

Content is often split up by formatting HTML markup. In a forum it is, for example, common to provide the user with a few limited possibilities for formatting his contribution. Such formatting nodes are not part of the page's structure. If they were handled like other HTML

nodes they would split content actually belonging together. In order to avoid such splits, a segment containing a continuous piece of content and possible formatting markup is called an *information container* and follows Definition 10.

Definition 10 *Information Container.* An information container is a node in an HTML tree satisfying two properties:

- It contains at least one direct child non empty text node or image node.
- It has no parent node which is an information container.

See the following code snippet for an example.

```

1 <div>
2   <div>
3     some text
4     <b> some internal text </b>
5   </div>
6 </div>

```

Listing 3.1: Example for an information container and non-information container

The node starting on line 1 is no information container since it contains no direct text child node. The node starting on line 2 is an information container since it contains a direct text child node and no parent is an information container. Since the node starting on line 2 is an information container, the node on line 4 cannot be one. It has a parent which is an information container. With this definition it is possible to prevent splitting of text due to inline formatting in most cases.

The following sections give a detailed explanation of all the features used for Web page classification by Effingo. At first, an overview of necessary preprocessing steps is provided, followed by information on how to use a page's HTML structure, URL, Metada, Interactive Elements, and the page's text to separate forum from non-forum pages.

3.4.1 Page Preprocessing

Not all of the content on an HTML page is equally important for the purpose of finding out whether or not it is a forum page. A usual HTML page contains advertisements, information texts, control areas, etc.. in addition to the actual main content. Therefore, some preprocessing is necessary prior to feature extraction. For data extraction, as described below in Section 3.9 and following, an exact identification of this main content area is important. However, the page classification described here is applied to a much larger amount of arbitrary Web pages. In order to reduce the computation per page, the following estimate is used to separate main content from the rest of the page. Initially, the tag paths for all information containers of the page are collected and grouped by length. Tag paths of equal length are compared pairwise and merged if it is possible to create a relative tag path representation

addressing the same nodes as the two individual tag paths. A set of tag paths P is created that way.

Consider the example HTML tree in Listing 3.2.

```

1 <html>
2 <body>
3 <h1>Title: Favorite Movies</h1>
4 <ul>
5 <li>
6 <div><b>Author: Max</b></div>
7 <div>
8 Title: What's your favorite movie?
9 </div>
10 <div>
11 My favorite movie is
12 <em>Star Trek: First Contact</em>
13 </div>
14 </li>
15 <li>
16 <div><b>Author: Moritz</b></div>
17 <div>
18 Title: Re: What's your favorite movie?
19 </div>
20 <div>I'd prefer Forrest Gump</div>
21 </li>
22 </ul>
23 </body>
24 </html>

```

Listing 3.2: Example for showing the generation of all absolute and relative tag paths.

Figure 3.3 shows the contents of P for that example after certain iterations. It contains tag

1st Iteration	
Length	Tag Paths
3	html[1]/body[1]/h1[1]
2th Iteration	
Length	Tag Paths
3	html[1]/body[1]/h1[1]
6	html[1]/body[1]/ul[2]/li[1]/div[1]/b[1]
4th Iteration	
3	html[1]/body[1]/h1[1]
5	html[1]/body[1]/ul[2]/li[1]/div[2]
	html[1]/body[1]/ul[2]/li[1]/div[3]
	html[1]/body[1]/ul[2]/li[1]/div
6	html[1]/body[1]/ul[2]/li[1]/div[1]/b[1]

Figure 3.3: Content of P for some example iterations on the HTML structure in Listing 3.2.

paths to the first two information containers after two iterations. During the fourth iteration a second path of length 5 is added. This additional path points to a sibling of the other path of length 5 and thus a relative tag path addressing both segments is also added to P . Figure 3.4 shows the final content of P .

Length	Tag Path	Name
3	html[1]/body[1]/h1[1]	p ₃
5	html[1]/body[1]/ul[2]/li[1]/div[2]	p ₁
	html[1]/body[1]/ul[2]/li[1]/div[3]	p ₂
	html[1]/body[1]/ul[2]/li[1]/div	p ₃
	html[1]/body[1]/ul[2]/li[2]/div[2]	p ₄
	html[1]/body[1]/ul[2]/li[2]/div[3]	p ₅
	html[1]/body[1]/ul[2]/li[2]/div	p ₆
	html[1]/body[1]/ul[2]/li/div[2]	p ₇
	html[1]/body[1]/ul[2]/li/div[3]	p ₈
6	html[1]/body[1]/ul[2]/li[1]/div[1]/b[1]	p ₉
	html[1]/body[1]/ul[2]/li[2]/div[1]/b[1]	p ₁₀
	html[1]/body[1]/ul[2]/li/div[1]/b[1]	p ₁₁

Figure 3.4: Resulting set P of paths for the example HTML structure in Listing 3.2.

Afterwards, tag paths sharing the same template container such as p_7 , p_8 and p_{11} in Table 3.4 are grouped together. Each such group is called a path group. Each path group usually consist of some main content, some important paths and perhaps some clutter. In order to separate paths into these three groups, a score is calculated for each path depending on the template nodes that specific path reaches. The score for a path k is calculated using Equation 3.1 with all paths from a path group $paths$ and the set of template nodes covered by those paths $nodes$.

$$\text{score}_k(\text{paths}, \text{nodes}) = \sum_{i=1}^M (\text{sign}(\text{paths}_k, \text{nodes}_i) (2 \cdot \text{covers}(\text{nodes}_i, \text{paths}) - N)) \quad (3.1)$$

$$\text{sign}(p, n) = \begin{cases} 1, & \text{if } n \text{ covers } p. \\ -1, & \text{otherwise.} \end{cases} \quad (3.2)$$

The equation uses the amount of information container paths covering a template node, which is provided by the function $\text{covers}(n, p)$. The result of this function is a value between 0 and N , with N being the number of paths in the path group. The result is used either to boost the overall score or to penalize it, depending on whether template node i covers path k . If it does not, the negative of the summand is used. This is controlled using the function $\text{sign}(p, n)$. The path to the template container with the highest score is considered as the *main path*; the path to the main content of the page. All paths (including the main path) sharing at least 90% of the main paths template nodes are considered as *important paths*.

For an example consider the path group shown in Table 3.2 with paths sharing the template container $ol[1]$. In addition to the relative paths from the group, the table also shows the amount of template nodes addressed by each relative tag path. In total, there are six li template nodes. Since not each path in Table 3.2 is part of six template nodes not each path occurs together with each template node. Meaning that some paths, such as p_1 , mark optional segments. Specifically all paths occurring only in combination with five template

Table 3.2: Example path group to show how selecting the main path works.

Name	Relative Tag Path	Template Nodes
p ₁	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/ul[2]/li[1]/a[1]	5
p ₂	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/div[1]/div[1]/p[1]	5
p ₃	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/div[1]/div[1]/p[2]	5
p ₄	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/h4[2]	6
p ₅	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/div[1]/div[1]/div[1]	5
p ₆	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/div[2]	6
p ₇	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/div[1]/h4[1]	6
p ₈	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/ul[1]/li[2]	5
p ₉	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/h4[1]	6
p ₁₀	/body[1]/div[1]/div[6]/ol[1]/li[1]/ol[1]/li/div[1]/ul[1]/li[1]	5

nodes are optional since they are missing as content of exactly one of them. All paths with six template node occurrences seem to be mandatory. They always occur as a part of the template container `ol[1]` and its template nodes `li`. The main path should be one that marks such a non-optional segment or at least a segment common to most paths in the same path group. In order to be selected as main path, a path should have a higher score than paths not occurring for each template node. Table 3.3 shows how to calculate the scores for all those paths. Paths such as p_1 , p_2 and p_3 are punished for not occurring together with all

Table 3.3: Example calculation to find the main path of a path group.

	P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇	P ₈	P ₉	P ₁₀
li ₁	$2 \cdot 10 - 10 = 10$	10	10	10	10	10	10	10	10	10
li ₂	$2 \cdot 10 - 10 = 10$	10	10	10	10	10	10	10	10	10
li ₃	$2 \cdot 10 - 10 = 10$	10	10	10	10	10	10	10	10	10
li ₄	$2 \cdot 7 - 10 = 4$	-4	-4	4	-4	4	4	4	4	4
li ₅	$-1(2 \cdot 7 - 10) = -4$	4	4	4	4	4	4	-4	4	-4
li ₆	$2 \cdot 10 - 10 = 10$	10	10	10	10	10	10	10	10	10
Score/Sum	40	40	40	48	40	48	48	40	48	40

template nodes, whereas paths such as p_4 get a high score. For this example any of the paths with score 48 are suitable main paths. In case of such a tie, the first is taken, i.e. p_4 . Remaining paths are categorized as important paths if they share at least 90% of the main path's template nodes, otherwise, they are categorized as clutter paths. For the example presented this means paths p_4 , p_6 , p_7 , and p_9 are important paths, while the remaining paths are clutter paths. In addition, we also count clutter nodes as the set of those template nodes from the important paths, not occurring together with the main path. There are no clutter nodes in the example above.

For each HTML page one path group is chosen as the page's main path group. Two alternative heuristics were considered for choosing the main path group. The first was to use the group covering the most text counted in words, which proved to be inadequate. As for the

second heuristic a measure was applied based on the product of the segments covered by the main path and the number of paths belonging to the group. The path group with the highest value for this product was used as main path group for each page.

The main path group and the separation of clutter paths and important paths is essential for calculating the features discussed in the following sections of this chapter.

3.4.2 HTML Structure

Based on the examination of several Web forums and in contrast to the research results presented by Chen & Choi (2008), apparently single HTML tags are not the only feature accessible from a page's HTML code. Forums are created by means of similar software systems and follow a common layout. As a result, common patterns inside their HTML code can be employed to separate forum pages from non-forum pages.

In order to capture those patterns important paths are examined for their content and categorized into one of several classes. Important paths are chosen according to the preprocessing explained in Section 3.4.1. There are two types of classes a tag path may belong to: Semantic classes and text classes.

A tag path in a semantic class is considered as belonging to the class if at least 50% of its segment's content matches the class description. For example, a tag path belongs to the class "Semantic, Number" if at least 50% of its content are numbers.

A tag path in a text class on the other hand belongs to its class if it matches a certain text length, which was found by an empirical examination of pages from the forums presented in Table 3.1.

The following classes are used to find forum pages.

Semantic, Equal timestamps The path contains dates or dates and times in a uniform format

Semantic, Ordered timestamps In addition to a uniform formatting these timestamps are ordered

Semantic, Number A number (i.e. page flipping links used for pagination)

Semantic, Image An image, for example a user's avatar

Text, short Arbitrary text spanning three or less words, for example user names or page navigation

Text, medium Arbitrary text spanning more than 3 but less than 25 words, for example thread titles

Text, long Arbitrary text spanning more than 25 words, for example user post body texts

Those classes are calculated for all tag paths in the path group containing the page's main path and aggregated to seven features, one per class.

In addition, all clutter paths, all clutter nodes for the page's main path and all remaining important paths are counted and used as three additional features. Furthermore the CSS classes common to all paths in the main group and the HTML Tags occurring after the template container are used as text features.

3.4.3 Uniform Resource Identifier

A Web page's URL often contains hints on whether it is a forum or not. Such hints can result from comparing the URL to lists of known Web forums. However, managing those lists is rather tedious. Clues with fewer manual effort can be retrieved from keywords occurring within the URL. The word "forum", for example, often occurs directly in the address, like `https://forums.oracle.com`. On the other hand, there are other pages such as `http://www.sachsen-forum-dresden.de` containing the word forum, which are no forums at all. As a result of a combined search on Google Web search and Google discussion search for pages containing the term "forum" in the URL (Query "allinurl: forum"), approximately 26% percent of pages containing the term "forum" are actual forums. A Google-analysis of the German World Wide Web reveals that approximately 3% of all pages and approximately 51% of all forums contain the term "forum" within their URL. So even though the occurrence of this feature is a good indicator, other features are required to eliminate false positives and to find the remaining 49%. However, presently it still remains unclear how to tokenize a URL since they usually do not contain whitespace. Nevertheless, some important keywords such as "thread", "user", or "post" occur in URL parameter names, such as "threadid" or "userid". This facilitates the separation of those keywords from the rest of the URL and thus parameters are used as features to represent each page's URL.

3.4.4 Metadata

Another interesting feature are meta data tags of a page. Since they are used by search engine robots as a means to identify the content of a Web page, they display the term "forum" quite often. They consist of key value pairs and are usually located in the header of an HTML page. An example taken from `http://forum.polar.fi` is shown in Listing 3.3.

```
1 <meta name="keywords" content="forum , bbs , discussion , bulletin_board" />
2 <meta name="description" content="This_is_a_discussion_forum." />
```

Listing 3.3: Examples for the usage of meta tags

The most interesting meta tags are the "keywords" and the "generator" tag. The keywords tag usually contains a very dense description of the site a page belongs to, while the non-standardized "generator" tag often contains the name of the forum software that generated the page. Another very interesting meta data tag is the "description" tag, which usually

contains a quite lengthy content. This lengthy content would generate much noise, which is why it is not included as a feature for focused forum crawling. Thus, only the words from the content attributes of the “keywords” and “description” meta tags are used as features to describe a page for the forum page classifier.

3.4.5 Interactive Elements

Forums provide a user with many possibilities to interact with the page, to publish new content and navigate on the site. By identifying those HTML elements which provide the interactive behavior the crawler gets additional clues whether it encountered a forum page or not.

Hyperlinks

Large parts of the interaction with a forum page are facilitated by hyperlinks. Forums can contain links to other sites, though most links are internal service links. Instances of such service links are links from list-of-thread pages to the individual threads, links from post-of-thread pages to authors or self referencing anchor links and links from one post-of-thread page to other post-of-thread pages belonging to the same thread. Therefore, the link density can be considered to be a feature which helps to identify whether a domain is a forum or not. Link density is a measure based on the amount of links and the distance of each link from the current page. As already mentioned, an accumulation of internal links can be a clue to having discovered a page from a Web forum. In reality there is no such thing as a distance between two URLs. However, assuming the resources on server side are categorized by a particular schema, it is possible to define one. Considering an example page like `http://example.com/test/foo.html?p=1` with an examined link inside a container under the identifier (i.e. HTML id attribute) “content”, enables us to define the distances presented in Table 3.4.

In order to calculate a feature from this information, the amount of links of each distance category in the main path group is counted and used as link density feature.

Link density, however, provides no notion of how large the amount of linked text is in comparison to all the text in the document: On the one hand, a link could be a single word in a long text while on the other hand, it could be a long thread title completely linked to its content. Therefore, the link ratio for each information container (see Definition 10) is calculated using Equation 3.3.

$$\text{linkratio} = \frac{\text{length}(\text{Text in Links})}{\text{length}(\text{Whole text from information container})} \quad (3.3)$$

An information container with a link ratio of more than 50% is called link dominant. Link dominant information containers are grouped into the same classes already shown for non-link dominant information containers in Section 3.4.2 and their counts are used as separate

Table 3.4: Possible link distances in ascending order.

Distance	Type	Example
1	selfreferencing link	<code>http://example.com/text/foo.html?p=1#content</code>
2	anchors	<code>http://example.com/test/foo.html?p=1#top</code>
3	Javascript function	<code>javascript:alert('test');</code>
4	same parameters	<code>http://example.com/test/foo.html?p=1</code>
5	different parameters	<code>http://example.com/test/foo.html?p=1&sort=asc</code>
6	same directory	<code>http://example.com/test/news.php</code>
7	same directory branch	<code>http://example.com/test/f/img.jpg</code>
8	same server	<code>http://example.com/other/page.pdf</code>
9	same server, different port	<code>http://example.com:8080/other/page.pdf</code>
10	same domain	<code>http://forum.example.com/index.php</code>
11	different server	<code>http://page.example.org</code>
12	different protocol	<code>mailto:max@mustermann.de</code>

features for link dominant content.

Form Elements

In addition to hyperlinks, many interactions on a forum are provided via HTML forms and their elements. Therefore, the frequency of occurrence of elements of the following type is used as another feature:

- Text Field
- Text Box
- Dropdown Box
- Check Box
- Radio Button
- Password Field
- File Uploader
- Button
- Form

3.4.6 Page text

As shown by Fetterly & Chien (2007), *NITLE Weblog Census* (2003), and Chen & Choi (2008) the page text also provides valuable information for site classification. Table 3.5 shows some examples for words that might be useful. Inline images are handled like word tokens since they usually are smileys or emoticons with a meaning similar to words.

The extraction of features from the page text is based on the information container according to Definition 10. At first, the text for an information container node is created by concatenat-

Table 3.5: Possible keywords, used as features for forum detection.

Keyword	Meaning
forum	common word within most forums
views	commonly used to show the amount of users who already viewed a thread
topics	used as heading
message	to send personal messages to other users
statistics	evaluation of forum usage
posts	commonly used to show amount of posts a user has created so far
new post/thread	to create new posts or threads
phpBB, Yet Another Forum, vBulletin, Jive Forums, Burning Board, ...	name of the forum software, used to create the forum

ing the text from all child text nodes. As a second step, the text is analyzed for the following tokens in the given order:

- Upper-/lowercase sentence beginnings
- Timestamps (dates and optional times)
- Numbers
- Pronouns
- other words.

If one of those tokens is found (except for sentence beginnings), it is removed from the text, so it can not reoccur in other categories. The occurrences for each category are saved.

Pronouns are separated into pronouns of first and second person and pronouns of third person. The first and second person pronouns usually occur in forums since they are much more personal and forums tend to be pages with personal salutations. The third person is more common on other pages. More specifically, the following list of words was used to indicate first and second person pronouns: "I", "me", "my", "you", "your", "yours", "we", "us". Additionally, "@nickname", the common salutation used in forums, was included. The list of third person pronouns consists of: "he", "him", "his", "she", "her", "hers", "it", "its", "they", "them", "their", and "theirs".

Page text statistics were calculated for the main path group as the fraction of text inside the main path group each type covers. For example, the feature for the "Numbers" class was calculated by counting the number of number-only tokens and by dividing this number by the number of all tokens covered by the main path group.

3.5 Performance of Focused Forum Crawling

The following section describes the performance of the focused forum crawler. Within this scope, it provides details about the dataset used for evaluation as well as about the ex-

perimental setup and it is closed by a discussion about the pros and cons of the proposed approach.

3.5.1 Dataset

The dataset used to evaluate the focused forum crawler consists of 1,326 pages from 71 different sites. The pages are equally distributed over forum and non-forum pages. In order to identify a representative set of pages from the Web, we applied two different approaches to find forum and non-forum pages.

Forum Page Collection

In order to generate a set of forum pages a list of forums was created. For that purpose the Web site directory dmoz.org was searched for the term “forum”. Subsequently the results were filtered manually to remove orphaned pages and pages containing the term “forum” without being one. Only forums spanning a whole domain were considered. Finally, up to 10 pages were retrieved from each of those forums until 663 pages were collected.

Non-Forum Page Collection

For non-forum pages a different approach was used. In order to achieve a broad spectrum of non-forum pages, the top 100 pages according to alexa.com were used. Since this thesis is focused on English pages, only pages from the US and GB lists were considered. Finally, a crawler was applied to retrieve pages from those sites. The crawler was configured to follow only external links, meaning links pointing to different domains.

3.5.2 Experiments

In order to test the feature set, the following classification algorithms from the Weka3 framework were evaluated using their standard settings:

Naïve Bayes see Section 2.3.1.

Support Vector Machines see Section 2.3.3

Voted Perceptron see Section 2.3.2.

JRip see Section 2.3.6

DecisionStump see Section 2.3.5

J48 see the C4.5 algorithm in Section 2.3.5

RandomForest see Section 2.3.5

REPTree see Reduced Error Pruning in Section 2.3.5

Table 3.6: Results of different classifiers using the features presented in Section 3.4. Each classifier was run for 10 times with 10-fold cross validation.

	Accuracy	Precision	Recall	F_1 -Measure
Naïve Bayes	0.81	0.78	0.87	0.82
Support Vector Machines	0.93	0.93	0.93	0.93
Voted Perceptron	0.75	0.76	0.75	0.75
JRip	0.90	0.90	0.89	0.89
Decision Stump	0.64	0.58	0.98	0.72
J48	0.92	0.91	0.92	0.91
Random Forest	0.95	0.94	0.97	0.95
REPTree	0.90	0.89	0.91	0.90

Each classifier was executed 10 times using a 10-fold cross validation with all features presented in Section 3.4. Thus finally each classifier was run 100 times. Table 3.6 provides the results for accuracy, precision, recall, and F_1 -measure.

As a means to evaluate the performance of single features, the three “winning” classifiers according to Table 3.6 (J48, support vector machine, random forest) were re-evaluated on five feature subsets.

Interactive Elements (IE) Amount of control elements and forms.

Page text (PT) Upper/Lowercase sentences, pronouns, in text images.

Metadata and URL (MU) Keyword and generator meta tag values, plus HTTP parameter names and whether the URL contains the term “forum”.

HTML Structure (HS) All HTML page structure data, like path counts and element sets.

HTML Content (HC) Used tags and CSS classes in the main group.

Each classifier was run again for 5 times with 10 fold cross validation using combinations of similar features. Table 3.7 shows the accuracy for all feature combinations.

3.5.3 Result Discussion

It is obvious that by applying the proposed features, focused forum crawling achieves very good results. The only comparable approach by Chen & Choi (2008) recognized 84% of forums correctly and reached a mean accuracy of 93,25%. Our classifier is able to recognize 94% of the forums and reaches a mean accuracy of 95%. Comparing the results, however, is not absolutely fair, since Chen & Choi (2008) used a different and not publicly available dataset.

Despite the high accuracy, there still was a number of error cases, which were analyzed in detail. The most interesting anomaly was the behavior of page text features. Those features did not only fail to improve the classification results, but decreased them in some cases.

Table 3.7: Accuracy for different feature combinations of the focused forum crawler classifier.

IE	81.39%	IE+PT+MU	91.93%
PT	73.24%	IE+PT+HS	90.71%
MU	86.49%	IE+PT+HC	93.69%
HS	88.12%	IE+MU+HS	93.17%
HC	91.51%	IE+MU+HC	95.07%
IE+PT	88.49%	IE+HS+HC	93.15%
IE+MU	89.68%	PT+MU+HS	93.06%
IE+HS	90.88%	PT+MU+HC	95.2%
IE+HC	93.05%	PT+HS+HC	92.52%
PT+MU	88.73%	MU+HS+HC	94.84%
PT+HS	88.01%	IE+PT+MU+HS	93.76%
PT+HC	91.93%	IE+PT+MU+HC	95.08%
MU+HS	93.05%	IE+MU+HS+HC	95.11%
MU+HC	94.3%	PT+MU+HS+HC	95.32%
HS+HC	92.14%	IE+PT+MU+HS+HC	95.37%

There is no definite explanation to this, but it should be taken as an advice to avoid these features. They are not strictly necessary and they might even lower the quality of results.

Further problems concerned almost empty pages like HTTP error pages. Those pages are often classified as forum pages. However, it should be easy to filter them out prior to classification in a future iteration of the focused forum crawler.

Pages available via https as well as http caused some problems, which may be caused by the URL distance measures. A link normalization approach could be helpful to remove such kinds of errors.

Finally, there are some pages which look like a forum but are none. These cases are rare but not easy to handle. One example is the site hotukdeals.com as shown in Figure 3.5. Furthermore, there are a few forums which are displayed as a tree structure rather than as a

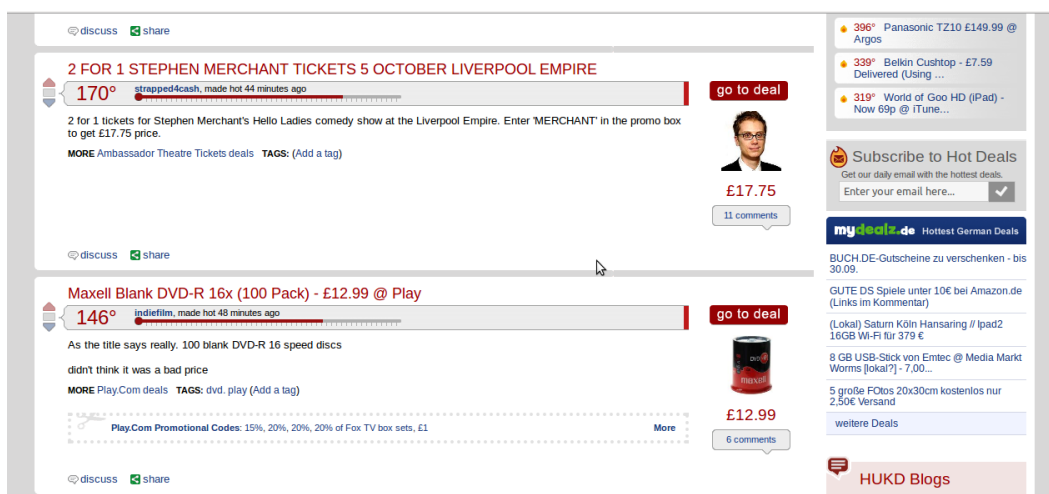


Figure 3.5: Example page from hotukdeals.com that is classified as forum page.

flat list of posts. Such forums are hard to detect for the current classifier.

3.6 Forum Page Clustering

Not all the pages crawled by the focused forum crawler from the previous section contain useful information. Among the less interesting ones are search interfaces, overview pages, and license as well as other legal information pages. It is necessary to filter out such pages before running data extraction (see Section 3.9). This step is marked as step (2) in Figure 3.1. The next sections provide further details.

3.6.1 Existing Approaches

As explained in Section 3.1, Cai et al. (2008) propose four categories of important pages. These categories are *list-of-board*, *post-of-thread*, *list-of-thread*, and *user-profile*. Since they contain the user generated content, post-of-thread pages are the most interesting pages for Effingo. In order to find those page types, Cai et al. (2008) present a *repetitive-region-based-clustering* approach combined with a *URL-based-sub-clustering*. The repetitive-region-based-clustering is based on the fact that forum Web pages are created by different templates. Following this assumption pages generated by the same template should contain regions which look pretty much the same and are located at similar positions on every page. Based on these repetitive patterns the authors identify two crucial findings:

- Repetitive regions within a forum Web page are able to describe the page's structure in a robust way.
- The location of links within a forum page is important.

The HTML structure of those repetitive regions is called repetitive patterns. Web forum pages may now be clustered using the repetitive patterns and their locations. The core of that algorithm is formed by "repetitive region detection", "records in region alignment", and "tree alignment cost calculation". The exact operating principles those algorithms are described by Zhai & Liu (2006) and by Zheng, Song, Wen & Wu (2007).

In a second step URL-based sub clustering is applied to further refine the created clusters. For this purpose the pages' URLs are examined, the interesting parts being the paths following the host name and the parameters separated from the path by a "?". Parameters are available as (key-value) pairs. Using this information it is possible to calculate the similarity of two URLs by comparing:

- the number and order of path elements
- the keys of the parameters.

Finally, the system created clusters of forum Web pages based on their repetitive regions and URL formats. Using these clusters the system created by Cai et al. (2008) is also able to

find an effective traversal path for a forum and to adjoin discussion threads distributed over multiple pages.

The analysis of forums carried out by Cai et al. (2008) is interesting and forms the foundation for the features we used for forum page classification. Unfortunately, they employed an unsupervised learning approach. Such approaches are not easy to apply in situations with a growing input set. As a consequence every time a new page is found the clustering needs to be adapted. Therefore, in this work we evaluate the performance of supervised learning algorithms to resolve this dilemma. Such algorithms are well suited for growing input sets, since they process one document at a time using a fixed classifier trained on some human-labeled data.

3.7 Features for Forum Page Classification

In order to identify crucial features for page type classification, the forums listed in Table 3.8 were used. The first four represent large forum software systems running many forums.

Table 3.8: Forums used to identify common forum page types and features.

Forum Software	Example URL
vBulletin Version 4.1.6 (Beta 1)	http://vbulletin.com/forum/forum.php
vBulletin Version 3.6.4	http://forum.spiegel.de
phpBB	http://phpbb.de/community
Phorum	http://phorum.org/phorum5/index.php
Individual	http://forums.d2jsp.org

The last one is a self-made forum to make sure small systems can be handled by page classification as well. The structure of the pages in those forums leads to the features explained in the following sections.

3.7.1 Page URL

As a simple approximation to the URL-based sub-clustering proposed by Cai et al. (2008) Effingo uses the number of parameters from each page's URL to classify pages into forum page types. An analysis of the sites from Table 3.8 results in the distribution of URL parameters shown in Table 3.9 for each page type and forum. Obviously this feature possesses a certain discriminatory power, although this does not apply to all forums. Some use restful URLs without parameters, like vBulletin 4.1.6 whereas for others, it is only useful as a means to distinguish certain types or groups of types. However, the separation of list-of-board pages as well as post-of-thread and user-profile pages seems to be most promising.

Table 3.9: Number of parameters used to access a page of each type in different forums.

	Phorum	phpBB
list-of-board	0	0
list-of-thread	1	1
post-of-thread	2	2
user-profile	2	2
	vBulletin 4.1.6	vBulletin 3.6.4
list-of-board	0	0
list-of-thread	0	1
post-of-thread	0	1
user-profile	0	1

Table 3.10: Average link-to-text ratio for different forum page types.

	vBulletin 4.1.6	vBulletin 3.6.4	phpBB	Phorum	Self-Made
list-of-board	44%	54%	34%	17%	51%
list-of-thread	47%	56%	37%	56%	39%
post-of-thread	49%	20%	33%	25%	28%
user-profile	34%	63%	-	49%	-

3.7.2 Link-to-Text Ratio

The example site analysis revealed some conspicuous patterns with regard to the distribution of linked and non-linked text over the different page types. Overview pages, like list-of-board pages and list-of-thread pages, are usually heavily linked whereas post-of-thread pages and user-profile pages do not contain as many links.

Table 3.10 shows the link density as described in Section 3.4.5, for the different forum page types. The table illustrates that the expected patterns exist but are not as conspicuous as initially expected. The problem is twofold. Some forums have mostly short threads and thus little unlinked text created by a user. This reduces the link-to-text ratio in post-of-thread pages. Secondly, there are forums having a lot of explanatory text for each sub forum which is not linked. In that case list-of-board pages show a similar or even smaller link-to-text ratio than post-of-thread pages and user-profile pages.

Nevertheless, the link-to-text ratio provides a valuable clue as to examine which page type a page belongs to and shall thus be considered as a feature for the page type classification.

3.7.3 Height of List Elements

Generally, pages are divided into *detail pages* and *list pages*. Detail pages such as user-profile pages display information for only one data entry. List pages contain information on multiple data entries, usually listed in consecutive order.

Data entries such as posts for post-of-thread pages, thread titles for list-of-thread pages, and sub forums for list-of-board pages are usually displayed that way. Figure 3.6 shows an example of a list for a list-of-thread page. Those lists usually look very different and the main distinguishing feature seems to be the height of each list element on the screen. This height is influenced by the content the list elements need to display and the CSS style applied. The post-of-thread page elements usually contain more content than the other classes and thus are much higher.



Figure 3.6: Example list structure for a list-of-thread page.

Finding the list with the main content is a demanding task. Any list consists of repeating tag patterns. In addition, we assume that the required main content list contains the most text in the majority of the forum pages. As a consequence, the search algorithm starts from the HTML element containing the most text. It assumes that this element is a list element, even if it only forms a list of length one. This assumption is helpful to handle threads having only one post or user-profile pages with only one large content block. Using those list elements, an HTML rendering engine is applied to render the HTML page as it would be rendered by a browser. It is now possible to calculate the height in pixels of the detected list elements from the rendered HTML representation. The mean height value of all list elements of one page is finally used as a feature for page classification.

The Identification of the height of the elements of the main content list consists of two steps. At first the systems searches for the main content list. In a second step the average height of the list elements for this main content list is calculated from the HTML source code of the page.

Each HTML page, regardless of its class, contains many lists. For example, the main menu or the page flipping links form a similar list structure.

3.7.4 Page Flipping Links

Especially list-of-thread and post-of-thread pages contain so called page flipping links providing access to pages showing old threads on list-of-thread pages or newer posts of long threads on post-of-thread pages. If these links are identified a separation of both types from the other three types is possible.

Page flipping links usually consist of an ordered sequence of numbers, with most of them embedded in `<a>` tags as shown in Figure 3.7. Page flipping links are used if there is at least a second page. Hence there must be an `<a>` node containing a numerical value x as well as a second `<a>` node containing a numerical value with $x + 1$. In the simple case shown in Figure 3.8(a) it should be possible to identify page flipping links by searching the siblings of all `<a>` tags containing a number x for other `<a>` tags containing the number $x + 1$.

Seite 1 von 121 1 2 3 11 51 101 > Letzte »

Figure 3.7: Example of page flipping links as usually encountered on “list-of-thread” or “post-of-thread” pages.

However, usually page flipping links do not occur as direct siblings. So for the example as shown in Figure 3.8(b) a different approach is required. To identify page flipping links in such cases it is necessary to consider `<a>` tags which are no siblings. The approach still uses only `<a>` tags containing numbers. However, additionally `<a>` tags with the same depth in the HTML tag tree are used.

If the algorithm finds page flipping links on a page the feature is `true`, otherwise it is `false`.

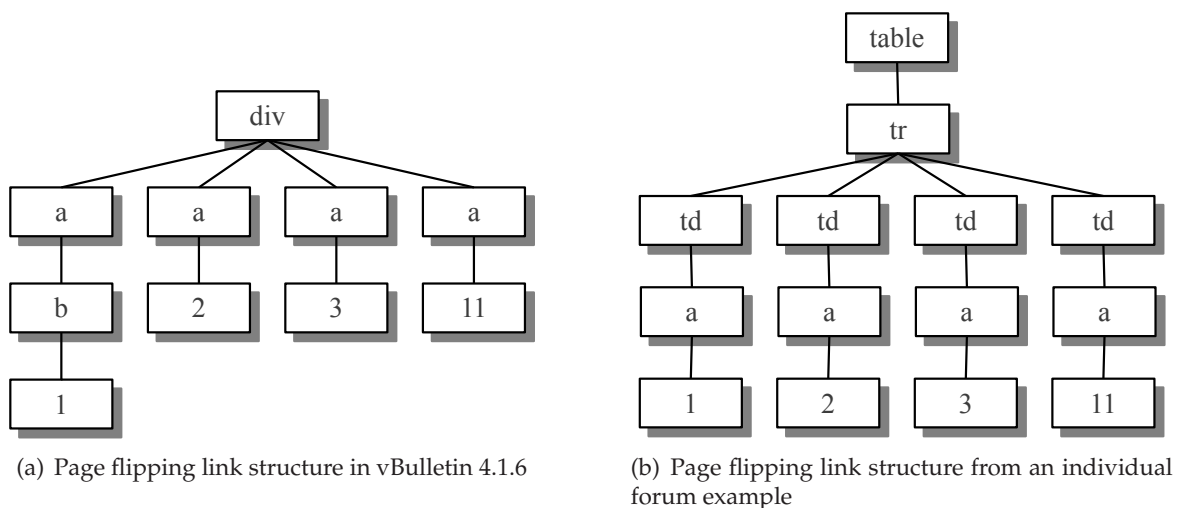


Figure 3.8: Examples for the structure of page flipping links

3.8 Results of Forum Page Classification

This section presents the evaluation of forum page classification using the features from the previous section. For that purpose a dataset labeled with the forum page types from Section 3.1 is required as gold standard. Different classifiers will be tested on that dataset with different feature combinations in order to measure the performance of the presented classification approach. Performance is measured using macro averaged precision, recall, and F_1 -Measure.

3.8.1 Dataset

A dataset created from the forums presented in Section 3.7 was used for the evaluation. We divided this dataset into a training set and a test set. The composition of the final dataset is

Table 3.11: Distribution of forum page types within the dataset used for evaluation

	list-of-board	list-of-thread	post-of-thread	user-profile	Σ
Training Set	4	178	200	101	483
Test Set	1	50	50	50	151

shown in Table 3.11.

3.8.2 Experiments

For evaluation purposes we ran experiments using a support vector machine (SVM) with RBF kernel, decision trees (DT), and the Naïve Bayes (NB) classifier. The parameters C and σ for the SVM were configured using a script provided by the LIBSVM library⁵. Both are explained in Section 2.3.3.

Table 3.12: Results for forum page classification using different feature sets and different classifiers; L=Link to Text ratio, H=Height of List Elements, F=Page Flipping Links, U=Page URL; P=Precision, R=Recall, $F_1=F_1$ -Measure

	Naïve Bayes			Decision Tree			Support Vector Machine		
	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1
L	0.74	0.50	0.60	0.70	0.40	0.51	0.70	0.50	0.58
H	0.75	0.50	0.60	0.50	0.41	0.45	0.49	0.40	0.44
L+H	0.74	0.50	0.60	0.49	0.46	0.47	0.74	0.50	0.60
L+F	0.74	0.50	0.60	0.75	0.50	0.60	0.74	0.50	0.60
U+F	0.17	0.25	0.20	0.17	0.25	0.20	0.33	0.25	0.28
L+H+F	0.74	0.50	0.60	0.75	0.50	0.60	0.73	0.50	0.59
L+H+U	0.74	0.50	0.60	0.49	0.47	0.48	0.49	0.49	0.49
H+U+F	0.74	0.50	0.60	0.50	0.72	0.59	0.62	0.69	0.65
L+H+F+U	0.75	0.50	0.60	0.70	0.38	0.49	0.74	0.50	0.60

Discussion

Table 3.12 presents the macro-averaged performance statistics using different classifiers and feature sets. It shows that the most important feature is the Height of List Elements (H). None of the other features seem to provide additional insight for the evaluated classifiers. Only the feature combination Height of List Elements, Page URL, and Page Flipping Links (H+U+F) improves on the recall in exchange for 10% of precision. The biggest problems are the separation of list-of-thread pages from list-of-board pages as well as the separation of post-of-thread pages from user-profile pages. Both problems are due to the fact that pages from those classes look quite similar. Most classifiers, after all, are able to separate user-profile pages from post-of-thread pages using the correct features, like for example Height

⁵<http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html>

of List Elements. Anyhow, none of the examined classifiers is able to distinguish between list-of-board and list-of-thread pages, which might be caused by the small amount of training data for that type of page as well as by their similarity. However, a forum's data are displayed on user-profile pages and post-of-thread pages. Therefore, these two types are the ones required for forum data extraction. The micro-averaged performance statistics (not shown in Table 3.12) show that it is possible to identify both with high confidence.

At this stage of the complete extraction process, the content pages are found. The next sections are going to describe how to extract the required data from those pages in order to fill the Effingo data storage.

3.9 Forum Data Extraction

Next, the step marked (3) in Figure 3.1 uses the pages identified as containing relevant data by the algorithms presented in the last sections. At this point the algorithm is able to identify which pages belong to a forum and which do not. With regard to the first, the algorithm is able to identify which of them contain content and which do not. The content pages are user-profile and post-of-thread pages. However, it still remains unclear how to use the information from user-profile pages for question answering. In addition, user-profile pages are not accessible on all forums. Therefore, they are excluded as well. The remaining page type is the post-of-thread-page type. Pages belonging to this type contain the user generated text and, as such, the questions and answers required for question answering. However, the user generated content as well as any other content on each post-of-thread page is still embedded in semi-structured HTML code. The last step is the extraction of information entities such as title, user generated text, author name etc.

This section describes an approach to automatically create extraction patterns for those fields from a set of post-of-thread pages. The approach is based on the detection of those patterns for a specific forum from a small set of pages from forum and to apply those extraction patterns to the remaining pages and pages generated in the future. The following sections start with a description of the entities which are supposed to be extracted and continues with an explanation of related approaches. Related approaches mainly originate from the area of general Web page data extraction, but there also is one work concerned with forum data extraction. The following two sections discuss the algorithm we used for data extraction as well as experiments to evaluate its performance.

3.9.1 Extracted Entities

A Web page such as a forum page consists of information *entities* and template, boilerplate, or structuring content. The template code consists of template tokens, while the information entities consist of data tokens. An entity is the smallest unit of content a page displays. The focus of forum data extraction refers to the extraction of the entities that are common to all

Web forums, thereby creating a unified database for Web forums. However, there usually is a large amount of entities per page which are either very forum-specific or do not contain useful information. Those entities are not relevant for forum data extraction. Although they hide the important entities and add noise to the page. This complicates the assignment of entity types to concrete entities on the page. For example, see the two forum posts in Figure 3.9. Additionally, entities might contain template text. Template text is static text usually

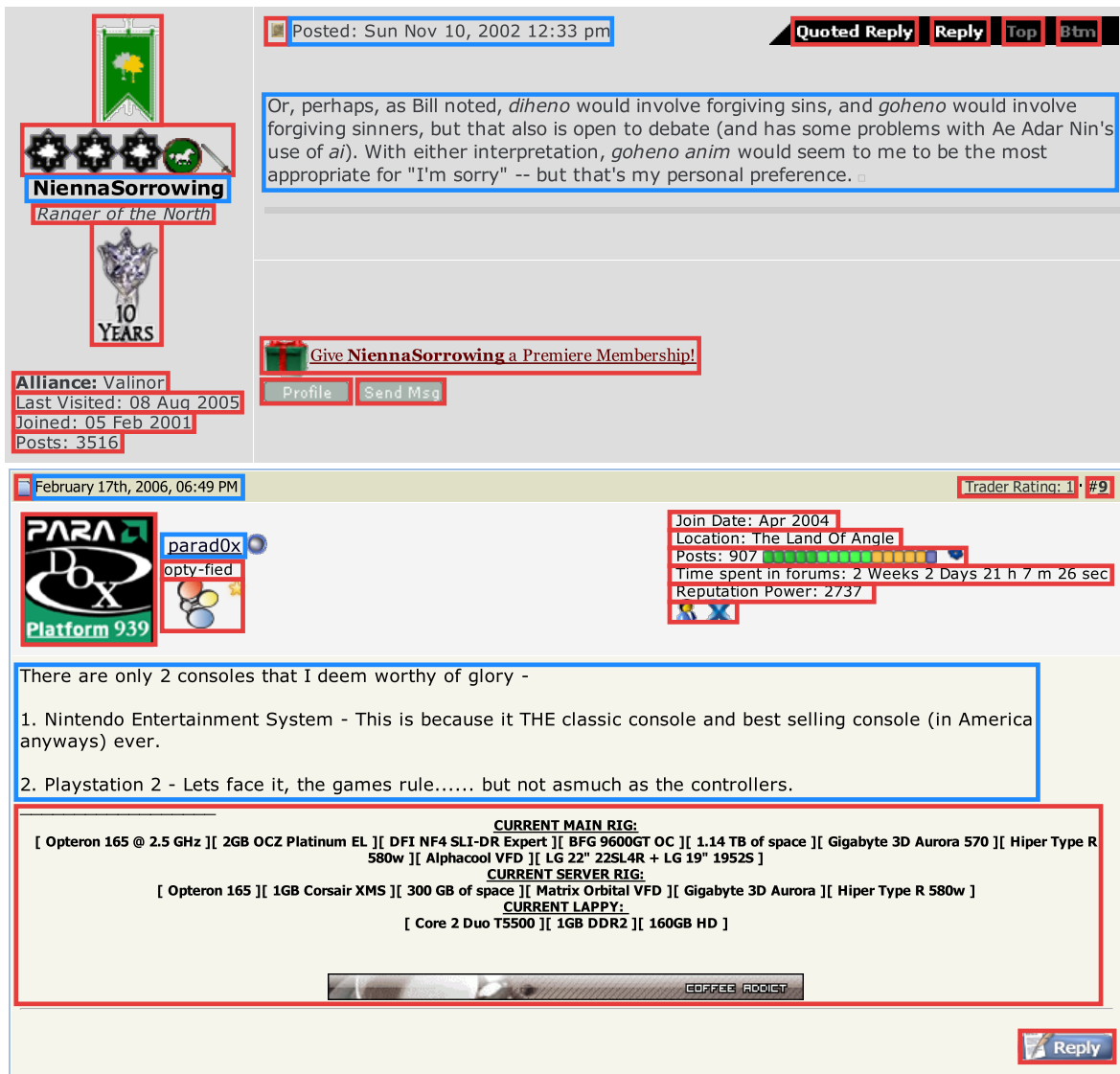


Figure 3.9: Examples for extraction candidates.

inserted to describe the entity and make the text readable. For example consider the two text nodes "username: nick" and "publication date: 07/04/2011". Both contain a template part and an entity part. With regard to the first example, the template part is "username: ", while the entity is just "nick". The second example has the template "publication date: ", marking the entity "07/04/2011". Even though the template text identifies the correct entity, it must

be separated from it during extraction.

In order to get a comprehensive impression of the entities which are common among Web forums and how they are represented we analyzed the HTML code generated by the ten most frequently used forum systems according to <http://www.big-boards.com/statistics>.

The analysis shows that there are two categories of forum data, which are user data and post data. Both are discussed in detail in the following paragraphs.

User Data

User data is divided into collected and aggregated user data.

Collected user data is provided by the user himself and usually presented on a user-profile page. Sometimes those entries are also displayed in combination with every post the user creates. This, however, usually depends on the user's choice about how much data they want to provide and show to the forum community. Examples of such data entries include the user's home town, their job, contact data, and, for some forums, e.g. product support forums, also topic specific fields like products the user owns. The only obligatory user data field is the post author's username. Table 3.13 shows the usual ways to display collected user data on post-of-thread pages.

Table 3.13: Analysis of how collected user data is presented on HTML pages for the ten most popular forum systems according to <http://www.big-boards.com/statistics>.

Posts Metadata	vBulletin 4	vBulletin 3	IPB 3	phpBB 3	phpBB 2
Username	T, L	T, L	T, L	T, L, P	T
Residence	T, P	T, P	T, P	T, P	T, P
Web Site	–	–	–	I	I
Signature	T	T	T	T	T
IM Contact	I	I	–	I	I
E-Mail address	–	–	–	–	I
Posts Metadata	SMF 2	SMF 1	WBB 3	Jive SBS	Jive CS
Username	T, L	T, L	T, L	T, L	T, L
Residence	–	–	T, P	–	–
Web Site	I	I	I	–	–
Signature	T	T	T	T	T
IM Contact	I	I	I	–	–
E-Mail address	–	I	–	–	–

Key:

T : Presented as text

L : Linked text

I : Presented as linked icon

P/S: Marked by prefix/suffix template text

– : not available

Aggregated user data is calculated from user activity on the forum and also presented on

user profile pages. Similarly to collected user data, some of these entries are presented with every post authored by the user. Examples for collected user data are the amount of posts a user has written, the user's registration date for the forum and their membership status. Some of the more common aggregated user data entries shown with each post and the way they are presented on post-of-thread pages are shown by Table 3.14. The table illustrates

Table 3.14: Analysis of how aggregated user data is presented on HTML pages for the ten most popular forum systems according to <http://www.big-boards.com/statistics>.

Posts Metadata	vBulletin 4	vBulletin 3	IPB 3	phpBB 3	phpBB 2
Registration Date	T, P	T, P	T, P	T, P	T, P
Posts Written	T, P	T, P	T, P	T, P	T, P
User Status	T	T	T, P	–	–
Online Status	I	I	I	I	–
Posts Metadata	SMF 2	SMF 1	WBB 3	Jive SBS	Jive CS
Registration Date	–	–	–	T, P	T, P
Posts Written	T, P	T, P	T, P, L	T, S	T, P
User Status	T	T	T	I	I
Online Status	–	–	I	–	–

Key:

- T : Presented as text
- L : Linked text
- I : Presented as linked icon
- P/S: Marked by prefix/suffix template text
- : not available

that user data is usually represented by an icon or text with some template text in front. As mentioned before, the username is the only entry occurring for all presented forums. It is usually presented as text linked to its user-profile page.

Post Data

The data directly associated with a forum post consist of the user-generated content separated into the main text or body and the post's title. Some forums are not supporting individual titles per post. Especially modern forum systems noticed that the common user has no intention of providing an individualized title for an answer post. Hence, titles are an optional post data entry.

In addition to the user-generated content a post is extended with meta data and service links. Meta data consists of the date the post was published on and an optional update date if the post was edited after its creation. Service links are either used to directly address a post within its thread, to reply to a post by citing it or to jump to the start of the thread.

An overview of post data and the way it is presented on a forum page is given by Table 3.15. Some of those data entries are optional and only displayed if the user provided them. Those

Table 3.15: Analysis of how post data is presented on HTML pages for the ten most popular forum systems according to <http://www.big-boards.com/statistics>.

Post Metadata	vBulletin 4	vBulletin 3	IPB 3	phpBB 3	phpBB 2
Title	T, O	T	–	T	T, P
Publication Date	T	T	T	T, P, M	T, P
Update Date	T, O, P	T, O, P	T, O	–	T, P
Cite	T	I	T, I	I	I
Direct Link	T	T	T	I	I
Start Link	–	–	–	I	T
Post Metadata	SMF 2	SMF 1	WBB 3	Jive SBS	Jive CS
Title	T, L	T, L	T	T	T
Publication Date	T, P	T, P	T	T	T, P
Update Date	T, P	T, P	T, P	–	T, P, O
Cite	–	T, I	T, I	–	T, I
Direct Link	T	I	T	–	–
Start Link	–	I	–	–	–

Key:

T: Presented as text

I: Presented as linked icon

P: Marked by prefix template text

O: Depends on author's choice

–: not available

entries are marked by an “O”.

In addition, Table 3.15 lists the posts’ title as an entity which sometimes is linked. This means that author usernames are not the only linked text within the meta data area of a post and thus, a link is not a feature unique to author usernames.

The analysis shows that besides the always available body, only the publication date exists in every forum. Hence, in addition to the author’s username, extraction will focus on publication date and body content.

3.9.2 Overview of Data Extraction Approaches

Early data extraction approaches are based on wrapper induction. This means that an extractor or wrapper for some semi-structured format such as HTML is created from example instances with manmade marks as to which entities are to extract. Examples are explained by Muslea (1999), Laender, Ribeiro-Neto, da Silva & Teixeira (2002), and Chang, Kayed, Girgis & Shaalan (2006). This approach is of course only viable for a small number of forums. For each new forum a human supervisor would be required and if a forum changed its markup, someone would need to update the wrapper. This problem is also called wrapper maintenance and discussed for example by Meng, Hu & Li (2003).

More recent works attempt to eliminate the manual step. They are divided into approaches for both page types as explained in Section 3.7.3. In order to find the template part of detail pages, multiple example pages are necessary. However, if the wrapper generator examines a list page and discovers an instance with enough entries, it might be able to create a wrapper from a single instance. Most content pages such as list-of-board pages, list-of-thread pages and post-of-thread pages in a forum belong to this type. Since post-of-thread pages contain the discussed topics in form of user-generated content and since the goal for the Effingo system is answering questions on the topics discussed in a forum, our discussion on existing data extraction approaches will only focus on list pages.

The following sections present crucial work from the two most closely related research areas the first of which is generic unsupervised data extraction. Most research has concentrated on this area. The second field of interest explicitly focused on forum data extraction, which will later be used as a baseline for the algorithms in this work.

3.9.3 Unsupervised Data Extraction

The goal of unsupervised data extraction is the separation of template and content parts on arbitrary Web pages. Important research in this area was provided by the RoadRunner system described by Crescenzi, Mecca & Merialdo (2001), EXALG and its extensions described by Zhai & Liu (2006), and Lo, Ng, Ng & Chan (2006) and FiVaTech developed by Kayed & Chang (2010). In order to understand how these systems work the following paragraphs provide detailed explanations of the above mentioned methods using the example HTML code from Figure 3.10.

1	<code><html></code>	1	<code><html></code>
2	<code><body></code>	2	<code><body></code>
3	<code><h1>Title: Favorite Movies</h1></code>	3	<code><h1>Title: Regular Expressions</h1></code>
4	<code></code>	4	<code></code>
5	<code></code>	5	<code></code>
6	<code><div>Author: Max</div></code>	6	<code><div>Author: Hans</div></code>
7	<code><div></code>	7	<code><div></code>
8	<code>Title: What's your favorite movie?</code>	8	<code>Title: RegEx Whitespace Modifier</code>
9	<code></div></code>	9	<code></div></code>
10	<code><div></code>	10	<code><div></code>
11	<code>My favorite movie is</code>	11	<code>Hi, how do I set the</code>
12	<code>Star Trek: First Contact</code>	12	<code>Ignore-Whitespace Modifier in Java?</code>
13	<code></div></code>	13	<code></div></code>
14	<code></code>	14	<code></code>
15	<code></code>	15	<code></code>
16	<code><div>Author: Moritz</div></code>	16	<code></body></code>
17	<code><div></code>	17	<code></html></code>
18	<code>Title: Re: What's your favorite movie</code>		
	<code>?</code>		
19	<code></div></code>		
20	<code><div>I'd prefer Forrest Gump</div></code>		
21	<code></code>		
22	<code></code>		
23	<code></body></code>		
24	<code></html></code>		

Document 1

Document 2

Figure 3.10: Example HTML pages

RoadRunner

In 2001, Crescenzi et al. (2001) presented a tool for automatic content extraction from HTML documents. RoadRunner works on multiple pages of equal class with the aim to find a common schema. For this purpose it linearizes the HTML tag tree as a list. Each position of that list is either an HTML tag or #PCDATA. Figure 3.11 presents the results of running the system on the example documents from Figure 3.10. The wrapper is generated as a regular expression from a line-by-line comparison of both documents. If two positions differ, they need to be inspected. If both contain #PCDATA like on line 4 of the example above, a simple placeholder is inserted. If there are different tags a cross search starts. For each document the tags following the mismatching line from the other document are compared to the tag at the mismatching line. If a match is found the area between the new match and the mismatching line is examined further. This area either is an optional element or an area with a different number of instances. The last tag of the spanned area is called the terminal tag.

The example contains non-matching areas between line 16 and 19. In this case the terminal tag is ``. To limit the non-matching area the corresponding start tag is discovered. Thereafter, the repetition of the non-matching area is examined. For that purpose the area between start and end tag is pushed up as long as the position of the last match between both documents has not been reached yet. If equal areas are identified by that means, a list of equal items is found. The example from Figure 3.11 shows this using the list items

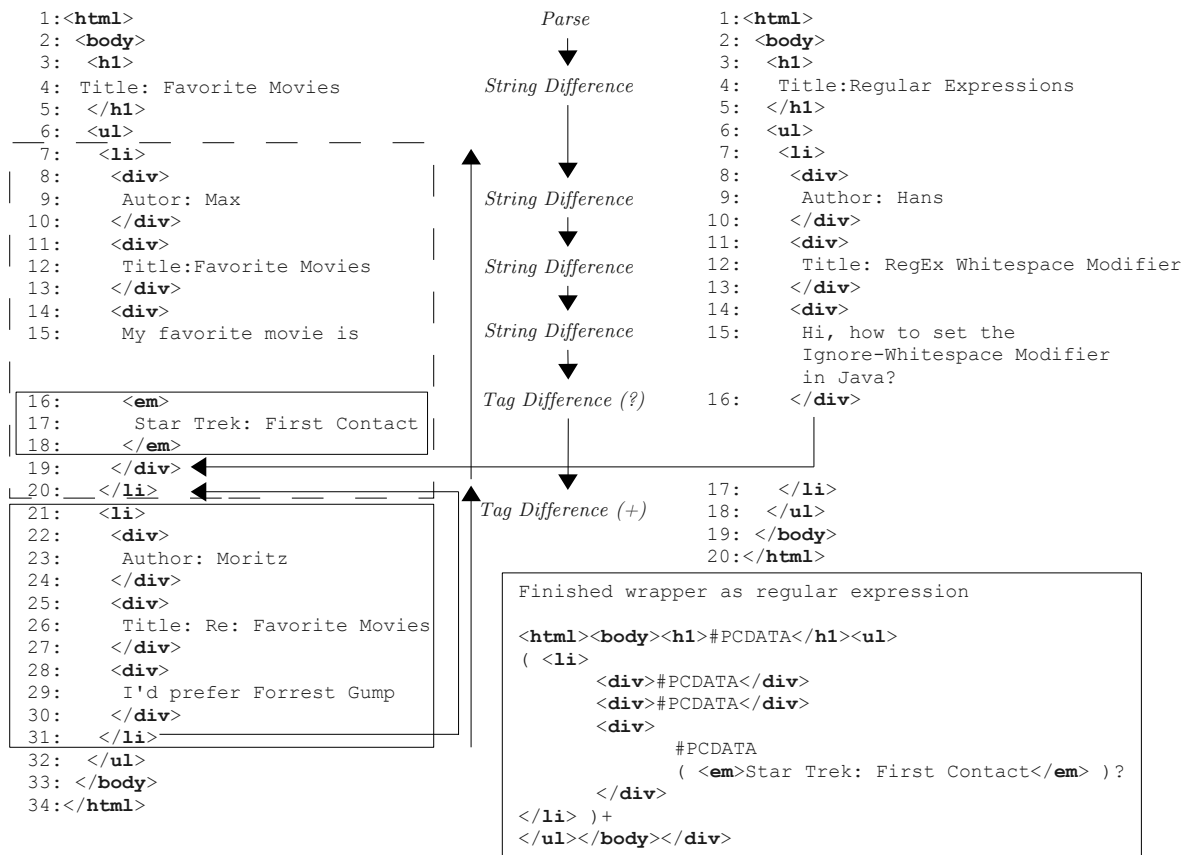


Figure 3.11: Creating a regular expression describing the template of two Web pages using Roadrunner (Crescenzi et al. 2001)

`` starting on line 7 and 21. Hence, a repetition is established within the resulting regular expression (+). If no repetition is found the area is marked as optional (?). Differences inside of equal repetitions are handled recursively. The wrapper is refined using additional documents. Its final form is shown as a regular expression in Figure 3.11.

However, RoadRunner as one of the early approaches is subject to several restrictions. Firstly, it considers every tag as part of the page template, thereby ignoring that user generated content may contain HTML tags as part of the page data. Furthermore, it considers `#PCDATA` as either belonging completely to the template or completely to the content. However, the example in Figure 3.10 clearly shows that text sometimes contains template parts as well as data parts, such as the “Author” or “Title” token.

EXALG

EXALG was developed by Arasu & Garcia-Molina (2003) and works on a finer level of granularity. It considers both, tags and text, as tokens either belonging to the template or data part of an HTML page. The basic idea is that tokens occurring with the same frequency belong to the same data structure. The algorithm tries to separate template tokens from data tokens by creating a schema for a Web page. The authors derive the problem of schema generation from the generic problem of regular grammar generation. They build upon previous work creating a DTD from a set of XML documents.

An EXALG-generated schema consists of a sequence of types T from the following list:

1. T : a simple type, consisting of one token
2. $\langle T_1, \dots, T_n \rangle$: a tuple containing instances of types T_1, \dots, T_n
3. $\{T\}$: a set containing an instance \mathcal{B} of type T .

The types 2 and 3 enable composite data records as well as optional, multiple or nested occurrences. They are called type constructor since they enable the creation of new types. Type 2 is called the tuple constructor while type 3 is called set constructor. The schema for the example presented by Figure 3.10 follows the schema $\langle \mathcal{B} \{ \langle \mathcal{B}, \mathcal{B}, \mathcal{B} \rangle \} \rangle$. The first \mathcal{B} marks the threads title, while $\langle \mathcal{B}, \mathcal{B}, \mathcal{B} \rangle$ stands for one forum entry.

The algorithm works as follows: A set of documents \mathcal{P} is used as input. The documents are tokenized and every token is assigned to an *occurrence vector*. Each occurrence vector has a length of $|\mathcal{P}|$ and contains token frequencies for each document. For example, the first document in Figure 3.10 contains the term `` two times, while it occurs only once in the second. So the occurrence vector would be $v_{\langle li \rangle}^T = (2 \ 1)$. A second step searches for *equivalence classes* based on their occurrence vectors. Tokens with equal occurrence vectors form an equivalence class. Each equivalence class preserves the order of tokens from the documents. With regard to the example in Figure 3.10 this means that the equivalence class (1 1) contains the occurrence vector for the token `<html>` as first element and for the token `<body>` as second element.

Large equivalence classes are marked as *large and frequent equivalence classes*. Based on the assumption that template tokens are very common among documents, they should occur only in such large and frequent equivalence classes. For their experiments the authors of EXALG regard every equivalence class of size 3 or higher as large and frequent. However, they report that this sometimes leads to tokens erroneously included in the wrong category. An iterative process follows the initial creation of equivalence classes. It creates new equivalence classes and extends existing ones. It checks for every existing equivalence class whether it occurs multiple times within the same document; i.e. whether the token occurs several times at different positions. If this is the case the context of the token is examined. The context is created using the path from the HTML parse tree and the position within the region. An equal token occurring at different positions has different roles. The token "Title:" for example has two different roles in Figure 3.10. The role of the first token is defined by the path `/html/body/h1`, while the role of the second token is defined by `/html/body/ul/li/div`. If a token has multiple roles it is removed from the previous equivalence class and added to another equivalence class matching the new occurrence vector. The token "Title:" initially had the occurrence vector (2 2) and belonged to the equivalence class {Title:}. After one iteration "Title:₁" has the occurrence vector (1 1) and "Title:₂" the occurrence vector (2 1) because "Title:" with context 1 occurs once in each of the two example documents whereas it occurs twice in document 1 and once in document 2 with context 2. Therefore, "Title:₁" is added to the large and frequent equivalence class `<html>, <body>, <h1>, Title:1, </h1>, ...` while "Title:₂" is added to another equivalence class. The algorithm stops as soon as there is no more need to further extend any of the existing (large and frequent) equivalence classes.

Finally, the template is reconstructed from all large and frequent equivalence classes. The template generation starts with the root equivalence class, which is the one consisting only of "1" elements (i.e. there is always one `<html>` token per page). If there is a gap between tokens from a specific large and frequent equivalence class, the algorithm continues with the large and frequent equivalence class the next token belongs to. Positions with no tokens from a large and frequent equivalence class are marked with a placeholder. These are the areas where data tokens are presumably located.

Even though the algorithm presented in this thesis is loosely based on ideas from EXALG, it is different since EXALG works on arbitrary HTML pages, while the proposed approach is focused on post-of-thread forum pages. Therefore it is possible to draw certain premises which the authors of EXALG could not assume. So, in contrast to EXALG, we do know that there is a certain amount of posts on each page and that each post contains exactly one instance of each data entity. In addition, EXALG has no capability to assign semantics to the extracted entities, whereas here we know that we need to find instances of usernames, publication dates and body texts.

FiVaTech by Kaye & Chang (2010) advances the algorithm of EXALG and avoids the problem that tokens are sometimes included in the wrong category. However, they reintroduce the drawbacks of RoadRunner. With this algorithm templates text is no longer separable

from entities in the same text segment and embedded HTML tags are always considered to be part of the template.

3.9.4 Existing Forum Data Extraction Approaches

One recent work specifically concerning forums was introduced by Yang, Cai, Wang, Zhu, Zhang & Ma (2009). Their approach is based on an analysis of the link structure of a forum site. Since the forum structure is analyzed, such a Site Level procedure requires at least 2,000 crawled pages in order to be successful. In contrast, the Effingo algorithms operate on Page Level and thus achieve good results with a smaller set of analyzed pages.

Based on the site structure Yang et al. (2009) apply rules formulated in predicate logic to additionally extract author, publication date, and body of a post. They assume that authors' usernames are always linked to their user-profile page, that bodies contain the largest text and that publication dates are parsable and available in ascending order. The analysis of forums from Section 3.9.1 proves this to be wrong for many forums. Especially if not logged in, the links to the user-profile pages are not accessible. The third assumption might be true for classical forums but extraction will fail on Q/A sites, where posts are ordered according to user rating and not by publication date. Also, since Yang et al. (2009) run no analysis of the text content they extract, their approach is unable to separate template tokens from data tokens inside the same HTML element, similar to RoadRunner.

They evaluated their approach on a dataset with pages from 20 different forums. The dataset uses mostly pages created by the same forum software. For this reason it might be heavily skewed. Nevertheless, the approach by Yang et al. (2009) is the closest to ours and thus used as a baseline in Section 3.11.5.

3.10 Forum Data Extraction Algorithm

The following sections are going to show how to reference data entities within a post-of-thread page. They also illustrate how to map a data entity to a post and how to map a thread to all its posts even if they are distributed over several pages. Finally, we assign one of four labels to an entity using a classification approach similar to the one we employed to detect forum pages and types of forum pages in the previous section. For this purpose a distinguishing feature set is created again. In contrast to existing approaches, this approach is not only able to extract data entities but also to assign extracted entities to the correct attribute such as the author's username, the body text etc. In addition, the presented approach is also able to distinguish data content from template content inside of segments and not only on segment level and also is able to handle custom user-provided HTML content as part of data entities.

The forum data extraction is separated into two parts. At first, a set of pages from one forum site is analyzed to find valid extraction patterns. This small set is called analysis

set. The analysis step is necessary only once per forum to create extraction patterns for all required entities. By means of those patterns an extractor is able to find the entities from new and unknown pages of the same forum. An extraction pattern is a tuple of three elements: The first part is an extraction path referencing the HTML tag which encompasses the text containing the entity. The second is a token pattern marking tokens inside the text as either belonging to the page's template text or to the data. The third part is a group index referencing the entities data part in the token pattern. This can be formalized to:

$$(Extraction\ Path, Token\ Pattern, Group\ Index)$$

Those extraction patterns are required to address all instances of a data entity on one post-of-thread page.

Technically, an extraction path is an XPath Expression, a token pattern is a regular expression and a group index is an index into a regular expression group enclosed in "(" and ")".

Thus for the example from Listing 3.4 the analyzer should create extraction patterns such as:

- **Author**

Extraction Path /html/body/div/div[contains(text(), {Posted, by, at, in, response, to})]/text()

Token Pattern Posted by (.*) at (.*) in response to (.*)

Group Index g = 1

- **Date**

Extraction Path /html/body/div/div[contains(text(), {Posted, by, at, in, response, to})]/text()

Token Pattern Posted by (.*) at (.*) in response to (.*)

Group Index g = 2

- **Post identifier**

Extraction Path /html/body/div[starts-with(@id, "post")]/@id

Token Pattern .*?([0-9]+).*

Group Index g = 1

- **Thread identifier**

Extraction Path /html/head/link[@rel="canonical"]/@href

Token Pattern .*?thread/([0-9]+).*

Group Index g = 1

The following sections explain the analysis process, for which an overview is shown in Figure 3.12. At first, the analysis set is selected from all post-of-thread pages of one forum. The post-of-thread pages identified by the process described in Section 3.6 could serve as an input to this step. Using their URL it is easy to find the pages belonging to the same forum. It is important to select maximum length analysis pages during this step. If only pages without the maximum amount of posts per page are selected, extraction pattern generation for the last posts will definitely fail. The second step can be defined as a segmentation process


```

1  <html>
2  <head>
3  ...
4  <link rel="canonical" href="http://example.com/thread/1111" />
5  </head>
6  <body>
7  ...
8  <div id="post4711">
9    <div>Posted by Moritz at 12/11/2011 04:35 pm in response to Frank</div>
10   ...
11  </div>
12  ...
13 </html>

```

Listing 3.4: Example page for showing the style of extraction patterns.

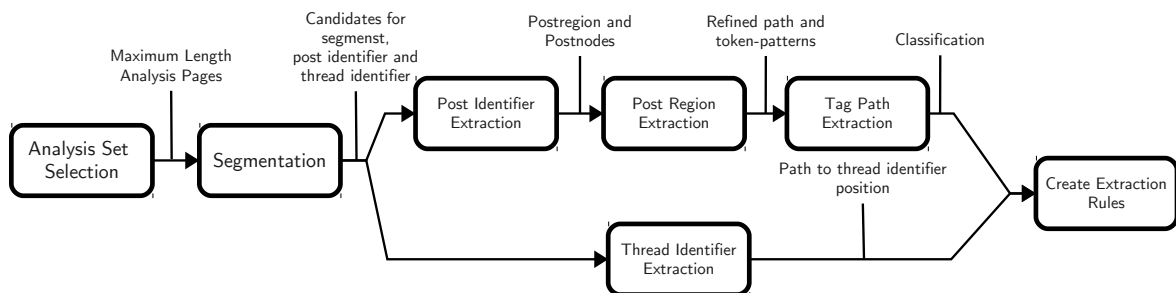


Figure 3.12: Overview of generating extraction patterns during the analysis process on a small set of analysis pages.

during which candidate segments are selected that possibly contain the required entities. The selection of segments is explained in Section 3.10.1. The process then splits into post extraction and thread identifier extraction, the latter of which is explained in Section 3.10.6 and is necessary to associate posts which belong to the same thread but are distributed over multiple HTML pages. The first step of the analysis process is split into three sub steps. Initially, the identifier for posts is extracted as a means to retrieve all segments belonging to the same post. Second the region surrounding the post is identified, and lastly, tag paths are extracted for all segments inside each post. Those tag paths are classified in order to find out which entity they contain. These steps are explained in Sections 3.10.2, 3.10.3, 3.10.4, and 3.10.5.

3.10.1 Segmentation

HTML unfortunately does not explicitly mark entities. Tags such as `<author>` would render the whole forum page extraction useless. However, entities are only surrounded with tags to create structure and formatting such as `<div>` or ``. Each such tag surrounds a segment of content as described in Section 3.2. Fortunately, these tags are good reference points for contained entities and sometimes even contain clues to their content. For this reason, as a first step candidate segments are extracted from the post-of-thread page. The list of candidates should not miss any required entities, but should exclude as many unnecessary segments as possible. In order to achieve this result the following requirements are applied

to select candidate segments:

- **Granularity** The granularity of a segment needs to be as small as possible to contain only one entity but at the same time as big as necessary in order to include all text belonging to the entity. Specifically, it is important that the segment does not split up user-generated text. An example segmentation might look like:

Max wrote on 11/11/2011: I have a question...

For this purpose it is important to know that HTML elements are divided by the W3C into block-level elements and inline-level elements⁶. Block-level elements are those HTML elements that generate a break after themselves. Examples are `div` and `p`. They are usually used for structuring a Web page. On the hand, inline-level elements are those elements occurring inside of continuous text. Examples are `strong` or `span`. They are used for formatting text and for providing interactivity to a Web page. Valid candidate segments are surrounded by HTML block-level elements satisfying the following properties.

- contains text
- contains inline-level elements

Each block-level element containing another block-level element is considered as valid segment if it satisfies the conditions as well.

- **Similarity** Candidate segments containing the same data entity shall have the same relative tag path. For two forum posts, for example, the segments surrounding the publication dates in both need to have the same relative tag path.
- **Features** Since segments are used for classification as explained in Section 2.3, it is necessary to keep additional features for each, such as:
 - HTML attributes and their values
 - Formatting
 - Links
 - Relative Position in Post

Listing 3.5 shows a small example page. Applying page segmentation results in the follow-

```
1 <html>
2   <head>...</head>
3   <body class="page">
4     <div id="4711">The first example.</div>
5     <div id="4712">A <strong>second</strong> example.</div>
6   </body>
7 </html>
```

Listing 3.5: A small HTML example

ing list of candidate segments:

⁶<http://w3.org/TR/html14/struct/global.html#h-7.5.3>

Segment S_1 The first example.

Segment S_2 A `second` example.

As shown by the example, two block level elements create two segments. It also illustrates that the second text would have been split up if the segments surrounding inline-level elements had been considered as well.

3.10.2 Post Identification

The list of segments spans the whole HTML page. For the purpose of data extraction only the segments pointing to the actual forum posts are important. Therefore, we now present an approach to select those segments that are part of a forum post.

The approach is based on the fact that every forum post is identified by a unique identifier within the forum system. This is necessary to save the post in the underlying database. In order to address a post, for example via service links, the forum software inserts the post identifier into the HTML markup. If we are able to find an instance of the post identifier and the corresponding HTML element, all tag paths containing that element are also part of a post.

Usually an identifier is expressed by a numerical value as part of an HTML id or name attribute. Therefore, all numerical values from all id attributes and name attributes are initially considered as post identifier candidates. All tag paths for the same identifier should be equal and form a relative tag path from the page root element to the id element. This relative tag path usually addresses the whole post. Unfortunately, there might be other identifiers such as the identifier of the thread on the page. However, as explained above, the pages from the analysis set are large post-of-thread pages. Since such a large page contains the maximum amount of posts and most of the post identifiers, we can assume that the post identifier will be the most frequent identifier on such a page. Therefore, from the candidate tag paths we choose the one addressing the most segments. An initial evaluation has shown that this approach already achieves 99% accuracy on classical forums.

Unfortunately the naïve approach is not applicable to social media platforms, especially question/answer sites such as StackOverflow⁷. The problem here is that pages from those sites do not follow the classical structure of a forum discussion, where one post follows the next. On question/answer sites first level posts are usually only questions and answers, while discussions are carried out via comments attached to the discussed post. It is very common that there are more comments than posts on such pages and each of those comments has its own identifier. Therefore, the comment identifier addresses a much larger section of the page and the approach discussed above fails. In addition, numbers are not the only type of identifiers. Other common identifier formats are hexadecimal strings or Universally Unique Identifiers (UUID⁸). Hence, the approach mentioned above at first is

⁷<http://stackoverflow.com>

⁸<http://tools.ietf.org/html/rfc4122>

Table 3.16: Identifier formats in the order they are searched for in a page.

Processing Step	Format
1	whole numbers
2	hexadecimals
3	UUIDs

extended to search candidates for the identifier formats presented in Table 3.16. The order is important since the alphabet of the first format is part of the alphabet of the second format and so on.

Again, for each identifier, an absolute candidate tag path is created and grouped with other tag paths for the same identifier as attribute representation α . That way all tag paths addressing the same identifier candidate are grouped as P^α . From the attribute representation we can create a relative tag path referencing all equal identifiers.

Consider the following example:

```

1 <div>
2 <div id="post4711">
3 ...
4 <div id="post4711_message">...</div>
5 </div>
6 <div id="post4712">
7 ...
8 <div id="post4712_message">...</div>
9 </div>
10 </div>

```

This results in two attribute representations α and β , each of which represents a relative tag path corresponding to two absolute tag paths.

α : /div/div[@id]:

P^α : {/div/div[@id="post4711"],
/div/div[@id="post4712"]}

β : /div/div[@id]/div[@id]:

P^β : {/div/div[@id="post4711"]/div[@id="post4711_message"],
/div/div[@id="post4712"]/div[@id="post4712_message"]}

Using a function `id()` the two sets $ID_\alpha := \{4711, 4712\}$ and $ID_\beta := \{4711_message, 4712_message\}$ are extracted as post identifier candidates. Since ID is a set duplicates are removed automatically. So if all post-of-thread pages contain an element of the form `div[@id="wrapper-1"]`, the cardinality of the resulting ID set still is only 1. If, however, every post-of-thread page has a different identifier at the same position, the ID set has a cardinality of $|\text{post-of-thread pages}|$. There are three possible relations between the cardinality of the post-of-thread pages and the cardinality of the ID sets.

- $|\text{ID}_\alpha| < |\text{post-of-threadpages}|$: The identifier either occurs only on some post-of-thread pages or the identifier is equal in all thread pages, if $|P^\alpha| = |\text{post-of-threadpages}|$.
- $|\text{ID}_\alpha| = |\text{post-of-threadpages}|$: This means that there is one identifier per post-of-thread page and it is a different identifier on each visited page.
- $|\text{ID}_\alpha| > |\text{post-of-threadpages}|$: The identifiers belong to a data structure with multiple data records per page.

Based on these observations we can build three identifier sets, namely: $\text{ID}/ <$ containing all identifier candidates with fewer instances than post-of-thread pages, $\text{ID}/ =$ with all candidates whose cardinality is equal to the number of analyzed post-of-thread pages and $\text{ID}/ >$ containing identifier sets with a cardinality higher than the number of analyzed post-of-thread pages. The set with the post identifiers is inside of $\text{ID}/ >$ since we can assume that at least one analyzed post-of-thread page has more than one post. So if $|\text{ID}/ >| = 1$ the post identifiers are found. This is the standard case for classical Web forums as described above. As already explained $|\text{ID}/ >| > 1$ for modern question answer sites, with comments and a question which is separated from the answers. For pages from such sites we apply a distance measure between tag paths. Since an ordered list of tag paths represents all segments from an HTML page, the distance can be calculated by comparing the distance of the indices of two tag paths in this list. As shown in Figure 3.13 the minimal segment distance between comments is much shorter than between posts. However, the amount of comments and posts varies on each post-of-thread page, so it is impossible to find a globally applicable measure to divide posts from other segments identified on the page. Due to this, we revert to a widely applicable heuristic, which weights the amount of identifiers with the pairwise minimal segment distance of identifiers from one P set. As a result of this heuristic the set of post identifiers ID_{Posts} is calculated using Equation 3.4 and 3.5

$$\text{weight}(P^\alpha) := |P^\alpha| \cdot \text{min-segment-distance}(P^\alpha) \quad (3.4)$$

$$\text{ID}_{\text{Posts}} := \max(\text{weight}(P^\alpha), \text{weight}(P^\beta), \dots) \quad (3.5)$$

The algorithm selects those identifiers occurring frequently albeit with maximum distance, which is a crucial feature of posts on post-of-thread pages.

For example, consider a post-of-thread page containing two posts and four comments. Post identifiers are called α while comment identifiers are called β . Table 3.17 shows the setup.

Assuming a minimal distance of 20 segments between two posts, the weighted cardinality of P^α becomes $20 \cdot 2 = 40$. In contrast, the minimum distance of two comments is only 3 segments and thus the weighted cardinality of P^β results in $3 \cdot 4 = 12$. Therefore, α 's relative tag path is chosen to select post identifiers.

By means of the post identifier's tag path the node of the post region and each individual post can be extracted. This is necessary in order to assign segments to posts. For this

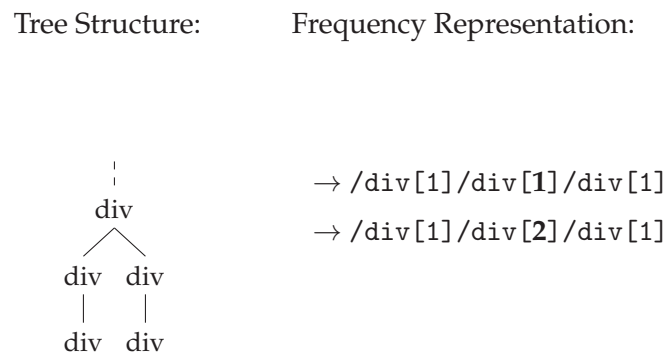


Figure 3.13: Distances of comments and posts on a typical question answer page.

Table 3.17: Example of identifier extraction on a page containing posts and comments for each post. AR = attribute representation

AR	P^{AR}	min. Distance	weight(P^{AR})
α	/div/div/div[@id="message-4711"]	20 Segments	40
	/div/div/div[@id="message-4712"]		
β	/div/div/div[@id="comment-23111"]	3 Segments	12
	/div/div/div[@id="comment-23112"]		
	/div/div/div[@id="comment-23113"]		
	/div/div/div[@id="comment-23114"]		

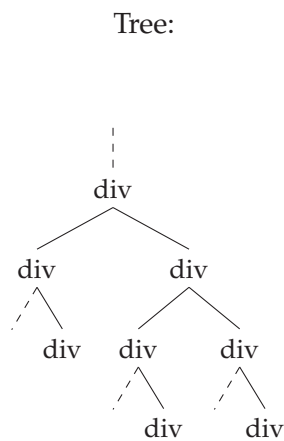
purpose we compare the frequency representations of the post identifier's tag paths. If the frequencies differ at a certain position of all paths, that position marks a split in the HTML tag tree, and thus the root nodes of the posts. That way, the previous element in the tag path constitutes the node marking the post region. See for example the following tree structure and the corresponding tag path frequency representation:



This leads to a trisection of tag paths into prefix and suffix parts surrounding the post node like:

$$\underbrace{/html/ \dots /Post-Node[i]}_{\text{Prefix}} \quad \underbrace{/ \dots /Leaf-Node}_{\text{Suffix}}$$

A final problem with regard to the location of the post region is the fact that some forums display the first post differently. Especially question answer pages usually show the question on a different layer of the tag tree hierarchy. For this purpose we can use the thread identifier group $ID / =$ which contains all identifiers with one instance per thread page, such as question identifiers. In order to check whether the tag paths belonging to such an identifier group point to question posts, their suffixes are compared to the suffixes of the answer posts. If their relative tag paths match, the identifiers belong to question posts. For example, consider the following HTML tree and the relative tag paths addressing the message body for the answer as well as the question:



Example Code:

```

1 <div>
2 <div class="question" id="post4711">
3 ...
4 <div id="post4712_message">...</div>
5 </div>
6 <div class="answers">
7 <div id="post4712">
8 ...
9 <div id="post4712_message">...</div>
10 </div>
11 <div id="post4713">
12 ...
13 <div id="post4712_message">...</div>
14 </div>
15 </div>
16 </div>

```

$$\alpha' := /div/div/\underbrace{div[starts-with(@id, "post") \wedge ends-with(@id, "message")]}_{\text{Suffix } \alpha'}$$

$$\alpha'' := /div/div/\underbrace{div[starts-with(@id, "post") \wedge ends-with(@id, "message")]}_{= \text{Suffix } \alpha'}$$

$$\rightarrow \alpha' \sim \alpha''$$

Even though the question is located in a higher hierarchy of the tree it is possible to merge it with the answers, since their relative tag path suffixes match. Of course this approach only succeeds when the markup of the questions is the same as the markup of the answers. This, however, is true for most of such pages. A detailed evaluation on how the heuristic works is shown in Section 3.11.

Predicate Based Relative Tag Path Refinement

If one of the relative tag paths is too relative it addresses more segments than intended. Such error cases must be ruled out by refining tag paths. For example, consider the HTML code in Listing 3.6: A relative tag path addressing both author elements would look like `/div/div` but would also wrongly address all other entities of the post. In order to solve this problem a relative tag path addressing the same child segment of both posts is required. For this purpose the expressiveness of tag paths using only tags is not enough. A possible representation of tag paths are XPaths, which provide a wealth of predicates for refinement, like:

1. Existence/Absence of attributes and/or their values.
2. Existence of a common string in all segments
3. Existence of element types, especially inline elements.


```

1 <div class="odd">
2   <div class="author">Max wrote:</div>
3   <div class="date">Posted: 05/10/2009</div>
4   <div>Registered: 12/27/2005</div>
5   <div>Location: Dresden</div>
6   <div id="message4725">
7     Lorem ipsum dolor sit amet
8     consectetur adipiscing elit.
9   </div>
10 </div>
11 <div class="even">
12   <div class="author">Moritz wrote:</div>
13   <div class="date">Posted: 05/11/2009</div>
14   <div>Registered: 07/12/2005</div>
15   <div>Location: Berlin</div>
16   <div id="message4731">
17     Aliquam nec dictum sapien.
18   </div>
19 </div>

```

Listing 3.6: Two example posts.

The most naïve way of refining the relative tag path with predicates for all attributes of all elements in each path would generate tag paths like:

```

1 /div[@class="odd"]/div[@class="author"]
2 /div[@class="odd"]/div[@class="date"]
3 /div[@class="odd"]/div
4 /div[@class="odd"]/div[@id]
5 /div[@class="even"]/div[@class="author"]
6 /div[@class="even"]/div[@class="date"]
7 /div[@class="even"]/div
8 /div[@class="even"]/div[@id]

```

Listing 3.7: Relative tag paths refined with attributes

These paths are still addressing the wrong number of segments. Note especially that, now, some of the relative tag paths address too few segments. This results from the separation of posts into an even and an odd class, which is usually done to improve the contrast between posts in a forum. In such a case, a reduction of relative tag paths is required in addition to the extension explained before. For this purpose, Section 3.10.3 presents the *Split/Merge/Refine - algorithm*, to generate tag paths like:

$$\begin{aligned} \alpha: & \text{ /div[@class]/div[@class="author" \wedge \text{ends-with}(\text{text}(), \text{"wrote:"})]} \\ \beta: & \text{ /div[@class]/div[@class="date" \wedge \text{starts-with}(\text{text}(), \text{"Posted:"})]} \\ \gamma: & \text{ /div[@class]/div[\text{starts-with}(\text{text}(), \text{"Registered:"})]} \\ \delta: & \text{ /div[@class]/div[\text{starts-with}(\text{@id}, \text{"message"})]} \end{aligned}$$

3.10.3 Split/Merge/Refine - Algorithm

The algorithm presented here is responsible for adding appropriate predicates to tag paths, thereby lending them an unambiguous character. A tag path is unambiguous if it addresses only one segment per post, which means in return that it reaches exactly as many segments per post-of-thread page as there are posts. There are two different cases in which the SMR algorithm is required to add predicates to an erroneous relative tag path: either the tag path addresses too many segments or too few. In the first case, the tag path must be split until it addresses just as many segments as there are posts. In the second case, the algorithm is supposed to find other relative tag paths addressing too few segments which can be merged until the result addresses as many segments as there are posts.

Sometimes merging will not be possible if the addressed segments span an optional entity. This applies, for example, if the user has the option but is not required to provide its residence. All posts of authors who provided their residence will have an additional entity showing this information within the page. The forum engine inserts such an optional entity as an additional segment moving all following segments by one step. In this case, indices referencing the same entity might become different for some posts.

If a tag path is not focused enough after a split or merge operation, this may lead to accidentally having the same form as a later merge or split result. This would require an expensive new split operation. Therefore, the results of a split or merge step should be refined in order to match the targeted segments as closely as possible.

The input for the SMR algorithm are relative tag paths as described above. Its output are refined relative tag paths, referencing all segments and only segments containing the same entity. As a means to simplify the remaining steps of the algorithm those tag paths are initially qualified with all attributes and their values. Only `id` attributes are left out. Since they always reference exactly one tag on an HTML page they are no good predicates to find relative tag path representations. In addition, for each tag path generated that way the amount of segments it addresses is recorded. The algorithm stops as soon as the set of relative tag paths stops changing.

Split/Refine Step

If a relative tag path addresses too many segments it must be extended with predicates. These predicates are based on appropriate template tokens. For this purpose the following XPath predicates are available:

- `starts-with(String, Sub-String)`
- `ends-with(String, Sub-String)`
- `contains(String, Sub-String)`

In addition, the content of a node is accessible using XPath functions like `text()` or `normalize-space()`.

In order to find the set of template tokens we start by tokenizing the content of each segment content. Then, for each token the number of segments it occurs in is counted. If a token occurs in as many segments as there are posts on the page it must be a template token. All tokens always occurring in combination with one another are grouped and used to refine the tag paths. A similar approach can be applied to the content of id attributes as shown by

```

...
<div>
the following example: <div id="post4711_author">Moritz</div>           → α: /div/div[@id]
<div id="post4711_message">Lorem ipsum ...</div>           → α: /div/div[@id]
</div>
...

```

$\implies_{split(\alpha)}$

```

α' : /div/div[starts-with(@id, "post") ∧ ends-with(@id, "_author")]
α'' : /div/div[starts-with(@id, "post") ∧ ends-with(@id, "_message")]

```

The example displays the markup of one post with two entities; the author's username and the message text. Both are initially addressed using the same relative tag path `/div/div[@id]`. Since every post contains an author and a message, the tag path addresses twice the number of posts per post-of-thread page and thus needs refinement. Tokenization transforms the content of the id tag to `{post. 4711, _author}` and `{post, 4711, _message}`. The second token 4711 is different for each post and thus not considered as an additional attribute. The tokens `_author` and `_message` occur as frequently as there are posts on the page. Thus, they are added as attributes to the refined tag paths. The post token is not absolutely necessary in this example but should still be added to avoid costly re-refinement steps.

Merge/Refine Step

Sometimes, for example, if a forum assigns different classes to a post to give every even post a different style to every odd one, a relative tag path addresses fewer segments than intended. In such cases we merge tag paths using the following five step approach, explained on example paths which are based on Listing 3.6:

1. Take an attribute representation from one of the tag paths addressing fewer segments than the amount of posts and remove all attribute values.

Example:

```

α' = /div[@class="odd"]/div[@class="author"]
α'* = /div[@class]/div[@class]

```

Here the "odd" and "author" values are removed.

2. Choose another tag path representation with the same reduced form, such as:

```

β' = /div[@class="even"]/div[@class="author"]
β'* = /div[@class]/div[@class]

```

3. Add the relative index of the leaf node for both merge candidates. Relative indices are counted from all siblings of a node. If the indices do not match, both tag paths reference different segments and should thus not be merged. If they match go on.

For the author entity in the example this results in the relative tag path `/div[@class]/div[@class][1]`, since the `div` tag surrounding the author entity is the first child of the `div` tag surrounding the post segment.

One should take into consideration that indices could have been slipped due to optional elements occurring in only some posts. In such cases tag paths to the same segment might still have different indices. Therefore, the algorithm should choose the most frequent index for its merge decision.

4. Merge both paths by reducing all elements with different attribute values to their attribute key. This is achieved by pairwise comparison of both tag paths.
5. Refine the path if possible by means of template tokens as already described for the split step. This prevents a further merge step from creating an equal tag path, which would have to be split in another split step.

Step four and five together produce a final tag path representation for the author entity, which looks like `/div[@class]/div[@class="author"]`. In order to create this final representation, step four has removed the "even" and "odd" attribute values and step five re-adds the "author" value.

3.10.4 Template Detection - Algorithm

The SMR algorithm creates tag paths referencing exactly the segment which contains equal entities. However, in addition to entities, the segment usually contains other text data belonging to the template. Consider the template tokens framed in Listing 3.8 and the marked data tokens.

```
1 <div class="postdetails">
2 <div>Posted: 10.05.2009 </div>
3 <div>Registered: 12/27/2005 </div>
4 <div>Location: Dresden </div>
5 <div id="message4725">
6   Nullam vehicula porttitor rutrum.
7 </div>
8 </div>
```

Listing 3.8: Detail information about post with highlighted template and entity tokens

The application of the SMR algorithm provides tag paths referencing the whole segment, including template and data tokens. This section explains how to split the content of the segments addressed by SMR-created tag paths.

The main difference between data and template tokens is that data tokens change their value for each instance of a relative tag path while template tokens always stay the same. It is possible to use that fact to create an extraction pattern for a data field. For example, consider the following two data fields S_1^α and S_2^α :

S_1^α : John earl of Doe wrote on 12/10/2011 01:12 pm
 S_2^α : Moritz wrote on 12/11/2011 04:35 pm (edited)

From these the following extraction pattern could be created:

$$pattern_\alpha = [* , wrote , on , * , pm , *]$$

Creating such an extraction pattern is a four-step process described by the following paragraphs:

Tokenization of Segment Content

The first step is to run a tokenization of the content of each analyzed segment. The tokenized content forms the foundation for the remaining steps. Using a simple whitespace tokenization, the example resolves to the following tokens:

$$T_1^\alpha = [John, earl, of, Doe, wrote, on, 12/10/2011, 01:12, pm]$$

$$T_2^\alpha = [Moritz, wrote, on, 12/11/2011, 04:35, pm, (edited)]$$

Mark Template Tokens

Template tokens are identified based on the tokenized content. For this purpose, the intersection of all segments is used. Tokens appearing in the intersection have occurred in every segment and thus are considered as template tokens. The set of template tokens would be generated from the example segments:

$$\{wrote, on, pm\}$$

Exchange Data Tokens by Placeholders

The instances of data tokens are irrelevant for extraction patterns. We only require the order of template and data tokens. Therefore, data tokens are replaced by Placeholders such as *. For the example presented above, this results in the following patterns.

$$T_1^{\alpha'} = [* , * , * , wrote , on , * , * , pm]$$

$$T_2^{\alpha'} = [* , wrote , on , * , * , pm , *]$$

Merge Continuous Placeholders

It is possible that a segment contains an arbitrary number of data tokens at a certain position. Only the start and end points of those positions are important. Everything in between will be extracted anyway. In order to unify all patterns for a relative tag path, continuous placeholders are merged to one. For the resulting patterns a placeholder represents an arbitrary number of data tokens. This results in the extraction pattern as intended for our example.

3.10.5 Tag Path Classification

At this point, the system is able to create extraction patterns for repeated data entities inside posts. However, it still does not know which pattern extracts which data entity. In order to assign an entity type to an extracted value a classification approach is applied again. For that purpose we need to find distinguishing features for the entity types. These features are created based on the analysis presented in Section 3.9.1.

Even though the task is to extract only a small subset of entity types, it can be helpful to provide the classifier with some information about additional entities in order to reduce noise. If a segment is classified as the user's home location with high confidence, it can be ruled out as anything else. Hence, the classifier is trained on the following list of entity types commonly found in Web forums:

- *Creation date*: The date the post was created.
- *Author*: The name of the post's author which might occur in several extraction patterns.
- *Body*: The user-generated content of the post.
- *Title*: A post's title, which is part of the posts in many forums.
- *Posts written*: The number of prior posts an author has published.
- *Home Location*: The home town of the author of the post if it was provided via the users profile.
- *Registration Date*: The date the post's author was registered for the forum.
- *User Status*: The author's status, such as "User", "Moderator" or "Administrator".
- *Signature*: An author's signature.
- *Update Date*: The date the post was last updated.
- *none*: A garbage class used to classify all of a post's remaining segments.

In order to classify extraction patterns into all these categories, the following features consider a broad set of entity types. All features are distributed over three categories. Syntactic features result from the text of a segment, context features stem from the HTML markup level, and semantic features are based on the meaning of a segment.

Syntactic Features

Syntactic features are calculated for every instance of a segment on a post-of-thread page and are averaged for the classification of an extraction pattern. The following syntactic features are used:

- *MeanLength*: The amount of characters.
- *MeanSpecialCharCount*: The amount of special characters.
- *MeanTokenCount*: The amount of tokens.
- *MeanWordCount*: The amount of tokens consisting only of letters.
- *TokenDiversity* Measures the frequency of a token in all instances of a segment. It is calculated based on the measure invented by Jaccard (1901):

$$td_{\alpha} := \frac{|D_{1j}^{\alpha} \cap \dots \cap D_{nj}^{\alpha}|}{|D_{1j}^{\alpha} \cup \dots \cup D_{nj}^{\alpha}|}$$

The equation uses the j^{th} data field of segment $1 \dots n$ from relative tag path α .

- *TokenCountDiversity*: A measure for the variation range of the number of tokens among segment instances.

$$tcd_{\alpha} := \max(|D_{1j}^{\alpha}|, \dots, |D_{nj}^{\alpha}|) - \min(|D_{1j}^{\alpha}|, \dots, |D_{nj}^{\alpha}|)$$

The equation again uses the j^{th} data field of segment $1 \dots n$ from relative tag path α

- *TemplateDataTokenRatio*: The ratio of template tokens to data tokens within segments.

Context Features

Contextual features describe features from a data fields environment. They are averaged over all segments addressed by an extraction pattern.

- *MeanRelativeSegmentNumber*: Segment's position within its post. This is expressed by the relative index of the segment counting all segments for that post.
- *MeanInlineTagCount*: Amount of inline elements within a segment.
- *MeanFormattingTagCount*: Amount of HTML formatting elements, for example, BB-Code in body segments.
- *MeanLinkTagCount*: Amount of link elements, i.e. HTML <a>, which is often found with author names which are linked to their profile.
- *TemplateTokenBeforeData*: The amount of template tokens in front of a data field. Considers only template tokens directly in front of a data field. Counting stops if another data field is found within the same segment.

Semantic Features

Semantic features provide information about the meaning of a data field or its surrounding template tokens. For this purpose they use different parsers and techniques.

- *ContainsDate*: Is true if the data field always contains a date. This is a useful feature for identifying dates like the registration date of a user or the creation date of a post.
- *TemplateContainsContext*: Template tokens are checked against a dictionary to examine whether it is a term describing a certain entity type. The terms "Location:" or "Written by:", for example, are good indicators for certain tokens. The problem is that this feature is very language-specific. Although this work is focused on English forums, it would be easy to extend those features using a forums language pack which provides translations for the most important terms into almost any language. To create a feature from template tokens each token becomes a boolean feature, so is either true or false.
- *AttributeContainsContext*: Similar to *TemplateContainsContext* but works on HTML class attribute values such as `class="author"`.

3.10.6 Thread-Identifier Extraction

Some forums distribute long threads over multiple HTML pages which results in disconnected posts. However, it is possible to reconnect them, if we are able to extract the thread identifier assigned by the forum software. This identifier should be equal on all pages belonging to the same thread and different on all other post-of-thread pages.

We used the same forum systems as for the entity analysis in Section 3.9.1 to test for occurrences of the thread identifier. The analysis results in our finding that the thread identifier usually occurs at least in one of the following places.

- *Canonical URL*: A markup used for search engine optimization, providing a unique URL for a page.
- *Open Graph URL*: A concept developed by Facebook as Social Graph to publish user relations. Open Graph is an extension by relations between users and Web sites. These meta data contains a primary URL similar to the canonical URL.
- *Feed URL*: The URL for a news feed of the current thread.
- *Form URL*: URLs in the action attribute of `<form>` elements, such as those of search buttons or send buttons.
- *Form Field*: If a `<form>`'s URL does not contain the thread identifier it is often provided as content of a hidden HTML field.

Table 3.18 shows where the thread identifier occurs with regard to the ten forums of our analysis set. If one of those occurrences exists, it is examined for one of the following terms:

{t, thread, topic, questions, discussion}

Table 3.18: Occurrences of thread identifiers

Position	vBulletin 4	vBulletin 3	IPB 3	phpBB 3	phpBB 2
Canonical URL	●	–	●	–	–
Open Graph URL	●	–	–	–	–
Feed URL	⚡	⚡	⚡	○	–
Form URL	●	●	○	–	–
Form Field	●	–	○	●	●

Position	SMF 2	SMF 1	WBB 3	Jive SBS	Jive CS
Canonical URL	○	–	–	●	–
Open Graph URL	–	–	–	–	–
Feed URL	⚡	–	●	○	●
Form URL	–	–	–	–	–
Form Field	○	○	●	○	–

Legend:

- : thread identifier is unambiguously available
- : thread identifier is available but mixed with other identifiers
- ⚡: position exists, but contains no thread identifier

Occurrences containing one of the terms plus “ID” are considered as well. They are examined further for occurrences of identifiers. The same rules as for post identifiers are applied (see Section 3.10.2). In addition, if one of the symbols ‘=’, ‘/’ or ‘-’ occurs between term and identifier it is ignored. This is very common since the identifier is either part of the URL or the URL parameters. Some extractable examples using those rules are:

```
thread/4711
threadID=4711
topic-4711
...
```

These rules are expressible using the following regular expression:

$$\underbrace{(?i)}_a \underbrace{(\^[^a-z]+)}_b \underbrace{(t | thread | topic | questions | discussion)}_c \underbrace{(id)?}_d \underbrace{ (= | / | -)}_e \underbrace{(ID-Format)}_f$$

With its individual parts having the following semantics:

- a: a Modifier, switching off case sensitivity
- b: prevents. “start=4711” from being detected as “t=4711”
- c: the terms mentioned above
- d: the optional “id” suffix for the recognition of forms like “topicID” or “threadId”
- e: the symbols connecting terms and identifier
- f: the identifier format like integer, hexadecimal or UUID

3.10.7 Summary

This concludes the description of the data extraction process introduced at the beginning of this section. The application of all of these steps to the analysis set results in a list of extraction patterns for one forum. These extraction patterns, as introduced in the beginning of this section, are triples of an *Extraction Path*, a *Token Pattern* and a *Group Index*. The *Extraction Path* points to the node or nodes in the HTML tag tree holding the extraction data. The *Token Pattern* identifies the tokens that make up the extraction data. Template tokens which are part of the pages' template and not of the data are removed. Finally, the group index points to the group of data tokens composing the target data. This is necessary since multiple data tokens might be part of the same text content, separated only by template tokens.

The application of these extraction patterns to previously unseen pages from the same forum results in candidate entities which are classified by the supervised classifier presented in this chapter. The classifier predicts which entities are publication date, author name or body of a post.

We also presented an approach to find thread identifiers to merge threads distributed over multiple HTML pages.

The next section shows an evaluation of applying such extraction patterns to different forums and a discussion on the results possibly achievable.

3.11 Evaluation of Forum Data Extraction

This section describes the extraction success achieved by means of the extraction patterns created by the approach described in the last section. With regard to the evaluation, we focus on four questions.

1. Is the extraction of post identifiers correct and complete?
2. Are the extraction patterns correct and unambiguous?
3. Are extracted entities classified correctly?
4. How well works the thread identifier extraction?

Post identifier extraction is crucial since it guarantees that all entities belonging to the same post are grouped together. It is also necessary for the SMR algorithm (see Section 3.10.3) in order to work correctly. Thread identifier extraction is important for grouping posts from long threads distributed over multiple HTML pages.

With the aim to answer these questions, four different datasets were applied to research different aspects of forum data extraction. The following list is an overview of these datasets. Detailed information on the forums used for each dataset are provided in Appendix A.

Fodex-Small: A dataset consisting of post-of-thread pages of two forums created by software from the following list: vBulletin, Jive Clearspace, phpBB3, phpBB2 Invision Power Boards. These are different classical forums. The dataset is used to examine how the algorithms work on those type of forums.

SiteLevel-FO: A dataset consisting of post-of-thread pages for the 20 forums also used by Yang et al. (2009). The dataset is used to compare the presented approach to the one created by Yang et al. (2009). As already mentioned, this dataset is very biased since it uses 15 forums created by vBulletin and only 5 created by some custom software.

Fodex-Large: A larger dataset including the pages from “SiteLevel-FO” and adding 10 forums created by phpBB2, phpBB3, and Invision Power Board. This dataset was created to restore the balance of “SiteLevel-FO” and see how this influences performance.

Fodex-QA: Contains post-of-thread pages from 10 different Q/A pages created by different software systems. Most of them are custom implementations but some are created using frameworks like QSQA, Question2Answer, or Jive Social Business Platform.

The Fodex datasets were created by Pretzsch (2012), which is one of the works this chapter is based on. The data is available upon request.

For each dataset 300 post-of-thread pages were downloaded and used as analysis set. Based on segment count (to make sure they contain a maximum amount of posts), the system always chooses the 50 largest pages. This is important since the successful generation of extraction patterns depends on repetition of posts.

3.11.1 Post Identifier Extraction

As already mentioned the correct extraction of post identifiers is crucial for the rest of the algorithms to work with a high success rate. Fortunately, the evaluation results show that from all 50 examined forums only one provides an incorrect number of posts. For the forum at <http://forums.dpreview.com> from the SiteLevel-FO dataset post identifier extraction returned 1,224 instead of the expected 612 posts. This is due to the fact that the post identifier in this forum occurs two times for two equal sibling HTML nodes. Since the selecting tag path is relative it selects both occurrences and thus identifies two posts when there is actually one. This problem could be solved by extending the selection predicate by the occurrence index to only select every odd or even-numbered position of the post identifier.

3.11.2 Extraction Patterns

We used the output of the SMR algorithm for all our datasets to check whether there are forums where no relative tag path was found for some entities. The results are shown in Table 3.19. There are two problems with these forums.

Table 3.19: Entities not addressable for certain forums using the SMR algorithm

Date	Author	Body
stackoverflow.com	stackoverflow.com	devhardware.com
	forums.d2jsp.org	bbs.cqzg.cn
	social.msdn.microsoft.com	forum.joomla.org

Table 3.20: Results of entity classification

Dataset	Date	Author	Body
Fodex-Small	9/10	9/10	10/10
SiteLevel-FO	16/20	19/20	17/20
Fodex-Large	9/10	9/10	9/10
Fodex-QA	7/10	4/10	10/10
Sum	41/50	41/50	46/50
Accuracy	82%	82%	92%

Firstly, if the content of the tag path contains no template tokens the split step of the SMR algorithm fails. One possibility to solve this problem is to split relative tag paths using path node indices. So if a tag path addresses two sibling nodes in the same post with the same relative tag path and both nodes contain no template tokens, the split step can qualify them using their indices from the HTML tree, thereby splitting both paths. However, this approach creates problems if there are optional elements in between the parent node and the node addressed by the index. In this case, the proposed approach fails because the index depends on whether the optional element exists or not.

Secondly, for the `social.msdn.microsoft.com` forum the author node is loaded dynamically via AJAX. The extractor is unable to extract such values since it does not process the page. A future implementation could do exactly that using for example the WebKit⁹ or Gecko¹⁰ engines.

3.11.3 Entity Classification

For the classification as presented here the Weka3 implementation of the RandomForest classification algorithm was used. We tried several of the other algorithms provided by Weka and chose RandomForest because of its good performance. Evaluation was carried out by means of a Leave-One-Out-Cross-validation on forum level. Since we evaluated on 50 forums this results in a 50-fold cross-validation. The results are shown in Table 3.20. One of the major problems was the correct classification of the publication date. On the one hand, the date parsers had problems parsing relative dates like “yesterday”, “5 hours ago” and so on. On the other hand, sometimes parts of the date are registered as template tokens, thus

⁹<http://www.webkit.org>

¹⁰<https://developer.mozilla.org/de/Gecko>

Table 3.21: Accuracy of thread identifier extraction

Dataset	Extracted
Fodex-Small	10/10
SiteLevel-FO	17/20
Fodex-Large	8/10
Fodex-QA	7/10
Sum	42/50
Accuracy	84%

splitting dates in half. This occurs if either the date contains a real template token like a '-' in 26 Aug 2011 - 10:10 or if the range of dates from the analyzed forum is too small. For example, if all analyzed dates are from 2011, the number 2011 is registered as a template token and the date is split. The split dates are parsed as two separate dates resulting in extraction errors. A solution to this problem would be a more intelligent tokenizer which recognizes dates as tokens themselves. The current tokenizer used white-space separation resulting in the errors mentioned above.

3.11.4 Thread Identifier Extraction

The results of the thread identifier extraction approach are shown in Table 3.21. The following URLs are examples for failing extractions:

- frageee.de/292562/pizzeria
- frageee.de/292534/playstation-3-problem
- getsatisfaction.com/mozilla/topics/poor_ram
- getsatisfaction.com/mozilla/topics/kde_integration

These URLs are optimized for search engines. So they do not carry the required identifiers along. As a solution to this problem we could run an analysis of a tokenized URL to see how often each part occurs, thus finding the part marking the thread identifier.

Figure 3.14 finally shows the location thread identifiers were found during our experiments.

3.11.5 Comparison to Existing Approaches

As a last step we compare our results to the results achieved by Yang et al. (2009). For this purpose, only results using the SiteLevel-FO dataset are included. Table 3.22 shows a direct comparison of the results achieved for the extraction of the three main entities. The approach of Yang et al. (2009) differs significantly since it evaluates link relations in addition to the instance level used by the approach presented in this chapter. For this purpose it recreates the forum's sitemap which facilitates the identification of the author if it is linked

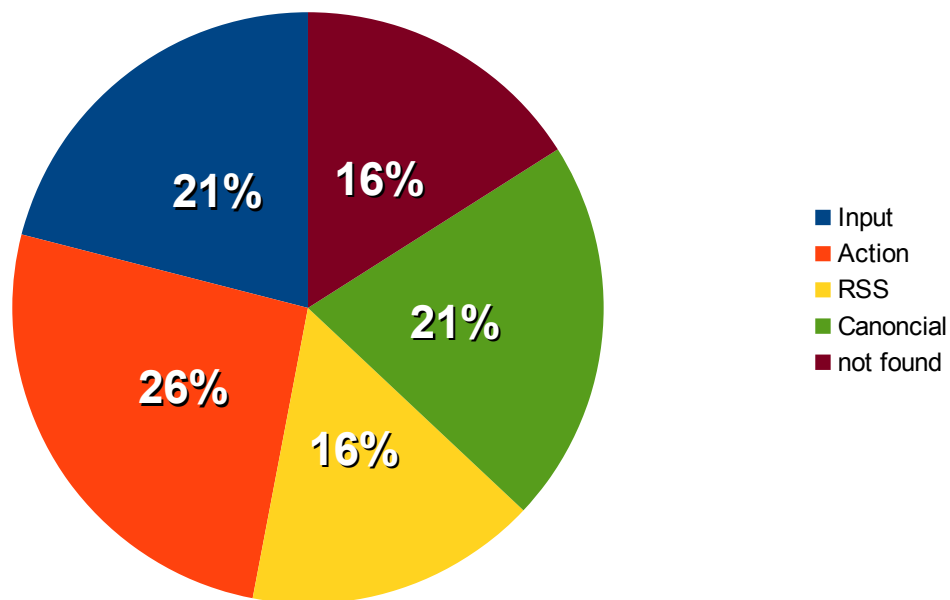


Figure 3.14: Distribution of locations the thread identifier was extracted from.

Table 3.22: Comparison of extraction results between this work and Yang et al. (2009).

	Date	Author	Body
Yang et al. (2009)	94 %	95%	88%
This work	85%	95%	90%

to its profile page. This explains the good extraction quality of 94% accuracy achieved for author entities.

Date extraction in both cases employs a date parser. All dates are ordered in order to separate the creation date of posts from the users registration date. The problem with the approach presented in this work is the same as already discussed in Section 3.11.3: If the dates do not differ significantly some parts of them are filtered out as template token.

Surprisingly, accuracy for body extraction by Yang et al. (2009) is only 88% even though they use a feature called "ContainLongText". The features in this work seem to be more appropriate for this type of entity.

However, it is necessary to repeat that even though both extractions were carried out on the same forums, Yang et al. (2009) conducted their experiments at a different time and thus their results are achieved on a different set of actual pages. Furthermore, 15 of the 20 forums are generated by the same forum software, having the same sitemap. This might skew the results in unpredictable ways.

3.12 Summary

This chapter describes a complete automatic approach for forum data extraction based on arbitrary Web pages as input. For that purpose, a focused forum crawler is applied, which classifies Web pages into forum pages and non-forum pages. The forum pages are classified further into list-of-board, list-of-thread, post-of-thread, user-profile, and miscellaneous. Finally, only the post-of-thread pages contain useful information and are processed by a forum data extractor. This extractor analyzes a small set of pages per site and creates extraction patterns applicable to remaining and new pages from the same site. It finally classifies the extracted entities into author, publication date, body and other and assigns the first three to the correct post and the correct thread. This final step is the desired output of the whole system.

Our experiments show that most steps fulfill the requirement of a low error extraction. However, they also show that there is room for improvement in future work.

Post extraction could be extended to additional optional entities. The problem is how to find out whether an entity is present or not. Additional entities include information such as a contribution's title or the points assigned to the contribution by other users. Some Web forums, for example, even provide explicit information on whether a contribution is a question or an answer. Especially Question/Answer sites like StackOverflow contain a rich amount of useful additional information, which reduces the necessary further processing steps and thus would reduce the overall error. Additional entities are already discussed in Section 3.9.1.

Another area open for further research is how to keep the data up to date. The explained components would need to run regularly to collect the most recent content as quickly as possible. This is a major problem since the amount of forums is very large and monitoring

all of them could prove to be quite expensive in terms of hardware resources. Fortunately, we can expect that there are some forums getting very few updates while others receive more in a shorter time. There are many sophisticated algorithms to monitor the update behavior of Web pages, feeds and databases. One example is presented by Urbansky, Reichert, Muthmann, Schuster & Schill (2011).

The next chapter explains how the extracted data is further processed in order to find question and answer posts.

4 Finding Question and Answer Posts

The application of the techniques from the last chapter results in a dataset of forum posts ordered by discussion threads and forums as shown in the introduction in Section 1.2. Each such post usually contains a user contribution to the forum in form of a freely formulated text. The second step of the Effingo process (see Section 1.3.1) requires a separation of these user contributions into questions, answers and other types.

Chapter 1 introduces two research questions related to Question/Answer (Q/A) detection in Section 1.5. Question 3 asks for a typology of forum contributions, including questions and answers as distinct types. Question 4 asks for an approach to automatically assign the types identified by answering question 3 to existing contributions. This chapter presents the basics of Question/Answer detection as well as a classification approach for detecting contribution types (i.e. questions, answers, etc.). The classification task is discussed in further detail as from Section 4.3 of this chapter.

The chapter also shows how contribution type detection performs and focuses especially on the quality of Question/Answer detection.

4.1 Question/Answer Detection

The research area of Question/Answer detection comprises the identification of questions and answers from unstructured text as well as the linking of each answer to the correct question. According to Allam & Haggag (2012), it combines research from Information Retrieval (IR), Information Extraction (IE) and Natural Language Processing (NLP). In contrast to current IR systems, Question/Answer detection not only retrieves documents to match them to a user query but extracts the answer from the document text and thus gives a reformulated answer to the user's query.

As stated by Allam & Haggag (2012) Question/Answer detection approaches are separated into open domain and closed domain approaches. Closed domain Question/Answer detection systems work on a closed domain of possible topics for questions and their answers (such as software development, running, science fiction movies, rock music, etc.). Since the domain is limited, they can make heavy use of natural language processing as well as domain-specific ontologies. Open domain Question/Answer detection systems answer questions about any topic. Since they may only use general knowledge about questions, answers and their relations, they are more limited with regard to available resources. A

Web-spanning system such as Effingo clearly is an open domain Question/Answer detection system.

The following subsections start with a discussion of existing research for Question/Answer detection in general. They also explain the speech act theory, which is a generalization of Question/Answer detection applied to e-mail conversations and Web forums. After that, the last parts of this section give an overview of Question/Answer detection with focus on social media content and present some existing systems.

This work’s contribution to the research in that area is presented in the following sections.

4.1.1 General Question/Answer Detection

The automatic detection of questions and answers is split into several subtasks. These tasks are steps in a common process that emerged from the numerous research activities in the area (for instance see Buscaldi, Rosso, Gómez-Soriano & Sanchis (2010) or Jijkoun & De Rijke (2004)) and was presented for example by Allam & Haggag (2012) and Ko, Si & Nyberg (2007). Figure 4.1 shows a merged version of both processes. Starting from a set of ques-

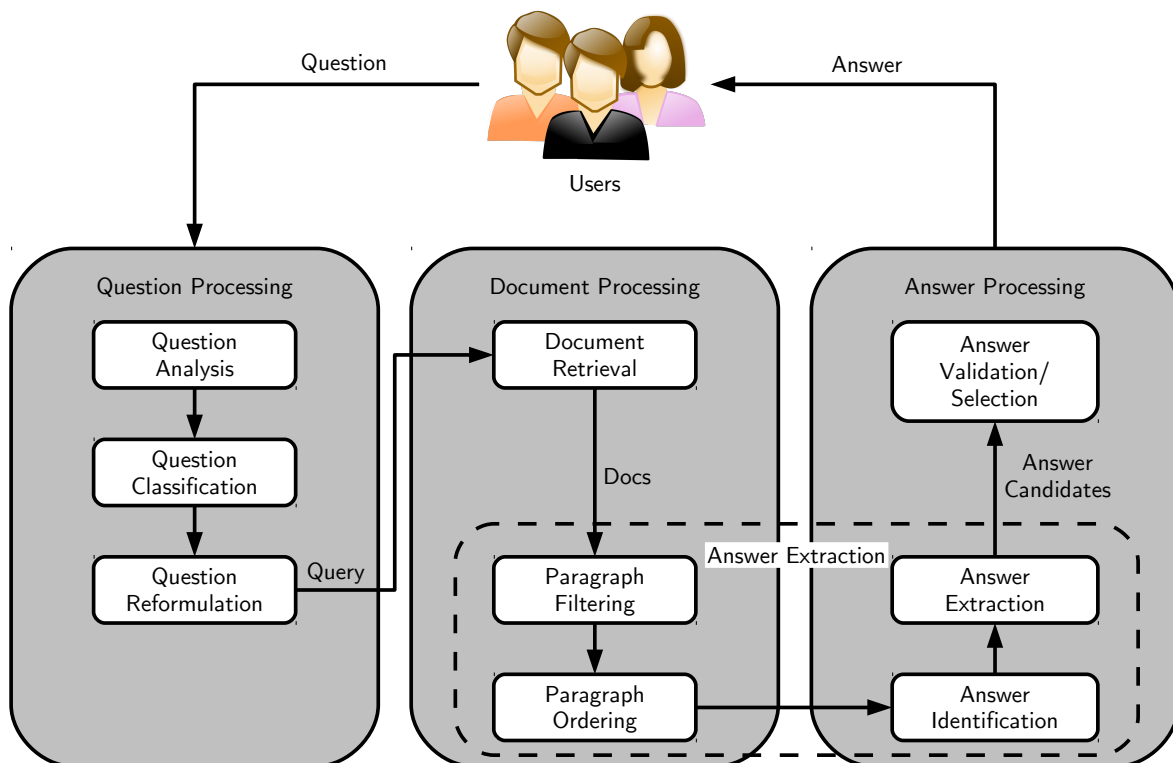


Figure 4.1: Question/Answer detection process based on the processes presented by Ko et al. (2007) and Allam & Haggag (2012).

tions, at first more precise subtypes are assigned. This is called *Question Analysis*. There is no common set of question types agreed on in the literature. Most QA systems work on factoid questions such as “Which city in China has the largest number of foreign financial

companies?”. Such questions are answered by one simple fact, such as “Shanghai” in this case. One reason for most QA systems to use factoid questions is the easy availability of the TREC QA dataset containing questions and answers for many factoid questions. It was created as a part of the TREC Question Answer Track to test systems’ ability to retrieve answers for factoid questions (Banerjee & Han 2007). This Track ran from 1999 till 2002.

Other question types we encountered in the literature are even simpler decision questions such as “Is the sun shining (on Octobre 31st 2012 in Dresden, Germany)?” or relational questions such as “What is the relationship between Alan Greenspan and Robert Rubin?”. Relational and boolean questions can be seen as a special case of factoid questions. According to Allam & Haggag (2012), there also are list questions such as “Which car manufacturers were founded in Germany?”, why-type questions like “Why was World War I begun?” and definitional question such as “Who is the President of Egypt?”. However, it could be argued that why-type questions and definitional questions are just factoid questions as well.

Effingo works on more complex questions such as the ones typically encountered in a Web forum. Such questions are not put into words easily by means of a single sentence. They require to explain some scenario or setting to provide a helpful answer. In this case, the answer might not be as simple as a fact, a relation, a list or just yes or no. In order to identify the type of a contribution, it is for example possible to apply machine learning techniques or pattern recognizers (Buscaldi et al. 2010).

While Ko et al. (2007) consider question analysis as a single step, Allam & Haggag (2012) split this into question analysis, question classification and question reformulation. In addition to question classification as described above, they prepend this first step with a question-analysis step which is responsible for finding the focus of a question. A question’s focus narrows down the information sought after. This enables the removal of ambiguous answers. The third substep is called question reformulation. This step is usually also designated as query expansion in Information Retrieval. The question is enriched with additional keywords like synonyms or hypernyms to expand on the set of matching answers. That way, alternative versions of the question, e.g. questions employing different wordings are considered as well.

According to Ko et al. (2007), the output of the first step is a query into a document retrieval system. Document retrieval refers to the search for documents containing possible answer candidates. This is usually achieved by Information Retrieval approaches using different feature sets to map the question as a query to a document corpus containing answers. The details of the different feature types examined in the literature are described further in the following paragraphs. While Ko et al. (2007) visualize this as an atomic step directly followed by an answer-extraction step, Allam & Haggag (2012) group it together with paragraph filtering and paragraph ordering. According to Ko et al. (2007) answer extraction refers to the creation of answer candidates from the retrieved documents text. Allam & Haggag’s (2012) paragraph filtering reduces the amount of text from each document. It selects a paragraph only if it contains keywords from the question which are closely clustered

together. Paragraph ordering finally ranks all selected paragraphs by the number of keywords matched and not matched and the distances between these keywords and finishes the document processing. Ko et al. (2007) groups paragraph filtering and paragraph ordering together with answer identification and answer extraction as a single step of extracting answers from the retrieved documents. The output of this step are answer candidates. Answer identification selects paragraphs matching the question type found during the question classification step. Answer extraction extracts the terms from the paragraph which are actually belonging to the answer.

The fourth step is equal for both processes and selects the correct answer from among the answer candidates. This can be achieved by means of a ranking algorithm, such as the one presented by Ko et al. (2007).

Both processes have an advantage over one another. Ko et al. (2007) clearly shows the input and output but provides a very abstract set of steps in comparison to Allam & Haggag (2012), who provide a fine grained process with a lack of information on the data exchanged between process steps.

Many approaches combine Ko et al.'s (2007) third and fourth step (Buscaldi et al. 2010, Jijikoun & De Rijke 2004) to a single step often only designated as answer extraction or according to Rinaldi, Hess, Mollá, Schwitter, Dowdall, Schneider & Fournier (2002) answer retrieval (AR). Researchers like Agichtein et al. (2008) or Jeon, Croft, Lee & Park (2006) provide a ranking based on answer quality. Bian, Liu, Agichtein & Zha (2008), for example, show how to establish a ranking function from training data obtained from Yahoo! Answers and the TREC Q/A corpus. By means of this ranking function they achieve precision@1 of over 76% for the factoid questions from the TREC data, which they believe are good values for factoid question answering.

The benefit to using a Web forum as the answer source is the fact that answers are already formulated by the answering user. Effingo only needs to select the correct contributions, without any reformulation.

Purely Syntax-Based Question/Answer Detection

Buscaldi et al. (2010) finds answers by forming answer candidates in the form of paragraphs. A paragraph consists of sentences matching tf-idf ranked keywords with the question. The number of sentences forming a paragraph is a parameter of the system. The authors provide evidence that three is a good value. In order to extract the answer from the candidate paragraphs questions are analyzed and classified into different types and subtypes. That way it is possible to assign patterns in the form of regular expressions to each question to find the questions' target and context. The target and context are summarized in the form of one target constraint and zero or more context constraints per question. The question "How many inhabitants were there in Sweden in 1994?", for example, contains the target constraint "inhabitants" and two contextual constraints "Sweden" and "1994". The answer

to this question is extracted from the candidate paragraphs by searching for a quantity adjacent or near to the target and contextual constraints. The system knows to search for a quantity, because the initial question classification step matched the question to belong to the quantity type. Overall they achieved only around 60% accuracy.

Light, Mann, Riloff & Breck (2001) relate answers to user questions based on the common feature of word overlap. Word overlap refers to the intersection of words between the question and a possible answer sentence. They experimented with different settings comparing absolute word overlap to relative word overlap. Absolute word overlap is defined by a simple count of all words common to question and answer, while the same count normalized with the similarity counts of other possible answer candidates is designated as relative word overlap. By means of using that feature, the authors show that it is possible to find a correct answer for a question in almost 80% of the cases for the TREC-8 and -9 and for a reading comprehension dataset (Canadian Broadcasting Corporation-CBC) developed by Dalmás, Leidner, Webber, Grover & Bos (2004). They demonstrate that absolute word overlap does not help to solve the task while relative word overlap does. They also present another interesting problem: If there is a question which has no answer in the available document collection, the results are very poor. This is due to the fact that the retrieval system will always return some results. It is easy to tell which result has the highest score for the current query. It is, however, not easy to tell that no result is relevant at all. This is a crucial problem for Question/Answer Detection in Web forums as Section 4.1.3 is going to show. Light et al. (2001) also demonstrate that it is not possible to find the remaining 20% of answers by means of simple lexical features, but conclude that it is necessary to apply semantical or grammatical features. Finally, they discovered that questions with multiple correct answer candidates cause better results. This suggests that redundancy is a helpful trait for answers. The discovery of answers based on redundancy was already attempted by Jijkoun & De Rijke (2004) and Ko et al. (2007). Jijkoun & De Rijke (2004) show, based on the Q/A sets from TREC 2002 and 2003, that extraction quality improves if aggregated results from multiple different Question/Answer retrieval systems are applied. Ko et al. (2007) confirm these findings using the same dataset adding questions from TREC 1999 to 2001. With the aim of aggregating answers Jijkoun & De Rijke (2004) used six answer streams. Each stream formulates answers from the TREC dataset to a query (i.e. question) by means of different features. The basic structure of the system is displayed in Figure 4.2. As shown, a question is sent to the system and gets answered by each of the six answer retrieval systems. Each of the systems is based on a different feature. These features are: the content from a knowledge base extracted from the answer corpus, two pattern based approaches, two n-gram based approaches and one approach using linguistic information. In order to filter out redundancy, they applied a simple syntactic approach using Levensthein distance, containment and absolute matches. Ko et al. (2007) aggregate four different answer sources to calculate an aggregated ranking of answer candidates and, finally, calculate a similarity score for answers to find redundancy. They boost redundant answers by ranking them higher and

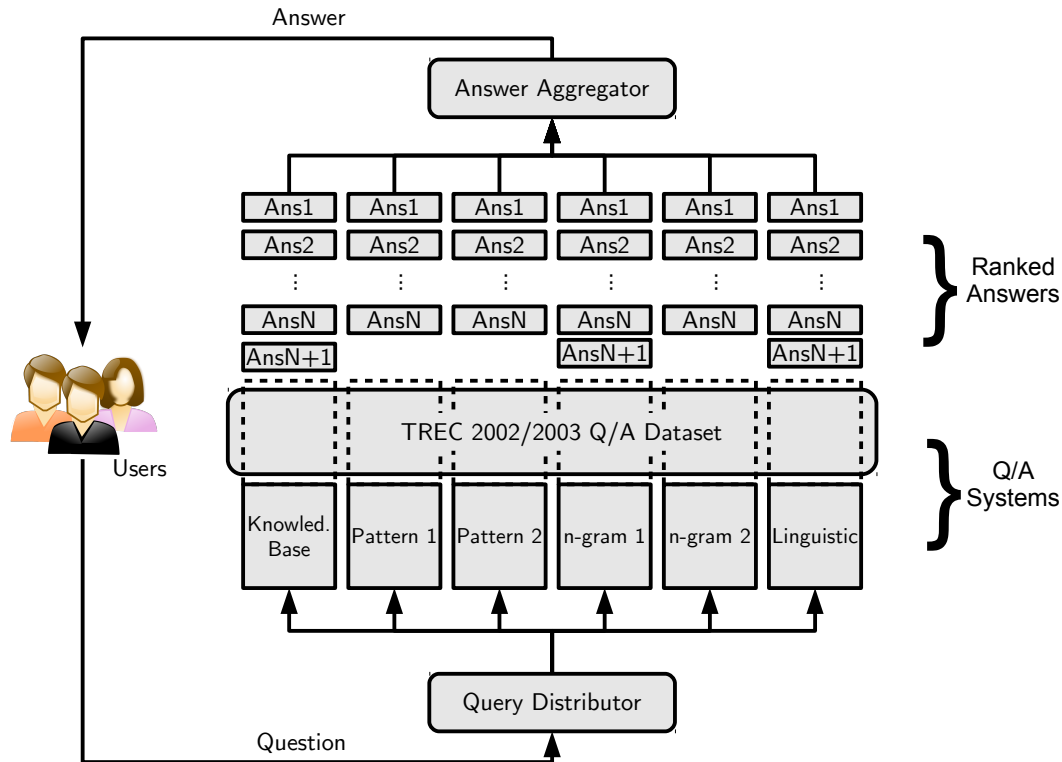


Figure 4.2: Overview of the Quartz system as proposed by Jijkoun & De Rijke (2004).

apply a combined score from all four strategies to find the final answer to a question. They show that a knowledge base like WordNet or Gazetteers for geographical information produce worse results than data-driven sources like Wikipedia and Google. This is due to the fact that knowledge bases do not cover as many questions as data driven sources do. However, both approaches are only evaluated on factoid questions from TREC, which is a kind of question not often appearing in a Web forum.

Semantic Approaches to Question/Answer Detection

Light et al. (2001) explain that semantic knowledge about relations of terms in both, question and answer would benefit the quality of Question/Answer detection. Such an approach was for example developed by Rinaldi et al. (2002) and Rinaldi, Dowdall, Schneider & Persidis (2004). Their system called *ExtrAns* incorporates semantics for the answer extraction step. They transform their queries as well as their documents into Minimal Logical Form (MLF) to match queries against phrases containing answers. MLFs are logic predicates representing the relations of the terms in a sentence. Their big advantage is that they are robust to ungrammatical sentences, providing a fallback form by means of simple keyword matching. They are also extensible. This means that it is possible to start with a few very general predicates and provide incremental extensions without the need to throw away the already working predicates. The creation of MLFs is described (in German) by Schneider, Aliod & Hess (1999). Like Light et al. (2001), they evaluated their approach on the TREC-8 and

-9 dataset and achieved quite high results, with precision and recall often ranging around 80%. They compared their system to a classic bag-of-words (see Section 2.2) system and concluded that it worked better. They especially showed that bag-of-words systems are inferior with regard to finding answers from short text segments with less than fifty words. Forums might contain such contributions, so it is important that Effingo works better than the bag-of-words approach as well. However, the evaluation of Rinaldi et al. (2002) used only 30 queries from a very limited domain with only 100 documents containing answers. A much bigger evaluation is necessary to actually prove the statements made in the paper. Earlier experiments attempting to incorporate semantic knowledge about the content of question and answer paragraphs were conducted by Narayanan & Harabagiu (2004). They use semantic frames such as the information that a theft has a thief, a victim and a stolen good, provided by knowledge bases such as FrameNet and ProbBank (Kingsbury, Palmer & Marcus 2002, Baker, Fillmore & Lowe 1998). By means of those semantic frames they infer the answer to a question from a corpus of documents. That way they are able to capture the meaning of the question and connect it to the answer.

Based on our observations with regard to the presented approaches we conclude that semantic approaches, such as the ones developed by Narayanan & Harabagiu (2004), Rinaldi et al. (2002) and Senellart & Blondel (2008) are unnecessarily complex for the domain of Web forums and show no significant improvement over simpler syntactic approaches. Semantic approaches always require some kind of ontology, knowledge base or semantic data structure describing the forum's context as well as common synonyms and antonyms, to provide terms with semantics. Such an ontology is hard to create. It requires large manual effort and it is neither language nor domain-independent. Semantic frames, as proposed by Narayanan & Harabagiu (2004), are potentially available for common concepts, such as a theft, but surely not for specific problems such as an explanation on how to install SAP NetWeaver software. Additionally, we observed text quality in Web forums to be low, containing many spelling mistakes and incorrect sentences.. Thus, we assume that it is too bad to automatically generate a suitable knowledge base from the existing forum content. Even if the knowledge bases are available it is hard to use them with "dirty" forum content.

4.1.2 Speech Act Analysis

Research on the detection of questions and answers in forums is based on classification approaches developed for a different, well known kind of asynchronous online discussion system: e-mail. E-mail discussions are very similar to forum discussions in many respects. They have a reply-response-structure and often contain informal discussions about almost all topics one can possibly think of. The technique used to classify e-mails is called speech act analysis. Early work on the theory of speech acts dates back to the middle ages but was defined more recently by Austin (1962). While Austin (1962) used very broad categories of speech acts, today more narrow categories are used for classifying natural language texts.

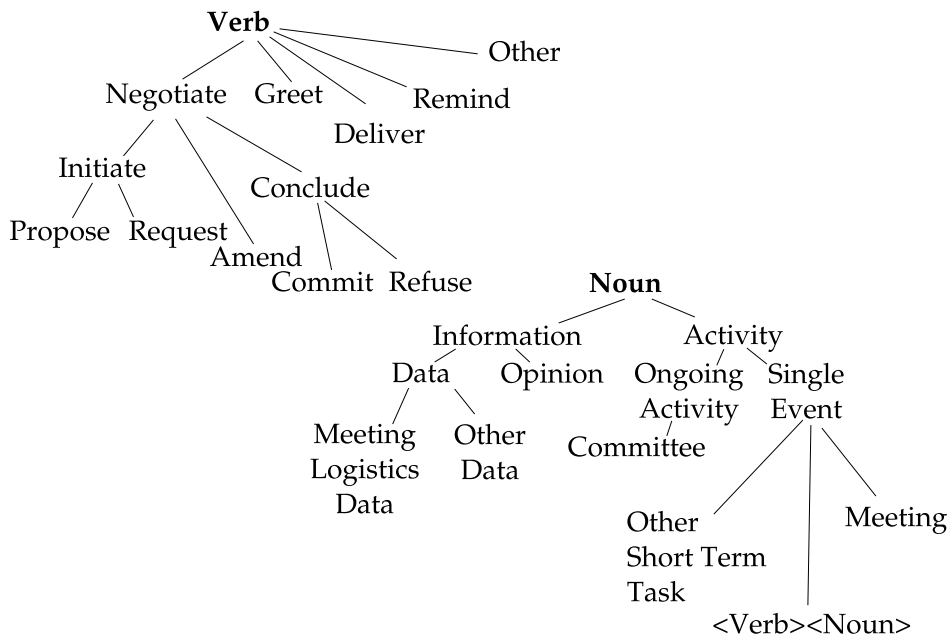


Figure 4.3: Ontology of e-mail conversation speech acts (Cohen et al. 2004).

Table 4.1: Substitutions for e-mail speech act analysis according to (Carvalho & Cohen 2006).

Symbol	Pattern
[number]	any sequence of numbers
[hour]	[number] : [number]
[wwhh]	"why, where, who, what, or when"
[day]	the strings "Monday, Tuesday, . . . , or Sunday"
[day]	the strings "Mon, Tue, Wed, . . . , or Sun"
[pm]	the strings "P.M., PM, A.M. or AM"
[me]	the pronouns "me, her, him, us or them"
[person]	the pronouns "I, we, you, he, she or they"
[aaafter]	the strings "after, before or during"
[filetype]	the strings ".doc, .pdf, .ppt, .txt, or .xls"

Carvalho & Cohen (2006) and Cohen, Carvalho & Mitchell (2004), for example, use speech acts to perform categorization of e-mail conversations according to verbs and nouns used in the e-mail's text. The authors define a small ontology of such speech acts shown in Figure 4.3. Each e-mail is defined by its verbs and nouns to belong to a composed speech act. The speech acts define the role the e-mail carries out in an e-mail conversation. Unfortunately, the ontology does not consider questions or answers explicitly.

In order to classify messages, the authors trained a linear support vector machine on the 1 to 5-grams from the e-mail's text. These n-grams are masked by means of the filters from table 4.1. Afterwards the most important features are selected using the information gain (Forman 2003) feature selection approach.

Carvalho & Cohen (2006) and Cohen et al. (2004) conducted several experiments with their approach on a set of 1716 manually labeled e-mails from the CSpace e-mail corpus (Minkov,

Wang & Cohen 2005). The labels are assigned according to Figure 4.3. They achieved a precision of around 60% together with a recall of 60% using the above mentioned features to classify into the presented ontology.

Kim, Chern, Feng, Shaw & Hovy (2006) picked up the approach of Carvalho & Cohen (2006) and Cohen et al. (2004) and applied it to a small Web forum they used to communicate with their students. The forum consists of approximately 200 threads produced by 98 students and their advisors. Most of these threads had a length of only two, but there were some up to a length of eighteen. In contrast to Carvalho & Cohen (2006) and Cohen et al. (2004), Kim et al. (2006) used a flat set of speech act classes consisting of:

Question Question to specific problems discussed in that forum.

Announcement Announcing some event or commanding some behaviour.

Answer/Suggestion A complex or simple answer or suggestion to a question.

Elaboration Elaborate on previous contribution.

Correction/Objection Correct or object to another contribution.

Acknowledge/Support Provide support for another contribution.

Most notably, they introduced questions and answers to the speech act theory. They also proposed a much wider set of categories but realized during their experiments that not even humans are able to separate them clearly.

They annotated the forum with correct speech acts from the list above using human annotators. This gold standard was used for the evaluation of all further experiments.

For speech act detection Kim et al. (2006) applied a classifier and three features. The first is a matrix of the probabilities of one speech act following another one. Since contributions are aligned in a sequence similar to the way words are aligned in a sequence within a sentence, this resembles word co-occurrence but with contribution speech acts. The matrix was created from their set of human annotated training data. In addition, they included the activity of users, meaning how many contributions a user wrote within that forum and a set of keywords per speech act similar to Table 4.1.

Ravi & Kim (2007) refined and evaluated this approach. Two human annotators manually annotated a dataset of 1834 messages from 475 threads with 133 authors. Their *kappa* agreement for questions is 0.89 and 0.72 for answers. The reason for this could be that there is a higher variance of answers than of questions. Some answers, for example, were part of questions again. The classifier was a linear support vector machine as explained in Section 2.3.3. The features are unigrams, bigrams, trigrams and quadrograms with a few preprocessing steps applied. The preprocessing steps are similar to the ones shown in Table 4.1, which were used by Carvalho & Cohen (2006). They removed the text from previous messages inserted due to the “reply to” function. They applied simple stemming, removing “s” and

“es” endings, exchanged programming code with the placeholder code, substituted “I”, “we” and “you” for `categ_person`, sequences of numbers by `categ_number_seq` and words like “which”, “where”, “when”, “who” and “how” by `categ_wh`. Similar substitutions were applied to filetype extensions, URL links, informal words and smileys. Furthermore, shortenings such as “you’re”, “I’m”, “we’ve” or “don’t” were replaced with their long forms. Finally, all technical terms were replaced by `tech_term`. Only the top 200 features per classifier (answer and question) according to Information Gain feature selection were used for classification. Tests were run using 5-fold cross validation, resulting in 88% accuracy for the question classifier and 73% for the answer classifier.

Besides automatic detection of correct speech acts, Kim et al. (2006) tried to improve question answering by finding the focus contribution within each thread. According to them, the purpose for this is to improve Question/Answer detection but neither do they define what a focus contribution is nor do they say how it supports Question/Answer detection. In order to detect the focus contribution, they calculate a connection strength from the connections between different contributions, based on the contribution author’s activity. After that, to calculate the final strength scores, they use the HITS and PageRank algorithms and achieved a precision of 70%.

The approach presented by Kim is very close to what Effingo requires for Question/Answer detection. However, the adaptations to the selected features are very specific to the used forum, the dataset is quite small and the whole approach was only tested on one forum.

4.1.3 Question/Answer Detection in User Generated Content

While the last section provided an overview of general Question/Answer detection, this one will provide insights into the state of the art with regard to Web forums and social media sites. Question/Answer detection within Web forums was first carried out by Cong et al. (2008). The task is different from classic question answering because Web forums are collections of user-generated discussions, with no clear separation of question content, answer content and glue text. In contrast, classic Question/Answer detection assumes the set of questions to be known. It is either a set of user queries or explicitly marked on FAQ pages, etc. So the task for classic Question/Answer detection is finding answers to these questions. User-generated content, on the other hand, does not make the location of questions explicit. Therefore, the process presented in Figure 4.1 must be extended by an additional “find the question” step in the beginning, as shown in Figure 4.4. This is not trivial. Every user contribution might contain a question and even though often it is the first contribution to a thread, this does not necessarily apply. A quick check on a dataset of 10,021 contributions from the SAP Developer Network forum and the Oracle Developer Network forum reveals that 7.23% of the answers are not the beginning of the thread. In addition, many threads are general discussions and do not even contain a question. Hence, Cong et al. (2008) propose an approach to first detect question sentences in a forum and second linking answer para-

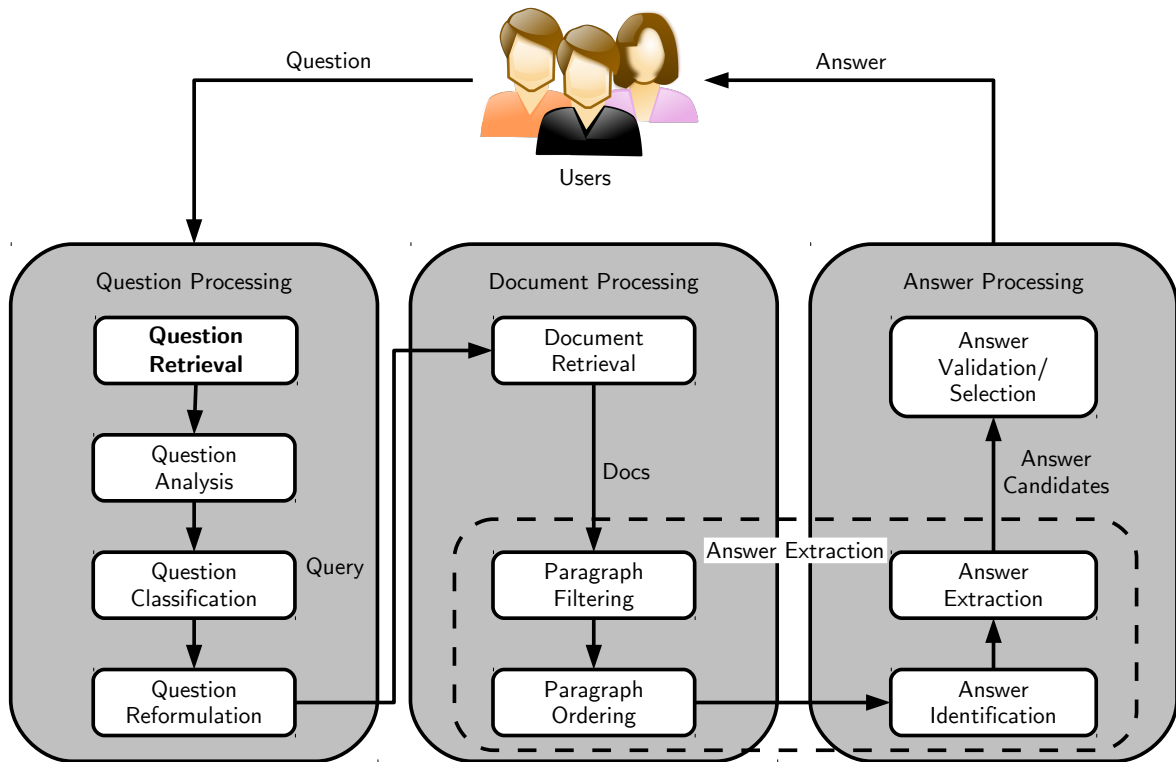


Figure 4.4: Extended Question/Answer Detection Process for Web forums based on the process presented by (Ko et al. 2007).

graphs to these questions. Their question-detection approach works on so called labeled sequential patterns (LSP). Such patterns are combinations of certain words such as 5W1H words, modal words, etc. and Part of Speech tags. LSPs are rules in the form of $LHS \rightarrow c$ where LHS is a sequence of words and PoS tags forming a pattern. Patterns may contain other patterns. So for example the pattern $\langle a, d, f \rangle$ contains $\langle a, f \rangle$. Each pattern formulates a rule that either denotes a question or not. Whether the pattern is a rule for a question or not is expressed by the value of the parameter c . So for the Question/Answer domain c can take on the values of Q (is question) or NQ (is no question). Each pattern B which contains A counts as instance for pattern A as well. That way, a database of LSPs is created. Since a pattern can occur multiple times (either as part of another pattern or as a stand-alone pattern), it is possible to calculate a confidence value and a support value for each pattern in context of the current database. The confidence is a statement about how probable a pattern denotes a question, based on how often a pattern has the value $c = Q$ versus how often it has $c = NQ$ in the database. The support value for a pattern measures the fraction of instances containing that pattern within the database. Using this information it is possible to calculate whether a specific sentence matching one or several patterns is a question or not. For example consider the following enumeration of LSPs:

1. $\langle a, d, e, f \rangle \rightarrow Q$

2. $\langle a, f, e, f \rangle \rightarrow Q$

3. $\langle d, a, f \rangle \rightarrow NQ$

Now we find a sentence matching pattern $\langle a, e, f \rangle$ and would like to know if it is a question (Q) or no question (NQ). It is obvious that $\langle a, e, f \rangle$ is part of LSP one and LSP two. So within the set of the three given LSPs, $\langle a, e, f \rangle$ has a support of 66.7% to be a question and a confidence of 100% since there is no rule stating $\langle a, e, f \rangle \rightarrow NQ$. Another example is the LSP $\langle a, f \rangle$ which has a support of 66.7% to be a question and a confidence of 66.7%. So the first pattern has a higher propability to be a question than the second one. Patterns are used as sparse features for a question classification algorithm. In order to remove bad patterns, two thresholds for support and confidence are applied. Cong et al. (2008) report a minimum support threshold of 0.5% and a minimum confidence threshold of 85%. Those thresholds are set to retrieve thousands of patterns, but are not optimized. Cong et al. (2008) use the Ripper classification algorithm, as outlined in Section 2.3.6, to process the LSPs and classify sentences as to whether they are a question or not.

In the next step, Cong et al. (2008) assign answers to the detected questions. The set of candidate answers is made up of all text paragraphs preceding the question within its discussion thread. Cong et al. (2008) apply a graph-based approach with answers as vertices connected by edges and labeled by means of different weighting strategies. The most successful approach is a Kullback-Leibler divergence model (see Section 2.2.3). For comparison they also apply a cosine similarity and a query likelihood model. A final score for each answer candidate is calculated by means of the popular PageRank algorithm (Page, Brin, Motwani & Winograd 1999) using the created graph. This results in a ranked list of answer paragraphs for each question.

Cong et al.'s (2008) approach achieves quite amazing results on their dataset. For question detection they report 97% recall and precision. The results for answer detection range between 60% and 88%.

However, it is important to consider that they used a relatively small set of only 650 threads with around 1,500 question sentences. In addition, their classification approach works on sentence level. However, observations on forum data show that the question sentence itself usually does not contain all the information that makes up the question. To address this issue, Ding, Cong, Lin & Zhu (2008) present an approach to associate context sentences with each question and improve answer identification based on these context sentences. On a corpus of approximately 2,300 forum items with 1,064 questions sentences they observe that only 26% of all questions do not need a context at all. In order to detect the context for the remaining ones they apply different conditional random fields (CRF) and compare those results with support vector machines and the C4.5 decision tree algorithm. CRFs are a kind of sequence learning mechanism, taking a sequence of items as input and providing an associated output sequence of the same length. Similar to a supervised classifier they are trained on a set of labeled sequences which then enables them to label new unknown sequences.

However, with results not even reaching 80% for either of precision, recall or F_1 -measure, this additional step introduces a big error into the overall process of Question/Answer detection.

Another interesting observation by Cong et al. (2008) illustrates that many errors are due to questions having no answer at all. Ignoring such questions improves answer detection results by 20% for some experiments. In order to solve this it would be necessary to detect whether a question has an answer or not. Cong et al. (2008) propose this as an open research problem. The classification approach presented in this thesis is able to master such cases gracefully by finding answers independently of the questions. The question-answer linking may be done in a future step.

A similar approach by Hong & Davison (2009) directly compares to the work from Cong et al. (2008), but has a different problem definition. They try to classify threads as either starting with a question or not. In addition, each question may only have one answer contribution. It was already mentioned at the beginning of this section that an analysis on our own dataset as presented in Section 4.2.1 shows that there are 7.23% questions which are not the first contribution in a thread. In addition, 8.54% of the answer contributions share a thread with another answer contribution. By applying the problem definition of Hong & Davison (2009) those additional questions and answers are lost.

If they find a thread starting with a question they try to find the answer post. In order to solve both tasks, they apply a support vector machine-based classification of contributions, using two sets of overlapping features: One for question detection, the other one for answer detection. The features for question detection are:

- Question Mark: This feature is a count of the number of question marks inside the contribution.
- 5W1H: This feature is a count of the number of 5W1H-words inside the contribution. It is derived from a theory about the typical set of questions used to get the core information about an event in journalism. The foundations for this theory stem from the greek rethorican Hermagoras of Temnos and are reformulated in a modern way for example by Flint (1917). The set of questions changes at least a little from source to source. Flint uses the set of "What", "Why", "When", "How", "Where" and "Who". Hong & Davison (2009) do not report which words they actually use, but since the set of Flint is the most current, it is reasonable to assume Hong & Davison (2009) use the same.
- Number of contributions in the thread: This feature concerns the number of the contributions in the thread the question candidate belongs to. Short threads tend to have clear questions while long ones are usually badly worded and drift away from the initial topic.
- Authorship: Experts usually provide answers while newbies ask questions. Authorship therefore results in two features: One is the number of threads the author of the

Table 4.2: Example n-grams used by Hong & Davison (2009) for question classification. All n-grams are stemmed and lower cased.

i do not know if	i wa wonder if anyon
what is the best way	i do not have
i am not sure	do not know what
i am look for	i can not
do not know	would like to

contribution has started and another is the number of replies they have created.

- n-gram: This is a sparse feature consisting of n-grams according to Speech Act analysis research by Carvalho & Cohen (2006) and Cohen et al. (2004). They used classic n-gram text classification but masked several special terms; e.g. “.pdf”, “.doc” was converted to “[fileending]” as already presented in Table 4.1. On these features they applied information gain feature selection to choose the most significant features. Table 4.2 shows some example n-grams Hong & Davison (2009) used for question classification.

For answer detection they run the classifier on all contributions in a thread previously classified as question and apply the following features:

- Position of the answer contribution: At which index in the thread is the contribution located.
- Authorship: The same as for question detection.
- N-gram: The same as for question detection.
- Stop words: Authors of answers usually use few stop words in favor of more expressive ones.
- Query Likelihood Model Score: Similar to Cong et al. (2008). Used as comparison. What is the probability of a question language generating the answer.

Even though the domain is quite restricted and Hong & Davison (2009) use a very limited set of features, they achieve quite good results. On their own dataset they achieve an effectiveness ameliorated by nearly 10% measured by precision and recall as compared to Cong et al. (2008). This, as a result, accounts for F_1 scores of around 80% up to 97%. They conducted a comprehensive analysis of the performance of different sets of their features. Their most interesting conclusion is the fact that the features performing best are authorship and position of a contribution. This is not surprising. A high authorship score is a sign for a very active user who knows much about the topic discussed in the forum and thus provides answers instead of posing questions. Second, the position of a contribution is important since answers usually do not occur as opening post of a thread while questions often do. There might be other interesting correlations between a contribution’s position in a thread and its type, which will be examined later.

4.1.4 Automatic Question/Answer Systems

There are some existing solutions for automatic question answering. Systems such as WEB-COOP (Benamara & Saint Dizier 2004) or TrueKnowledge (Tunstall-Pedoe 2012) require an underlying ontology to answer questions. These ontologies form huge databases filled with facts to answer questions. The problem with ontologies is always how to create the data necessary to answer a critical mass of questions. Usually an ontology engineer is necessary to solve this problem. TrueKnowledge provides an interface for users to enter new facts but is limited to information added to the system that way and focused on factual answers. WEBCOOP does not incorporate knowledge provided by the Internet community. Another instance of a knowledge based Q/A system is Wolfram Alpha (*Wolfram | Alpha: Computational Knowledge Engine* 2012). This, however, is focused on computable answers since it uses the math library mathematica as a kind of ontology. That way it is able to answer many factual questions that are computable, such as currency conversions.

Question/Answer communities like Stackoverflow¹, Yahoo!Answers² or Answers.com³ focus on the community aspect of question answering. They, however, are only able to provide the answers generated by their community. A user trying to find an answer on such a Q/A portal needs to choose among the portals available to find his answer. Since there usually are multiple portals for similar questions, it is tedious to search all of them for the correct answer. If an answer does not exist in the database of such a system, the user can post the question and wait for an answer. Analyses on our data show that this usually takes around 10 hours (modal value on 23,277 threads). However, if, for example, the expert able to answer a certain question is on another forum, the questioner is out of luck.

Systems like ChaCha⁴ try to decrease the time between posting the question and getting the answer by forwarding the question to a real person to provide a direct answer. However, there are many questions and only very few persons answering those questions. Therefore the questions are limited in size and complexity.

All of the presented systems are either focused on automatic answering of factoid questions or on manual answering of more complex questions. Automatic answering of complex questions is only possible by aggregation of the large knowledge base provided by user-generated content on the World Wide Web. This is the aim of Effingo.

4.1.5 Summary

Summing up question-answer detection based on the research questions formulated in Section 1.5, classic Q/A detection so far only works for factoid questions. Ding et al. (2008) prove with a corpus of 1064 questions taken from the TripAdvisor⁵ that only 10% of those

¹<http://stackoverflow.com/>

²<http://answers.yahoo.com/>

³<http://www.answers.com/>

⁴<http://www.chacha.com/>

⁵<http://www.tripadvisor.com/ForumHome>

questions are factoid. It is reasonable to assume that a similar percentage applies to most other forums as well. In addition, no approach considers the identification of questions but only works for extracting and assigning answers to already known questions. As already explained, this does not apply to the forum domain, since there, questions are hidden among all the other contributions.

The most promising approach on finding a valid typology for forum posts was presented by Kim et al. (2006) and is based on speech acts. Those or similar types should be applicable to other forums as well. However, Kim et al. (2006) left out an evaluation of their classification approach.

Question and answer identification on the forum domain was carried out by Cong et al. (2008) and Hong & Davison (2009). They already achieved reasonable performance on some forums. However, they have a different problem definition. Cong et al. (2008) try to detect only question sentences and answer paragraphs. Typical forums usually require the whole contribution to get a meaningful question or answer. Otherwise, they suffer from a lack of context. Consider, for example, questions like “Any idea on this?” or “Anyone could help me to solve this scenario?” Both are questions extracted from real forum data and both do not make sense without the contribution they are extracted from. Cong et al. (2008) combine answer identification with answer extraction. The problem is that answers are extracted based on the text in the question, so questions like the ones above would not produce any reasonable results. In addition, the approach assumes that an answer to a question is always available. This is not true. Cong et al. (2008) describe that a manual filter for those questions with no answer improved their results by around 20% F_1 measure. They also describe the creation of an automatic filter for questions with no answers as an open research problem. We are going to show a solution to this problem in the remainder of this chapter.

Hong & Davison (2009), despite working on complete contributions, make some limiting assumptions as well. They consider questions only in the first position of a thread and they assume there is only one answer contribution per thread. That way, they would miss 8.54% of the answers and 7.23% of the questions of our manually annotated dataset. Apart from these facts, both systems are evaluated on a small dataset and might not generalize well.

4.2 Contribution Typology

One important dimension of each Web forum contribution is its author’s intent. Research question 3 in Section 1.5 asks for a typology to identify an author’s intent in the form of a label assigned to a forum contribution. This section explains the typology we found while reviewing a large set of contributions from real forums.

4.2.1 Data Used to Identify the Forum Contribution Typology

The dataset used consists of a crawl of posts from two different forums. It includes posts from the Oracle Developer Network Forum⁶ and the SAP Developer Network Forum⁷. The distribution of posts from both forums is presented in Table 4.3.

Table 4.3: Absolute distribution of posts, threads, channels and users in the dataset used for experimentation.

SAP Developer Network				Oracle Developer Network			
Posts	Threads	Channels	Users	Posts	Threads	Channels	Users
45,216	11,247	137	8,674	30,742	12,030	89	7,211

In total the dataset consists of 75,958 different posts written by 15,885 different users organized in 23,277 discussion threads which again are organized in 226 different topic channels. In order to identify the typology, two human annotators examined a randomly chosen sample of 1,088 contributions from the SAP Developer Network (SDN) and the Oracle Developer Network (ODN). Since it is not clear whether an approach for one forum generalizes to other forums of the same domain, multiple forums were chosen. So if Users on the SDN and the ODN discuss similar topics and show similar behavior, all algorithms developed and evaluated on this dataset should generalize well to both. That way, we can assume that the concepts are applicable to a multitude of other forums.

Some contributions do have more than one intent. In this case, the annotator was asked to put it into a dominant category. That way, multi-type contributions are ignored throughout this work and all types are mutually exclusive with each contribution being assigned to exactly one type. As a result of this methodology, we identified nine common contribution types that fit into three broad groups: *questioning*, *answering* and *adversarial* contributions. The types we found are similar to those found by Kim et al. (2006).

4.2.2 Contribution Types

The following enumeration describes the identified types.

Questioning contributions The following three contribution types refer to contributions that ask, request or elaborate an issue:

Question Most posts belonging to this type are the start posts of a thread. However, occasionally during the discussion on a question, a new question occurs or the initial question is refined into even more detailed questions. Such posts also belong to this type. Finally, sometimes a user kidnaps an old thread to pose a new related question. Such posts also belong to the Question type.

⁶<https://forums.oracle.com>

⁷<http://forums.sdn.sap.com>

Request Sometimes, an expert does not have enough information for answering a question. In that case they write a *Request* post, asking for additional information.

Elaboration Often, the initial question does not contain enough information for the expert users to answer it. If the user posing the question notices this problem or if experts request additional information, the author of the question usually elaborates on his initial question, effectively posing the same question again with additional information. Such posts belong to the *Elaboration* type.

Answering contributions The next contribution types refer to posts that give an answer or confirm a helpful answer.

Answer Any description or reference leading to a question's solution. Often, *Answer* contributions contain an ordered list of instructions or links to helpful resources.

Thx Is for any contribution with the sole intention to thank others. This is usually a reply to authors that provided a helpful answer.

Adversarial contributions Often, users try to manipulate the attention of other users for disseminating an information or for raising their attention to an unsolved thread. We also identified three typical contribution types for this group:

Affirmation If someone encounters a problem that is already described, they write an affirmative post stating that they have the same problem to thereby increase the priority for answering that question.

Bump Most Web forums show threads on an overview page ordered by date of the last contribution. Threads without activity are pushed down the list until no one pays attention to them any more. Some users try to increase attention to their problems by writing posts with no additional content just to bring their thread back to the top of the list. This is usually called a bump post and thus, any contribution whose sole purpose is to bring the whole thread back to the attention of the forum community belongs to this category.

Description Besides the usual-question answer cycle in Web forums, some contributions are simply used to give some information to the community. This type thus covers stories, tutorials, advices, announcements and so on.

Other contributions There are some rare cases in which a post belongs to none of the types in this enumeration. Such posts belong to the *Other* type.

Table 4.4 shows the contribution text of an example per contribution type to illustrate the definitions. All examples are taken directly from our dataset.

Table 4.4: Examples for contribution types.

Contribution type	Example
<i>Question</i>	<p>Hi Mustafa</p> <p>I have a problem which can be related to you answer. When you define the path it's my understanding that a HTML file is generated from the XML and put into the defined folder. Is that right?</p> <p>Presuming i'm right why is my news put into the folder /documents/news when something else is defined in the path. Do I need to make some other configurations in order to make it work?</p> <p>Best regards Ole Mose Nielsen</p>
<i>Request</i>	<p>Hello Kwabla,</p> <p>we would need more information ot this one. Looks like the connection to your jdbc datasource does not work properly.</p> <p>By the way, there is a new Sneak Preview Edition of Web AS Java based on Web AS 6.40 available. You may want to download and use this edition?</p> <p>Many greetings</p> <p>Ivo</p>
<i>Elaboration</i>	<p>Hi,</p> <p>I have set the personalization class.But i dont think the problem is because of that.</p> <p>The requirement is to open the personalization class in the same iview and not as a popup.</p> <p>Thanks, Vivek</p>
<i>Answer</i>	<p>Hi Leander,</p> <p>Check Chapter 6 from the SAP Exchange Infrastructure 2.0 Configuration guide. The RFC Adapter Configuration is explained in details in this chapter.</p> <p>Thanks Prasad</p>
<i>Thx</i>	Thank you very much. It works!
<i>Affirmation</i>	I have the same problem. Can sb help us / give a working example for a credit card payment ?
<i>Bump</i>	(bump)

Table 4.4: Examples for contribution types.

Contribution type	Example
<i>Description</i>	<p>SDN is hosting the webinar “SAP Exchange Infrastructure, XI-30 Overview” as part of The SAP NetWeaver Know-How Network Webinar series. On Wednesday, March 10, SAP’s Prasad Illapani will discuss this XI overview. We invite you to post your questions to the presenter prior to the Webinar taking place and carry on the discussion afterwards.</p> <p>...</p> <p>A replay of this call is available for five days following each conference. Please access this recording by dialing one of the following numbers and using the replay access code 720129.</p> <p>...</p> <p>We encourage you to join this Webinar, educate yourself and collaborate with SAP experts and peers.</p>
<i>Other</i>	<p>Hi Karin,</p> <p>sorry for my impatience :-)) I guessed the posting might have gone lost somehow at the point of software change.</p> <p>Best regards</p> <p>Stefan</p>

4.2.3 Preparing the Dataset

In order to prove or refute thesis two and thus answer research question 4, a so called gold standard or ground truth is required. This means a human annotator is required to manually provide the true labels for a significant set of forum posts. Hence, a subset of the dataset was manually labeled by human annotators. The labeled dataset for the forum post classification experiments is a random subset of the complete set of 75,958 contributions. A human annotator independently labeled 10,029 posts from random threads, using only the definitions given in Section 4.2.2. For that purpose, a tagging tool was created and deployed as a Web application as shown in Figure 4.5. The human annotator used this tagging tool to annotate forum posts. The tool iterates through the whole dataset and presents random threads to the annotator. The annotator decides for each post of the thread to which type it belongs. After having finished the tagging of one thread, the tool presents the next thread. At the top of the page the annotator was provided with the same definitions for each contribution type as presented in Section 4.2.2. The box to the right of a post shows the contribution types while the left box contains the controls as well as statistics about the current progress of labeling the dataset. In addition to saving the labels for a thread the annotator was also allowed to skip a thread and go on with labeling the next one.

[hide label definitions](#)

Label Definitions

AFFIRMATION If someone encounters a problem that is already described, he writes an affirmative post, stating that he has the same problem, to increase the priority for answering that question.

ANSWER Any description or reference leading to a solution to a question.

BUMP Most web forums show threads on an overview page ordered by date of the last contribution. Threads without activity are pushed down the list until no one pays attention to them any more. Some users try to increase attention to their problems by writing po

DESCRIPTION Besides the usual question answer cycle in web forums some contributions are simply used to give some information to the community. This type thus covers stories, tutorials, advices, announcements and so on.

ELABORATION Often the initial question did not contain enough information for the expert users to answer it. If the user posing the question notices this problem or if experts request additional directly, the author of the question usually elaborates on his initial qu

OTHER There are some rare cases in which a post belongs to non of the types in this enumeration. Such posts belong to the Other type.

QUESTION Most posts belonging to this type are the start posts of a thread. However it happens that during the discussion on a question a new question occurs or the initial question is refined into smaller detail questions. Such posts also belong to this type. Fin

REQUEST Sometimes an expert does not have enough information for answering a question. In that case he writes a Request post, asking for additional information.

THX Is for any post with the sole intention to thank others, mostly the authors that provided a helpful answer.

Tagging Statistics

Count of labeled posts: 1044
 Percentage of corpus tagged: 3%
 Items in this thread: 4
 Current threads source:
 SAP Developer Network Forum

AFFIRMATION	4
ANSWER	464
BUMP	7
DESCRIPTION	4
ELABORATION	114
OTHER	21
QUESTION	358
REQUEST	34
THX	38

Tagging progress:

Attila	22
Azara	496
Beth	10021
Epsi	2490
sky	16

Bart Visser
03.03.2004

SAP Java Proxy Generator
[link](#)

Hi all,

Has someone any idea about the existence of a Java Proxy Generator for BAPI's en RFM's? I saw it already mentioned when they introduced the SAP Jco in 2001 but it was then together with the IDE of Borland (JBuilder). So I presume it was specific for this IDE.

Also it seems that the J2EE part of the WAS (have no idea about release) will have a proxy generator.

But if i download the SAP Jco from the SAP Service Marketplace it seems not included. Can I get it somewhere else ? Or is it not available anymore?

Best regards,

Annotation Name:

- AFFIRMATION
- ANSWER
- BUMP
- DESCRIPTION
- ELABORATION
- OTHER
- QUESTION
- REQUEST
- THX

Figure 4.5: Tagging tool for tagging forum posts.

Henceforth, the set of all 10,029 manually labeled contributions will be referred to as the “full dataset”.

4.2.4 Calculation of Inter-Annotator Agreement

It is important to note that not all contributions are clear and thus even humans are making errors while labeling. We cannot assume that a machine classifier produces better results, than a human annotator. Therefore, it is important to find out which labels are clear and which not. For that purpose, we use multiple labels on the same contributions provided by different human annotators. Hence, the tagging tool was designed to show contributions which were already labeled by other annotators to each new annotator before showing new contributions. Nevertheless, annotators are independent of each other, which means annotator A does not know which label was given to a contribution they are labeling by another annotator. The purpose of these independent double annotations will be explained during the remainder of this section.

Before concluding any experiments on the labeled dataset, it is important to evaluate the label quality. To this end, usually the inter-annotator agreement of labels added by different annotators to the same data object (i.e. forum contribution) is calculated. For this purpose a smaller subset of forum contributions was labeled twice by different persons. The set consists of 1,088 randomly chosen contributions from the full dataset. Henceforth, this set will be referred to as “benchmark dataset”, since it will be used for benchmarking the classifier

Table 4.5: Inter-annotator agreement for contribution types ordered by *kappa* value.

Post type	<i>kappa</i> value	first annotator	second annotator
<i>Thx</i>	0.92	40	39
<i>Question</i>	0.89	378	396
<i>Answer</i>	0.87	489	433
<i>Bump</i>	0.87	7	9
<i>Elaboration</i>	0.71	117	100
<i>Description</i>	0.5	4	9
<i>Request</i>	0.49	38	70
<i>Other</i>	0.44	22	38
<i>Affirmation</i>	0.16	8	9

prior to the final evaluation.

In order to measure inter-annotator agreement, the well known *kappa* measure (Manning et al. 2009, p. 164) is applied. It is calculated using Equation 4.1.

$$kappa = \frac{P(A) - P(E)}{1 - P(E)} \quad (4.1)$$

The probabilities $P(A)$ and $P(E)$ correspond to the probability that both annotators agreed and to the probability that both annotators agreed by chance. Please refer to Manning et al. (2009, p. 164) for details on how to calculate these probabilities. So Equation 4.1 subtracts the by-chance agreement from the actual agreement and normalizes with the ideal case of an absolute agreement minus the by-chance agreement.

The calculation and results for *kappa* are shown in the next paragraphs. According to Manning et al. (2009), a *kappa* value above 0.8 is considered as good agreement while a *kappa* between 0.67 and 0.8 is considered as a fair agreement. If *kappa* is below 0.67, the data is not really fit for evaluation since not even humans can agree on how the data should be labeled. The literature provides other scales for assessing *kappa* as well (Fleiss 1981, Landis & Koch 1977). The scale provided by Manning et al. (2009) seems to be the most conservative. So we stay with it for now as a worst case scenario.

Table 4.5 shows the *kappa* values for all contribution types. It is evident that not all of them are clearly distinguishable. The types of *Description*, *Request*, *Other* and *Affirmation* have a *kappa* value that is far smaller than 0.67, which means the annotators did not agree as to which contributions belong to these types and which do not. However, most important is that the central types of *Questions* and *Answers* show a fairly high agreement. It is interesting to see that there are some additional classes with a high *kappa* value. It seems that *Thx* contributions are easy to distinguish as well as *Bump* contributions. *Elaborations* are more difficult but it seems to be possible to label them with some agreement. This is interesting, especially since it often is not easy to distinguish between an *Elaboration* and a *Question*. The macro-averaged *kappa* value according to Table 4.5 is 0.65. This means that the overall confusion between the two annotators including all post types is a little too high. Agreement

was especially low on very small classes with few instances. It is reasonable to assume that those *kappa* values are not statistically significant though.

Only the labels for *Questions* and *Answers* are really important for contribution type classification. Therefore, it is reasonable to aggregate all other classes using the label *Other*. That way, the classifier only needs to distinguish between the three types *Question*, *Answer* and *Other*.

4.3 Feature Engineering for Question/Answer Detection

Feature engineering defines the set of features that are employed by Effingo's algorithms to distinguish between contribution types. A feature is a distinguishing piece of information that is used as evidence to assign labels to posts. That way, it is possible to define a function assigning labels based on different feature sets. Each feature should possess a similar value or range of values for at least one contribution type and different values or range of values for the *Other* contribution types. Ideally it even has a certain value for each contribution type.

In classic information retrieval, features are simply all keywords from a document. Their values are either simple booleans stating the presence or absence of the word in a document. More sophisticated values are based on the tf-idf measure as described in Section 2.2.2. That way, words that are very common over the whole corpus are voted down, while words that are common for a document but not for the whole corpus are voted up. Domain-dependent information retrieval systems can use additional domain-specific feature types. For forum contributions for example, we have additional knowledge about each document and thus the ability to choose feature types fitting exactly that domain, which might be useful for classification. Choosing the correct ones is crucial for the success or failure of the system.

As a requirement for Effingo the approach has to work for a large number of forums discussing different topics. Hence, selected features must be domain-independent with respect to the forum's topic. This task is particularly difficult since discriminative features may be sparse for some contribution types.

Effingo's feature types are organized into a hierarchy of categories based on the information unit the feature originates from. The categories are *token level*, *sentence level*, *thread level* and *forum level*. The following sections present the domain-independent feature types, ordered by these feature levels. They also show the distribution of each feature type's value on the benchmark dataset of 1,088 in Figures 4.6, 4.7 and 4.8.

4.3.1 Token Level features

Token level features are created from the text of a post's body or title. A token normally is a word, but some tokens are only numbers, control characters, links etc.

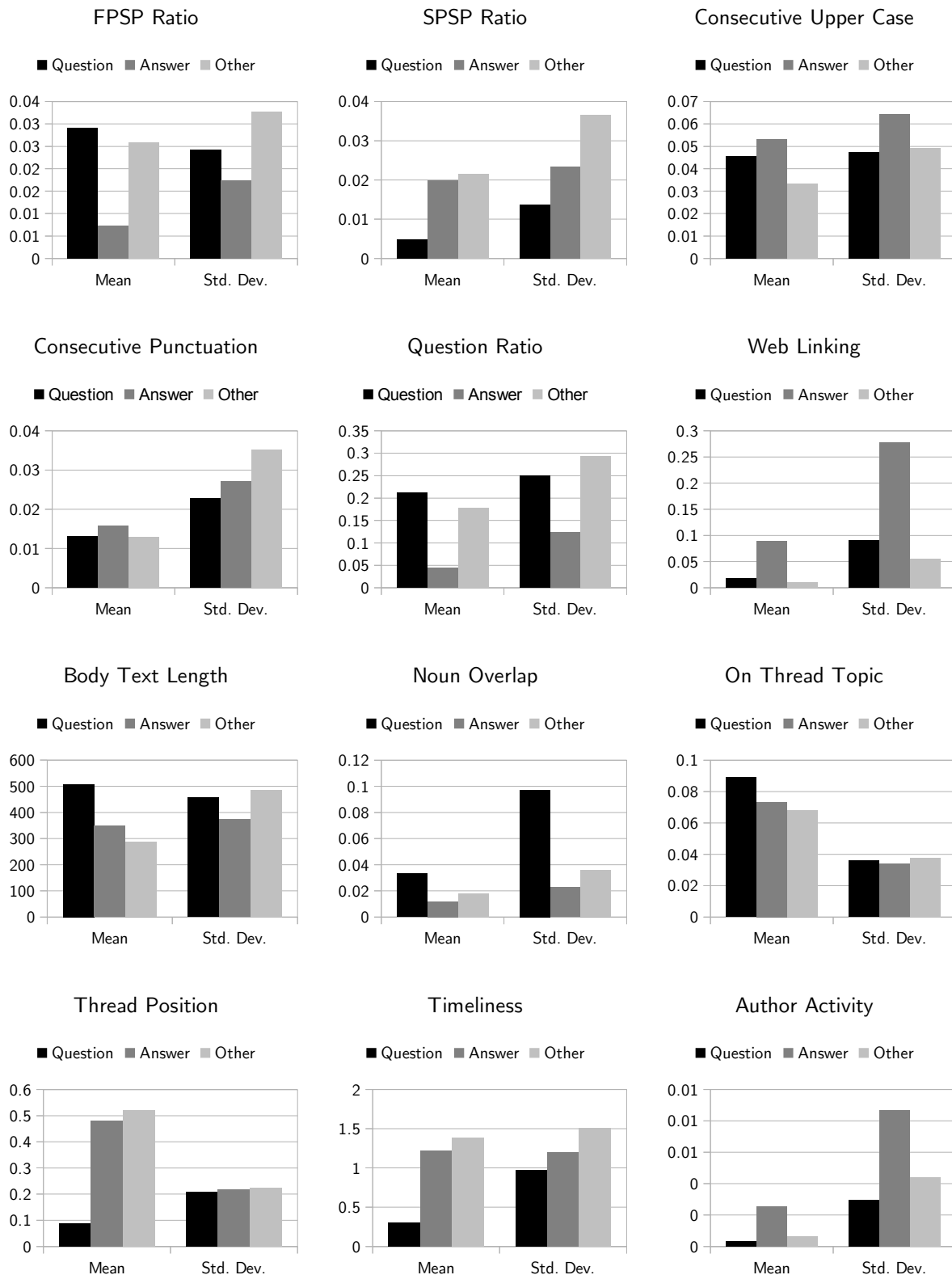


Figure 4.6: Distribution of numeric feature values per *Question*, *Answer* and *Other* contribution types. The distributions are calculated over a set of 1,088 manually labeled contributions from the SAP Developer Network and the Oracle Developer Network forums. SPSP is short form for Second-Person Singular Pronoun while FPSP is the short form for First-Person Singular Pronoun.

First-Person Singular Pronoun Ratio Calculates the fraction of first-person singular pronouns (e.g. I, my, ...) in proportion to all terms in the same contribution. Authors posing a *Question* usually talk about themselves and their problem using many first-person singular pronouns. Such pronouns are easily detected by means of a part of speech tagger. Figure 4.6 in the upper left corner shows this to be true. The mean value is much higher for *Questions* than *Answers* and even a little higher than for *Other* contributions.

Second-Person Singular Pronoun Ratio The Second-Person Singular Pronoun Ratio is similar to the ratio for first-person singular pronouns, except that it calculates the value for the second-person singular (e.g. you, yours, ...). We assume that *Answers* usually address the questioning user, employing mainly pronouns in the second-person singular. This is true for the benchmark dataset as shown in Figure 4.6 in the middle of the first row of diagrams.

Ratio of Consecutive Upper Case Letters Consecutive capitals are used for acronyms, shouting and inside of source code. Acronyms are often used for technical and domain-specific terms similar to how *Noun Overlap* will be used as a feature to identify technical terms below. Shouting, on the other hand, is mostly used in low quality contributions which rarely qualify as an *Answer*. Figure 4.6 shows the distribution of consecutive upper case letters in the upper right corner. *Answers* tend to have a higher value in that feature but they also have the highest variation. It is not clear from that figure whether the feature will be really helpful for classification.

Ratio of Consecutive Punctuation Consecutive punctuation is a feature often used for source code. Source code appears quite often in *Questions* and *Answers* but rarely in *Other* contributions.

Consecutive punctuation can be transformed to a feature by counting all consecutive occurrences of punctuation (2 or more punctuation characters) and setting this value in proportion to all terms from the same contribution. That way, a ratio between 0.0 and 1.0 is generated. The distribution on the left of the second line in Figure 4.6, however, shows that there is no clear difference concerning this feature for the different contribution types.

The feature will be kept for the evaluation. However, we assume it will not be very helpful for the classifier in the end.

4.3.2 Sentence Level Features

Sentence level features are created from the sentences of a post's body or title. Such features depend on a good sentence detector and are easily deteriorated by badly written text or non-natural language content such as source code.

Each sentence can be classified to belong to a type, such as a question or an imperative. Some sentence types are co-located with some post types. Question sentences are a good indicator for *Question* contributions. However, they are not necessarily an indicator and not every question sentence belongs to a *Question* contribution.

Sentences also have different meanings based on their position in the post text. Question posts, for example, often either start with a question followed by an explanation of the problem or they end with a question preceded by a long explanation.

Question Ratio Not all, but many contributions stating a *Question* actually contain a question sentence. Such sentences are identified by searching for sentence endings with a question mark and/or starting with one of the famous 5W1H words already introduced in Section 4.1.3. The 5W1H words are interpreted differently depending on the reference. This thesis applies “What”, “Who”, “Where”, “How”, “Why” and “When” in accordance with Hong & Davison (2009). Question sentences identified this way are set in proportion to all sentences from the same contribution to calculate a feature with a value between 0.0 and 1.0.

Most question sentences occur either directly at the start of a *Question* contribution or at its end. This is due to the fact that users either poses a *Question* and gives a lengthy explanation or they start with an explanation of their problem finishing it with a question sentence.

Figure 4.6 shows this feature in the middle of the second line. It is obvious that *Answers* contain a lot less question sentences than *Questions* and *Other* contributions.

Web Linking Links to Web pages with further information are an indicator for an *Answer*, where some user provides a reference to a page containing the actual information. Using this intuition, we can calculate a feature by at first extracting all links from a contribution and then creating the ratio of those extracted links in proportion to the number of sentences from the same contribution. That way, the feature assumes the value 1.0 if all sentences contain a link and 0.0 if there are no links. Figure 4.6 shows the distribution for the Web Linking feature on the right side of the second row. It is evident that a high value clearly distinguishes *Answers* from all other types of contributions. An extension to this feature would be to separate forum internal links from external links. While external links usually point to resources like documentations or files, internal links usually point to other threads. However, since it is unclear how this influences the contribution types it is not considered for contribution type classification.

4.3.3 Post Level Features

Features which are calculated in isolation of the surrounding thread but which are valid for a whole post are post level features.

Body Text Length The length of the body text is usually an indicator for posts with some content. Although there are also short *Questions* and even short *Answers*, long posts have a higher probability of not being a simple *Thx* statement or an *Affirmation*. Therefore, this feature concerns the length of the body text in characters. As Figure 4.6 (third row on the left) shows, the assumption clearly can be maintained for *Question* contributions, while for *Answers* the difference to other contributions is not that clear.

Lexical Patterns An important feature for distinguishing post types from each other is given by generic type-specific words. The phrase “thank you” might be typical for *Thx* contributions while *Questions* usually describe scenarios of what an author has done. Phrases of the form “I am getting ...” are typical for such posts (such as in “I am getting an exception while starting SAP Netweaver”). It is not really feasible to capture all these rules manually. As a solution, patterns similar to the ones adapted by Hong & Davison (2009) from Carvalho & Cohen (2006) can be employed as a language model, trained from labeled training data. In order to avoid over-specification, these patterns mask all words with part of speech tags. That way each contribution is transformed to its pattern representation, which in turn is transformed to lexical n-grams. Similar to the approach presented by Carvalho & Cohen (2006), we keep some words instead of substituting them with their PoS tag. These words are the 5W1H words presented for the Question Ratio feature as well as the closed word class of modal words. They are generic enough to be relevant for patterns individually, but we want to avoid that they get substituted to only one PoS tag. The language model created from those lexical n-grams is a matrix of the size $M \times N$ with M being the size of the distinct set of lexical n-grams from all training contributions and N being the number of contribution types (i.e. nine based on the types presented in Section 4.2 or three if we only consider *Question*, *Answer* and *Other*). Based on this language model, it is possible to identify the crucial lexical n-grams per type and classify new unknown contributions (i.e. a test set) according to patterns known from the training data. Each n-gram forms its own very sparse feature and not all are useful for classification. Section 4.4.3 explains how to select the most important lexical n-grams by means of a χ^2 test.

4.3.4 Thread Level Features

Features on thread level provide statistics on the whole thread and require information from the whole thread.

Noun Overlap The idea for this feature is based on the assumption that *Questions* and *Answers* contain many technical terms. Nouns are a first approximation of technical terms. Further refinements could be based on named entity recognition techniques as explained for example by Nadeau and Sekine (Nadeau & Sekine 2007). Assuming that the first contribution of a thread sets that thread’s topic, noun overlap is calculated

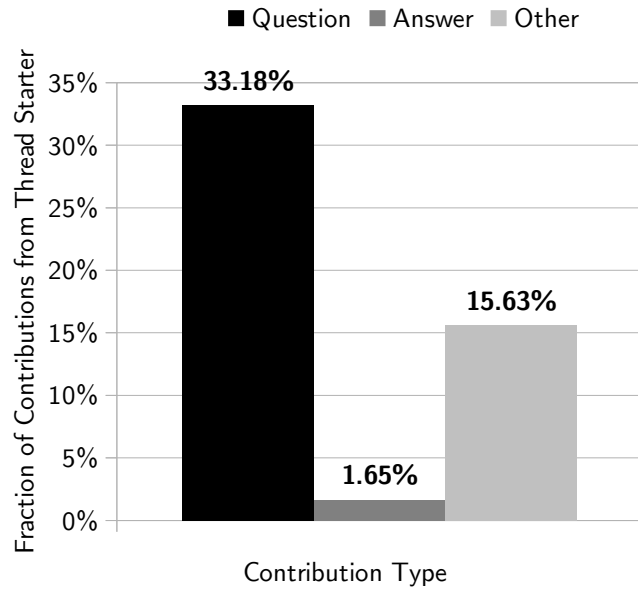


Figure 4.7: The fraction of contributions of different types created by the initial author of a thread. This distribution was calculated on the benchmark dataset.

as the Jaccard distance (Jaccard 1901) between a contribution and the start contribution from the same thread. This, of course, means that the first contribution always gets a value of 1.0, which explains the high mean for the noun overlap of *Questions* shown in the middle of the third row of Figure 4.6. The figure also shows that, surprisingly, *Answers* have a lower noun overlap than other contributions. This fact might cause the feature to fail during classification.

Post is by Thread Creator A contribution created by the author who started a thread has a high probability of being a *Question* and a lower probability of being an *Answer*. This is due to the fact that most threads start with a *Question* whose author rarely answers himself. This is a boolean feature since each contribution either is created by the same author who created the thread or not. Figure 4.7 shows the fraction of contributions of one type taking on the value “true” for the benchmark dataset. The feature is true for one third of all *Questions*. This is no surprise since a large fraction of *Questions* starts a thread. Interestingly, the feature is well suited to distinguish *Answers* from *Other* contributions. There are more than seven times as many *Other* contributions created by the thread starter than *Answers*.

Deviation from Thread Topic A post that has a high deviation from the thread’s topic has a high probability of being spam instead of a *Question* or an *Answer*. This deviation can be measured by calculating the distance between a post and all other posts in the same thread. Distance can be measured using the Jaccard coefficient (Jaccard 1901), for example. In order to reduce computation costs it also suffices to calculate the deviation from the start post instead of from all other posts. The Jaccard coefficient, for this feature, is calculated on the sets of words from a contribution and the thread’s starting

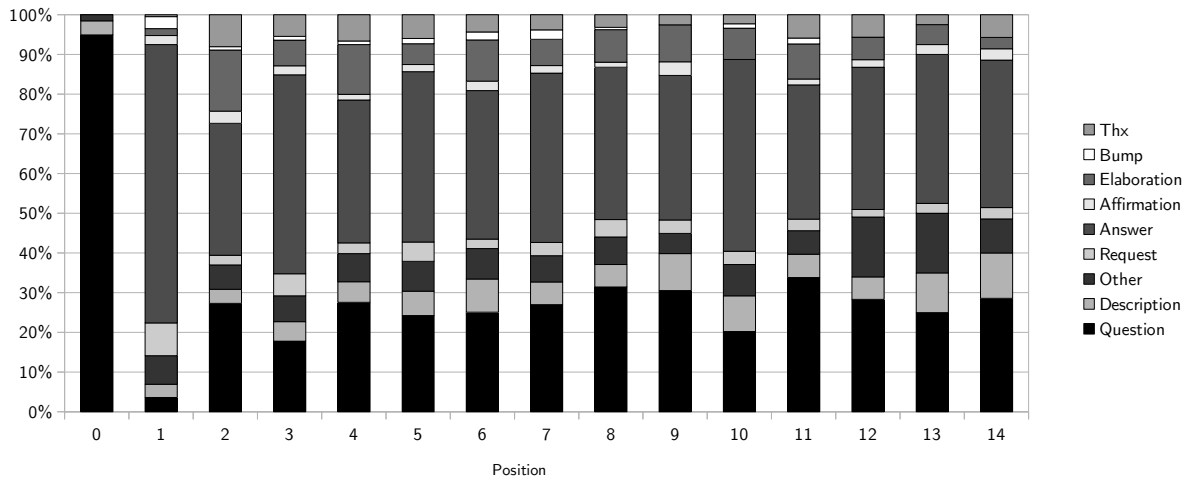


Figure 4.8: Distribution of the post types at the first 15 thread positions from a human labeled dataset of around 10,000 posts as described in Section 4.2.1.

contribution, similar to the Noun Overlap feature although including all words. Figure 4.6 shows the distribution for this feature on the right of the third row. It is evident that *Questions* and *Answers* are more on thread topic than other contributions. However, the difference is small and the variation very similar. It is hard to say whether the difference is going to suffice for a classifier. The experiments in Section 4.4 provide some information on the quality of the feature.

Post's Position in Thread The contribution's position in a thread is an important feature for detecting the contribution type. Figure 4.8 shows the distribution of post types at each position within a thread of our labeled data set. The figure includes all annotated contribution types. Obviously, posts at the starting position 0 are mostly of the type *Question*. Another observation is that posts of the type *Thx* do not appear among position 0 and 1. Moreover, *Questions* tend not to be frequent at position 1, but appear in higher positions with an equal likelihood. *Answers* are the most frequent contribution type in positions 1 and higher. *Elaborations* appear, to a lesser extent, also in higher order postings.

Figure 4.6 shows the distribution only for the types *Question*, *Answer* and the aggregated *Other* type. It is evident that *Questions* appear at very low positions while *Answer* and *Other* contributions tend to appear later in a thread.

Answers usually appear before *Other* contributions. This illustrates that long threads usually lose focus.

In order to create a feature, the contribution's post position is normalized by the thread length to a value between 0 and 1.

Timeliness Sometimes users revive old threads since they think their *Question* belongs to the one that was discussed there before. In this case, a new *Question* appears somewhere

in the middle of a thread but with a large deviation between the time passed since the last contribution and time passed between other contributions in the same thread.

At the bottom in the middle of Figure 4.6 the distribution of a contribution's timeliness is shown. Timeliness is calculated using Equation 4.2 and denotes the time it took for a contribution to be created in proportion to the mean time a contribution was created in that thread.

$$T(c_j) = \frac{\text{AbsT}(c_j)}{\sum_{i=0, i \neq j}^N \text{AbsT}(c_i)} \quad (4.2)$$

Timeliness is calculated based on absolute timeliness AbsT , which simply refers to the time in seconds since the last contribution and equals 0 for the first contribution of a thread. The feature is calculated from the absolute timeliness of a contribution in proportion to the mean timeliness of the whole thread. The distribution shows that it is very low for *Questions*, which is not very surprising since most *Questions* are the first contribution of a thread ($T = 0$). The distribution also shows a slightly lower timeliness of *Answers* in contrast to *Other* contributions. This corresponds to our expectations since a high timeliness often indicates some disturbing contribution. However, whether or not the difference is large enough will be shown in Section 4.4.

4.3.5 Forum Level Features

User statistics and overall statistics are created and managed on forum level. The relationship of a post and its thread to these statistics provides valuable information, grouped by this category of features.

Activity A user's activity can be denoted by the fraction of posts created within the forum. For the following considerations activity (A) for an author (a_i) in a forum (f_j) will be calculated using Equation 4.3.

$$A(a_i, f_j) = \frac{|F(a_i, f_j)|}{|G(f_j)|} \quad (4.3)$$

The function F refers to all contributions of author a_i in forum f_j . The function G provides all contributions from forum f_j .

To make our experiments repeatable, a static dataset is used for evaluation. Since it takes very long or is even—with the available resources—impossible to crawl a complete forum and since some authors might have created new posts during this process the true values of $|f_j|$ and F are impossible to measure. Hence, a small adaptation is applied to Equation 4.3 to calculate an author's activity. Instead of including all contributions created by an author we only consider the contributions inside the dataset. The same is true for the f_j value which denotes the size of the dataset. For our example dataset the complete set of 75,958 contributions from the SAP Developer Network and

the Oracle Developer Network Forum was used. We assume that this sample is large enough to approximate the true distribution of an author's activity at the time of our data collection.

As shown in the lower right corner of Figure 4.6, *Answers* are usually created by authors with a high activity value, while *Other* contributions and *Questions* are created by low-activity users. This is no surprise since expertise and dedication are required to *Answer* contributions. More active users tend to have both.

4.3.6 Summary

This list of feature types might not be complete and is created from observations of existing forums and partly compiled from related work adapted to the forum domain. In order to clarify which feature results from which source, Table 4.6 presents an overview of the discussed features and the sources used to acquire them. Section 4.4.4 shows results of detecting forum contribution types by means of either single features or different combinations.

4.4 Experiments

The following section explains the experiments carried out to evaluate the forum contribution type classifier. It begins with an explanation on how to select an appropriate feature set from the features discussed in the previous section. After that, the actual classification experiments on the large dataset are shown. Finally, a comparison to a classifier from the related work is presented.

4.4.1 Experimental Setup

Forum contribution type classification is evaluated by means of different classification algorithms and different feature subsets. The feature sets are based on the features presented in Section 4.3. The datasets are based on the manually labeled contributions as described in Section 4.2.3. For evaluation purposes, the two datasets, as explained in Section 4.2, were used. The benchmark dataset is used to tune the classifier and it is the fundament for finding correct values for parameters and to evaluate different feature sets as well as classification algorithms. The full dataset is only used to report final numbers and never used to tune the classification approach. That way, the final results are prevented from overfitting. Thus, we assume the results in this chapter to be applicable to larger samples at least from forums with similar user behavior and topics as the SAP Developer Network and the Oracle Developer Network.

For the evaluation of the contribution type classifier we adopt the measures (P)recision, (R)ecall and the derived (F)₁-Measure as defined by Grishman & Sundheim (1996). In addition, we report on the (A)ccuracy, which is defined for example by Manning et al. (2009, p.

Table 4.6: Overview of the features used for the classification of forum posts and their sources.

Token Level	
First-Person Singular Pronoun Ratio (FPR)	New feature
Second-Person Singular Pronoun Ratio (SPR)	New feature
Ratio of Consecutive Upper Case Letters (CU)	Wanas, El-Saban, Ashour & Ammar (2008) used this as "FormatCapitals" feature for determining a post's quality.
Ratio of Consecutive Punctuation (CP)	Wanas et al. (2008) used this as "FormatPunctuation" feature for determining a post's quality.
Sentence Level	
Question Ratio (QR)	Adapted from Hong & Davison's (2009) "Question Count" and "5W1H Word" features
Web Linking (WL)	Adapted from Wanas et al. (2008) WebLinking feature.
Post Level	
Body Text Length (BL)	Based on "Lengthiness" feature proposed by Wanas et al. (2008).
Lexical Patterns (LP)	Based on the n-gram feature presented by Hong & Davison (2009), which is based on the e-mail classification approach proposed by Carvalho & Cohen (2006).
Thread Level	
Noun Overlap (NO)	New feature
Post is by Thread Creator (TC)	New feature
Deviation from Thread Topic (TT)	Wanas et al. (2008) used this as OnThreadTopic feature for determining a post's quality.
Post's Position in Thread (Pos)	New feature
Timeliness (Time)	Wanas et al. (2008) used this feature for determining a post's quality.
Forum Level	
Activity (Act)	Based on "Authorship" feature proposed by Hong & Davison (2009).

155). For a brief summary of those measures refer to Section 2.1.1. Specifically, (P)recision, (R)ecall, and (A)ccuracy are defined for a binary classification problem (i.e., a posting belongs to a target class, for example, *Question* or not). To use them for multi class classification (i.e. *Question*, *Answer*, *Other*) we adapt them as follows: For each target class we consider as true positives (tp) all cases where both, the human annotator and the classifier, assign the postings to the target class, true negatives (tn) refer to the cases where both, the human annotator and Effingo, do not assign the postings to the target class, false positives (fp) concern the cases where Effingo assigns the post to the target class which is not considered correct by the human annotator, and false negatives (fn) refer to the cases where Effingo does not assign the post to the target class although it is correct according to the ground truth. Then, precision is defined as $P = \frac{tp}{tp+fp}$, recall as $R = \frac{tp}{tp+fn}$, and accuracy as $A = \frac{tp+tn}{tp+tn+fp+fn}$. F₁-Measure is defined as the harmonic mean of P and R.

All experiments are carried out using the classifier implementations provided by the Weka Machine Learning Library⁸ in version 3.6.8 or LibSvm⁹ in version 3.1. Although there is a LibSvm wrapper in Weka we directly used the LibSvm implementation without that wrapper. For feature extraction the pipelining functionality of the Palladian¹⁰ library was used and extended by several additional feature extraction components.

4.4.2 Classifier Selection

Section 2.3 shows several possible classification algorithms. Some of them demonstrate a better runtime behavior in certain cases, while others show higher quality results. Some are prone to underfitting, while others are prone to overfitting. The most recent implementations of the algorithms try to overcome their drawbacks as well as possible. However, it is still unclear which classifier is supposed to be employed for contribution type classification. Before tuning a classifier, a reasonable selection of state-of-the-art algorithms is run with their default settings including all features presented in Section 4.3. Since there are many classification approaches available, we tried a selection of common algorithms, using their implementation as provided by Weka with their default settings. As already explained we tested only the benchmark dataset for these initial evaluations in order to avoid overfitting. The classifier with the most promising results during this first evaluation is used for further refinements of the classification approach. The results on the benchmark dataset are shown in Figure 4.9.

As the figure indicates, all measures (i.e. precision, recall, accuracy, F₁ measure) are nearly the same for most classifiers. The bagging classifier provides the best results with around 88% on all measures, while random forest is by far the worst. This leads to the conclusion that the bagging classifier should be used for all further evaluations.

⁸<http://www.cs.waikato.ac.nz/ml/weka/>

⁹<http://www.csie.ntu.edu.tw/~relaxcjl/in/libsvm/>

¹⁰<http://palladian.ws/>

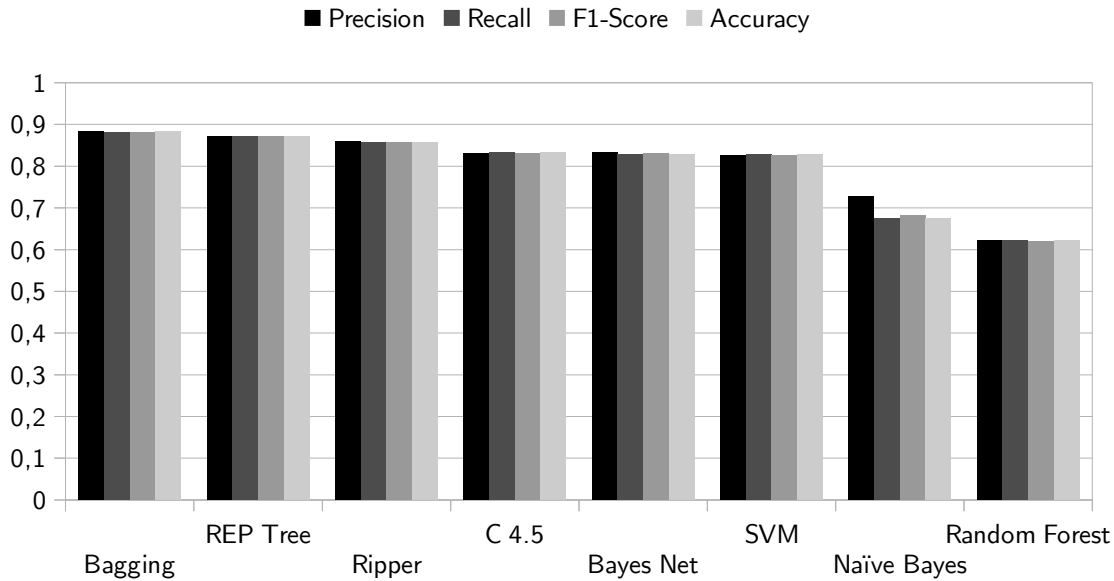


Figure 4.9: Results for different classification algorithms using 5-fold cross validation on the benchmark dataset.

4.4.3 Feature Selection

Section 4.3 describes a set of features which might be helpful to separate *Question* and *Answer* contributions from each other and from *Other* contributions. Since these features are discovered by means of an empirical study of a subset of forum contributions we do not exactly know whether and how useful each feature is. We could simply use all features and not investigate any further. This approach, however, has two drawbacks. At first, we do not know whether all these features are really helpful to identify the target classes. Some of them might even be destructive and confuse the classifier more than they are helping. Second, using redundant features or features carrying no helpful information is not harmful to the classifier's quality but increases the feature space unnecessarily and thus decreases classification speed. The features selected for classification need to be able to distinguish the classes presented in Section 4.2 but need not be redundant or reduce classification quality. That means each feature must have a statistically significant association with some class. This association may either be the presence or absence of that feature for a particular class. This section evaluates the feature set and shows how useful each feature actually is.

On its own, each feature achieves micro-averaged accuracy, recall, precision and F_1 score as presented in Table 4.7. It is interesting to see that some features achieve quite good results on their own. The timeliness feature, for example, reaches an F_1 score of more than 60%. The question now is as to whether a combination of those features improve these numbers and which features actually carry additional information.

There are two different types of features and feature selection works differently for both.

Categorical features are countable, they only have a fixed set of possible values. One ex-

Table 4.7: Performance of the features presented in Section 4.3 in isolation.

Feature	Accuracy	Recall	Precision	F ₁ score
Noun Overlap	0.815	0.727	0.702	0.667
Timeliness	0.802	0.705	0.675	0.664
Deviation from Thread Topic	0.795	0.695	0.666	0.659
Post is by Thread Creator	0.852	0.756	0.605	0.646
Post’s Position in Thread	0.808	0.719	0.667	0.638
First-Person Singular Pronouns Ratio	0.707	0.578	0.557	0.529
Second-Person Singular Pronouns Ratio	0.681	0.543	0.572	0.507
Question Ratio	0.713	0.586	0.497	0.502
Body Text Length	0.636	0.496	0.502	0.487
Lexical n-grams	0.607	0.443	0.415	0.412
Activity	0.626	0.485	0.388	0.371
Ratio of Consecutive Upper Case Letters	0.567	0.402	0.375	0.359
Ratio of Consecutive Punctuation	0.591	0.443	0.407	0.348
Web Linking	0.595	0.449	0.370	0.281

ample of such a feature are the lexical n-grams presented in Section 4.3. To select the most important n-grams we can either apply a so called χ^2 test or the Kullback-Leibler divergence (also known as Information Gain) as explained in Section 2.2.3. The χ^2 test is based on the assumption a combined occurrence of each n-gram with a specific class takes place only by chance. This hypothesis is called *null hypothesis*. The goal of the χ^2 test for feature selection is to disprove this hypothesis for important n-grams. If the null hypothesis is disproved for the correlation between an n-gram and a class, that pair is statistically significant and should be considered as a feature for forum contribution type classification.

Information Gain feature selection is, for example, well explained by Yiming & Pedersen (1997). The idea there is to measure the occurrence distribution of each feature in respect to the target class distribution. According to Yiming & Pedersen (1997), the Information Gain for a term t and a set of target classes (c_1, c_2, \dots, c_m) is calculated using Equation 4.4.

$$G(t) = - \sum_{i=1}^m P(c_i) \log P(c_i) + P(t) \sum_{i=1}^m P(c_i|t) \log P(c_i|t) + P(\bar{t}) \sum_{i=1}^m P(c_i|\bar{t}) \log P(c_i|\bar{t}) \quad (4.4)$$

The first sum calculates the distribution of the target classes using the probability $P(c_i)$ that a target class occurs at all in the dataset. The second sum refers to the probability $P(c_i|t)$ that a term occurs in combination with any class, multiplied by the probability $P(t)$ that the term occurs at all. The third sum refers to the probability $P(c_i|\bar{t})$ that term and class do not occur together, multiplied by the probability $P(\bar{t})$, that this term does not occur. All probabilities are calculated based on some training or validation dataset.

The second type of features are numeric features. All features presented in Section 4.3 except for lexical n-grams are numeric features. Since those features have a continuous range of values, it is not so easy to calculate a correlation value. It is possible to define buckets and

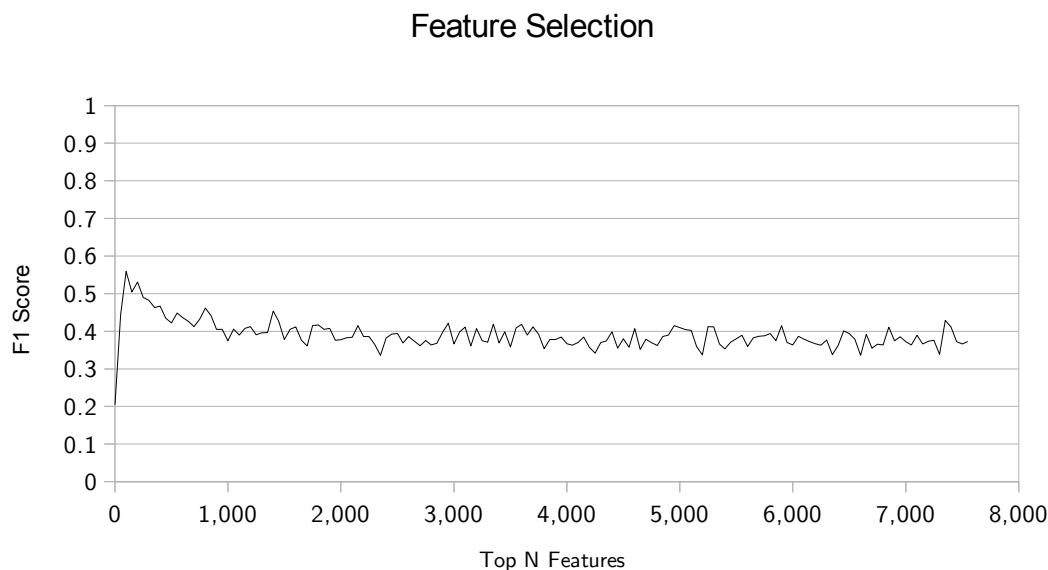


Figure 4.10: F_1 scores for different feature selection runs on lexical n-grams using increasing cutoffs.

calculate the χ^2 value for those buckets. Such an approach is for example described by Fayyad & Irani (1993). However, an easier method is brute force feature selection, where the classification performance is calculated for all feature combinations, filtering out those which do not improve or even lower performance. The drawback of brute force feature selection is the immense complexity. It increases exponentially with the amount of features. This means if there are n features, there are 2^n different combinations and all are checked using cross validation on some large dataset. However, since the numeric feature set in this case is very limited, the computational overhead can be neglected.

Lexical N-gram Selection

Table 4.8 shows the ten most important n-grams according to their χ^2 values for *Questions*, *Answers* and the aggregated *Other* contribution type. In total, there are 7,528 different n-grams. Different n-grams are important for different contribution types and thus have a different χ^2 value for each contribution type. In order to get a global ranking for each lexical n-gram, we selected the top n-grams for each class in a round-robin fashion.

In order to select a set of n-grams for classification, an ever increasing set is singled out and micro-averaged accuracy, precision, recall and F_1 score are calculated. The aim is to find a maximum F_1 score and employ that set of n-grams for all further classification tasks. Figure 4.10 shows the results of those runs. The runs start only with the top feature increasing the set in steps of 50, which means that the next run used 50, the one thereafter 100, 150 a.s.o. This was repeated until the last run included all 7,528 lexical n-grams. Figure 4.10 clearly shows that the peak in F_1 score is reached very early. Therefore, we conducted another set of cross validation runs, evaluating increments of one over the top 1,700 features. Figure 4.11 shows the results.

Table 4.8: The top 10 most significant lexical n-grams for the benchmark set of 1,088 forum contributions. The n-grams are chosen from all 2 and 3-grams extracted from those contributions.

<i>Question</i>	<i>Answer</i>	<i>Other</i>
MD P	MD P	CS QL R
P VB	P VB	DET SPEC Please
P HV	you MD	CS QL
DTI N	You MD	DTI i IN
TO VB	P HV	DTI i
MD P VB	you MD VB	CS JJ CS
P VB AT	MD P VB	N MD CC
VB TO	Thanks IN	DOD nothing
VB TO VB	DET SPEC	R DOD
VB AT N	my N	WRB JJ IN

The following abbreviations are used:

AT Article (the, an, a, ...)

CC Conjunction, Coordinating (and, or, but, plus, ...)

CS Conjunction, Subordinating (that, as after, ...)

DET Determiner (quite, all, both, many, other, ...)

DOD Form of "to do"

DT* Determiner/Pronoun (any, some, ...)

HV Form of "to have"

IN Preposition (of, into, against, over, ...)

JJ Adjective (recent, possible, hard, meager, fit, such, ...)

MD Modal Auxiliary ('ll, 'd, ...)

N Noun

TO Infinitive marker to

P Pronoun (something, everything, ...)

QL Qualifier (very, little, still, ...)

R Adverb (further, loudest, formally, ...)

SPEC Special Character (., ;, (, ...)

VB Verb

WRB WH-adverb (however, when, where, why, ...)

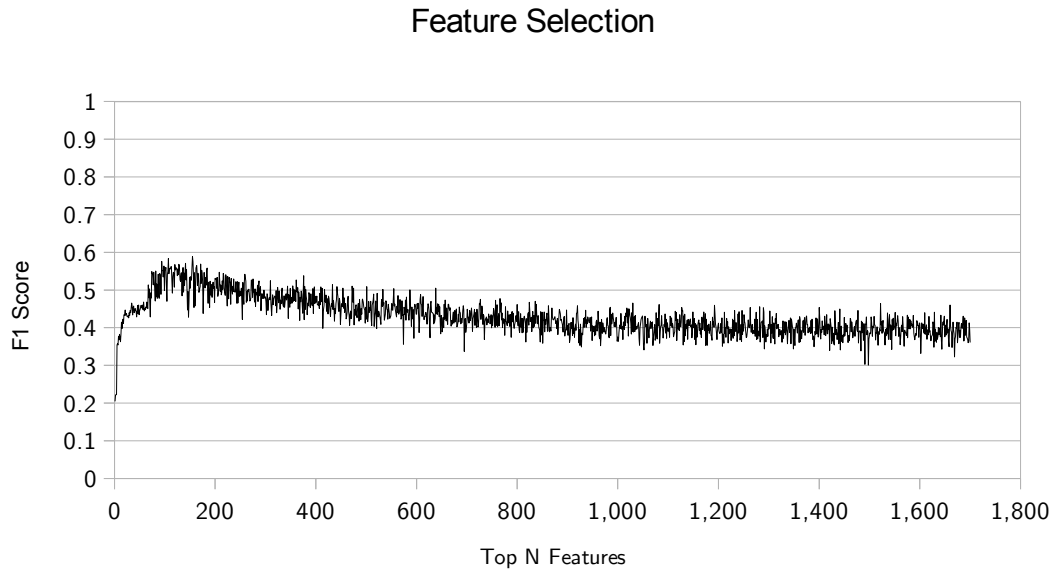


Figure 4.11: Zoomed in to the performance on the first 1,700 lexical n-grams.

The oscillating behavior of the graph is due to the random selection of the cross validation cuts on each run and only represents static noise.

The peak at the beginning is reached after only 155 features. This means that the most distinguishing 155 features according to χ^2 measure are the best for contribution type classification. Adding further features only increases confusion for the classifier. These 155 features are going to be used for the final run on the test dataset.

Brute Force Numeric Feature Selection

Brute force feature selection was carried out using the features presented in Section 4.3 with two exceptions. Lexical n-grams were selected as described in the previous paragraph and are thus excluded at this point. In addition, we excluded the imperative ratio feature, since early experiments showed that our algorithm was unable to extract imperatives from any contribution in our dataset. The remaining thirteen features were evaluated by means of five-fold cross validation on the benchmark dataset.

Since $2^{\text{amount of feature-types}} = 2^{13} = 8,192$, the complete experiment consists of 8,192 runs, each one calculating micro-averaged accuracy, precision, recall and F_1 score. Of course, the run with no features is unnecessary and thus excluded, which is why 8,191 runs remain. The micro-averaged F_1 score was used to assess the runs and, thereby, the feature combinations. Since it is rather unfeasible to display the details for all 8,187 remaining runs in a comprehensive table here, Table 4.9 only shows the five best combinations.

In order to assess a feature's usefulness for contribution type classification, a record of how often that particular feature contributes to high quality runs is required. Figure 4.12 shows the distribution of a feature to occur in a run, which resulted in a certain F_1 score range. Since each feature must occur exactly 4,096 times, each graph sums up to 4,096. For comparison,

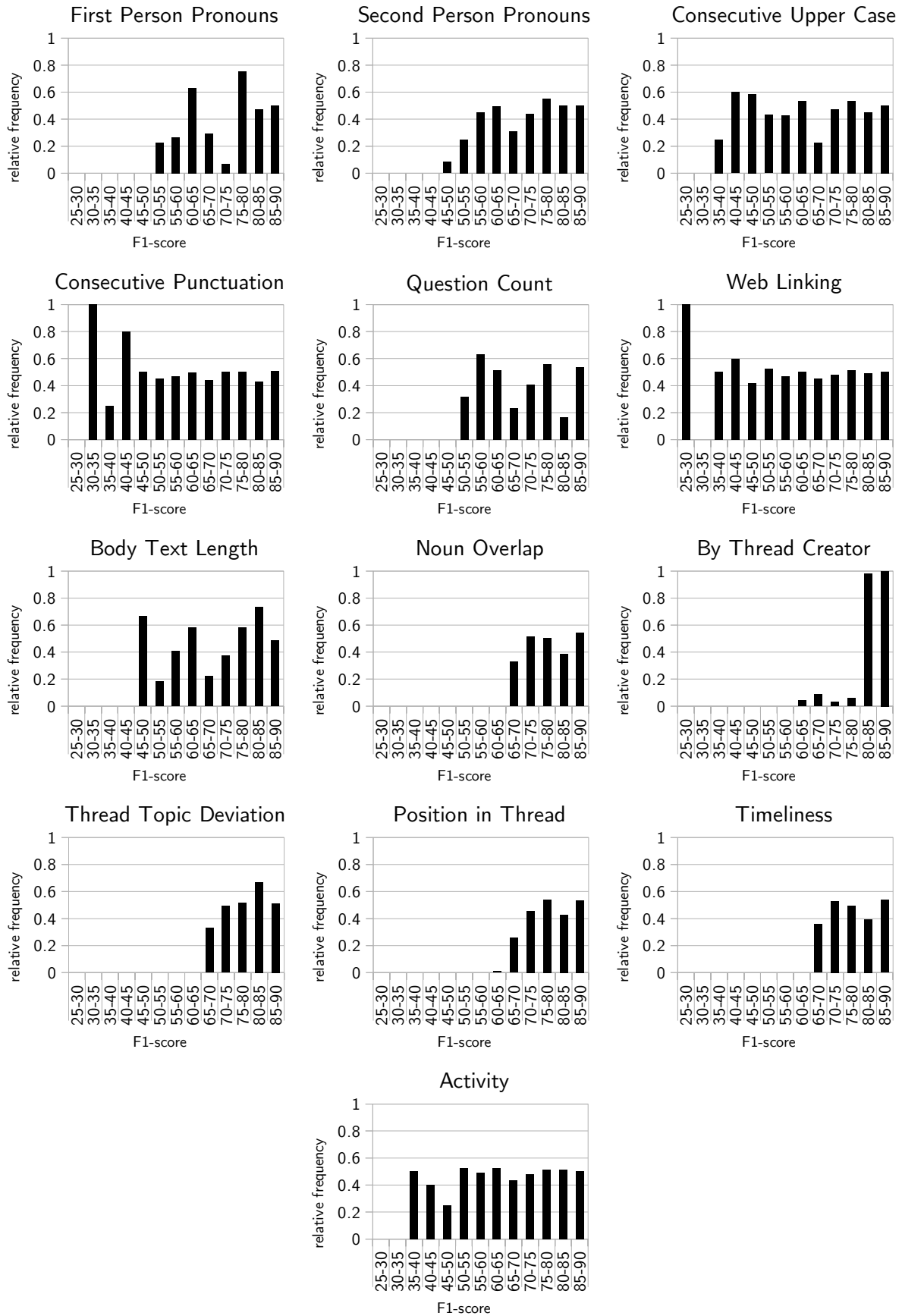


Figure 4.12: Absolute distribution of F_1 scores over brute force feature selection runs per feature. If the histogram is left-skewed the feature has a bad influence on classification quality, if it is right-skewed it has a good influence. The lowest right distribution shows the distribution of all 8,142 runs.

Table 4.9: The five best feature combinations according to brute force feature selection. The abbreviations are defined by Table 4.6.

Feature Kombination	Accuracy	Recall	Precision	F ₁ score
FPR+BL+WL+QC+CP+Act+TC+Pos	0.924	0.878	0.883	0.880
FPR+SPR+BL+WL+QC+CU+NO+Act+TC+Time	0.924	0.877	0.885	0.880
BL+WL+QC+CU+NO+Act+TC	0.925	0.878	0.883	0.880
BL+QC+CP+TC+Pos	0.925	0.879	0.883	0.881
FPR+BL+QC+CP+TC+Pos	0.925	0.879	0.885	0.881

the lowest left graph in Figure 4.12 shows the distribution of all 8,142 runs over the F₁ scores. The skewness of a distribution of a feature provides an understanding of how useful it is for classification. A feature has a high usefulness if the distribution is skewed to the right, which means that the feature occurs more often in high quality runs. A feature has a low usefulness if its distribution is skewed to the left. This means that the feature occurs more often in low quality runs. It is evident that most features have a quite similar distribution and no feature shows a left-sided concentration. Thus, all features seem to have some value for classification and are not disturbing the classifier. However, since most runs are already high quality runs—compare the total distribution on the lower right—a consideration of only viewing the absolute distributions is not enough. Figure 4.13 shows the relative distribution. Each bar refers to the relative amount of runs which the specific feature was part of in a certain range of F₁ scores. Again, a right-sided distribution is better than a left-sided one. However, it should not be overestimated that for example “Web Linking” occurs for all runs in the range of 25%–30% F₁ score since, as can be seen from the distribution of all runs on the lower right of Figure 4.12, there are very few (namely exactly one) runs in that range. Still, it is possible to estimate from a wide distribution that such a feature has much less discriminative power than one which is clearly shifted to the right. The figure also shows that “Post is by Thread Creator” is by far the most powerful feature and we most definitely should not leave it out.

In order to get a feeling for the consequences of the exclusion of a feature, Table 4.10 shows the micro-averaged quality measures for all feature combinations missing exactly one feature. The table also shows the results using all features at the bottom. Except for “Post is by Thread Creator” the exclusion of a feature seems to have no recognizable effect.

From these observations we can conclude that some features are more powerful and other less so. Since it is the only feature occurring during all runs which achieved an F₁ score of between 85% and 90%, the information on whether a contribution was created by the same author, who started the thread or not, sticks out. Nevertheless, all other features seem to at least have no destructive effect on classification and the exclusive use of “Post is by Thread Creator”, as shown in Table 4.7, results in very low precision and clearly inferior results.

For the classification we are going to test the best feature combination according to Table 4.9, but we also will provide results for the application of all features since it is not clear whether

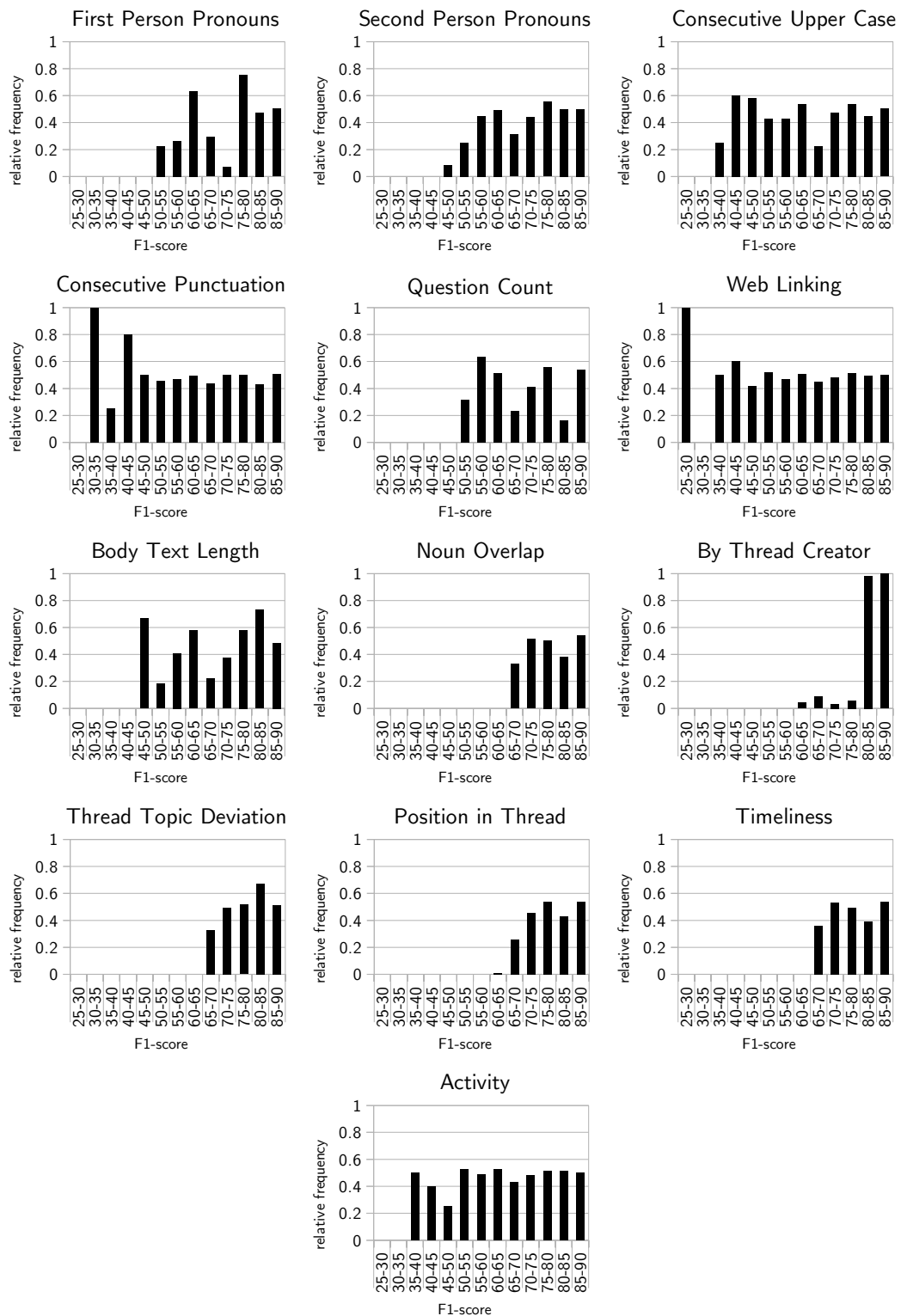


Figure 4.13: Relative distribution of F1 scores over brute force feature selection runs per feature.

Table 4.10: Classification results on the benchmark dataset leaving one feature out.

Missing Feature	Accuracy	Recall	Precision	F ₁ score
Body Text Length	0.921	0.872	0.879	0.875
Ratio of Consecutive Punctuation	0.919	0.870	0.877	0.873
Web Linking	0.919	0.868	0.874	0.870
Second-Person Singular Pronoun Ratio	0.917	0.866	0.873	0.869
Deviation from Thread Topic	0.916	0.865	0.873	0.869
Timeliness	0.917	0.865	0.869	0.867
Post's Position in Thread	0.915	0.864	0.867	0.865
Ratio of Consecutive Upper Case Letters	0.914	0.862	0.869	0.865
Activity	0.914	0.861	0.865	0.862
First-Person Singular Pronoun Ratio	0.910	0.855	0.861	0.857
Noun Overlap	0.911	0.855	0.858	0.856
Question Ratio	0.908	0.854	0.855	0.852
Post is by Thread Creator	0.862	0.787	0.790	0.787
	0.917	0.865	0.870	0.867

Table 4.11: Classification results on five fold cross validation of the full dataset using the top 155 lexical n-grams together with all thirteen dense features.

Contribution Type	Prior	Accuracy	Recall	Precision	F ₁ score
<i>Answer</i>	0.356	0.864	0.771	0.878	0.821
<i>Other</i>	0.245	0.825	0.691	0.517	0.592
<i>Question</i>	0.399	0.902	0.866	0.893	0.880
Micro Average	-	0.870	0.796	0.790	0.788

there are any disturbing features at all. However, remember that each feature adds another dimension to the feature space and thus increases computational costs. If, in practice, fast computation is more important than correctness to a certain point, the figures and numbers above can help to make an educated decision on which features are to be left out and what impact this might have on the expected results.

4.4.4 Results and Discussion

After deciding on a classifier and a final feature set the performance was evaluated employing the full dataset. For this experiment the top 155 lexical n-grams from the benchmark dataset were applied in unison with the thirteen other dense features. In order to get the final values, the classifier ran on the full dataset using five fold cross validation. The results are displayed in Table 4.11 The table also shows the prior probability of encountering each class. A comparison of that probability to the results achieved emphasizes the improvement in contrast to a less to a naïve random classifier.

As the confusion matrix of the bagging classifier in Figure 4.12 shows, the most error-prone type is the *Other* type. Individually it only reaches a F₁ score of around 70% while the other

Table 4.12: Confusion matrix for the classification of the full dataset into *Question*, *Answer*, *Other* using all features presented in Section 4.3.

	Classified as		
	<i>Other</i>	<i>Question</i>	<i>Answer</i>
<i>Other</i>	1270	451	734
<i>Question</i>	233	3575	194
<i>Answer</i>	334	101	3128

Table 4.13: Counts of classification errors by bagging classifier broken down for the classes presented in Section 4.2.

	Classification Result		
	<i>Question</i>	<i>Answer</i>	<i>Other</i>
<i>Question</i>	–	194	233
<i>Answer</i>	101	–	334
<i>Other</i>	66	218	–
<i>Elaboration</i>	127	66	–
<i>Thx</i>	4	31	–
<i>Request</i>	54	138	–
<i>Description</i>	108	237	–
<i>Affirmation</i>	63	42	–
<i>Bump</i>	29	2	–

two classes are above 80%. Thus, the discrimination for the class *Other* needs to be improved. Since we know that the *Other* class consists of several subclasses, we can analyze which of those subclasses is the most error-prone. For that purpose Table 4.13 shows the misclassifications broken down to the classes explained in Section 4.2. Of course, with regard to the overall number of questions and answers there was no change, whether they were attributed to the wrong classes or not. The table shows that most errors result due to the misclassification of *Descriptions* as *Answers*. This is not surprising since *Description* contributions share many commonalities with *Answer* contributions. Both usually contain long explanatory texts addressing another individual. The only difference is that descriptions are usually destined to the whole community while an *Answer* is for one individual. Patterns that address a general audience might be added as an additional feature similar to the current first and second-person pronoun features to avoid this error.

The second most common error results from the non-aggregated *Other* contribution type as *Answer*. Finding a feature to reduce this error is much harder on intuition.

As a means to point out individual features relate to the correctly classified vs. the incorrectly classified ones, the same plots as in Section 4.3 for the distribution of the features are presented in Figures 4.14 for all erroneous cases and 4.15 for all correct cases.

As can be seen most features show a different distribution for the correctly classified contributions versus the incorrect ones. Correct distributions have at least one value that is

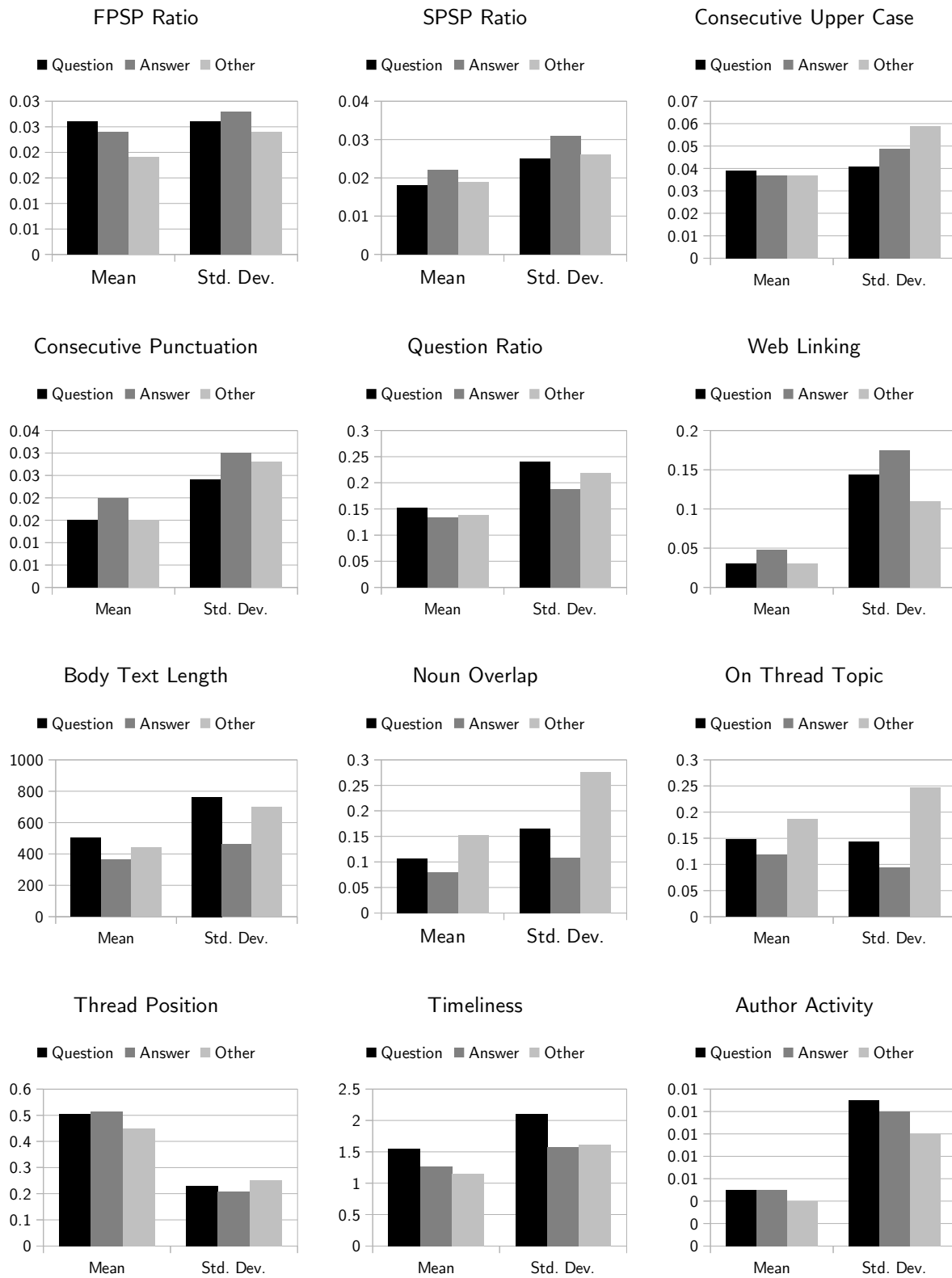


Figure 4.14: Distribution of dense feature values for all erroneous classifications.

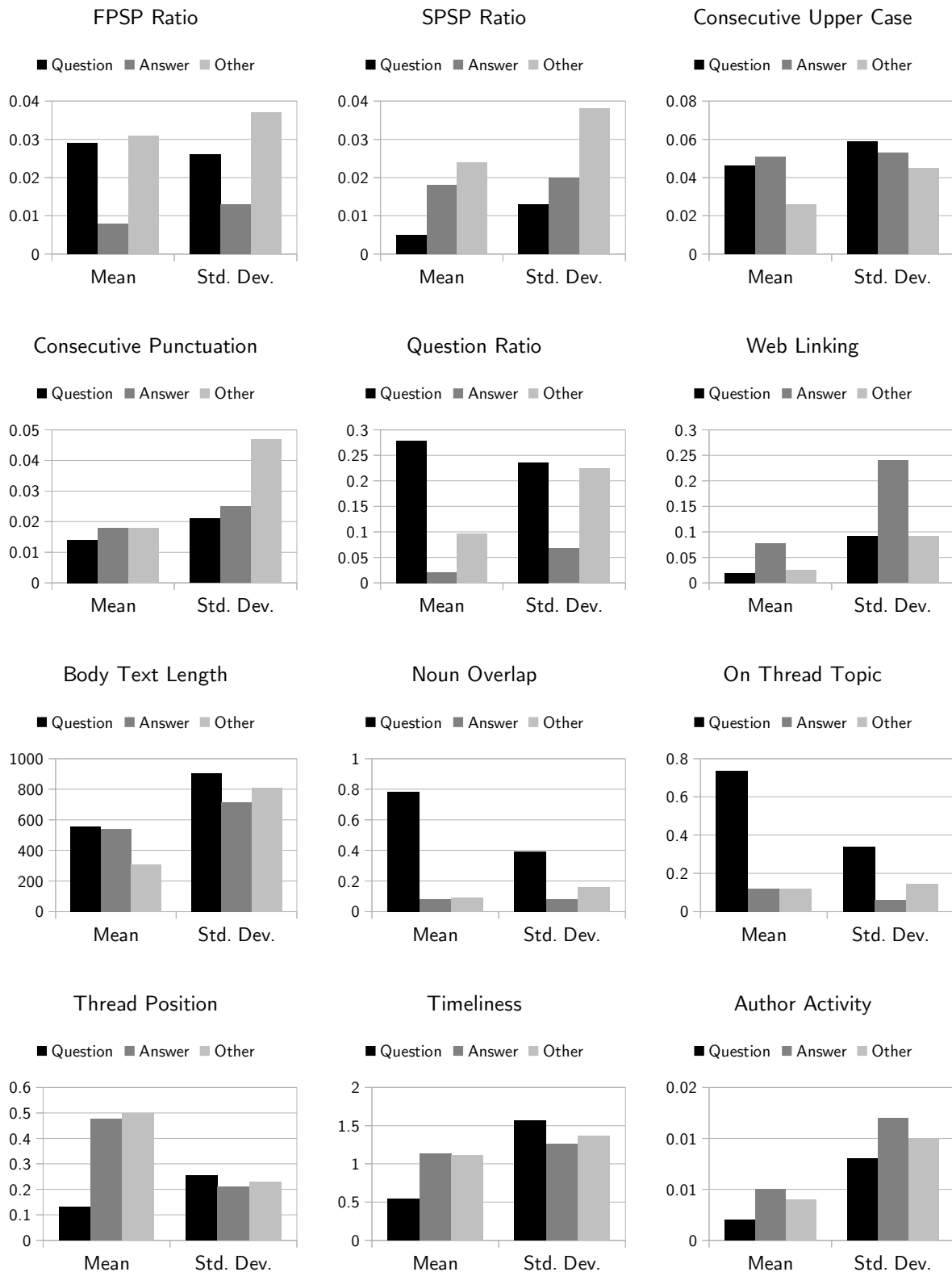


Figure 4.15: Distribution of dense feature values for all correct classifications.

clearly distinguishable from the other two types. Incorrect distributions are almost equally distributed. This suggests that additional features are required to help improve classifier performance in contrast to providing additional training examples. Whether this is true or not can be shown by means of so called learning curves as explained by Professor Andrew Ng from Stanford University¹¹. For this purpose we split the full dataset in half. The first half was used as training set, while the second was used as validation set. From each set we took the first example and trained a classifier as explained at the beginning of this section with the instance from the training set. We classified both instances by means of that classifier and measured the average squared error for both, the validation and the training set. In the next iteration the first two instances from the training and the validation set were included, then three, then four and so on until both sets were consumed. The average squared errors are plotted for each iteration to create the learning curves. The training curve is expected to start at a very low average squared error since it is easy to perfectly fit a classifier to the same set it was trained on if that set is small. The validation curve, on the other hand, starts at a very high average squared error since it is hard to train a classifier that fits unknown examples well on a small training set. So iteration after iteration, both curves are going to gradually close in on each other. If they are close to touching one another, this indicates that an underfitting or high bias problem exists. In this case, additional training examples are not useful since the classifier already fits the training examples to its best and will not improve on additional examples. Only additional features are capable of increasing extraction quality in such cases. If, on the other hand, there is a large gap between both curves, we are experiencing an overfitting or high variance problem. This may result if there are multiple features which allow the classifier to match the training set exactly but no other instances. In such cases, adding additional training examples helps to improve the generalizability of the classifier.

Figure 4.16 shows the learning curves for the full dataset. For low training set sizes the error bounces around wildly. It is interesting to see that the bagging classifier is not even able to classify its own training set in such a case. The error rate stabilizes at around 0.3 for larger training sets. Since both, the validation and the training set error rate, stabilize at around the same value, it is safe to conclude that adding further training examples would not be helpful in this case. In order to classify the remaining error cases correctly, additional features are required. This confirms the observations from the feature distributions in Figure 4.14 and Figure 4.15.

Still, the effect of choosing fewer features is unknown. According to brute force feature selection (see Section 4.4.3), there is a subset of dense features producing higher quality results on the benchmark dataset as compared to the application of all features. In a final experiment we are going to show how this subset works for the full dataset. We conducted the same experiment as at the beginning of this chapter, yet including only the dense features

¹¹<https://class.coursera.org/ml/lecture/64>

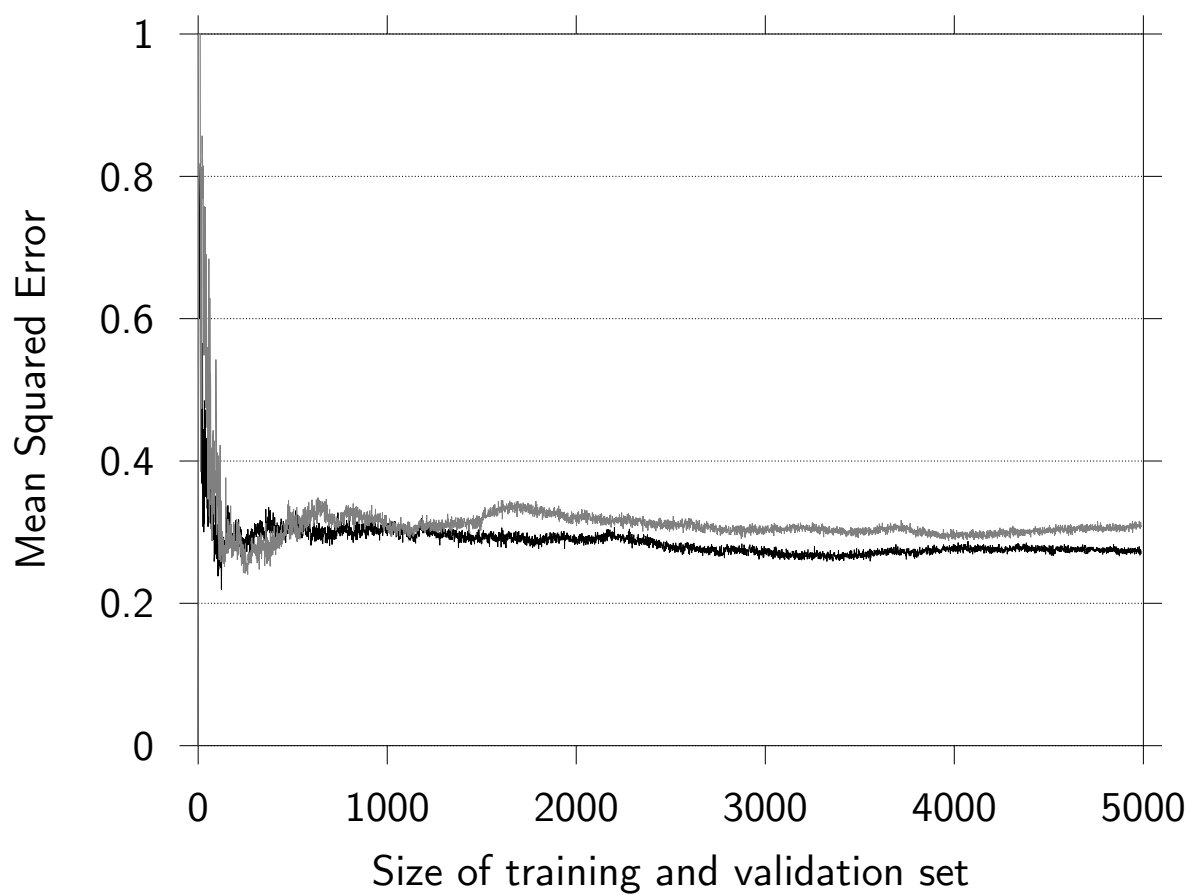


Figure 4.16: Learning curves showing the mean squared error rate for the full dataset split half into a training set and half into a validation set.

Table 4.14: Results of a five fold cross validation run on the full dataset using the top 155 lexical n-grams together with the top features according to brute force feature selection.

Contribution Type	Prior	Accuracy	Precision	Recall	F ₁ score
<i>Answer</i>	0.356	0.856	0.760	0.869	0.811
<i>Other</i>	0.245	0.813	0.653	0.500	0.566
<i>Question</i>	0.399	0.882	0.846	0.861	0.854
Micro Average	-	0.856	0.768	0.775	0.767

from the top run in Table 4.9 together with the top 155 n-grams. Table 4.14 displays the results in the same format as before. A comparison of this table with Table 4.11 shows that all results are slightly worse. So instead of improving the classification the subset created worse results. This is not absolutely unexpected, since even the improvement achieved on the benchmark dataset was very small and might have occurred because of noise. However, Table 4.14 also reveals that even the employment of a much smaller feature set produces comparable results.

Performance of Two Class Classifiers

Hitherto, we always used a three-class classifier (i.e. separating contributions into *Questions*, *Answers* and *Other* contributions). As an alternative, Table 4.15 and Table 4.16 show the performance on two two-class classifiers.

Table 4.15: Performance of classifier using only types for *Questions* and *Other* contributions. *Answer* contributions are marked as *Other* contributions in this case.

Contribution Type	Prior	Accuracy	Precision	Recall	F ₁ score
<i>Other</i>	0.601	0.904	0.918	0.924	0.921
<i>Question</i>	0.399	0.904	0.884	0.875	0.880
Micro Average	-	0.904	0.904	0.904	0.904

Table 4.16: Performance of classifier using only types for *Answers* and *Other* contributions. *Question* contributions are marked as *Other* contributions in this case.

Contribution Type	Prior	Accuracy	Precision	Recall	F ₁ score
<i>Answer</i>	0.204	0.903	0.772	0.742	0.757
<i>Other</i>	0.796	0.903	0.935	0.944	0.939
Micro Average	-	0.903	0.901	0.903	0.902

The first classifier separates *Questions* and *Other* contributions. Answers are grouped together with the remaining *Other* contributions. The second one separates *Answers* and *Other* and groups *Questions* together with the remaining *Other* contributions.

The expectation with regard to such two-class classifiers is the following. Since there are only two classes, the confusion for the classifier is much smaller. Therefore, the classification

results should improve. However, in comparison to Table 4.11 all measures for *Question* and for *Answer* contributions lost significantly. The *Other* type, including *Questions* and *Answers* respectively, improved by a considerable amount to approximately 90%.

The improvement for the *Other* type and the loss for both *Questions* and *Answers* in this case is, however, probably not due to a decreased classifier confusion. In particular, it results from the very high prior probability of *Other* contributions. This means that since the probability of encountering an *Other* contribution for both classifiers is so high, the probability of the classifier assigning the *Other* label to borderline cases also increases significantly. That is also the reason why *Questions* and *Answers* lose so dramatically.

However, there might be application scenarios for this kind of classifiers which are able to separate *Other* contributions with high confidence. On the other hand, the use of two classifiers may involve another disadvantage: Since both are applied to the same dataset there might be disagreements between them on the same contribution. Disagreement refers to the scenario that one classifier classifies a contribution as *Question*, while the other classifies the same contribution as *Answer*. For the experiment presented here this occurred in only approximately 0.2% of the classified instances. For such cases it would be possible to apply the three-class classifier as a tie-breaker for example.

4.4.5 Comparison of Results from Cong et al. (2008) and Hong & Davison (2009) to Effingo

Both publications, the one by Cong et al. (2008) as well as Hong & Davison (2009) report better results to the those shown in the previous chapter for the Effingo classifier. Both approaches also differ from our approach but provide good benchmark results. Unfortunately, neither Cong et al. (2008) nor Hong & Davison (2009) were able to provide their dataset or their implementation. For this reason, a reimplementation was used to create the benchmark. To reduce the necessary effort, only the approach of Hong & Davison (2009) was implemented. As explained in Section 4.1.3, Hong & Davison (2009) already compared their results to the ones by Cong et al. (2008). This comparison justifies omitting a separate comparison to Cong et al. (2008).

Since Hong & Davison (2009) created two distinct two-class classifiers for *Questions* and *Answers*, the labels from the dataset used in this thesis were filtered accordingly. For the *Question* classifier all labels except the *Question* label are mapped to *Other*, while for the *Answer* classifier all labels except the *Answer* one are mapped to *Other*. In addition, since Hong & Davison (2009) assume *Questions* to occur only at the first position in a thread, only first-position posts are used for training. Classification is carried out over items from every position (no filtering of only first-position posts is done) to examine whether the classifier created by Hong & Davison (2009) would be able to detect *Questions* occurring in other positions as well. In accordance, all first posts are filtered out for training the *Answer* classifier. Additionally, Hong & Davison (2009) assume each thread to only contain one *Answer*.

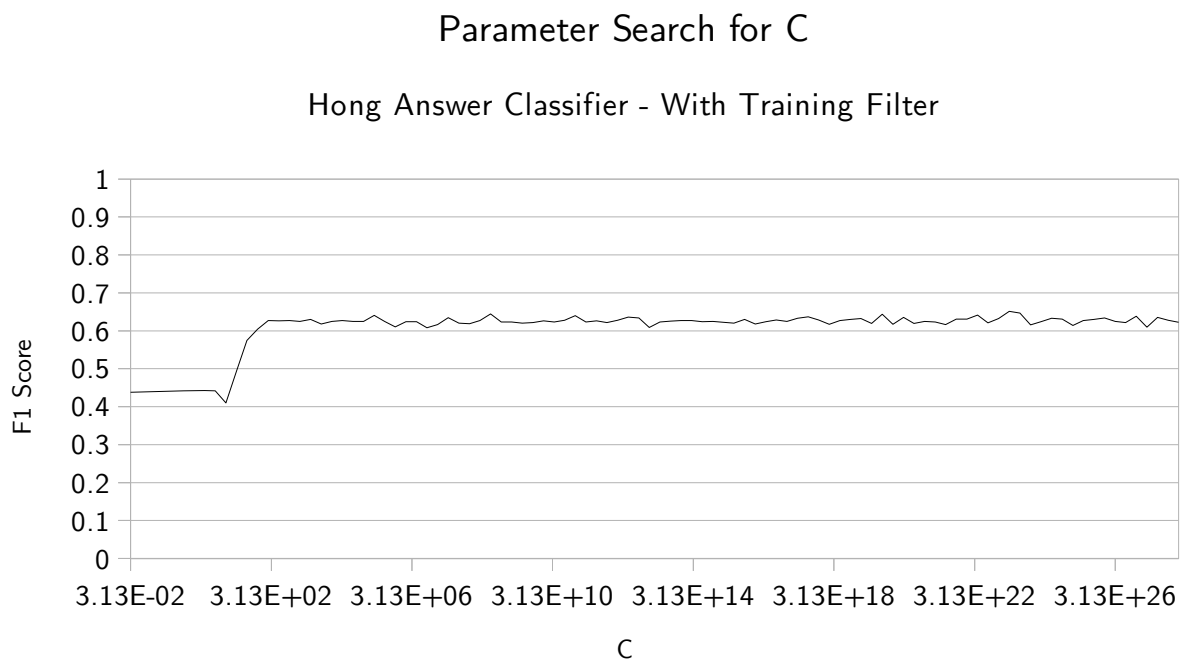


Figure 4.17: F_1 score values dependent on the value of C – the regularization parameter for the (Hong & Davison 2009) *Answer* classifier.

Therefore, only the first *Answer* is used for training. Remaining *Answers* are also mapped to the *Other* label. As explained for the *Answer* classifier, no filtering is done during classification as well. We denominate this procedure of filtering *Questions* at the start of a thread and including only first *Answers* a training filter since it is only applied during training.

We tried to keep as close to the description as possible, however Hong & Davison (2009) leave out several implementation issues. Firstly, they use LIBSVM as classifier, without providing its detailed settings. The most important parameters are the value C used for regularization and the kernel, as explained in section 2.3.3. For this experiment, a linear kernel was used as proposed by the LIBSVM documentation (Hsu, Chang & Lin 2003) for problems with large training sets and large feature sets. The regularization parameter was determined by searching the parameter space for the best value according to F_1 measure and five-fold cross validation on the benchmark dataset. Following the advice from the LIBSVM documentation (Hsu et al. 2003), parameter search started with a value of $2^{-5} = 0.03125$. The value was increased by powers of two e.g. 2^{-4} , 2^{-3} . . . and so forth. Figure 4.17 shows the development of C for the *Answer* classifier over 100 increments. The F_1 score remains relatively constant up to a regularization parameter of 16. After this value the score suddenly drops by almost 5% and then rises to reach a value slightly above 60% at around $C = 256$. It stays at this height for the remainder of the search scope and thus we are going to use $C = 256$ for the remaining experiments with the *Answer* classifier.

There is no figure to illustrate the development of C for the *Question* classifier. This is due to the fact that the *Question* classifier shows no significant change in F_1 score, no matter which value is used. It stays at around 0.25% and thus the initial value of 0.03125 will be used.

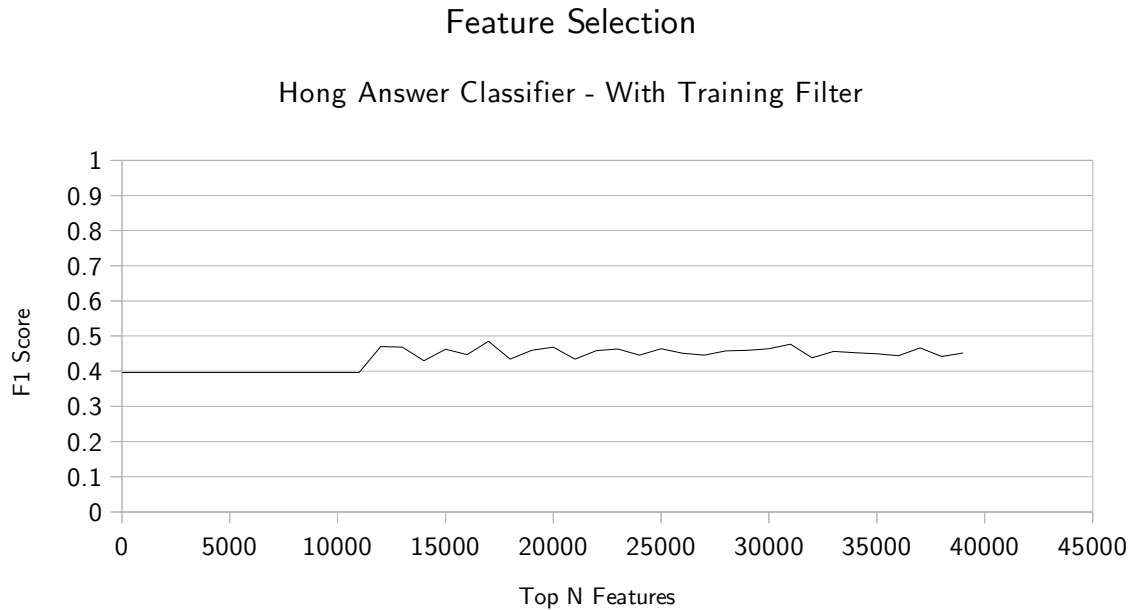


Figure 4.18: F_1 score of different Information Gain cutoff points for Hong & Davison’s (2009) *Answer* classifier.

Secondly, Hong & Davison (2009) describe their application of the approach developed by Carvalho & Cohen (2006) as explained in Section 4.1.2. This approach uses Information Gain feature selection (see Section 2.2.3), which requires a threshold value to select the most important features. However, no such value is provided by either paper. Therefore, we need to find a reasonable one independently. It can either be an absolute cutoff point such as the 1,000 most discriminative features, or an Information Gain score such as taking features with a score above 1.3. In order to find a valid amount of the top features to use, we tried various absolute cutoff points and measured the change in F_1 score to find the best. As regularization parameters for *Question* and *Answer* classifier we used the ones determined above. In total, the benchmark dataset contains close to 240,000 Carvalho n-grams. Figure 4.18 shows the results of Information Gain feature selection for values between 1 and 40,000 in increments of 1,000 using our implementation of Hong & Davison’s (2009) *Answer* classifier. The figure for the *Question* classifier is not displayed, since it is a straight line similar to parameter search which alternates between 0.2549 and 0.2554. It is therefore not clear which cutoff point to use. For this reason, feature selection for Hong & Davison’s (2009) *Question* classifier experiments is ignored and all n-grams are used.

The *Answer* classifier has a sudden increase at around 12,000 n-grams with an F_1 score of 0.47. After this, there are only slight changes which can be attributed to noise. In order to select the most appropriate set of top n-gram features we zoomed into the area around 12,000 top n-grams. Figure 4.19 shows the development of the feature selection curve from 11,000 top features to 13,000 top features in increments of 100 features. The peak is reached at 12,200 top n-grams, which consequently are used for further experiments.

Table 4.17 shows the results for both Hong classifiers in comparison to Effingo results pre-

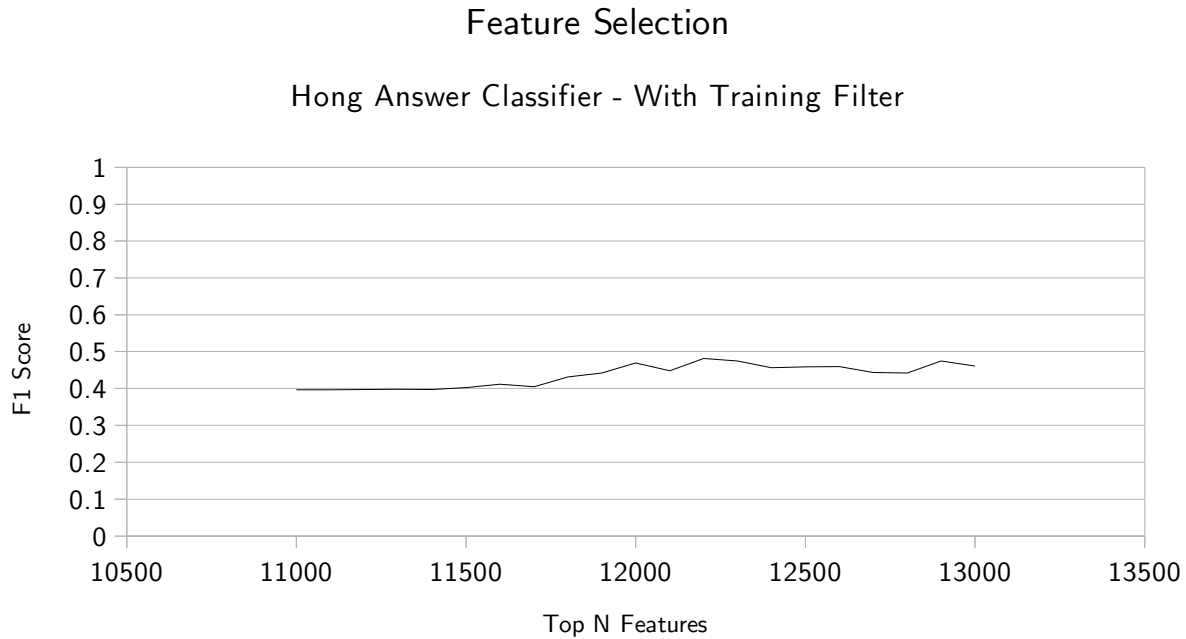


Figure 4.19: Development of F_1 score for feature selection using the *Answer* classifier as proposed by Hong & Davison (2009).

sented in the previous section.

Table 4.17: Comparison of *Question* and *Answer* Classification by Hong & Davison (2009) to the Effingo results.

	Accuracy	Precision	Recall	F_1 score
Hong & Davison (2009)				
<i>Question</i>	0.40	0.40	1.0	0.57
<i>Question</i> on first posts	0.95	0.95	1.0	0.97
<i>Answer</i>	0.80	0.69	0.54	0.61
<i>Other</i> vs. <i>Question</i>	0.40	1.0	0.0008	0.002
<i>Other</i> vs. <i>Question</i> on first posts	0.95	1.0	0.02	0.04
<i>Other</i> vs. <i>Answer</i>	0.80	0.83	0.90	0.87
Effingo				
<i>Question</i>	0.90	0.89	0.87	0.88
<i>Answer</i>	0.86	0.88	0.77	0.82
<i>Other</i>	0.83	0.52	0.69	0.59

The abysmal performance of the Hong features on our dataset might be caused by the artificial reduction of training items for the two classes of *Questions* and *Answers*. This is noticeable especially for *Questions*. Since only first post contributions are used for training and since almost all first post contributions are *Questions*, the classifier has a large bias towards *Questions*. Therefore, it classifies nearly everything as a *Question*. This is not absolutely fair since it was never intended to classify anything but first position posts by Hong & Davison (2009). For comparison we tried the same classifier only on first post contributions. The

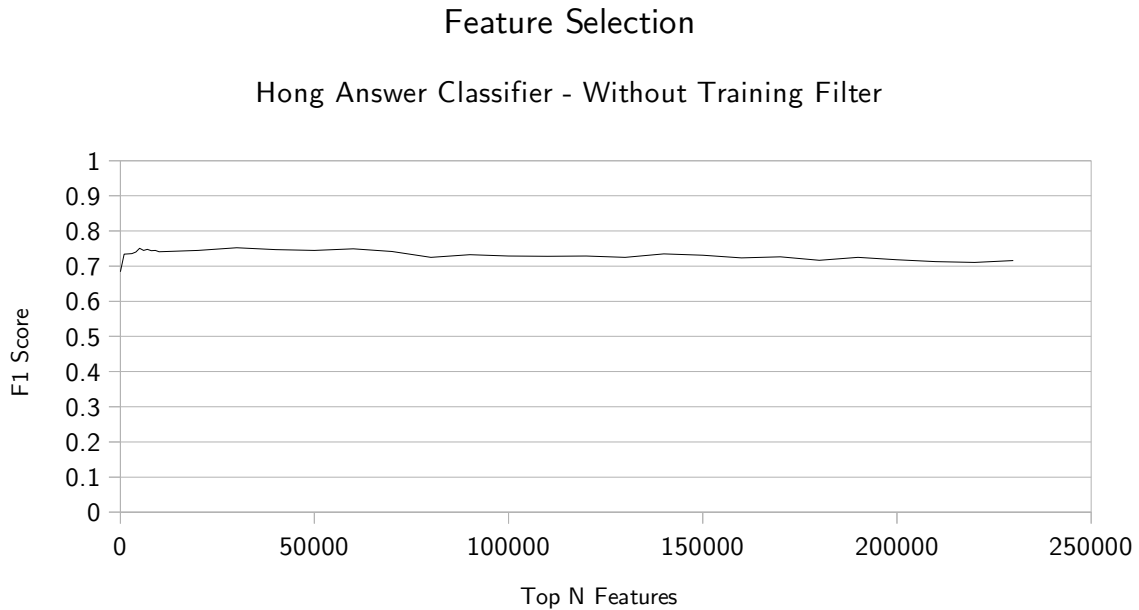


Figure 4.20: F_1 score of different Information Gain cutoff points for Hong & Davison’s (2009) *Answer* classifier without using the filtering as applied by Hong & Davison (2009).

results—also provided by Table 4.17—do ameliorate little, although not significantly. Recall for *Other* contributions is still very bad, because the classifier predicts nearly everything to be a *Question*. We therefore conclude that this artificial filtering of posts for *Answers* as well as for *Questions* is not beneficial to the classification results since it reduces the training set size and especially its variance drastically. Therefore, we repeatedly conducted the same experiments, but left out the training filter explained in the beginning of this section.

The *Question* classifier manifests a similar behavior without training filter. No value of C and no n-gram feature set size has any influence on the classifier’s performance. However, the performance is constantly above 40% by a small amount. Thus again we will use a regularization parameter of 0.03125 and include all n-grams.

The parameter search for the *Answer* classifier without training filter reaches its maximum very early as displayed in Figure 4.20. It only shows a zoom into the first few runs. After these runs the curve is a straight line at F_1 score of 0.75. Its maximum is reached at 1.0, which basically means the optimum regularization parameter effects no regularization at all.

The feature selection curve in Figure 4.21 shows feature selection for all the 240,000 Carvalho n-grams. However, the maximum is also reached very early at approximately 30,000 features. From that point on, it only decreases. This means, only the top 30,000 n-grams according to information gain scores from the benchmark dataset are helpful for *Answer* classification, whereas the remaining 210,000 only confuse the classifier. Thus, these 30,000 features are applied to generate the final results on the complete dataset.

The final results using the parameters determined so far are shown in Table 4.18.

Omitting the training filter results in much better classification quality. The classification of *Question* and *Answer* contributions is still not as good as the Effingo classifier. However,

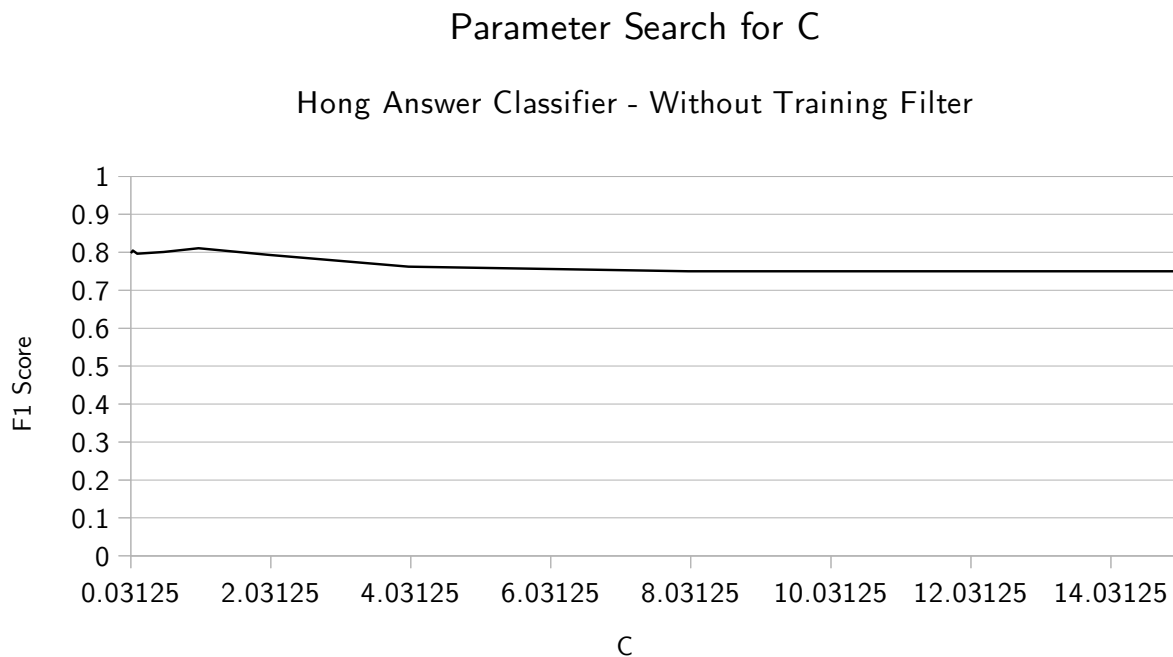


Figure 4.21: F_1 score values dependent on the value of C —the regularization parameter for the (Hong & Davison 2009) *Answer* classifier without filtering of contributions, like done by Hong & Davison (2009).

Table 4.18: Comparison of *Question* and *Answer* Classification by Hong & Davison (2009) to the Effingo results without filtering of contributions like done by Hong & Davison (2009).

	Accuracy	Precision	Recall	F_1 score
Hong & Davison (2009)				
<i>Question</i>	0.63	0.85	0.08	0.14
<i>Answer</i>	0.82	0.76	0.75	0.75
<i>Other vs. Question</i>	0.63	0.62	0.99	0.76
<i>Other vs. Answer</i>	0.82	0.86	0.87	0.86
Effingo				
<i>Question</i>	0.90	0.89	0.87	0.88
<i>Answer</i>	0.86	0.88	0.77	0.82
<i>Other</i>	0.83	0.52	0.69	0.59

they achieved much better results on the *Other* type using both classifiers. As shown at the end of Section 4.4.4, this is probably a result of employing the two-class classifier approach in contrast to the one three-class classifier approach preferred by Effingo. In comparison to the Effingo two-class classifier presented in Table 4.15 and Table 4.16, the results for both *Other* types are still worse, while the performance of both two-class classifiers for *Questions* as well as for *Answers* is comparable.

4.4.6 Discussion

The results presented in Section 4.4.4 and the comparison in Section 4.4.5 illustrate that the Effingo Question/Answer classifier is capable of finding Question/Answer contributions from Web forums with high quality. Especially the classification quality for *Question* contributions is quite high.

The low scores for the *Other* type indicate that many *Other* contributions are erroneously categorized as *Question* or *Answer*. However, if we look at the errors in detail, it seems that this is mostly due to ambiguities between whether a contribution is really of the *Other* type or not. For example the following contribution was labeled as *Description*, so it actually belongs to the *Other* type but was classified as *Answer*.

```
you both are right in different ways:  There will be an upgrade path to
mySAP ERP, but there will be no way to run R/3 Enterprise 4.7 on WebAS 6.40.
Or to make it more easy:
```

```
R/3 Enterprise runs on WebAS 6.20 mySAP ERP runs on WebAS 6.40
```

```
Regards,
Benny
```

One could, however, argue that the label is wrong and this contribution is no *Description* but an *Answer* and thus the classifier was correct.

The classifier also seems to have problems with very short contributions consisting of one or two words. Such contributions seem to cause confusion quite easily. They usually require the context of another contribution they are referring to which the classifier does currently not use.

Answers are often wrongly classified if they contain first-person personal pronouns. Such *Answers* do not fulfill the expectations that *Answers* usually refer to the asking user employing second-person pronouns such as 'you'. This often happens if the answering user provides an example explaining how the problem is solved in their case.

Another problem is the occurrence of code blocks, which is quite common in the examined forums. Features dependent on natural language processing such as the lexical n-grams fail on such texts.

Some of the strongest evidence for a *Question* is the position of the post in the thread and its author's activity. The classifier easily confuses *Questions* not located in the beginning of the thread or *Questions* by very active users. This is well illustrated by Figure 4.14 and Figure 4.15.

The comparison to the results achieved by Hong & Davison (2009) shows that the Effingo classifier is better suited for the complete dataset presented in this work. This, however, does not rule out that Hong & Davison's (2009) algorithm was better suited for the dataset used in their work. The experiments however showed that their classifier is probably not generalizable to all Web forums.

4.5 Conclusion

This chapter started with an overview of automatic question answering for the generic case and also for approaches with a focus on Web forums.

We showed that classic question answering is not sufficient to find *Questions* and *Answers* in a Web forum and that approaches focused on forums still have some severe drawbacks. Most important are the assumptions as to which contributions may be *Questions*, that each *Question* may only have one *Answer* and the complete ignorance of a question's context. Furthermore, existing approaches only used very small datasets not openly available.

Based on existing work we developed a typology of forum contributions and showed several features of the different contribution types. The features focus mainly on *Question* and *Answer* identification, which are the contributions most easily distinguished from the *Other* types as our inter-annotator agreement analysis shows. We also presented a means to select the most significant features from all proposed ones and, finally, used those features to evaluate different classifiers on our manually labeled dataset. For that purpose we used a smaller benchmark dataset to choose the best classifier and examined the parameter space for that classifier in more detail.

The results on our complete dataset of 10,021 contributions from the SAP Developer Network and the Oracle Developer Network show that it is possible to achieve usable results with the proposed approach. It might be possible to improve the classification by means of another or an extended selection of features. However, such a selection requires careful consideration since each feature could also decrease the classification accuracy already achieved. So instead of applying additional features, fixes and improvements on the existing features might be a next step to take. The POS tagger for the lexical n-grams, for example, currently works on the brown corpus tag set. A tagger trained specifically for forum data might produce more reliable results. Also, the question tagger and the imperative tagger are very basic implementations applying simple rules. More elaborate implementations could provide improved values for these features and might be able to improve extraction quality. Different types of sentences are, for example possible, to detect applying supervised learning as well. Techniques such as Hidden Markov Models are possible proposals for such

algorithms.

We tried to setup all experiments in such a way as to avoid any overfitting. All parameters are set using the benchmark dataset and the labels for the final evaluation are provided by a student annotator who was not involved in the creation of the classifier and its evaluation. Thus, we believe that the results are applicable at least to other forums which show a similar behavior to the SDN and the ODN. Such forums include question-focused forums, which usually present one question per thread. Among them are probably Question/Answer forums such as the network provided by StackExchange¹²

Finally, we also showed Effingo's behavior in comparison to one of the most recent related approaches. The results indicate that the related approach developed by Hong & Davison (2009) is inferior with regard to the presented dataset. We were, however, able to raise its results to comparable values with a few small adaptations that alleviated the constraints Hong & Davison (2009) applied during their research.

Finally, we provided a discussion of the most obvious and common error cases we encountered during the final analysis of the classification results. These observations can serve as the foundation for future research.

¹²<http://stackexchange.com/>

5 Example Applications

After the previous two chapters have shown how to extract forum posts from the Web and how to find question and answer contributions, this chapter finally presents an overview of applications for the crawled and classified forum contributions. The chapter is separated into two parts.

The first part is an application for question aggregation, as explained in Section 1.3, with a discussion of how to achieve this task. In the discussion a semantic approach for the task of finding near-duplicate questions is compared to a syntactic one.

The second part of the chapter explains several other applications which are possible to build upon the Effingo system as presented in Section 1.3.1.

5.1 Near-Duplicate Question Aggregation

Finding near-duplicate questions is an application of text topic detection. Current algorithms for topic detection however mostly ignore semantic similarities for simpler syntactic ways of matching similar questions as the following paragraphs show. Syntax in this context refers to matching word occurrences in two near-duplicate question candidates, possibly modified by some scoring of the word's importance for the question. In contrast, semantic similarities are relations between words as well as semantic connections like synonymy and hypernymy. We assume that such relations are important to find near-duplicate questions with the same content phrased by different people, who might not even be aware of each other. For this reason, a semantic topic detection approach on Web forum questions is presented in the following sections. The approach considers semantic relations using the generic knowledge base provided by Miller (1995). The proposed approach finds questions with near-duplicate content to a query question. That way, near-duplicate questions are easily grouped and, in a condensed way, shown to users with a similar question.

The contribution of this section is the examination of shallow semantic topic detection on question posts from Web forums using questions from different forums. It also includes a detailed discussion of the possible results. In addition, a comparison to naïve implementations and a state-of-the-art approach is provided. The results were created during the supervision and as part of the diploma thesis of Christian Hensel (Hensel 2013).

5.1.1 Topic Detection

The area of Topic Detection was established with the Topic Detection and Tracking (TDT) conferences by Allan, Carbonell, Doddington, Yamron & Yang (1998). Topic Detection finds topic groups in a collection of news stories and assigns them to the correct group.

Stories are clearly marked pieces of text as short as a single paragraph, such as a forum post, or as long as a complete text, such as can be found on a Web page. A story is defined as *A Seminal event or activity along with all directly related events and activities*. Allan et al. (1998) describe the research problems of Story Segmentation, First Story Detection, Cluster Detection, Tracking and Story Link Detection. Allan et al.'s (1998) research focuses on transliterations of television news shows. Since a forum is—in contrast to such a transliteration—no continuous stream of text with hidden stories occurring one after another, Story Segmentation, First Story Detection and Tracking are not relevant for this work. It rather focusses on Cluster Detection and Story Link Detection. In addition, Allan's definition of a topic does not fit the problem of question aggregation well since it requires an event or activity. Our definition for questions, formulated by Definition 5, in contrast requires a forum user with an Information Need.

The publications of TDT propose probabilistic and vector space-based approaches for topic detection on news stories from the TDT2 corpus.

The probabilistic approach models each topic as a language model, such as a Hidden Markov Model used by Leek, Schwartz & Sista (2002), or uses simple unigram models as developed by Yang & Ng (2008) and Yamron, Gillick, van Mulbregt & Knecht (2002). These approaches generalize well to all texts and are even language-independent. With regard to news stories the approach proposed by Leek et al. (2002) achieves only mediocre results. However, with their precision of 53% and a recall of 67% for the cluster detection task, their Hidden Markov Model approach performs better than the ones proposed by Yang & Ng (2008) and Yamron et al. (2002).

In contrast to the probabilistic approach, the vector space approach models topics and news stories as vector from the vector space of all words. Vector space approaches are discussed in Yang & Ng (2008), Franz, McCarley, Ward & Zhu (1999), Eichmann & Srinivasan (2002), Levow & Oard (2002), Allan, Lavrenko & Swan (2002), (Schultz & Liberman 2002) and Chen & Ku (2002). Both approaches were only tested on news topics using an edited and limited vocabulary.

Topic detection for online discussions is examined for example by Tang (2008). She builds graph structures by matching keywords. By means of these keyword graphs her approach is capable of clustering the blogosphere, online discussion boards or similar Web 2.0 applications into topics. However, the detected topics only build an overview of hot topics and are not associated to groups of similar questions. Bengel, Gauch, Mittur & Rajan (2004) use a set of predefined concepts and a keyword index for each concept to categorize chat messages into hot topics similar to Tang. Even though such messages show similar properties

as forum posts, such short utterances are not the focus of this work.

Work on Semantic Topic Detection was introduced, for example, by Wang, Huang, Guo & Li (2009). They extract terms from research papers, considering synonyms and hypernyms using data collected from WordNet. This approach demonstrates that a semantic topic detection approach is able to achieve high-quality results on research papers. The approach presented in this section will build on some of these ideas but add word co-occurrences and show performance on Web forums.

Topic Detection in Web forums was examined by Wu & Li (2007), Xu & Ma (2006) and Yang & Ng (2008). However, these algorithms only consider forum pages or whole threads—not question contributions—as stories.

5.1.2 Question Aggregation

The goal of the algorithm proposed in the following sections is to assign similarity scores to question contribution pairs such as the following pair of questions:

<p><i>I can add nodes to my JTree, but after expanding a node, I can not add any child nodes to that node, or at least they dont show on the screen? Actually I can use the .add(new DefaultMutableTreeNode("Label")); to add a node to a parent node, but the new child node doesn't show on the display of the JTree? I am having trouble displaying a JTree with newly added nodes.
 ...</i></p>	<p><i>Using the same exact routine as above - I add additional new children to the root, the new children do not display in the Jtree. ...</i></p>
--	--

Our semantic similarity detection approach starts with a training phase using a large corpus of questions. The initial step is to calculate a co-occurrence matrix for terms occurring together in the same question. In order to reduce the amount of relevant terms, each question is preprocessed with a stemmer and a stop word filter. Repeated co-occurrences are counted multiple times. The absolute frequencies of each word pair are added to the corresponding field in the co-occurrence matrix. The matrix is a representation of typical word relations and provides the input for the similarity calculation of two questions. The similarity calculation considers two factors. The next paragraphs present a formalized notation of both. An example will be provided afterwards for improved understanding.

The first factor is the relation of matching term co-occurrences from both questions. At first, an importance measure i is calculated for each co-occurrence based on the value in the co-occurrence matrix M according to Equation 5.1. The importance value of i for a term co-occurrence c and a question Q refers to the absolute co-occurrence frequency of both co-occurring terms $c(1)$ and $c(2)$ normalized by the maximum absolute frequency for any two

terms in Q .

$$i(c, Q) = \frac{M_{c(1),c(2)}}{\max(M_Q)} \quad (5.1)$$

The result of i is high for co-occurrences which are frequent in the training set and low for infrequent ones. That way, all co-occurrences are ranked in order of importance for a question with values ranging from 0 to 1.

The similarity score consists of an aggregated relation score for all co-occurrences from both questions. This relation score is calculated for one pair of co-occurrences using Equation 5.2. It builds the ratio of the smaller importance value to the larger one. That way, the domain of rel is $(0, 1]$. For important co-occurrences, according to the co-occurrence matrix M , rel takes on higher values than for unimportant ones.

$$rel(i_1, i_2) = \begin{cases} \frac{i_1}{i_2}, & \text{if } i_1 < i_2 \\ \frac{i_2}{i_1}, & \text{otherwise} \end{cases} \quad (5.2)$$

Equation 5.2 is only applicable for two co-occurrences with the same terms. This means that both terms $c(1)$ and $c(2)$ must be equal for both co-occurrences with importance values i_1 and i_2 to make them comparable. However, as already mentioned the relations between terms are more complex on a semantic level. That is why a second, diminishing factor is added for co-occurrences with terms having a similar meaning. This factor results from semantic similarities between the terms of two co-occurrences according to the word database WordNet developed by Miller (1995). Current research provides different similarity measures based on word relations from WordNet. There are for example approaches developed by Wu & Palmer (1994), Lin (1998), Resnik (1995) and Leacock & Chodorow (1998). Since the approach by Lin (1998) has proved to produce high quality measures, it will be used as a function called lin . Its domain is defined as a value of 1 for equal words and a value close to 0 for rather unrelated words. There are four possible relations between the four words from two co-occurrences shown in Figure 5.1. Since lin is defined over two words, and not

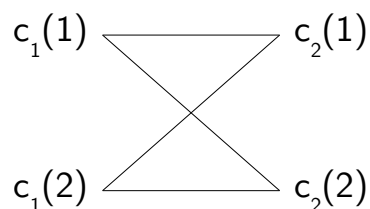


Figure 5.1: Possible relations between the words of two co-occurrences.

on co-occurrences, lin is calculated as the average of all four possible relations for our purposes. Since lin never reaches 0, a threshold is defined to filter out all similarities below a certain value. All values of lin below this threshold are set to 0. These two factors result in Equation 5.3 which calculates the similarity between two questions Q_1 and Q_2 as the sum of

Table 5.1: Co-occurrence matrix and importance values i for Q_1 and Q_2 .

	awt	concept	component	swing	understand	comprehend
awt	9	3	2	3	1	3
Q_1			(2/6)	(3/6)		(3/6)
Q_2		(1/7)	(2/7)	(3/7)	(1/7)	
concept	3	18	3	7	2	8
Q_1						
Q_2			(2/7)	(7/7)	(2/7)	
component	2	3	13	6	4	1
Q_1				(6/6)		(1/6)
Q_2				(6/7)	(4/7)	
swing	3	7	6	19	4	1
Q_1						(1/6)
Q_2					(4/7)	
understand	1	2	4	4	11	1
comprehend	3	8	1	1	1	12

the two presented factors over all co-occurrences from Q_1 and Q_2 .

$$\text{sim}(Q_1, Q_2) = \frac{\sum_{c_1 \in Q_1} \sum_{c_2 \in Q_2} (\text{rel}(i(c_1, Q_1), i(c_2, Q_2)) \cdot \text{lin}(c_1, c_2))}{|Q_1| \cdot |Q_2|} \quad (5.3)$$

Equation 5.3 also contains a factor normalizing the result with the product of the lengths of the two questions.

For an example consider the following two term sets, resulting from two questions after stop word filtering and stemming:

$$Q_1: \text{awt, swing, concept, awt} \quad Q_2: \text{concept, awt, swing, component}$$

Also consider the co-occurrence matrix presented by Table 5.1. The matrix presents the co-occurrences (the integer numbers) in the first line of each cell. In order to simulate a larger training set, some additional words not in Q_1 or Q_2 are shown as well (i.e. “understand” and “comprehend”). In addition, the importance values i for each co-occurrence in each query are shown in the second and third line of each cell, but only if co-occurrence applies. This means the value in the first row, fourth line, results from the term “swing” occurring three times together with “awt”, for instance. The most important co-occurrence for the second question is “swing” and “component”, since among the co-occurrences from that question it has the largest value in the co-occurrence matrix. Thus, all other values are normalized with 6 and the importance of “swing” and “awt” for question two computes to 3/6. Populating Equation 5.3 with these values results in Equation 5.4.

$$\begin{aligned} \text{sim}(Q_1, Q_2) &= \frac{\left(\frac{2}{7} + \frac{3}{7} + \frac{2}{7}\right) \cdot 1.0 + \left(\frac{1}{7} + \frac{1}{6} + \frac{1}{6}\right) \cdot 0.9}{4 \cdot 5} \\ &= 0.183154762 \end{aligned} \quad (5.4)$$

5.1.3 Evaluation

This section provides details on the performance of the similarity detection algorithm presented in the previous section. It focuses on two areas. First, it explores the influence of the threshold value for the semantic similarity as proposed in the previous section and second, it compares the results generated by the similarity measure to human assessments.

Dataset

Near-duplicate questions are very sparse among arbitrary sets of Web forum contributions. This makes it hard to evaluate any near-duplicate question detection approach. Nevertheless, in order to test the performance of the approach presented in Section 5.1.2, a new dataset was generated and enriched with duplicates. The dataset consists of 315 question posts with a shared domain, namely Java programming. We started with a collection of 15 seed questions and added near-duplicates using Google and diverse forum search engines. We also added non near-duplicates from the same forums just including the most recent threads. As most recent threads we used the threads displayed on top of the first list-of-thread page from each forum. That way, we created 15 sets of 20 posts for each seed question, totaling 315 posts (each group consisting of the seed question and 15 near-duplicates as well as non near-duplicates).

Three human annotators evaluated each of the 15 sets and created a ranking of how similar to the seed question they thought each question to be. Furthermore, they were asked to mark a question if they thought it was a near-duplicate. This gold standard was used for all further evaluations presented in this section. General agreement between the human annotators was quite good. All annotators recognized nearly all near-duplicate questions and ordered them at the top of the list. However, even though relative positions of posts were assigned similarly by all annotators, the absolute order varied heavily. All annotators tended to attribute shorter questions to higher ranks. In addition, long and complicated near-duplicate questions with the actual question sentence at the end of a long descriptive text were not recognized correctly. The evaluation results are calculated using each annotator's assessment separately, averaging the individual results to generate the final values.

Evaluation Measures

The results are evaluated based on the measures presented in Section 2.1.1. However, since this is no classification task, they required some adaptation. For the purpose of measuring the correctness of ranked lists precision, recall and F_1 score are adapted as proposed by Vaughan (2004).

Precision is calculated by means of the simplified Spearman coefficient according to Equation 5.5.

$$p = 1 - \frac{6 \cdot \sum_i [\text{rg}(x_i) - \text{rg}(y_i)]^2}{n(n^2 - 1)} \quad (5.5)$$

The value of n refers to the amount of posts in the list and thus always equals 20 for our evaluation. The result of $\text{rg}(x)$ refers to the rank of the post x either in the gold standard or as provided by the automatic system it is compared to. This automatic system, for example, can be the algorithm presented in Section 5.1.2. The Spearman coefficient takes on values between -1 for no agreement between the ranking from the gold standard and the human labeler up to 1 for complete agreement.

Recall is calculated according to Vaughan (2004) by counting the correct results from the top k results, with k being the amount of expected results. As a result recall is 1 if the top k are the expected semantically similar questions and gradually decreases to 0 as fewer of the top k results are relevant results.

F_1 score is the known harmonic mean of precision and recall as proposed for MUC (Grishman & Sundheim 1996).

Those three—precision, recall and F_1 score—are used for the evaluation results presented in the following sections.

Threshold Evaluation

As a first step we tried to find the optimal threshold for semantic similarity. A low threshold would include terms rather distantly related in semantic terms, while a high threshold would reduce the approach to a simple syntactic co-occurrence comparison. Figure 5.2 shows the development of the F_1 score on our dataset with regard to different threshold values. The optimal value for our dataset is located, according to Figure 5.2, at approximately 0.74. This value will be employed for the following experiments.

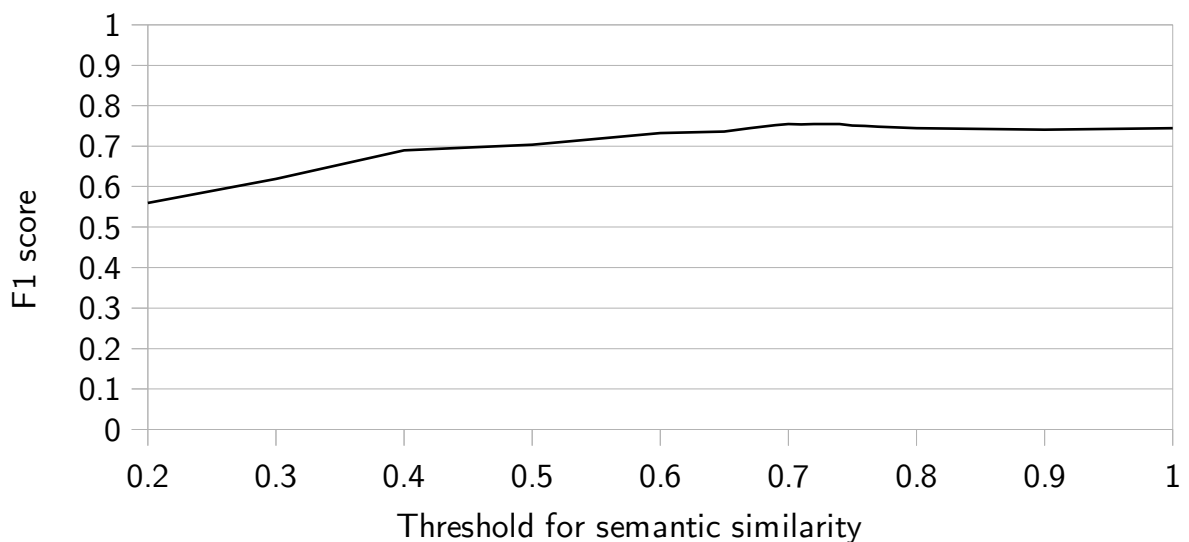


Figure 5.2: Development of F_1 score for different thresholds.

The small hump around the maximum can be explained as follows: For very low thresholds, words which have almost no relation are considered for similarity calculation. Words such as “help” and “use” might occur by chance reducing result quality. At around 0.74 the lin

Table 5.2: Results for applying different algorithms to the evaluation dataset.

	Spearman/Precision	Recall	F ₁ score
Semantic Baseline	0.133	0.497	0.210
Syntactic Baseline	0.589	0.790	0.675
Effingo	0.670	0.864	0.755
Lucene	0.667	0.867	0.754

algorithm returns only direct hypernyms and synonyms. These are the kinds of relations which have a positive effect on the assessment of whether two questions are near-duplicates or not. At around 0.75 the lin algorithm returns only synonyms and thus, the F₁ score drops again.

Results

The final results of the algorithm proposed for this work were compared to three other approaches: a semantic baseline, a syntactic baseline and a state-of-the-art syntactic algorithm. The semantic baseline calculates semantic similarity between all words from Q_1 to Q_2 according to Lin (1998) and sums them up.

In addition, two syntactic approaches were compared. The syntactic baseline is a simple tf-idf approach. It creates two term vectors with tf-idf values for Q_1 and Q_2 and compares them using their scalar product. This comparison is conducted for all pairs of two questions, the first question being the seed question and the second being one of the 20 other questions from the same group. The calculated scores are used to rank all questions with respect to the seed question. The tf-idf values are calculated from the complete evaluation dataset including training and test data.

As state-of-the-art syntactic algorithm we used a well established Information Retrieval framework: Apache Lucene (*Apache Lucene* 2013). In this approach the 20 questions from each group are indexed by Lucene. The Lucene index is queried using the seed question and Lucene creates a ranking of all 20 questions according to the seed question.

Table 5.2 presents the results of those three approaches and the approach explained in this work, namely Effingo.

As expected, the semantic and the syntactic baseline produce lower scores than the Effingo and the Lucene approach. The baseline even resembles an equal distribution where all questions are randomly distributed over all 20 ranks. The expected recall for an equal distribution is 0.5 since the relevant documents in this case are equally distributed over the 20 ranks. The Spearman coefficient for such a distribution is 0.135. The result according to Table 5.2 is 0.133, which indicates a rapport.

Both approaches, the semantic and the syntactic baseline, have the problem that long texts tend to have a higher probability of containing many relevant terms with low scores. Such long texts tend to swallow shorter ones and, thus, are usually ranked higher. The algorithm in this case shows the same behavior as the human annotators, who tended to rank long

texts higher than short ones.

The syntactic Lucene algorithm's results are similar to the results of the Effingo approach. This may be due to the way the gold standard was created. Since a search engine was used to find the near-duplicates in the first place, it is not surprising that another search engine such as Lucene is capable of finding those texts again.

The Effingo approach was also capable of finding a significant portion of the relevant texts. It identified 128 of the 150 near-duplicates correctly. False positives resulted from questions with a similar length as the seed question, which were capable of moving weak true positives down below themselves. The false positives usually contain some low ranking similar words which suffice to boost the similarity above the similarity achieved by some of the weak true positives.

For high thresholds the Effingo approach reduces to a syntactic co-occurrence measure of terms between questions. This occurs if the lin measure is below the threshold for all except equal co-occurrences. The hump in Figure 5.2 occurs directly before this threshold is reached. It is located at an F_1 score of around 0.7. Since the F_1 score drops after this threshold, it seems that semantic relations do indeed have an influence on the quality of similarity detection.

Another advantage of the Effingo approach over Lucene is that it returns a normalized score. Lucene ranking scores are not comparable among different queries. The Effingo scores are constant, no matter which questions are compared for a whole dataset. This is important as soon as rankings must be merged or extended.

The Effingo approach still has some limitations which prevent it from finding all near-duplicates. One is that WordNet is a generic knowledge base which knows nothing about the target domain. On the one hand, this makes the approach domain-independent. On the other hand, domain-dependent terms provide useful information, which consequently is not taken into consideration with Effingo. Thus, instead of associating Java with a programming language, it might relate it to coffee or the term "island". Information about such terms in the form of a structured knowledge base could solve or at least reduce the problem of long questions swallowing smaller ones. How can this be facilitated? A controlled vocabulary in the form of a knowledge base could provide the terms used for semantic near-duplicate detection and their meanings. That way, the term Java would not be associated with an "island" or coffee but only with a programming language in a programming forum. In addition, the amount of terms would be drastically reduced and, since not every random word would be matched, random matches from long texts would no longer occur. The problem with this approach is the cost associated with the creation of such a knowledge base for each domain. There is a large research area regarding this topic which is not part of this thesis.

5.1.4 Summary

This section explained an approach to find semantic near-duplicate questions from social media sites. It is based on the idea that co-occurrences of words are important to identify two differently phrased questions with the same content. In addition, it incorporates the knowledge base WordNet to identify similarities between semantically correspondent terms.

The section shows that the approach allows for better results than a semantic and a syntactic baseline and reaches the performance of current state-of-the-art information retrieval systems, measured by means of established scores.

It is still uncertain whether the results of the state-of-the-art algorithm and the Effingo algorithm would find the same results or whether the union of the result sets would create a higher quality ranking. If the overlap between both was smaller than one, it would be possible to take the correct results from both algorithms and combine them to an aggregated correcter result. First impressions from the result set suggest that the overlap is not one, which means there are unique true positives only detected by one of both algorithms. In future work the overlap of the Effingo algorithm and the state-of-the-art syntactic algorithm needs to be examined and a method combining both has to be developed thus fully exploiting the advantages of both approaches with the aim to create better results than one single algorithm alone.

The goal is to create a system which is capable of aggregating questions from all over the Web, thus providing users with an overview on a question and its answers.

Additionally, a better gold standard could facilitate differentiation between the semantic and the syntactic approaches. The dataset should contain near-duplicate pairs with fewer common words. However, the creation of such a gold standard requires a manual comparison of huge amounts of question pairs to find the sparse set of semantic duplicates.

Finally, the problem of long texts swallowing short ones still persists. This is hard to solve without the use of a knowledge base. However, it might be possible to split up contributions into paragraphs and sentences and compare them sentencewise, thus reducing the amount of text compared against each other. It would then be possible to create an aggregated score from the sentence and paragraph scores.

5.2 Other Applications

The following section presents additional applications which could be built on top of the Effingo system. Each application is motivated by a different scenario shown in Table 5.3, and each application is important for a different set of stakeholders. The following section introduces the stakeholders and then presents six possible applications. They are summarized in Table 5.3.

Table 5.3: Possible Effingo applications

Functionality	Beneficiary	Scenario	Application
Answer Provider	Member Moderator Operator	"Before I ask... maybe there's already an answer?"	Tool to suggest existing answers to new threads.
Subforum Advisor	Member Moderator Operator	"Should post be into this forum?"	Tool to detect content in wrong forum.
Thread Recommender	Member	"Nice thread. Any others like it around?"	Tool to show related threads.
Expert Assistant	Expert Moderator Operator	"Which thread is most urgent to answer?"	Tool to detect most pressing questions.
Spam Detector	Member Moderator Operator	"Are there spam threads to delete?"	Tool to detect spam threads and show to moderators.

5.2.1 Stakeholders

Many people produce and consume content on Web forums. Others care for the correct operation and earn money with running forums. Each of them takes up one of the four roles presented by the following list:

Member

Forum members are typical registered users who pose questions to the forum community. The average member typically provides no answers. They found the forum via Google or a link on some support page, pose their question, read the answer and probably never return.

Expert

Experts are forum members who tend to answer many questions. They usually have some deeper understanding of the topics discussed in the forum and can identify topic duplicates with their inherent knowledge. They are also a valuable asset for answering new and complicated questions. Therefore, they should be shielded from trivial questions or questions already answered several times before.

Moderator

Moderators are people responsible for keeping the quality of forum discussions. They can close completed or hijacked topics, delete unwanted contributions and move contributions and threads to the correct place. In some forums, experts get the right to carry out this job. In other, mostly commercial forums, paid workers moderate the forum content.

Operator

Operators are people or companies running a forum. They own the forum, provide the hard- and software needed, pay the moderators and sometimes even the experts, and are responsible with regard to legal issues.

5.2.2 Applications

The Effingo system, as described in Section 1.3.1, is applicable to realize different systems. Those applications are presented in the following paragraphs. They are intended as prospect for further development of the system.

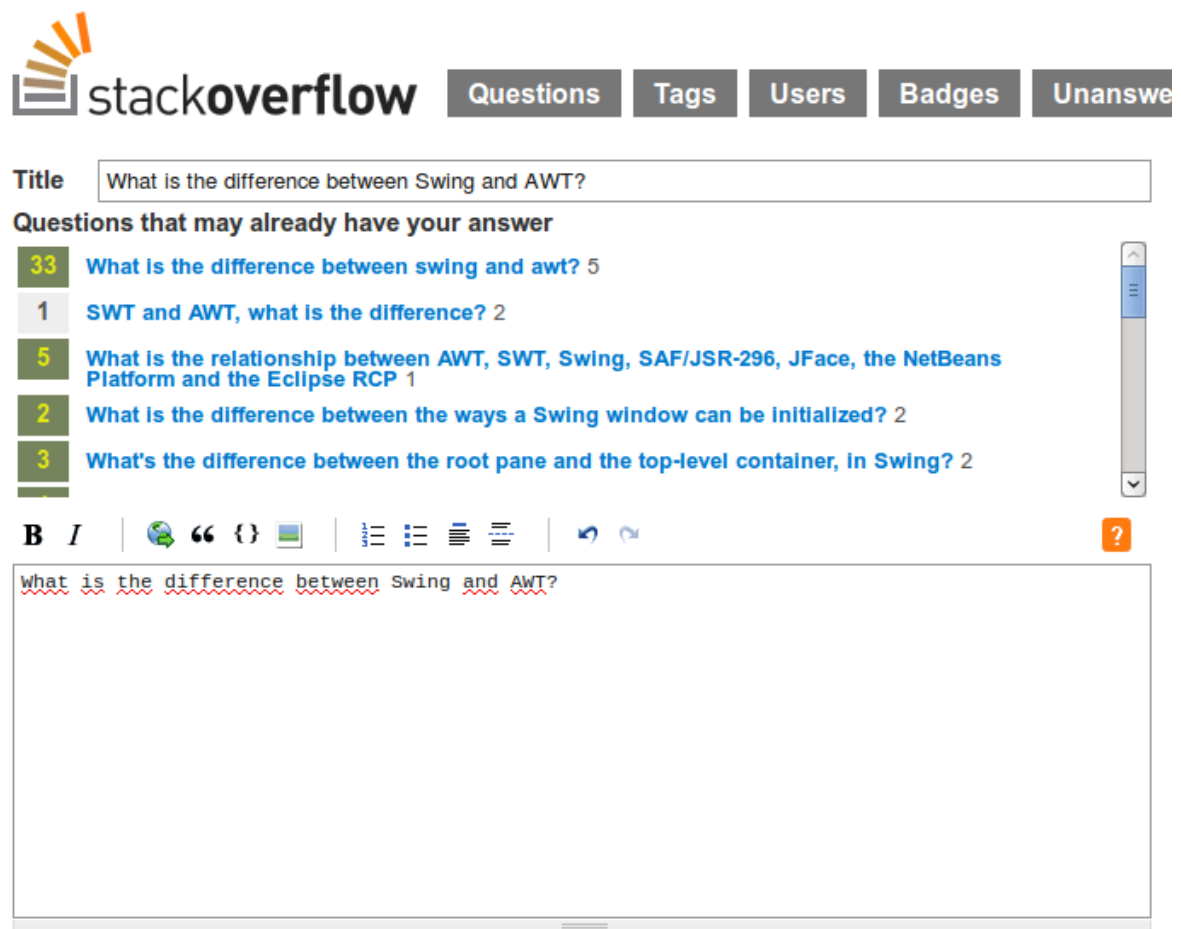
Answer Provider

When people use a forum they usually search answers, opinions or want to read about some event or topic. If they cannot find threads covering their intention they start creating new ones. In order to find threads they use either the forum's search engine or a global one. Such search engines usually take a list of key words and match that list against an inverted index. All threads containing these keywords are shown as results ordered by the number of keywords they contain. Threads matching the searchers question but not containing these keywords are not shown. Thus, if someone used a different vocabulary to describe the same problem, that search engine can not provide the answer.

In contrast to the keyword-based query formulation provided by search engines, users provide much more information upon writing a new question. During this step they describe their problem in whole sentences in a way other people can understand. Therefore, a tool could be created that analyses the new question before it is converted to a new thread and propose existing questions on the topic. If the users find a solution they do not need to create a new thread with their question and thus, a new near-duplicate has been avoided. If they find no solution they can still create a new thread from their already entered question. Similar tools are already used on <https://launchpad.net/>, <http://digg.com/> and <http://stackexchange.com>. An example for the Stackexchange Network is shown in Figure 5.3.

Subforum Advisor

Many forums, such as <https://forums.oracle.com/forums> contain a large list of subforums for several topic areas. This is helpful to structure the information but often also confuses new users. Since they do not know where to post their question they just post where it seems appropriate, often in the wrong place. It is the moderators' job to move such threads to the correct place. Currently this is hard manual work. One could imagine a tool that calculates the similarity between all threads and tells moderators which threads might be



The screenshot shows the Stack Overflow interface. At the top left is the Stack Overflow logo. To its right are navigation buttons: Questions, Tags, Users, Badges, and Unanswered. Below this is a search bar with the text "What is the difference between Swing and AWT?". Underneath the search bar is a section titled "Questions that may already have your answer". This section contains a list of five questions, each with a green box indicating the number of answers and a blue link to the question. The questions are: 1. "What is the difference between swing and awt? 5", 2. "SWT and AWT, what is the difference? 2", 3. "What is the relationship between AWT, SWT, Swing, SAF/JSR-296, JFace, the NetBeans Platform and the Eclipse RCP 1", 4. "What is the difference between the ways a Swing window can be initialized? 2", and 5. "What's the difference between the root pane and the top-level container, in Swing? 2". Below the list of questions is a rich text editor for writing an answer. The editor has a toolbar with icons for bold, italic, link, quote, code, image, list, and table. The text in the editor is "What is the difference between Swing and AWT?".

Figure 5.3: Example for an Answer Provider on the Stackoverflow Forum

in the wrong place or a tool that tells users which subforum their thread belongs to. If its precision is high enough, it might even insert threads automatically to the correct subforum. Extrapolating from this idea, it would be possible to generate a list of subforums dynamically from the user-generated content. This list could evolve and thus represent the users' understanding of the forum's content in contrast to the limited understanding of the operators or moderators.

Thread Recommender

Often, a user finds an interesting discussion and wants to know what else is discussed about this topic. Some forums, such as <http://social.msdn.microsoft.com/Forums/>, or those run on the phpBB forum software provide a basic implementation of this feature. Both implementations use keyword queries¹ to retrieve threads containing similar words and show them as similar thread proposals to the user.

Expert Assistant

A forum might contain questions that repeatedly occur but never receive an answer. Such questions might be too complex or not of interest to the people knowing the answer. Usually it is, however, in the interest of the forum community to solve such pressing questions. This increases the community of happy forum users who might provide answers in the future. It also extends the forum's shared knowledge.

However, experts should not waste time answering questions that were already answered in other threads. By detecting and grouping similar questions it is easier to find groups which never received an answer and those already having multiple answers in another thread. Thus, it is easy to point expert users to groups of the first kind and keep them away from groups of the second kind.

Spam Detector

Sometimes users post their questions multiple times to heighten their visibility. It is usually possible to find such duplicates with existing techniques, but it becomes more complicated if one such near-duplicate thread actually gets an answer or is modified by its creator. In this case, the changed thread is not equal to the other near-duplicates anymore and thus is harder to find. Large forums have moderator teams to remove such posts, but they might miss occurrences, they are expensive and therefore better employed for more important work and might not be available for smaller forums. By detecting near-duplicate questions automatically one can move them together and possibly even send a warning to the creator to discourage such behavior. Alternatively, the system could send a warning to a moderator, who can delete the spam threads and warn the spamming user to prevent him from

¹https://www.phpbb.com/customise/db/mod/precise_similar_topics_ii/

continuing the spamming.

5.3 Summary

This chapter shows several possible applications for the concepts presented in this thesis. Some of those applications already exist in rudimentary forms on today's web forums. All of them can be realized based on the approach presented in Section 1.3.1.

This shows that the algorithms and methods presented in the previous chapters are of practical relevance.

6 Conclusion

To this point the thesis has developed the foundations for a question answering system based on Web forum discussions.

This chapter starts with a brief summary of the most important results of the work. It provides an overview about the thesis's contribution to the research community. Thereafter, it explains further steps to build on the thesis and expand its results to an integrated working solution.

The three main contributions this thesis provides to the research community are:

- An approach to find user discussions on the Web and to extract the relevant data for automatic question answering from these discussions automatically;
- A classifier to separate user contributions into questions, answers and other contributions;
- A large dataset of manually annotated contributions to measure and evaluate the classifier and to be used as baseline for future works, with a similar approach.

6.1 Summary

This thesis presented an approach to **collect user contributions from Web forums** and to **classify these contributions into questions, answers and other contributions**.

Collecting contributions and classifying those contributions are the first two steps necessary to realize a question/answer system based on questions and corresponding answers explicitly formulated by human beings for human beings. Such a system, called Effingo, was presented in the introductory chapter in Section 1.3.1. Effingo is a proposal for a system to help manage and reduce the information overload a user is presented with when searching for answers on the World Wide Web, using online discussions such as those prevalent in Web forums as a knowledge base.

Research Theses

Two research thesis have been derived from the description of the Effingo system.

Thesis 1 The first states that user contributions all over the Web follow a similar structure and thus, it is possible to crawl and extract them automatically. While we were using different kinds of Web forums it became clear that this statement is at least true with regard to

visual evidence. The look and feel of most forums is similar and a human user who is capable of using one forum usually feels at home in another one pretty fast. However, automatic systems have a much harder time finding these structural similarities. Thus, nowadays data is usually collected by handcrafted wrappers. These wrappers require maintenance. First, each time a Web page changes the wrapper must adapt and second, adding new Web pages requires a partly or complete rewrite of the wrapper. Both tasks are tedious, unexciting and expensive manual work. Improvements are possible by applying semi-supervised approaches. The approach presented in Chapter 3 is intended to go one step further and examine the possibilities of collecting forum data fully automatically. Two research questions are asked to guide our research in that area. The first asks how forum content can be identified among all the content on the Web. The second asks for a common schema among forum posts from different Web forums. With answers to both questions it is possible to crawl for forum posts and to extract those posts to a single database.

Thesis 2 Thesis two asserts that user contributions contain questions, answers and other content and that it is possible to separate all three from each other. Only by identifying questions as opposed to answers are we able to build a question/answer system based on user formulated questions and answers. A supervised classifier ordering forum contributions into these three types was presented in Chapter 4.

The Main Difference to Existing Question Answering

Existing question/answer systems usually try to formulate an answer based on a query entered by a user and the facts it knows about this query. That way, so called factual questions are easy to answer. It is however still not possible to answer problem oriented questions automatically by means of a knowledge base with a collection of facts. Since problem oriented questions require longer explanations, a human is necessary to formulate the answers. The advantage of Effingo is the usage of the readily available questions and answers formulated by forum users. Thus, the Effingo system does not need to understand the question and create an answer from the facts it knows. It only needs to find a question matching the user's query question and show the answer belonging to the question. An approach on how to match such topical near-duplicate questions was presented in Chapter 5. Since problem oriented questions that exceed simple fact questions are hard to answer by an automatic system, this approach extends the possible range of questions computer systems can answer with support of the crowd. A crowd-based question/answer system, such as Effingo, can have a much higher topic coverage than a traditional system working on a fact knowledge base. Chapter 5 also provided an overview of possible applications with regard to this concept.

6.1.1 Focused Forum Crawling and Extraction

Our research in Chapter 3 rendered answers to research questions one and two. It provides a feature set to identify Web forums among other Web pages and even offered a second feature set to identify the content pages from among those pages belonging to a Web forum. **Our main research contribution** for the first two steps are the feature sets developed and evaluated to create successful classifiers.

It also answered research question two by a detailed analysis of different Web forums and demonstrated that the common schema consists of only three entities. Finally, an approach to extract those entities fully automatically was discussed. The workflow regarding the retrieval of relevant forum pages on the Web to extracted data was structured into three distinct steps.

Step 1: Focused Forum Crawler

The focused forum crawler and extractor presented in this thesis was structured into three parts. At first we examined crawling and developed a classifier for forum pages. This crawler works like a normal Web crawler but includes a filter for non-forum pages. This filter only passes forum pages to the crawler's index and rejects all pages that are classified as non-forum pages.

The crawler was examined on a dataset of pages from diverse forums, mixed with randomly chosen non-forum pages. Overall, the best classifier—Random Forest—achieved an accuracy of 95%.

Step 2: Forum Page Classifier

There are five types of Web forum pages. During the analyses in Chapter 3 we found that only post-of-thread pages contain valuable information and thus created a second classifier for identifying the post-of-thread pages. Post-of-thread pages are those pages containing the actual contributions. The forum page classifier was evaluated on a selection of pages created by different forum systems. Our classifier only achieved an F_1 score of 65% for classifying these forum pages into all five identified page classes. However, the classifier was able to distinguish post-of-thread pages with high accuracy.

Simplifying Steps 1 and 2 in Future Work

In a not yet developed advanced approach we would combine the focused crawler and the page classifier to one step. This would result in a crawler with a filter passing only post-of-thread pages to the index and rejecting all other pages, even if they are from a Web forum. On the one hand, such a classifier needs to find a much smaller set on a much wider domain (post-of-thread pages among all Web pages), so classification accuracy could be expected to

drop. On the other hand, the classifier only needs to find differences between two classes (post-of-thread vs. non post-of-thread), which might again increase accuracy.

The existing classifiers for forum pages and entities could be further analyzed by means of similar approaches as presented for the contribution type classifier in Chapter 4. Learning curves and feature selection could show whether it is necessary to remove any features or to add new ones to improve their performance.

Step 3: Post-Of-Thread Page Information Extraction

The final step was the extraction step. This step takes a set of post-of-thread pages as input and outputs the raw data extracted from those pages. The first step to a solution for such an algorithm was to answer research question two and find a *common schema* for Web forum posts. It turned out that the only constant fields for all Web forum posts are the *publication date*, the *author's username* and the *body text*. Consequently, the extraction approach focused on these three entities. Our algorithm's goal was to find extraction patterns for entities occurring in all posts from the same forum and to classify these extraction patterns in order to find out which pattern points to which of the target entities. The approach we presented in Sections 3.9, 3.10 and 3.11, requires a certain amount of example pages per forum to learn the extraction patterns from. Since it is possible to create such a set of example pages automatically, the system works fully automatically. The simplest approach for the collection of example pages is to wait until the crawler has added a certain amount of post-of-thread pages from the same domain. The data extractor would begin creating its extraction patterns as soon as that amount is collected. This approach assumes that only one forum is operated under each domain. Since this is not always the case, a more sophisticated approach might consider similarities in sub-domains and URL paths as well.

For the extraction of the most important part of a forum post—the body—the system achieved 92% accuracy while it still passed the 80% for the other two entity types.

An important problem addressed by the approach regards scattered threads. Scattered threads are threads that have so many posts that the forum template engine decided to split the thread into two or more pages. While many existing multi-purpose data extraction approaches, extract data from only one Web page, our approach finds the thread identifier and links scattered threads back together.

Differences to Related Work

Altogether Chapter 3 provided concepts for all the steps necessary to create a fully automatic forum post retrieval and extraction system. Our extractor is not only able to find the data entries on a Web page like existing automatic data extraction approaches, it is also able to find and assign a meaning to the top three data entities.

The chapter also provided evidence for the truth of thesis one. However, it is restricted evidence since our approaches are unable to achieve 100% accuracy. On the one hand, a fully

automatic approach based on statistical classifiers will never reach 100% accuracy. On the other hand, since the markup used for Web forums is very diverse, it is impossible to determine rules or a complete semantic description for all Web forums that would allow to extract them with absolute certainty. Thus, the statistical approach is a very reasonable compromise between engineering complexity and result quality. If the Semantic Web achieved broader coverage in the future and forum posts would be annotated with a unique semantic tag, an easier approach with higher accuracy would become possible. Until then, the current fully automatic approach can be used in addition to the existing manual approaches. In such a scenario the automatic retriever and extractor would be used as fall-back solution for pages which are unknown by the manually created wrappers.

6.1.2 Classification of Forum Contributions

In order to prove or disprove thesis two, a forum contribution classifier has been described in Chapter 4. Similar to the classifiers developed for forum post retrieval and extraction, no new classification algorithms were developed for contribution type classification.

Classification Results

The classifier achieves good results when trying to find question and answer contributions. Other contributions, however, are often wrongly classified as questions and answers. We divided these other contributions into additional classes. With regard to these additional classes it is evident that some are very close to questions and answers and thus pose a challenge to the classifier. However, in a real application a human would probably accept them as questions and answers and thus not recognize the error. With our question/answer classifier we were able to prove thesis two and answer research questions three and four. Just as for the page classifiers and entity classifier, improvement is possible. Since the classifier is a statistical approach and works on complex data not even clear to humans, it does not achieve 100% classification quality.

Baseline Comparison

We also compared our approach to the most recent similar approach from Hong & Davison (2009). As a result, Hong performs better for some special restricted cases but in general shows a worse performance. It however was able to achieve a higher F_1 score on contributions belonging to the *Other* type.

Research Contribution

Our research contribution in this area is the feature set and the careful analysis of the classifier created from this new and extended feature set and its comparison to existing approaches. This carefully crafted and explained feature set based on our observations on real

forum data. It was derived from existing work and extended with additional or adapted features. Different classifiers and feature combinations were examined and limitations of the classifier were shown by means of established and widely applied techniques from the machine learning community.

6.1.3 Dataset for Forum Contribution Types

For feature engineering as well as for the contribution type classifier's evaluation a dataset was developed as part of this thesis. The dataset consists of content from two forums stemming from the same domain. Approximately 10,000 contributions were manually and carefully annotated by a human labeler.

The dataset features question and answer labels and divides all other contributions into seven distinct labels. This results into 10,000 contributions labeled with nine different labels. The labels are the result of careful observation on approximately 1,000 benchmark contributions. These 1,000 benchmark contributions form a benchmark dataset.

Calculation of Inter Annotator Agreement

The benchmark dataset is a subset of the full dataset but was labeled a second time by a different annotator. Hence, it was possible to calculate the κ score as a measure of inter-annotator agreement. Inter-annotator agreement was very good for some labels and very bad for others. This means that humans agreed on some labels more than on others. Most important was the fact that both annotators achieved a very good agreement on the classification of labels as questions and answers. Due to the partially bad agreement, all other labels were summarized as an aggregated *Other* label.

The benchmark dataset was used as source for the features engineered for the contribution type classifier. The complete dataset was applied to calculate the final performance statistics of the developed approach. It also served as a common ground truth to compare the classifier developed in this thesis with the classifier developed by Hong & Davison (2009).

Research Contribution

The dataset will be provided for free, so that future research on forum contribution classification has a point of reference to start from. Aside from its usage in our evaluation it contains a typology for forum contributions with nine different classes and many examples per class.

6.2 Future work

The following section describes ideas we developed during our work on this thesis. They are supposed to inspire future research building on this work. Using and developing these

ideas will enable the implementation of the complete Effingo system and even extend it with additional new and helpful features to overcome the information overload many users are faced with while searching solutions on the World Wide Web.

Additional Input Sources

Today, social media is spreading from the domain of mere forums to new platforms like Twitter, Facebook, Google+ and similar applications. Such platforms do not use threads but activity streams, which nevertheless are similar to forum threads. They also form a discourse of multiple authors on some topic. The difference is that they are not limited to one discourse but may contain multiple mixed discourses and run forever. So even though they might contain questions and answers as well, finding them requires an additional step of separating the intermingled discussions. This is similar to the Story Segmentation task proposed by Allan (2002) for news transcripts.

Even more promising as a source for community-contributed questions and answers are question/answer sites like those operated by the StackExchange Network. In the course of this work we handled them like ordinary forums. That way, we were able to use the knowledge stored inside. However, question/answer sites contain a richer amount of meta information about questions and answers. They provide points for questions and answers as well as for each user. Items with more points are rated as more valuable or as being of a better quality than items with fewer points. Since they support a wiki style, it is possible for each user to edit and extend existing questions and answers so that the quality of the knowledge base increases over time. There also are explicit links between topical near-duplicates, reducing the necessity to detect such anomalies automatically. Even a semi-automatic approach would be possible in this case, highlighting possible topical near-duplicates and waiting for a critical mass of users to acknowledge the near-duplicate before grouping both questions together. A question/answer classifier has no high relevance for question/answer sites, since questions and answers are marked explicitly. This fact qualifies the data on question/answer sites as an ideal candidate for training such a classifier and use it on unlabeled data sources such as ordinary Web forums. In addition, the different ranking schemes for questions, answers and users could be used to create an automatic ranking for a hypothetical question/answer system using such platforms as knowledge base.

Improving Community Based Question Answering with User Feedback and Active Learning

In a production environment constant maintenance is necessary for the classifier to adapt to changing user behavior. This is only possible via user feedback. The problem with user feedback is that users usually do not like or do not care to provide information about their contribution. If they asked a question they expect an answer while they do not want to tell the system that it was a question. If they provide an answer they usually also do not want

to tell the system that it is an answer. However, this information would be very helpful to improve the classifier and adapt it to changing requirements. One possibility to reduce the amount of required user feedback is called *Active Learning*. By means of Active Learning users are only asked for feedback if knowledge about their contribution improved the classifier significantly. This would mostly be the case for borderline cases where the classifier is unsure about the classification. Feedback could be asked for via direct feedback. This means the users would get some kind of popup asking them for the correct type to use. A less invasive way would be to use indirect feedback and add the process of creating a question or an answer directly to the workflow of creating a contribution. This is similar to what the question/answer sites are doing. In this case, the user would click explicitly on "create new question" or "create new answer". Although this is the safer way for actually getting feedback, the users will require reimbursement for their effort. This reimbursement usually comes in the form of great user experience or fast and very helpful answers. In this case, the information concerning the advantage must be available to the users just the same moment they create the question or answer. One will possibly face a cold start problem with such a system. Early users might not see the benefit if the knowledge base is still empty and thus provide no content. A solution might be to use the content from existing question/answer sites as bootstrap dataset.

A Complete Architecture for Effingo

The Effingo system as envisioned in Section 1.3.1 requires some additional steps which are not discussed in this thesis. A complete implementation could follow an architecture as proposed in Figure 6.1. As can be seen not only forums are considered as input, but also question/answer sites and FAQs. FAQs are a very valuable knowledge base since they are authored by experts and usually contain explicitly marked questions and answers. Unfortunately, they do not share the structure of a typical forum post. They usually have no date or author and contain a large collection of question and answer pairs on the same page. Different retrievers and extractors are required for this type of input source. The component *Content Discovery* covers the first two steps from Chapter 3 and delivers retrieved forum pages to a raw data storage. *Content Processing* covers the extractor from Chapter 3 as well as the classifier from Chapter 4. Some threads contain multiple questions and answers per thread. In such cases it is important to have a component that links each answer to its question. The *Question-Answer Linker* component would solve this problem. It was already mentioned briefly in Section 1.3.1 as a part of the "GROUP Answers BY Question Group" step. Extracted question/answer pairs are stored to a central knowledge base. The knowledge base is used by a *Question Aggregator* which can build on the topical near-duplicate detection presented in Section 5.1 and an *Answer Summarizer* which creates a summary from all the answers available for one grouped question. Topical near-duplicate detection was not completely solved by the approach presented in Section 5.1. However, we found evidence

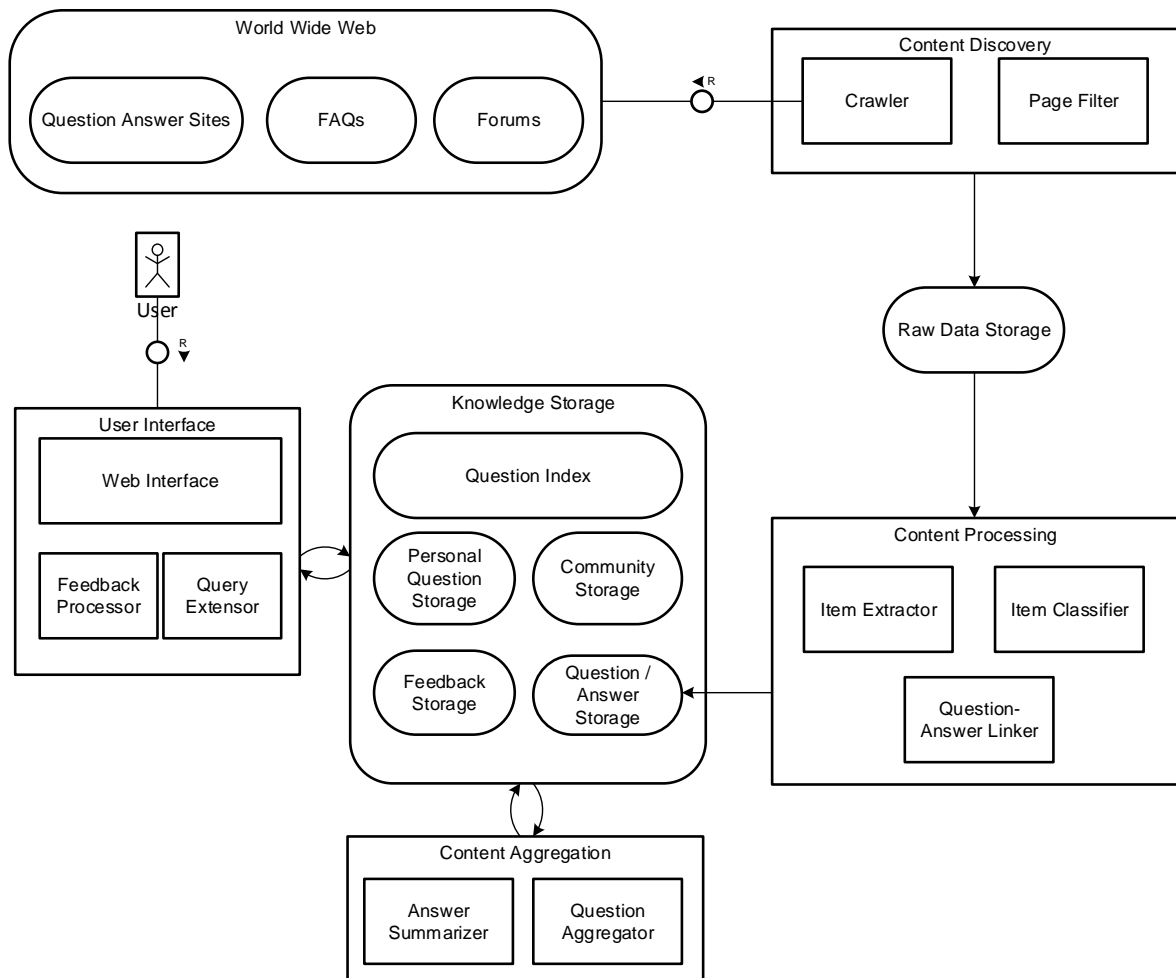


Figure 6.1: A hypothetical architecture for a complete implementation of the Effingo system.

that approaches using semantic information might be unnecessarily complex. Further investigations in this area should examine the possibilities of syntactic topical near-duplicate detection and of combinations of both. The *User Interface* presents the user with a search interface to ask questions and get available answers. In order to get the correct question it might be necessary to extend the user's query with additional terms, such as synonyms and related entities. This can be handled by the *Query Extensor*. It is also possible to include some of the applications proposed in Chapter 5 into this interface. User feedback can be processed using a *Feedback Processor*.

Automatic Discourse Systems

An interesting extension to the proposed Effingo system could build on recent research on automatic discourse management. Such a system does not simply return an answer to a question, but tries to initiate a dialogue with the user. An example for a first partial implementation is the Siri system from Apple. For unclear or ambiguous concepts it asks the user for further information and is thus able to narrow down the set of possible answers, even telling the user with high confidence that no answer is available yet. In such a case the users could pose their questions with high confidence that they are not creating a topical near-duplicate.

Thread Development Analysis

A user who really needs to ask a question because it was never answered before could be supported by an analysis of existing threads and a feedback as to how long they can expect to wait until an answer arrives. It would even be possible to provide recommendations on how to alter a question in order to improve the chance of receiving an answer. Algorithms for this task can be based on time line and development analyses on existing threads. The most obvious feature would be the time between posts. Additionally, the classifier developed in Chapter 4 can identify the type for each post. That way it is possible to predict which type of post to expect next and how long it may take until it appears.

Type and time might not be enough information. We could again analyze the contribution itself and rank it according to its quality. An approach to this was shown by Agichtein et al. (2008). High quality contributions are more likely to attract a quick high quality answer. Such a system could help to facilitate the usage of forums and explain well known pitfalls to new users. In addition, a filter for low quality content might be useful to improve all algorithms working on forum data.

Handling Obtrusive Content

A final note on contribution content concerns obtrusive content which is not comparable to text. Examples for such content are code snippets, signatures, and citations from previous

contributions. Often, such content is marked by HTML tags, other times it is not and even if it is, the markup usually changes from forum to forum. All approaches presented in this thesis ignored such content and handled it just like normal text. Filtering such content might improve algorithm performance. Further experiments are necessary to prove or disprove this idea.

A Dataset Details

Datasets used for Forum Data Extraction

Table A.1: Web forums used for the SiteLevel-FO dataset.

Forum	Software
avsforum.com	vBulletin
boards.cruisecritic.com	
codeguru.com	
computerforum.com	
devhardware.com	
disboards.com	
flyertalk.com	
forum.gsmhosting.com	
photography-on-the.net	
forums.photographyreview.com	
ubuntuforums.org	
howardforums.com	
pctools.com	
phpbuilder.com	
sitepoint.com	
forums.d2jsp.org	custom
bbs.imobile.com.cn	
forums.asp.net	
www.dpreview.com/forums	
bbs.cqzg.cn	Discuz!

Table A.2: Web forums used for the Fodex1-FO dataset.

Forum	Software
dslr-forum.de	vBulletin
forums.steampowered.com	
area51.phpbb.com	phpBB3
tt-forum.co.uk	
forums.gentoo.org	phpBB2
forums.theonering.com	
forums.sdn.sap.com	Jive Clearspace
forums.oracle.com	
forums.syfy.com	Invision (IPB)
community.invisionpower.com	

Table A.3: Web forums used for the Fodex2-FO dataset.

Forum	Software
creditboards.com	Invision
diskusjon.no	
neowin.net	
www.bzpower.com	
forums.xkcd.com	phpBB3
forums.mozillazine.org	
www.dmtabs.com	
car-pc.info	phpBB2
forum.joomla.org	
onlinebanking-forum.de	

Table A.4: Web forums used for the Fodex2-QA dataset.

Forum	Software
de.answers.yahoo.com	custom
stackoverflow.com	
social.msdn.microsoft.com	
ask.sqlservercentral.com	
getsatisfaction.com	
discussions.apple.com	
gutefrage.de	
lockergnome.net	QSQA
community.jivesoftware.com	Jive Social Business Platform
frageee.de	Question2Answer

B Bibliography

- Agichtein, E., Castillo, C., Donato, D., Gionis, A. & Mishne, G. (2008), Finding high-quality content in social media, *in* 'International conference on Web search and web data mining', ACM, pp. 183–194.
- Allam, A. M. N. & Haggag, M. H. (2012), 'The Question Answering Systems: A Survey', *International Journal of Research and Reviews in Information Sciences* 2(3).
- Allan, J. (2002), Introduction to topic detection and tracking, *in* 'Topic detection and tracking', Kluwer Academic Publishers, pp. 1–16.
- Allan, J., Carbonell, J., Doddington, G., Yamron, J. P. & Yang, Y. (1998), Topic detection and tracking pilot study: Final report, *in* 'Broadcast news transcription and understanding workshop', Vol. 1998, Citeseer, pp. 194–218.
- Allan, J., Lavrenko, V. & Swan, R. (2002), Explorations within topic tracking and detection, *in* 'Topic detection and tracking', Kluwer Academic Publishers, pp. 197–224.
- Amit, Y. & Geman, D. (1997), 'Shape Quantization and Recognition with Randomized Trees', *Neural Computation* 9(7), 1545–1588.
- Apache Lucene* (2013).
URL: <https://lucene.apache.org>
- Arasu, A. & Garcia-Molina, H. (2003), Extracting structured data from Web pages, *in* 'International conference on Management of data', ACM, pp. 337–348.
- Austin, J. L. (1962), *How to do Things with Words*.
- Baker, C. F., Fillmore, C. J. & Lowe, J. B. (1998), The berkeley framenet project, *in* 'International conference on Computational linguistics', Association for Computational Linguistics, pp. 86–90.
- Banerjee, P. & Han, H. (2007), Drexel at TREC 2007: Question answering, *in* 'Text Retrieval Conference'.
- Bayes, T. (1763), 'An Essay Towards Solving a Problem in the Doctrine of Chances', *Philosophical Transactions of the Royal Society of London* pp. 370–418.

- Benamara, F. & Saint Dizier, P. (2004), Advanced relaxation for cooperative question answering, in 'New Directions in Question Answering', Citeseer, pp. 263–274.
- Bengel, J., Gauch, S., Mittur, E. & Rajan, V. (2004), Chattrack: Chat room topic detection using classification, in 'Intelligence and Security', Springer, pp. 266–277.
- Berners-Lee, T. & Mark, F. (2000), *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*, HarperInformation.
- Beyer, M. (2011), Fodex2 - Automatische Typisierung von Forenseiten, Study thesis, Technical University Dresden.
- Bian, J., Liu, Y., Agichtein, E. & Zha, H. (2008), Finding the right facts in the crowd: factoid question answering over social media, in 'International conference on World Wide Web', ACM, pp. 467–476.
- Breiman, L. (1996), 'Bagging predictors', *Machine learning* **140**, 123–140.
- Breiman, L. (2001), 'Random Forests', *Machine Learning* **45**(1), 5–32.
- Buscaldi, D., Rosso, P., Gómez-Soriano, J. M. & Sanchis, E. (2010), 'Answering questions with an n-gram based passage retrieval engine', *Journal of Intelligent Information Systems* **34**(2), 113–134.
- Cai, R., Yang, J.-M., Lai, W., Wang, Y. & Zhang, L. (2008), iRobot: An intelligent crawler for Web forums, in 'International conference on World Wide Web', ACM, pp. 447–456.
- Carvalho, V. R. & Cohen, W. W. (2006), Improving email speech acts analysis via n-gram selection, in 'Workshop on Analyzing Conversations in Text and Speech', Association for Computational Linguistics, pp. 35–41.
- Chang, C.-H., Kaye, M., Girgis, R. & Shaalan, K. F. (2006), 'A Survey of Web Information Extraction Systems', *IEEE Transactions on Knowledge and Data Engineering* **18**(10), 1411–1428.
- Chen, G. & Choi, B. (2008), Web page genre classification, in 'Symposium on Applied computing', ACM, pp. 2353–2357.
- Chen, H.-H. & Ku, L.-W. (2002), An NLP & IR approach to topic detection, in 'Topic detection and tracking', Kluwer Academic Publishers, pp. 243–264.
- Cohen, W. W. (1995), Fast effective rule induction, in 'ICML', pp. 115–123.
- Cohen, W. W., Carvalho, V. R. & Mitchell, T. M. (2004), Learning to classify email into "speech acts", in 'EMNLP', Vol. 4, pp. 309–316.

- Cong, G., Wang, L., Lin, C.-Y., Song, Y.-I. & Sun, Y. (2008), Finding question-answer pairs from online forums, in 'International conference on Research and development in information retrieval', ACM, pp. 467–474.
- Crescenzi, V., Mecca, G. & Merialdo, P. (2001), Roadrunner: Towards automatic data extraction from large web sites, in 'VLDB', Vol. 1, pp. 109–118.
- Dalmas, T., Leidner, J., Webber, B., Grover, C. & Bos, J. (2004), 'Annotating CBC4Kids: A Corpus for Reading Comprehension and Question Answering Evaluation', *Institute for Communicating and Collaborative Systems* .
- Ding, S., Cong, G., Lin, C.-Y. & Zhu, X. (2008), Using conditional random fields to extract contexts and answers of questions from online forums, in 'ACL-08: HLT', number June, Association for Computational Linguistics, pp. 710–718.
- Drescher, R. (2010), Entwicklung eines fokussierten Crawlers für Internetforen, Diplomathesis, Technical University Dresden.
- Duong, L. T., Hall, M. R., Mayfield, J. C., McNamee, P. J. & Piatko, C. T. (2002), 'Directed web crawler with machine learning'.
- Eichmann, D. & Srinivasan, P. (2002), A cluster-based approach to broadcast news, in 'Topic detection and tracking', Kluwer Academic Publishers, pp. 149–174.
- Fayyad, U. M. & Irani, K. B. (1993), 'Multi-interval discretization of continuous-valued attributes for classification learning'.
- Fetterly, D. C. & Chien, S. S.-T. (2007), 'Identifying a web page as belonging to a blog'.
- Fleiss, J. L. (1981), *Statistical methods for rates and proportions*, John Wiley.
- Flint, L. (1917), *Newspaper writing in high schools, containing an outline for the use of teachers*, Department of journalism press in the University of Kansas.
- Forman, G. (2003), 'An extensive empirical study of feature selection metrics for text classification', *The Journal of Machine Learning Research* **3**, 1289–1305.
- Franz, M., McCarley, J. S., Ward, T. & Zhu, W.-J. (1999), 'Segmentation and detection at IBM: hybrid statistical models and two-tiered clustering', *Topic detection and tracking* .
- Freund, Y. & Schapire, R. E. (1999), 'Large margin classification using the perceptron algorithm', *Machine learning* **37**(3), 277–296.
- Fürnkranz, J. & Widmer, G. (1994), Incremental reduced error pruning, in 'International Conference on Machine Learning', Morgan kaufmann, New Brunswick, New Jersey, pp. 70–77.

- Grishman, R. & Sundheim, B. (1996), 'Message understanding conference-6: A brief history', *COLING* **96**, 466–471.
- Hensel, C. (2013), Themenbasierte Aggregation von Fragen in Sozialen Medien, Diploma thesis, Technical University Dresden.
- Ho, T. K. (1998), 'The random subspace method for constructing decision forests', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(8), 832—844.
- Hong, L. & Davison, B. (2009), A classification-based approach to question answering in discussion boards, in 'International conference on Research and development in information retrieval', ACM, pp. 171–178.
- Hsu, C.-W., Chang, C.-C. & Lin, C.-J. (2003), 'A practical guide to support vector classification'.
- Iba, W. & Langley, P. (1992), Induction of one-level decision trees, in 'The Ninth International Conference on Machine Learning', Vol. ML92: Proc, Citeseer, pp. 233–240.
- Jaccard, P. (1901), *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*, Vol. 37.
- Jeon, J., Croft, W. B., Lee, J. H. & Park, S. (2006), A framework to predict the quality of answers with non-textual features, in 'International conference on Research and development in information retrieval', ACM, pp. 228–235.
- Jijkoun, V. & De Rijke, M. (2004), 'Answer selection in a multi-stream open domain question answering system'.
- Kayed, M. & Chang, C.-H. (2010), 'FiVaTech: Page-level web data extraction from template pages', *IEEE Transactions on Knowledge and Data Engineering* **22**(2), 249–263.
- Kim, J., Chern, G., Feng, D., Shaw, E. & Hovy, E. (2006), Mining and assessing discussions on the web through speech act analysis, in 'Workshop on Web Content Mining with Human Language Technologies at the 5th International Semantic Web Conference', Citeseer.
- Kingsbury, P., Palmer, M. & Marcus, M. (2002), Adding semantic annotation to the penn treebank, in 'Proceedings of the Human Language Technology Conference', number 36, Citeseer, pp. 252–256.
URL: ftp://ftp.cis.upenn.edu/pub/ace/public_html/HLT2002-propbank.pdf
- Ko, J., Si, L. & Nyberg, E. (2007), A probabilistic framework for answer selection in question answering, in 'Proceedings of NAACL-HLT', pp. 524–531.
- Kullback, S. & Leibler, R. A. (1951), 'On information and sufficiency', *The Annals of Mathematical Statistics* **22**(1), 79–86.

- Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S. & Teixeira, J. S. (2002), 'A brief survey of web data extraction tools', *ACM SIGMOD Record* **31**(2), 84—93.
- Landis, R. J. & Koch, G. G. (1977), 'The measurement of observer agreement for categorical data.', *Biometrics* **33**(1), 159–174.
- Leacock, C. & Chodorow, M. (1998), 'Combining local context and WordNet similarity for word sense identification', *WordNet: An electronic lexical database* **49**(2), 265–283.
- Leek, T., Schwartz, R. & Sista, S. (2002), Probabilistic approaches to topic detection and tracking, in 'Topic detection and tracking', Kluwer Academic Publishers, pp. 67–83.
- Levow, G. A. & Oard, D. W. (2002), Signal boosting for translingual topic tracking: Document expansion and n-best translation, in 'Topic detection and tracking', Kluwer Academic Publishers, pp. 175–195.
- Light, M., Mann, G. S., Riloff, E. & Breck, E. (2001), 'Analyses for elucidating current question answering technology', *Natural Language Engineering* **7**(04), 325–342.
- Lin, C., Yang, J.-M., Cai, R., Wang, X.-J., Wang, W. & Zhang, L. (2009), Simultaneously modeling semantics and structure of threaded discussions: a sparse coding approach and its applications, in 'Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 131–138.
- Lin, D. (1998), An information-theoretic definition of similarity, in 'Proceedings of the 15th international conference on machine learning', Morgan Kaufmann Publishers Inc., pp. 296–304.
- Lo, L., Ng, V. T.-Y., Ng, P. & Chan, S. C. (2006), Automatic Template Detection for Structured Web Pages, in '10th International Conference on Computer Supported Cooperative Work in Design', IEEE Computer Society, pp. 1–6.
- Manning, C. D., Raghavan, P. & Schütze, H. (2009), *An introduction to information retrieval*, number c, Cambridge University Press.
- McLachlan, G. J., Do, K.-A. & Ambrose, C. (2005), *Analyzing microarray gene expression data*, Wiley-Interscience.
- Meng, X., Hu, D. & Li, C. (2003), Schema-guided wrapper maintenance for web-data extraction, in 'Proceedings of the fifth ACM international workshop on Web information and data management', ACM, pp. 1–8.
- Miller, G. A. (1995), 'WordNet: a lexical database for English', *Communications of the ACM* **38**(11), 39–41.
- Mingers, J. (1987), 'Rule induction with statistical data—a comparison with multiple regression', *Journal of the operational research Society* pp. 347–351.

- Minkov, E., Wang, R. & Cohen, W. W. (2005), Extracting personal names from emails: Applying named entity recognition to informal text., *in* 'HLT-EMNLP'05'.
- Mitchell, T. M. (1997), *Machine Learning*, McGraw-Hill.
- Muslea, I. (1999), Extraction patterns for information extraction tasks: A survey, *in* 'The AAAI-99 Workshop on Machine Learning for Information Extraction'.
- Nadeau, D. & Sekine, S. (2007), 'A survey of named entity recognition and classification', *Linguisticae Investigationes* **30**(1), 3–26.
- Narayanan, S. & Harabagiu, S. (2004), Question answering based on semantic structures, *in* 'Proceedings of the 20th international conference on Computational Linguistics', Association for Computational Linguistics, p. 693.
- NITLE Weblog Census (2003).
URL: <http://www.blogcensus.net>
- Page, L., Brin, S., Motwani, R. & Winograd, T. (1999), 'The pagerank citation ranking: Bringing order to the web', *World Wide Web Internet And Web Information Systems* .
- Pretzsch, S. (2011), Fodex - Datenextraktion aus Webforen, Study thesis, Technical University Dresden.
- Pretzsch, S. (2012), FODEX2 - Informationsextraktion aus Social Media Diskussionen, Diploma thesis, Technical University Dresden.
- Quinlan, J. R. (1979), Discovering rules by induction from large collections of examples, *in* 'Expert systems in the micro electronic age', Vol. 174, Edinburgh University Press, pp. 168–201.
- Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann.
- Ravi, S. & Kim, J. (2007), 'Profiling student interactions in threaded discussions with speech act classifiers', *Frontiers in Artificial Intelligence and Applications* **158**, 357.
- Resnik, P. (1995), Using information content to evaluate semantic similarity in a taxonomy, *in* 'Proceedings of the 14th international joint conference on Artificial intelligence-Volume 1', Vol. 1, Morgan Kaufmann, pp. 448–453.
- Rinaldi, F., Dowdall, J., Schneider, G. & Persidis, A. (2004), Answering questions in the genomics domain, *in* 'Proceedings of the ACL 2004 Workshop on Question Answering in Restricted Domains', pp. 46–53.
- Rinaldi, F., Hess, M., Mollá, D., Schwitler, R., Dowdall, J., Schneider, G. & Fournier, R. (2002), Answer extraction in technical domains, *in* 'Computational Linguistics and Intelligent Text Processing', Springer, pp. 360–369.

- Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* **65**(6), 386–408.
- Rosenblatt, F. (1961), Principles of neurodynamics. perceptrons and the theory of brain mechanisms, Technical report, DTIC Document.
- Rudolph, M. (2011), Automatische Erkennung von Internetforen mittels Maschinellen Lernens, Study thesis, Technical University Dresden.
- Schneider, G., Aliod, D. M. & Hess, M. (1999), Inkrementelle minimale logische Formen für die Antwortextraktion, in 'Proceedings of 4th Linguistic Colloquium', pp. 7–10.
- Schultz, M. J. & Liberman, M. Y. (2002), Towards a "Universal dictionary" for multi-language information retrieval applications, in 'Topic detection and tracking', Kluwer Academic Publishers, pp. 225–241.
- Senellart, P. & Blondel, V. D. (2008), Automatic Discovery of SimilarWords, in 'Survey of Text Mining II', Springer, pp. 25–44.
- Tang, X. (2008), Approach to detection of community's consensus and interest, in 'Advanced Web and NetworkTechnologies, and Applications', Springer, pp. 17–29.
- Tunstall-Pedoe, W. (2012), 'TrueKnowledge'.
URL: <http://www.trueknowledge.com/>
- Urbansky, D., Reichert, S., Muthmann, K., Schuster, D. & Schill, A. (2011), An Optimized Web Feed Aggregation Approach for Generic Feed Types, in 'Proceedings of the Fifth International Conference on Weblogs and Social Media', pp. 638–641.
- Vaughan, L. (2004), 'New measurements for search engine evaluation proposed and tested', *Information Processing & Management* **40**(4), 677–691.
- Wanas, N., El-Saban, M., Ashour, H. & Ammar, W. (2008), Automatic scoring of online discussion posts, in 'Proceeding of the 2nd ACM Workshop on information Credibility on the Web', ACM, pp. 19–26.
- Wang, H.-C., Huang, T.-H., Guo, J.-L. & Li, S.-C. (2009), Journal Article Topic Detection Based on Semantic Features, in 'Next-Generation Applied Intelligence', Springer, pp. 644–652.
- Wolfram | Alpha: Computational Knowledge Engine (2012).
URL: <http://www.wolframalpha.com/>
- Wu, Z.-L. & Li, C.-h. (2007), Topic Detection in Online Discussion Using Non-negative Matrix Factorization, in 'Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops', IEEE, pp. 272–275.

- Wu, Z. & Palmer, M. (1994), Verbs semantics and lexical selection, *in* 'Proceedings of the 32nd annual meeting on Association for Computational Linguistics', Association for Computational Linguistics, pp. 133–138.
- Xu, G. & Ma, W.-Y. (2006), Building implicit links from content for forum search, *in* 'Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 300–307.
- Yamron, J. P., Gillick, L., van Mulbregt, P. & Knecht, S. (2002), Statistical Models of Topical Content, *in* 'Topic detection and tracking', Kluwer Academic Publishers, pp. 115–134.
- Yang, C. C. & Ng, T. D. (2008), Analyzing content development and visualizing social interactions in Web forum, *in* 'IEEE International Conference on Intelligence and Security Informatics', IEEE, pp. 25–30.
- Yang, J.-M., Cai, R., Wang, Y., Zhu, J., Zhang, L. & Ma, W.-Y. (2009), Incorporating site-level knowledge to extract structured data from web forums, *in* 'Proceedings of the 18th international conference on World wide web', ACM, pp. 181–190.
- Yiming, Y. & Pedersen, J. O. (1997), A Comparative Study on Feature Selection in Text Categorization, *in* 'ICML', Morgan Kaufmann, pp. 412–420.
- Zhai, Y. & Liu, B. (2006), 'Structured data extraction from the web based on partial tree alignment', *IEEE Transactions on Knowledge and Data Engineering* **18**(12), 1614–1628.
- Zheng, S., Song, R., Wen, J.-R. & Wu, D. (2007), Joint optimization of wrapper generation and template detection, *in* 'Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 894–902.