The Infocious Web Search Engine: Improving Web Searching through Linguistic Analysis

Alexandros Ntoulas
Infocious Inc. &
UCLA Computer Science
ntoulas@cs.ucla.edu

Gerald Chao
Infocious Inc.
gerald@infocious.com

Junghoo Cho
UCLA Computer Science
cho@cs.ucla.edu

Abstract

In this paper we present the Infocious Web search engine [23], which currently indexes more than 2 billion pages collected from the Web. The main goal of Infocious is to enhance the way that people find relevant information on the Web by resolving ambiguities present in natural language text. Towards this goal, Infocious performs linguistic analysis to the content of the Web pages prior to indexing and exploits the output of this analysis when ranking and presenting the results to the users.

Our hope is that this additional step of linguistic processing provides Infocious with two main advantages. First, Infocious tries to gain a *deeper understanding* of the content of Web pages. and to match the users' queries with the indexed documents better, improving the relevancy of the returned results. Second, based on its linguistic processing, Infocious tries to *organize* and *present* the results to the users in a structured and more intuitive way.

In this paper we present the linguistic processing technologies that we investigated and/or incorporated into the Infocious search engine, and we discuss the main challenges in applying these technologies to Web documents. We also present the various components in the architecture of Infocious, and how each one of these components benefits from the added linguistic processing. Finally, we present preliminary results from our experimental study that evaluates the effectiveness of the described linguistic analysis.

1 Introduction

Millions of users today use Web search engines as the primary (and oftentimes the sole) means for locating relevant and interesting information. They rely on search engines to satisfy a broad variety of informational needs, ranging from researching medical conditions to locating a convenience store to comparing available products and services. The most popular of the search engines today (e.g. Ask [3], Google [21], MSNSearch [38], Yahoo! [55], etc.) maintain a fresh local repository of the ever-increasing Web. Once a user issues a query, search engines go through their enormous repository and identify the most relevant documents to the user's query.

While the exact process of identifying and ranking the relevant documents for current major Web search engines is a closely-guarded secret, search engines generally match the keywords present in the user's query with the keywords in the Web pages and their anchor text (possibly after stemming) in order to identify the pages relevant to the query. In addition, search engines often exploit the link structure of the Web to determine some notion of "popularity" for every page, which is used during the ranking of results. In most cases, simple keyword-based matching can work very well in serving the users' needs, but there are queries for which the keyword matching may not be sufficient.

As an example, consider the query *jaguar* that a user might issue to a search engine. Typically, the major search engines may return results that deal with at least three disjoint topics: (1) Jaguar - the car brand name, (2) Jaguar - one version of MacOS X, (3) jaguar - the animal. As one can imagine, it is highly unlikely that a user is interested in all three of the above at the same time.

The query *jaguar* is an example of an ambiguous query because it is associated with multiple senses, each one pertaining to a different topic of interest. As a consequence, Web pages that discuss distinct topics but all share the same keywords may be considered relevant and presented to the user all at the same time. In this scenario, the user has to wade through the results searching for the topic that is of interest to her, or augment her query with additional keywords in the hope of retrieving more relevant results. In the first case, the user is wasting time looking at results that

are of no interest to her, while in the second case the user needs to think and identify appropriate additional keywords that will hopefully lead to retrieving the desired results. In both cases, if we could determine that the user's query is ambiguous we could potentially use this fact in returning better results. For example, we could notify the user that there are multiple possible topics regarding her query and ask her to be more specific, or we could organize the results and let the user navigate to the desired topic.

Resolving such ambiguities has long been the study of a field called Natural Language Processing (NLP). In this paper we present Infocious, a new Web search engine built with the primary goal of improving the users' search experience by reducing ambiguities through the use of NLP techniques. At a high level, Infocious applies linguistic analysis to the Web pages that it indexes in two major ways:

1. First, through language analysis Infocious resolves ambiguities within the content of Web pages. Currently Infocious focuses on three types of ambiguity: (1) part-of-speech ambiguity, (2) phrasal ambiguity, and (3) topical ambiguity. By resolving such ambiguities in the text, our goal is to provide more precise searching and to enable the users to locate the information they seek more quickly.

2. Second, the language analysis performed by Infocious contributes to the ranking of the results that are presented to the user. At a high level, one can see this as rating the overall quality of the text within a Web page. More specifically, Infocious promotes well-written, content-rich documents while conversely demoting lower quality text. Our hope is that by promoting such Web pages we are presenting the users with more interesting and better results, while at the same time we demote noise pages (such as spam pages [40]).

While building Infocious we encountered various issues and challenges in applying NLP towards Web searching, including scalability, efficiency, usability, and robustness. In this paper, we document the NLP techniques that we investigated and decided to incorporate in our search engine and we discuss the challenges involved in applying them to Web text. It is our hope that our work will help and motivate other researchers and shed some light in the challenges involved in bridging the gap between NLP research and large-scale Web-text searching.

The remainder of the paper is structured as follows. In the next section we present an overview of the techniques within the NLP research field that can be of potential benefit to Web searching. In Section 3, we discuss our implementation and the decisions that we made regarding these NLP components. In Section 4, we describe the overall architecture of the Infocious search engine and how our linguistic analysis components are used throughout the system. In Section 5, we present a preliminary experimental evaluation of the linguistic components that we have implemented within Infocious along with two of the tasks that Infocious performs in an attempt to improve the search results: Web page categorization and query categorization. Finally, we review the related work in Section 6 and we conclude in Section 7.

2 Benefits of NLP towards Web Searching

The main goal of the Natural Language Processing (NLP) field is to understand the process of information exchange between humans when they communicate using natural languages. A better understanding of this process would allow computers to extract and operate on information and knowledge represented using natural languages in a more reliable way. The field of NLP has a long and rich history, encompassing linguistics, computer science, psychology and even neuroscience. Over the years, a variety of approaches and methodologies have been used to better resolve ambiguities in order to extract the semantics within natural language text.

Our goal is to build upon NLP and create a better Web search experience for the users. For our task, we focus on statistical, data-driven techniques, which have experienced a surge of progress in the last decade (e.g., [27] and [33]). The reason for this choice is threefold. First, data-driven NLP requires minimal human effort for creating statistical models. This is particularly important in the case of Web-scale processing because of the voluminous and dynamic nature of Web content.[1] Second, statistical models are very robust in the sense that they will generate an interpretation regardless of the input. This is of paramount importance when coping with the heterogeneous and unpredictable nature of the Web. Third, statistical models are currently the most accurate in resolving ambiguities within natural language text, which is the primary task but also the main challenge of NLP.

[1] Focusing on statistical models also gives Infocious the ability to scale to the different languages available on the Web with minimal effort. For example, for every different language, Infocious requires a set of training documents which is, in most cases, already available from researchers in the NLP field.

Here, we present an overview of certain NLP tasks that can be of potential benefit to Web search. In most cases, these NLP tasks are aiming at resolving ambiguities within textual information. For the NLP tasks that we consider, we also discuss how a statistical approach can be applied in each of them.

2.1 Part-of-speech Disambiguation

Consider a Web page that contains the two words *house plants*. Depending on the context around it, this phrase may have multiple interpretations. For example, the Web page may be about *plants for inside the house* or it may be about *objects or methods to house plants*. The difference in the meaning comes from the fact that in the first case the word *house* is used as a noun, while in the second case it is used as a verb. A search engine based on keyword matching would not be able to distinguish between the two cases and the returned results might contain a mix of both uses.

Part-of-speech (POS) disambiguation is the process of assigning a part-of-speech tag (such as *noun, verb, adjective*, etc.) to each word in a sentence. By assigning POS tags to each word, we can determine how the word functions within its context. In doing so, we can determine whether *house* is used as a noun or as a verb in the previous example. A search engine can exploit this POS tagging information by restricting the use of the query keywords to a particular POS tag, thus providing results that are more specific to the desired meaning of a query.

2.2 Word Sense Disambiguation

In many cases, words take on a multitude of different meanings (or senses). Such words are called *polysemous*. For example, the word *jaguar* may refer to a car brand-name, an operating system[2] or an animal.[3] The task of distinguishing between the meanings of a polysemous word, is called *word sense disambiguation (WSD)*. Having the word senses disambiguated would allow the users to search for a specific sense of a word, thus eliminating documents containing the same keyword but that are semantically irrelevant.

2.3 Phrase Identification

Multiple words are typically grouped into phrases to describe a concept more precisely. As individual words are overloaded with multiple usages and meanings, phrases are important for describing a concept more precisely. For example, in the phrases *motor oil* and *cooking oil* the words *motor* and *cooking* are used to describe a more specific type of oil. Phrases, however, are not simply words occurring next to each other. Take for example the sentence *"In the profession of cooking oil is the most important ingredient"*, where *cooking* and *oil* do not form a phrase. Correctly identifying phrases would be beneficial to a search engine in order to return more accurate results to a user's query. More specifically, in the previous example a search engine should not consider the given sentence as relevant to the phrase *cooking oil*, but should ensure that the two words appearing next to each other are indeed parts of a phrase. In general, in order to properly identify phrases it is necessary to perform linguistic analysis with a broader context, a task referred to as shallow parsing or chunking.

2.4 Named Entity Recognition

Named entities refer to names of people, companies, locations, dates, and others. Recognizing the difference between *Jordan* being a person versus a country is the process of *named entity recognition* (NER). A search engine capable of distinguishing different types of name entities would enable users to search specifically for the person or for the country, for example. NER can also be used to extract particular named entities of interest to the user, such as all the companies or locations mentioned in a business article, or all the people mentioned in a newsletter.

2.5 Full Sentential Parsing

Parsing is the process of decomposing a sentence into smaller units, as well as identifying the grammatical role of each and its relationship to other units. Parsing is a well studied problem with many grammar formalisms and parsing algorithms. Parsing is very important for extracting the semantics of sentences precisely.

[2] MacOS version X.

[3] Scientific name: *Panthera onca*.

Query: *house plants*	**Query: *V:house plants***
House Plants - pictures types indoor **House Plants** **House Plants** ... Bring the beauty of **plants** and flowers indoors with **house plants**. Check out this *www.homeandfamilynetwork.com/gardening/houseplants.html* **House Plant** Care and Cultivation Guides Caring for Flowering and Foliage **House Plants** Most **houseplants** are hybrids of **plant** species... *www.thegardenhelper.com/houseplants.html* Troubleshooting and Solving **House Plant** Problems receive pertain to problems with **house plants**.... **House plants** are all hybrids or species **plants** which grow wild somewhere in *www.thegardenhelper.com/troubleshooting.html* gardening, **house plants**, country flower farms a category to browse our **house plant** section.... us — contact us — gardening links — greenhouse tour — directions — weekly specials... weekly **plant** care tips — **house plants** *countryflowerfarms.com/holiday_plants.html*	Hope Grows the Good Samaritan Inn will **house** up to 150 people, making... trees, shrubs, sod and other **plants**, along with the walking trail,... *www.cals.ncsu.edu/agcomm/magazine/spring04/hope.htm* Life History and Ecology of Cyanobacteria the same photosynthetic pigment that **plants** use.... Many **plants**, especially legumes, have formed symbiotic... their roots or stems to **house** the bacteria, in return for... *www.ucmp.berkeley.edu/bacteria/cyanolh.html* Conservatory of Flowers: Inside the Conservatory The Potted **Plants** gallery will feature many flowering... other gesneriads, and will also **house** an interesting array of large... Exhibits gallery is intended to **house** miniblockbuster exhibits themed around particular... *www.conservatoryofflowers.org/insidetheconservatory/index.htm* Mainwaring Wing and Stoner Courtyard The Stoner Courtyard garden, featuring **plants** from three continents, is an... and storage facility, built to **house** the Museum's most at risk... *www.museum.upenn.edu/new/about/mainwaring/newwing.shtml*

Figure 1: Sample search results from Infocious for the query *house plants*, with the default results on the left and the results for *house* used as verb, done via the query *V:house plants* on the right.

Consider as an example the sentence *the man who bought shares of Apple fell*. In this case, a parser would be able to determine that *who bought shares of Apple* is a modifier for *the man*, and that it is the man who fell. In the case of simple keyword matching this article may have been returned as a result for the query *shares of Apple fell*. Additionally, parsing can enable very precise searches, since it would allow the user to specify queries based on subjects (e.g., only *Apple* as the subject), main verbs (e.g., only *bought* as the main verb), or even combinations of these linguistic units. This is especially powerful since many structural constructs can be used to express the same semantics, such as *the man, who owns some Apple shares, fell*.

We have presented a very brief overview of the most common types of language ambiguities. Interested readers may refer to [27] and [33] for a more comprehensive treatment of the subject. In the next section, we present the approach that we have taken in Infocious while investigating each one of the linguistic analysis components that we have just discussed.

3 Linguistic Analysis of Web Text

In this section, we document the linguistic analysis techniques that we investigated and decided to incorporate within Infocious and we discuss the challenges involved in applying them to Web text. At a high level, the task of applying linguistic analysis to Web searching involves two main challenges:

1. The first comes from the massive scale and diversity of Web content, making the issues of balancing cost versus

benefit of linguistic analysis highly important. That is, given the enormous size of the Web, what linguistic analyses should be performed in an efficient, robust and reliable manner that would potentially maximize their utility for improving information retrieval?

2. The second is how to exploit this linguistic analysis to best benefit the user. That is, given that we have resolved various ambiguities through linguistic analysis, how can this improve the way users find information, while making the system simple and intuitive to use?

In this section we discuss the first challenge of Web-scale linguistic analysis, and in Section 4 we address the second challenge: how Infocious leverages this analysis to best benefit the user. We should stress that our focus of linguistic analysis is placed more on the *content* of Web documents and less on the queries. This is because most queries are too short to provide a meaningful context for reliable disambiguation. Instead, ambiguities in the query terms are resolved through examining the *results of queries*, a process described in Section 4.6.1 and is experimentally studied in Section 5.3.

3.1 Part-of-speech Tagging

We treat POS tagging as a probabilistic classification task, i.e.,

$$\tilde{T} = T_{best}(S) = arg\ max_T P(T|S),$$

where S is the input sentence, and T is the set of POS tags assigned to each word of the sentence. In this formulation, the POS assignment for each word w_i in the sentence is treated as a random variable T_i. Each variable can take on the values $\{1, \ldots, |N|\}$, where $|N|$ is the number of different POS tags. Therefore, the task is to determine the instantiations of T such that $P(T|S)$ is maximized.

POS tagging is one of the best studied problems in NLP and is also one where great advances in providing accurate solutions have been made over the years (e.g., [7], [43], and [48]). In Infocious, POS tagging is the first step in the linguistic analysis of every Web page. Our state-of-the-art statistical POS tagger was implemented with efficiency and robustness in mind [8] such that it operates at crawling speed. In particular, a pre-compiled dictionary is used to improve efficiency. If a word does not appear in the dictionary, we calculate its POS tags based on its prefix or suffix. The Viterbi algorithm [52] is used to determine the most probable tag assignments across a sentence, and this probability is recorded for each sentence in every Web page.

By assigning POS tags for each keyword in the Web pages that it indexes, Infocious can offer its users the choice between different word classes (nouns, verbs, and adjectives) of their ambiguous query words. An example comparison of the search results for *house plants* is shown in Figure 1, with and without distinguishing the POS for the word *house*. On the left side of the figure, results that match *any* POS for the words *house plants* are returned, while on the right side of the figure, the user can restrict the word *house* to be only verb by prepending the *V:* directive before it. This directive is a shortcut for experienced users, since knowing and specifying the POS tag for a query keyword may be burdensome for the average user. Because of this reason, Infocious provides illustrative textual prompts to let the users select the POS tag of interest via hyperlinks, as we will show in Sections 4.6.1 and 4.7.

3.2 Phrase Identification

Infocious performs phrase identification (also called chunking in the NLP literature) right after POS tagging. Our statistical chunker also treats this task as a probabilistic classification problem, i.e., it assigns a phrase tag for every word (e.g. whether a word is the start of a noun phrase or the end of a verb phrase), so that the overall probability is maximized across the sentence. For each sentence, this outcome probability is combined with the one from POS tagging to reflect the confidence of both disambiguation processes. For an introduction and additional details on chunking and POS tagging, please see [8] and [50].

Based on the chunker's outputs, we extract what we refer to as "concepts" by combining phrases via a set of rules, such as noun-preposition-noun phrases (e.g., *United States of America*), verb-noun phrases, (e.g., *build relationships*), and verb-preposition-noun phrases (e.g., *tossed with salad dressing*). These rules can be specified either manually or can be automatically extracted from annotated collections of text.

We refer to these constructs as concepts because the phrases are reduced to their main components only, i.e., they are stripped of any intervening modifiers or quantifiers base on their part-of-speech. For example, the set of phrases

Coral Springs Restaurant Reviews

served over Romaine lettuce and **tossed with a Sesame Teriyaki Dressing**. This is a fine...
coralsprings.com/dining/metropolitan.htm

Moab Brewery - Fine Dining And Beers in Moab Utah

roma tomatoes, black olives & croutons **tossed with our special Caesar Dressing**.... black olives, fresh parmesan & croutons **tossed with our special Caesar Dressing**....
www.themoabbrewery.com/menu.htm

pinocchio's

It was **tossed with a tasty, homemade dijon vinaigrette dressing**. We also...
www.jour.unr.edu/outpost/Dining/Reviews/din.gosen.pinocchio.html

Figure 2: Sample search results from Infocious for the concept *tossed with dressing*.

lightly tossed with oil and vinegar dressing is reduced to the *"tossed with dressing"* concept. Similarly, the set of phrases *tossed immediately with blue-cheese dressing* is converted to the same concept. Therefore, a user would be able to find all documents describing the concept of *"tossed salads"*, irrespective of the dressing used. A sample of Infocious' search results for this concept is shown in Figure 2.[4]

In effect, this concept extraction process compresses a document into a list of linguistically sound units. This list of concepts is created for every Web page and is used in two ways. First, it is used *externally* as a search aid for the users. We will show how the extracted concepts blend with Infocious' user interface in Section 4.7. Second, the list of concepts is used *internally* to improve the accuracy of determining the topic of a Web page and to detect pages with very similar (or identical) content.

3.3 Named Entity Recognition

Based on the phrases extracted by the chunker, NER is largely a classification task of labeling the noun phrases in a document. This task is again modeled as a statistical tagging problem, calculating $P(E|p)$, where E is the entity tag given a phrase p. A gazette, which is an entity dictionary that maps a phrase to its entity type E, is compiled from the Web and is used to simplify the NER task. For each proper noun and noun phrase in a document, the NER classifier computes the probability of the phrase belonging to an entity type. Finally, the type e with the maximum probability $P(E = e|p)$ is chosen as the correct named entity tag.

3.4 Word Sense Disambiguation and Page Classification

We experimented with a statistical WSD model with state-of-the-art accuracy rates. While our model is sufficiently efficient for Web-scale disambiguation, there were two problems that we encountered: the accuracy of determining the correct sense of a word and the presentation of a word's different meanings to the user. Although our model's accuracy [9] is comparable to the current best, this accuracy remains relatively low compared to other NLP tasks. Additionally, in the hypothetical case that one could perform WSD correctly, there is still the challenge of how to engage users into specifying which sense they are interested in. Our feeling is that users would not be inclined to read a list of definitions before choosing the desired sense for each of their ambiguous query words. Due to these two issues, we decided to put WSD on hold for the moment. Instead, we opted for an intermediate solution for distinguishing between keyword senses through the use of automatic text categorization.

We use classification as a way to organize results and hide the complexities involved with various linguistic ambiguities. That is, instead of prompting the user with a list of definitions, Infocious simply organizes the results into

[4]Concept-based searching in Infocious is not identical to traditional phrase searching. Concept-based searching is designed to help the user better organize and navigate search results via our user interface, which is described in Section 4.7. The results shown in Figure 2 can be reproduced via the following URL: `http://search.infocious.com/q?s=%60tossed+ with+dressing%60&c0=cab81178c`

categories. Therefore, in the example case of the *jaguar* query, pages about Jaguar cars would fall under the *Automobile* category, whereas pages about the software would be under the *Computers* category. The users can then choose one of these categories to narrow their search.

This feature is made possible by classifying every page within Infocious' index into categories prior to querying. To train our classifier, we have used the category hierarchy from the DMOZ directory [17], along with the documents organized into each of the categories. The classification process is described in more detail in Section 5.

3.5 Parsing

From our prior experience with statistical, lexicalized parsers, we believe that full sentential parsing remains too expensive for Web-scale deployment. Having a complexity of $O(n^3 \cdot |G|)$, where n is the number of words in a sentence and $|G|$ is the number of rules in a grammar,[5] one can see that for sentences of non-trivial length, parsing can be quite expensive. While parsing can provide useful information to improve ranking of results, we believe that at present the computational cost does not justify the benefits. Furthermore, parsing also presents the issue of user interface, in that in order to tap into the preciseness of searching parsed data, users may have to master a query language. These are interesting challenges we wish to address in the near future.

3.6 Calculating Text Quality

As Infocious processes the text probabilistically, the resulting probabilities are combined and saved for each sentence. These probabilities are then factored into an overall score for the entire document, which we refer to as the textual quality (or *TextQuality*) of the page. This probability is used during ranking to promote pages with high-quality textual content, as well as during indexing to weigh the relative importance of anchor texts.

In Figure 3 we illustrate the influence of our TextQuality measure on the ranking of search results based on the textual portion of the documents. On the left of Figure 3 we show the results for the query *britney spears* without the TextQuality metric. As seen from the summaries around these results, these pages are mainly composed of secondary phrases containing popular keywords. On the right of Figure 3 we show the results for the same query (i.e., *britney spears*) but we factor the TextQuality metric into the ranking. In this case, the results presented are considered to contain more coherent text which we believe the users would find more informative and useful.

4 The Architecture of the Infocious Search Engine

We now describe the overall architecture of Infocious and how our linguistic analysis is used throughout the system to improve Web searching. An overview of Infocious' main modules and their interaction is shown on Figure 4.

4.1 Crawling the Web

The crawler is the part of a search engine that handles the task of retrieving pages from the Web and storing them locally for processing. Our distributed crawler behaves like other typical crawlers in the sense that it discovers and follows the links inside a Web page in order to download other Web pages. However, we have extended our crawler according to recent research results such that it can provide us with a fresh subset of the Web with a minimal overhead [14], as well as retrieve pages from the so-called Hidden Web [41]. Crawling is a broad topic and readers interested in the topic please refer to [11, 12, 13].

Once the crawler downloads a page from the Web it performs two tasks. First it extracts all the links from the page and sends them to the link database, described next. Second, it hands the page off to the linguistic analysis component.

4.2 Linguistic Processing

This module performs the linguistic analysis that we described in Section 3 for every page that the crawler downloads. More specifically, every page that is downloaded and sent to this module, is first stripped from the HTML markup and then its content is POS-tagged. Once the content of the page is annotated with the POS tags we perform phrase

[5] A grammar is a set of rules describing the legal construct of the sentences in a given language. One example rule for English is that verbs follow subjects.

Britney Spears Pictures - **britney spears** pictures,... picture of **britney spears**, hot pictures of **britney spears**... *britney-spears-pictures.hotyoungstars.com/nude/*	Is **Britney Spears** over the edge? Is **Britney Spears** over the Edge?... **Britney Spears** is a singer.... *azwestern.edu/modern_lang/esl/cjones/mag/spring2004/britney.htm*
Britney Spears Breasts - **britney spears** breasts, pics... breast implant, pictures of **britney spears** breasts, **britney**... *britney-spears-breasts.hotyoungstars.com/nude/index2.html*	Best Pictures Of **Britney Spears** + wallpapers, facts and ... **Britney Spears** comes to us from... **Britney** was a performer since a... *keanu-reeves.best-pictures.com/spears/britney.html*
Britney Spears Photos - **britney spears** photos,... **spears, britney spears** nude photos, nude photos of... *britney-spears-photos.hotyoungstars.com/nude/*	IMPERSONATORS - **BRITNEY SPEARS** Is Proud To Present! Contact: Gary Shortall Back... *www.impersonators.com/brittany/brit.htm*
Hot **Britney Spears** Pics - hot **britney spears** pics,... **britney spears**, new hot pics of **britney spears**,... *hot-britney-spears-pics.hotyoungstars.com/nude/*	**Britney Spears'** Coke Habit **Britney Spears'** Coke Habit Destroys Her... *www.emptyv.org/britney_spears.htm*

Figure 3: Sample search results for the query *britney spears*, comparing Infocious' relevance ranking *without* our TextQuality measure on the left and the ranking when it is *included* on the right.

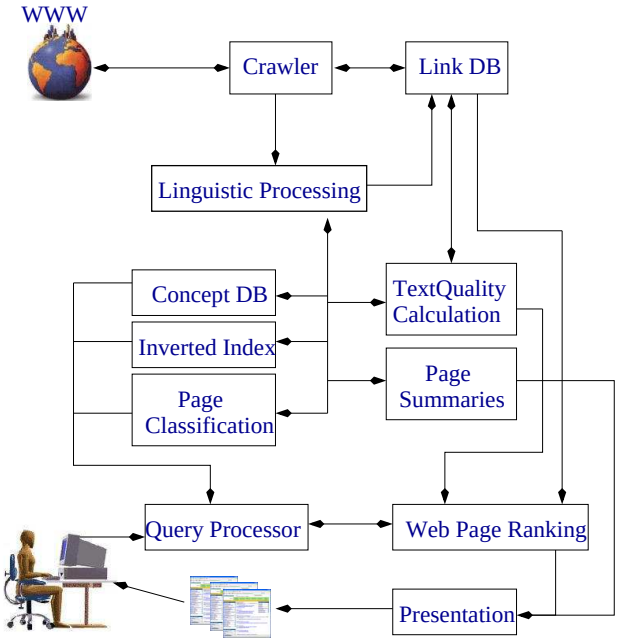

Figure 4: The architecture of Infocious.

identification and named entity recognition. Finally, we resolve any ambiguous words in the page and we attempt to identify the topic of the page's content through the use of a classifier, which we will describe in more detail in Section 5.

We have developed the linguistic analysis module in such a way so as to keep pace with the crawling speed. This module, the heart of Infocious, resolves language ambiguities, appends the linguistic information to the content of a Web page, and sends documents to other modules for processing.

4.3 The Link Database

The link database performs two functions. First it manages the task of assigning a globally unique ID for every link that the crawler has identified. The second functionality is to store various static properties of the URLs that Infocious is aware of. Such information includes the number of incoming links to a Web page, the number of outgoing links, a content signature, a concept list signature, and the quality of the text described earlier. This information is used during the ranking of results, as well as for rescheduling crawls.

4.4 Inverted Indexes

An inverted index is a typical data structure used for retrieving documents that are relevant to one or more particular keywords. Given a collection of Web pages, the inverted index is essentially a set of lists (called inverted lists), one for each term[6] found in the collection. For every term, a record (also called a posting) is maintained at its corresponding list. Every posting stores the document ID that contains a particular term. Additionally, every posting contains the number of occurrences of the term in the document, along with a list of positional information of the term.

Along with every positional record, we maintain information regarding the formatting of a term (i.e., whether the term should be rendered in bold, what is the font size, the color, etc.) Furthermore, the index stores functional attributes such as whether the term appears in the URL of the document, whether it appears in the title, etc.

Finally, for every term occurrence in a document we store in the index any associated NLP annotations as identified by the linguistic analysis component. This records any ambiguities resolved by the linguistic analysis module, such as whether a term is used as a noun or a verb. This enables Infocious to return only the documents with the correct meaning to the user.

4.5 Page Summaries

This module stores and retrieves the NLP-annotated version of the Web pages that Infocious indexes. The module takes as input the data from the linguistic processing modules and stores the pages along with the annotation of their content in a compressed format. Upon returning a document as a search result, the document's content is retrieved from this module. After that, the query keywords are identified within the retrieved text in order to display a short context (typically in the order of 5-10 words) around the query words.

Additionally, this module stores and retrieves the list of concepts extracted by the NLP module for every document. These concepts are used as navigational aids for users, as well as for improving text categorization, described later.

4.6 Answering a Query

Infocious supports the standard keyword searching, phrase searching,[7] as well as searching based on the concepts described earlier. Furthermore, a mixture of keywords, phrases, concepts, and categories is supported, including the ability to exclude concepts or categories deemed undesirable by the users. For example, a user searching for *jaguar* the animal can either select the *Animals* category, or choose to exclude the *Computer* category instead. In addition, the user can specify the part-of-speech tag for any query keyword. For example, the query *V:house plants* will only match documents where the word *house* is used as a verb. On the other hand the query *N:house plants* will retrieve documents where *house* is used as a noun. We should note that the default query semantics in our search engine is the ANDing of the keywords. That is, similarly to other Web search engines, we return documents which must contain all of the keywords that the user specified.

Given a list of documents that contain the user's keywords and any additional directives (e.g., exclusion or POS tags), Infocious ranks and sorts the result list so that the most relevant documents are displayed first. Ranking is probably the single most important component of a search engine and can be considered as one of the most challenging. It is also an ongoing process that needs to be constantly tuned and tailored to the dynamic nature of the Web.

Within Infocious we take a variety of factors into account for ranking the results. Such factors include the frequency of the keyword in a document, whether the keyword appears in the URL of a page, whether it appears in the title of the page, its relative font size to the rest of the document, etc. We also incorporate link-based properties of the Web

[6] Term is used loosely in this context. It can refer to either a single word, a phrase or a concept. In its inverted indexes, Infocious keeps all three kinds of terms.

[7] Phrase searching consists of keywords surrounded by quotes and they are to be matched in-order and next to each other within a Web page.

pages. More specifically, within Infocious , pages which are highly linked are in general considered more important than pages with fewer incoming links.

In addition to the above, we leverage our NLP technology in an attempt to return more relevant pages to the user. More specifically, our ranking algorithm uses the results from POS-tagging, word sense disambiguation, classification etc., so that when a page is composed of well-written textual content, it will be promoted, while the opposite will happen for a page with poor textual content.

The process of ranking results is summarized as follows:

1. **Step 1.** Retrieve a list of document IDs that satisfies the user query from the inverted index.
2. **Step 2.** For each document ID, retrieve its document attribute values from the Link Database, such as its TextQuality and crawl date.
3. **Step 3.** Normalize the values for each attribute across all result documents.
4. **Step 4.** Compute an overall score for each document based on a linear combination of normalized attribute values with its associated weight.
5. **Step 5.** Sort based on the overall score, retrieve the document summary for the top-N documents, and format the results for display to the user.

4.6.1 Automatic Query Disambiguation

We also utilize the NLP annotation stored in our index to perform a form of automatic query disambiguation, which is then used to dynamically rank documents according the most likely meaning of a keyword for which the user is querying.

Instead of performing linguistic analysis on the query strings, which are usually too short to establish a reliable context, we instead use the result documents themselves. That is, by gathering statistics on how the query terms are used in context within complete documents, Infocious can disambiguate the query terms based on how people use these query words within the same context.

For example, we can establish that in a majority of documents where the words *train* and *engines* are discussed, *train* is most often used as a noun. We then rank the results based on this meaning of the query word, i.e., promoting documents with the noun usage of *train*. The same principal applies for the opposite case, such as for the query *train pets*, where the verb sense would more likely be used.

Taking this example a step further, consider a more ambiguous query *train chiefs* or a seemingly non-sense query *train grass*. In these cases, there might not be enough evidence in the documents as to decide which of the two senses the word *train* refers to. In such cases Infocious does not assume a particular meaning. Instead, it presents the user with intuitive examples of different usages so he or she can choose the desired meaning.

We conjecture that our method of query disambiguation is more reliable because it draws upon the great number of instances of Web documents where the query words are used in context. On the other hand, directly performing disambiguation on the user's query cannot be as reliable since the context that the user provides is typically very limited. Note that our method of disambiguation comes nearly for free because the NLP analysis is performed and stored in the index ahead of querying. We experimentally study query disambiguation in Section 5.3.

4.7 User Interface

When Infocious presents the results to the user, we again tap into our NLP technology to further help users navigate and manage search results. An example of our user interface is shown in Figure 5 for the query *lesson plans*. This is how the search results are presented to the user (along the center), plus any additional search and navigational aids designed to help users in their search quests. We briefly describe each of these aids: [8]

- Infocious presents, right above and below the search results (Figure 5-1), the categories that the current search results fall into. In our particular case for the query *lesson plans*, these categories include *Education/Educators*, *Education/K through 12*, etc. By hovering over these categories the user can see in real time what category each

[8] For more detailed information on the Infocious' user interface, please visit `http://corp.infocious.com/tech_overview.php`.

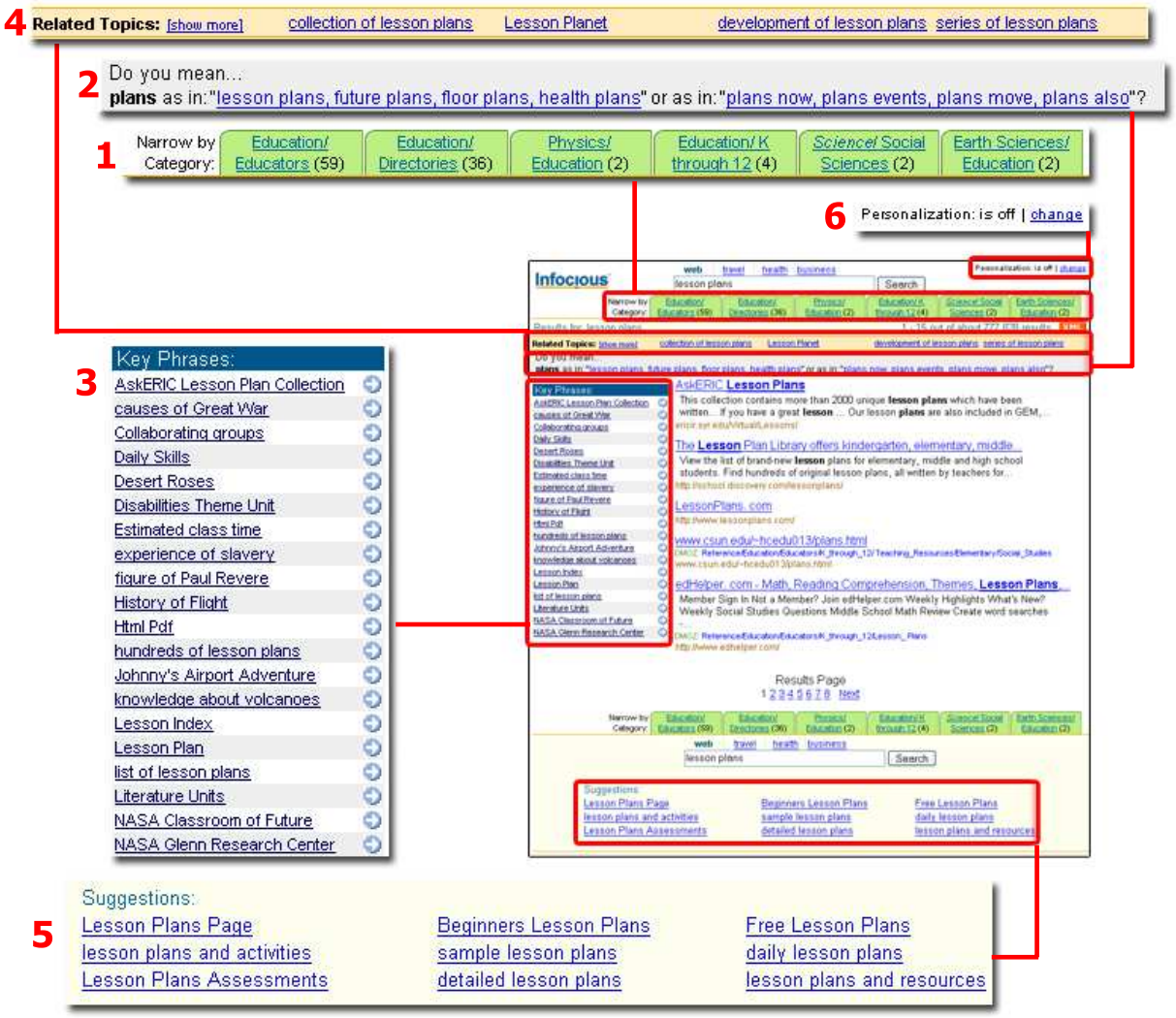

Figure 5: The Infocious user interface for the query *lesson plans*. Each section of the interface is zoomed in and numbered: (1) Categories of the results, (2) Disambiguation suggestion, (3) Key Phrases, (4) Related Topics, (5) Suggestions, (6) Personalization.

of the results falls into. At the same time, by hovering over one result, its respective categories are highlighted in order to give the user an idea of the topic of every result.[9]

- In this example, the word *plans* is ambiguous and can be a verb or a noun. Because for this query both meanings of *plans* are deemed as probable, Infocious provides the users with links to more precisely specify which meaning of *plans* they are interested in. This is shown on Figure 5-2.

- On the left side of the search results (Figure 5-3), the user can find the "Key Phrases" list. This list presents the concepts culled from the Web pages during the NLP stage. The "Key Phrases" is similar to an index at the end of a book, listing the important concepts along with their location in the text. This list provides users with a quick overview of the important concepts within search results, and gives them context in advance about these results before having to visit them.

[9]The reader may visit `http://search.infocious.com/q?s=lesson+plans&ISbtn=Infocious+Search` to see this interactive feature.

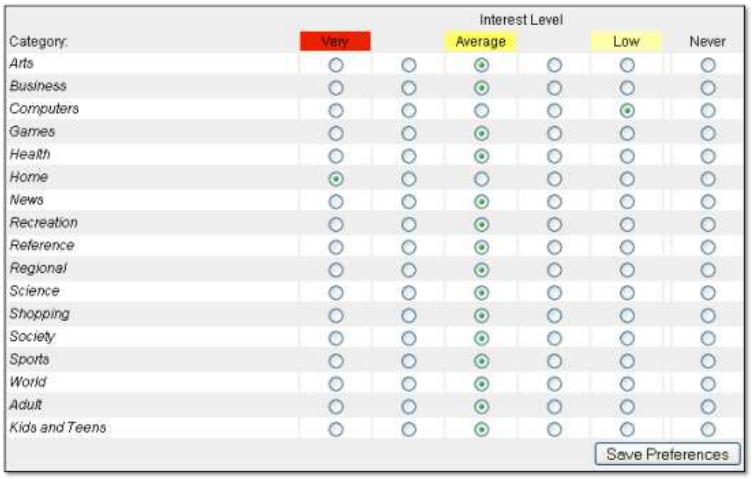

Figure 6: The personalization user interface. The user specified that is very interested in the *Home* category and cares very little about *Computers*.

- Above the search results, there is a list of "Related Topics" (Figure 5-4), which is compiled during the concept extraction phase. This list is provided to help the user expand their search in case they are unfamiliar with the topic they are researching on. For this example Infocious suggests concepts such as *collection of lesson plans* or *series of lesson plans*. The user may examine more related topics (e.g. *classroom activities* or *social studies*) by clicking on the *[show more]* link. We have found this feature to be particularly useful when the user wants to explore some generic area of interest such as *fuzzy logic*, *information retrieval*, *data mining*, etc.

- Right below the search results there exists a "Suggestions" list (Figure 5-5). This particular list contains suggestions of longer queries that, when employed, would make the current search more specific. For the current example, Infocious suggests queries such as *sample lesson plans* and *daily lesson plans*, which will help the users further hone their original search.

- Finally, because Infocious classifies every Web page into categories, it is capable of offering the users the ability to personalize their search results and tailor them to their particular interest. Right next to the search box (Figure 5-6) the user can enable or disable personalization. For example, the user might be an avid connoisseur of arts and may not be interested at all in sports. By using personalization, users can restrict the results to be within the categories that are of interest to them. This is done through an intuitive interface, shown in Figure 6 where the user can specify the level of interest for each category.

We have presented the major features of Infocious and how they are utilized in an attempt to provide the users with a better, and easier experience in finding what they are looking for. As one can see, most of these features are either enabled by ("Key Phrases", and "Do You Mean"), made better ("Related Topics" and "Suggestions"), or made more accurate (Categories and Personalization) because of our NLP technology. We further support this claim by demonstrating the benefits of NLP analysis in improving the accuracy of two classification tasks in the next sections.

5 Experimental Evaluation

Ideally, the best way to evaluate the effectiveness of the linguistic analysis techniques employed by Infocious is to measure how much it helps users in finding what they want in the minimum amount of time. Measuring this overall improvement, however, requires a large user base and extensive user survey, which is difficult for us given our limited resources. Given the difficulty of such evaluation, in this section we present a preliminary experimental evaluation of the individual linguistic components within Infocious. In Section 5.1, we first evaluate the accuracy of our part-of-speech tagging and phrase-identification modules. In Section 5.2, we evaluate the accuracy improvement in the page

Model	Known Word	Unknown Word	Overall
JMX [43]	97.3%	88.3%	96.7%
Infocious	97.8%	92.3%	97.4%

Table 1: Comparison of POS tagging accuracy between Infocious' and Ratnaparkhi's JMX tagger.

	Precision	Recall	$F_{\beta=1}$	Techniques
Infocious	92.74%	92.86%	92.80	Maximum Entropy models
Kudoh et al.[29]	93.45%	93.51%	93.48	SVM + model combination
van Halteren et al.[51]	93.13%	93.51%	93.32	WPDV + model combination + post error correction
Li et al.[31]	93.41%	92.64%	93.02	SNoW + CSCL
Sang et al.[49]	94.04%	91.00%	92.50	MBL + model combination

Table 2: Accuracy results of chunkers trained and tested on the CoNLL-2000 data.

categorization when we use the results from NLP analysis during the categorization task. Finally in Section 5.3, we evaluate the effectiveness of our query categorization module.

5.1 Evaluation of the Linguistic Analysis Component

The linguistic analysis components used within the Infocious search engine were evaluated using established benchmarks commonly employed by researchers in the field of computational linguistics.

In the experimental evaluation that we present in this section we used a corpus where the documents contained in it have been manually annotated with the associated linguistic information, such as a word's part of speech, a phrase's boundaries and its type, a word's sense, and a sentence's syntactic parse tree. Given such an annotated corpus, it is divided into pairs of non-overlapping training and evaluation sets. For every pair, the training set was used to train a NLP model (such as POS, word sense disambiguation etc.). Once trained, it is tested on the evaluation set of the pair to determine its accuracy. The accuracy of every NLP task is scored with standardized metrics, such as precision and recall, using the annotated data as the gold standard.

For the purpose of training and evaluation of the Infocious NLP models, we employed two corpora which are widely used as benchmarks in the NLP field: the Penn Treebank [34], which contains full syntactical parse trees for each sentence in the corpus, and SemCor [37], which provides the WordNet sense for every word within the corpus.

5.1.1 Part-of-speech Tagging

For the evaluation of part-of-speech tagging accuracy, sections 1-19 of the Wall Street Journal portion of the Penn Treebank were used for training, and sections 20-22 for testing. This is an established procedure and thus we can directly compare to the precision of other POS tagging algorithms. In Table 1 we compare Infocious' tagger results to Ratnaparkhi's tagger [43], which is still considered to be one of the most accurate taggers. For additional experimental details and comparisons to other POS taggers, please see [8], Chapter 3.

5.1.2 Phrase Identification

For the phrase identification[10] task, we used the evaluation procedure that was developed as part of the CoNLL-2000 workshop [50]. We compared our method to several other models that followed the same evaluation procedure. More specifically, we evaluated Infocious' phrase identification against other single-model phrase identifiers and we present the results in Table 2. It is worth noting that the most accurate models employ ensemble of a large number of classifiers, optimizing for accuracy at the cost of efficiency and scalability, making the approach less desirable for Web-scale processing.

[10] Phrase identification is also referred to as "chunking" in the NLP research field.

5.2 Evaluation of Automatic Categorization of Web Pages

To better address the word-sense disambiguation problem, one of our goals is to automatically classify every Web document into a pre-defined category hierarchy such as the DMOZ directory as accurately as possible. In doing so, Infocious enables users to narrow their search to a particular topic, or to personalize the ranking of search results to better match their interests.

What Infocious has in addition to other text classification methods is its large repository of NLP annotated Web pages. In this section, we illustrate through a classification experiment that the additional information that NLP provides can actually improve classification accuracy and therefore can help Infocious to better organize search results.

The text classification problem can be stated simply as follows: given an input document d, find the class c it belongs to. A probabilistic formulation of the problem can be: $max_{c \in C} P(c|d)$. However, because DMOZ has close to $600,000$ categories (i.e., $|C| \approx 600,000$), Infocious uses a hierarchical algorithm that employs a different classifier for every parent node in the DMOZ hierarchy. In this section, we report our experimental results on classifying Web pages into one of the top-level categories of DMOZ, since the results on the top-level classification accuracy is sufficient in bringing out the influence of NLP annotations on classification accuracy.

5.2.1 Experimental Setup

In order to measure our classification accuracy we need an annotated collection of Web pages. For the purpose of this experiment, we examined the crawled pages within the Infocious repository and we determined which ones of them were also present within the DMOZ directory. For every such Web page, we generated its NLP-annotated version. This data is used as our training corpus to evaluate classification accuracy, i.e., to reproduce the classification done by the DMOZ volunteers given a new Web document.

We should note that in the DMOZ directory there are 17 top-level categories. Since DMOZ is organized hierarchically, we not only include documents listed within each top-level category, but also pages from all of its sub-categories. Overall we identified 981,890 pages that we used for our evaluation. Table 3 shows the top-level categories along with the number of documents within each of the categories.

For each document, our preprocessor first discards all formatting elements, tokenizes the document, and detects sentence boundaries. The NLP module then performs POS tagging and phrase detection, and appends the tagging to each word. Lastly, concepts for each document are extracted, sorted based on their *tfidf* values [44, 45], and the top 50 concepts are added to the documents. The number is rather empirical as we have found that adding 50 concepts generally produces the best results.

For each experiment we performed 10-fold cross-validation to generate the accuracy rates, with 90/10 split of training and testing data. For classification, all tokens are converted to lower case and words that occur less than five times are replaced with the "unknown" token which is not used in the final classification. We have empirically found that by removing words occurring less than five times we are not harming the classification accuracy of our algorithms and, at the same time, we are reducing the vector dimension that our classifiers have to work with.

For our experiment here, we chose the Naive Bayes classifier [18] because of its efficiency, important for Web scale processing, and for its accuracy. We compared Naive Bayes to maximum entropy, expectation maximization, and tfidf on a subset of our collection and Naive Bayes was either comparable to or more accurate than the other classifiers.[11] We have also found that Support Vector Machines [15], well known for their classification accuracy, are too computationally expensive for our task.

5.2.2 Results

We trained four classifiers with increasing amount of NLP annotations: (1) words only (i.e., no NLP information), (2) words plus POS tagging, (3) words plus extracted concepts, and (4) words plus POS tagging and extracted concepts. The first classifier serves as our baseline since it does not rely on any NLP information. Classifiers (2) and (3) add partial linguistic information and their purpose is to demonstrate how much each of the POS-tags and extracted concepts may improve accuracy when used in isolation. The last classifier combines both additional annotations.

The overall accuracy results are shown in Table 4. In this table, the "Accuracy" column shows the fraction of all the pages that were classified correctly from the respective classifier. In Table 5 we present the accuracy rates for

[11] We plan to report on a detailed study comparing the performance of different classifiers in future work.

Category	Number of documents	Avg # of sentences
Arts	48,568	88
Business	93,936	71
Computers	43,293	104
Games	8,499	92
Health	18,759	115
Home	6,031	116
News	2,859	151
Recreation	23,239	95
Reference	15,204	146
Regional	258,543	75
Science	23,029	136
Shopping	43,284	83
Society	52,178	114
Sports	23,223	94
World	308,975	49
Adult	8,969	69
Kids and Teens	3,301	86
Total	981,890	

Table 3: Statistics on the collection of Web pages used for evaluating classification accuracy.

Classifier	Accuracy	stdev
(1) words only	64.9%	0.03%
(2) words plus POS tags	66.1%	0.03%
(3) words plus extracted concepts	66.3%	0.04%
(4) words plus POS and extracted concepts	67.6%	0.04%

Table 4: Accuracy results from four classifiers trained on varying amounts of NLP annotations. All accuracy improvements are less than 0.1% likely to be due to statistical variations according to the t-test statistical significance test [53].

each individual category for the baseline Classifier 1 and Classifier 4, which uses POS tags along with the extracted concepts.

5.2.3 Discussion

The overall accuracy results show that POS tags and extracted concepts individually improved classification accuracy, and by combining both the accuracy improved by 2.7%, i.e., we observed a 7.7% reduction in error rate. While this improvement is modest, we demonstrated that NLP annotations do provide valuable context for improving text classification accuracy.

In order to ensure that our observed classification accuracy is not due to statistical variation we have performed the t-test statistical significance test [53]. The test showed that the accuracy numbers reported in Table 4 are less than 0.1% likely to be due to statistical variation.

Table 5 shows the accuracy rates of each top-level DMOZ category. The most accurate category is *World*, which benefits from the English/non-English distinction. The worst is *Kids and Teens*, a relatively recent addition to DMOZ that has a limited number of documents. When comparing between Classifier 1 and 4, one can see a uniform improvement of classification accuracy, with the *Arts* category benefiting from NLP annotations the most. One exception is the category *News* which seems to have a modest improvement due to the fact that this is a very broad category containing a large number of specialized sub-topics.

While these accuracy rates leave room for improvement, it is worth mentioning that the baseline accuracy is

Category	Classifier 1	Classifier 4	% Accuracy Increase
Arts	52.01%	59.59%	14.57%
Business	56.65%	60.58%	6.93%
Computers	58.14%	61.03%	4.97%
Games	61.92%	62.54%	1.00%
Health	62.10%	67.23%	8.26%
Home	32.24%	35.88%	11.29%
News	46.35%	46.74%	0.84%
Recreation	46.75%	51.57%	10.31%
Reference	60.75%	65.52%	7.85%
Regional	51.16%	52.64%	2.89%
Science	39.89%	45.64%	14.41%
Shopping	58.79%	64.00%	8.86%
Society	45.14%	51.20%	13.42%
Sports	64.38%	69.80%	8.41%
World	91.37%	92.24%	0.95%
Adult	62.44%	63.27%	1.32%
Kids and Teens	11.40%	13.86%	21.57%

Table 5: Comparison of average accuracy rates and reductions in error rates between individual categories for the classifiers without (Classifier 1) and with NLP annotations (Classifier 4). All accuracy improvements are less than 0.1% likely to be due to statistical variations according to the t-test statistical significance test [53].

comparable to other large-scale text classification studies with a complete set of categories [10, 22, 30, 39].

Inside Infocious, we store both the classification outcomes and their corresponding probabilities in our indexes. Upon ranking of results, pages with higher classification confidence are prioritized over more ambiguous pages, thus reducing the likelihood of erroneous categorization appearing early in the results. This is done in the hope of presenting the users with results that are more relevant to their queries.

We now turn to studying the problem of disambiguating the users' queries.

5.3 Evaluation of Automatic Query Disambiguation

As we discussed in Section 4.6.1 we leverage the linguistic information stored in our index to perform automatic query disambiguation. The results of the query disambiguation can be used in a variety of ways: e.g to deduce the user's intention in finding information or to rank documents based on the most likely meanings of the keywords in the user's query.

Within Infocious, instead of performing linguistic analysis on the query strings which are typically very short, we use the result documents themselves. That is, we collect statistics on how the query terms are used in-context within complete documents and then we disambiguate the query terms based on how the query words are used within the same context. In this section we experimentally study the accuracy of our query disambiguation.

5.3.1 Experimental Setup

In order to study the accuracy of our query disambiguation, we used a query set of 800 queries, manually tagged by three human experts. This query set is the one used in the recent KDD Cup 2005 competition.[12] Every query is manually annotated with up to 5 categories from a pre-defined set of 67 categories.

Given this set of queries along with their categories, our goal is to use Infocious to correctly predict the categories. For this task, we operated as follows:

1. **Step 1.** Retrieve one query from the test file.

[12]The data can be downloaded from: `http://www.acm.org/sigs/sigkdd/kdd2005/kddcup.html`.

2. **Step 2.** Send the query to Infocious and identify the categories for the top-K results. In the results that we report here, K was set to 1000.

3. **Step 3.** Compile a list of attributes for every category that was identified from the previous step. In particular, along with every category label, Infocious provides us with a number showing how many of the 1000 returned documents belong to that category, and a floating point number reflecting the average probability that those documents belong to the given category. That is, every category from Infocious is represented as a tuple *(Category-name, Number-of-docs, Average-probability)*. The highest the *Average-probability*, the more certain we are that the documents within this category are classified correctly. This step is described in more detail in Section 5.3.2.

4. **Step 4.** Since Infocious uses the DMOZ hierarchy, the *Category-name* that Infocious generates through the previous step is one of DMOZ's. However, in the KDD Cup dataset there are 67 predefined categories that are not necessarily identical to DMOZ. For this reason we generated a mapping from the DMOZ categories to the ones provided by the KDD Cup. Thus, as Infocious returns DMOZ categories for a query, this mapping is used to generate the KDD Cup categories, which are eventually ranked based on a number of factors as we will describe in more detail in Section 5.3.3.

5. **Step 5.** Print out the top-N KDD Cup categories that were identified from the previous step.

6. **Step 6.** If the test file has more queries repeat from Step 1.

In the above algorithm the most crucial steps for the performance of our method are 4 and 5. In the next sections we describe these steps in more detail.

5.3.2 Answering Queries through Infocious

For the task of retrieving the categories for a query, we used an internally developed API that would return only the categories. This was done for efficiency and simplicity, so that we can avoid parsing the HTML output of the Web interface. Since the Infocious' Web page classifier works probabilistically, it returns not only the number of documents assigned to a category, but also the average probability of these documents assigned to that category. For example, for the category *Hardware/Components*, Infocious returned the tuple *(Hardware/Components*, 358, 0.74*)*, which means that this particular category contained 358 documents out of the 1000 results that we retrieved, and that the average probability of these 358 results belong to the *Hardware/Components* category is 0.74. Note that within Infocious, documents can be categorized in more than one categories and therefore the sum of the *Number-of-docs* can be more than 1000. These tuples, one for each category, are used to rank the final categories, described next.

5.3.3 Mapping and Ranking of Categories

As we have already discussed, the categories that Infocious outputs are the ones from the Open Directory Project. In order to convert them to the KDD Cup categories, we have constructed a mapping between the two sets of categories. The mapping was constructed manually and a sample is shown in Table 6. The first column shows the DMOZ category and the remaining columns show the KDD Cup categories that it maps to.

Since no example documents were provided to help us assess the nature of each KDD Cup category, our mapping is mainly based on the similarity of the names between the two sets of categories. In order to have as complete of a mapping as possible we assigned up to three KDD Cup categories for each DMOZ category. Additionally, we considered up to two levels of the DMOZ categories in our mapping. Should Infocious return a third-level DMOZ category we would use only the first two levels for the mapping.

With the DMOZ categories to the KDD Cup mapping in place, we then convert the tuples replacing the DMOZ *Category-name* with the mapped one, while maintaining the *Num-of-docs* and *Average-probability* components. Since we used a one-to-many mapping, the final set of categories that we have may contain one or more categories multiple times.

If we assume that a KDD Cup category C_i has k tuples after the mapping, we use the following ranking function in order to identify the most important categories:

DMOZ Category	KDD Category 1	KDD Category 2	KDD Category 3
News/Magazines and E-zines	Living Book & Magazine	Information References & Libraries	
Shopping/Health	Living Health & Fitness		
News/Internet Broadcasts	Computers Internet & Intranet	Computers Networks & Telecommunication	Online Community Chat & Instant Messaging
Kids and Teens/Arts	Living Family & Kids	Information Arts & Humanities	

Table 6: Example of category mapping from DMOZ to KDD Cup

$$Rank(C_i) = Pr_{trainsample}(C_i) \cdot (\sum_{j=1}^{k} Num(C_i, j))^2 \cdot \sum_{j=1}^{k} (Num(C_i, j) \cdot AvgProb(C_i, j)) \quad (1)$$

In the previous function, the numbers are normalized. In addition:

- $Pr_{trainsample}$ is the probability of category C_i within the train sample (CategorizedQueries.txt) that was provided by the KDD Cup. We made the assumption that the given training sample is representative of the final test set, therefore we applied a prior probability of the categories based on their appearances within the training sample.
- $Num(C_i, j)$ is the number of documents that exists in the *Number-of-docs* field of tuple j (out of the k ones) for the KDD Cup category C_i. We favor categories that have more documents assigned to them.
- $AvgProb(C_i, j)$ is the average probability that exists in the *Average-probability* field of tuple j (out of the k ones) for the KDD Cup category C_i. We favor categories that have a higher average probability and we take into account the number of documents within the category.

Once we calculate the value $Rank(C_i)$ for every KDD Cup category we keep the top-5 (since the testing set contains up to 5 categories). Within those top-5 we ultimately identify the minimum k ($1 \leq k \leq 5$) that the top-k categories cover 80% of the documents. For example, if we determine that based on the rank function we should keep the top-5 categories C_1, C_2, C_3, C_4 and C_5 and we know that C_1 and C_2 together cover, say, 83% of the returned documents, we will only return C_1 and C_2. Admittedly this final heuristic may introduce some bias in the final results. The intuition behind it however is that it can improve recall accuracy, based on the fact that we will essentially discard categories which are not covering a significant fraction of the returned documents, even though they managed to be in the final top-5.

5.3.4 Results

In order to evaluate our results, we compared the categories assigned by our algorithm to the manually assigned categories from the three experts. Each one of the three experts tagged all 800 queries with up to 5 out of the 67 pre-defined categories. Here we report our performance using standard precision and F1 metrics averaged over the three data sets from the experts. This is essentially the same metrics used during the KDD Cup competition:

$$Precision = \frac{1}{3} \sum_{i=1}^{3} \text{(Precision against human labeler i)} \quad (2)$$

$$F1 = \frac{1}{3} \sum_{i=1}^{3} \text{(F1 against human labeler i)} \quad (3)$$

Our performance results are shown in Tables 7 and 8. Table 7 shows the precision performance of our algorithm along with the top-10 precision values that were achieved by the 37 KDD Cup participants. Overall, we achieved an average precision of 33.9% over the three human labelers and we ranked 7th. Similarly, Table 8 shows our performance under the F1 metric. Overall, our algorithm achieved an F1 value of 35.7% and we ranked 5th under the F1 metric.

It is worth mentioning that the performance of our query disambiguation is comparable to the overall performance achieved by the winners of the KDD Cup with precision of 42.3% (Id 37 in Table 7) and F1 of 44.4% (Id 22 in

Submission ID	Precision	Rank
25	0.753659	1
33	0.469353	2
8	0.454068	3
37	0.423741	4
22	0.414067	5
21	0.340883	6
Infocious	**0.339435**	**7**
10	0.334048	8
13	0.326408	9
35	0.32075	10
...	...	...
16	0.050918	37

Table 7: The precision performance of our query disambiguation algorithm.

Submission ID	F1	Rank
22	0.444395	1
37	0.426123	2
8	0.405453	3
35	0.384136	4
Infocious	**0.357127**	**5**
10	0.342248	6
21	0.34009	7
14	0.312812	8
3	0.309754	9
9	0.306612	10
...	...	...
16	0.060285	37

Table 8: The F1 performance of our query disambiguation algorithm.

Table 8).[13] The winners of the KDD Cup used a 2-phase ensemble classification: in the first phase the queries are sent to number of search engines from where page categories and page content are retrieved. In the second phase this data is passed through synonym-based and statistical classifiers in order to produce the final query categories.[14] Although the results from all participants reported in Table 8 are promising one may argue that the values achieved are generally low for a production search engine since about 60% of the queries will be misclassified. We also believe that query classification is a promising problem to work on with space for improvement and deserves further research.

6 Related Work

Some of the earliest research into searching textual information is in the field of information retrieval [5, 54, 45]. Certain approaches proposed by the information retrieval field have been incorporated into the modern Web search engines. One promising approach is the so-called *latent semantic indexing* (LSI) [16, 19], which is capable of locating semantically similar documents in a textual collection. Unfortunately, at present LSI is a very computationally expensive technique to be applied to the scale of the Web.

Web search engines have made significant progress in the last few years. Arguably the very first search engine on the Web was the World Wide Web Worm [35]. The paradigm of Web searching was followed by a variety of search engines such as Altavista [2], Lycos [32], Excite [20], etc. In the last few years, an innovative approach to ranking of the Web pages was introduced by Google [42] and the area of Web searching has advanced even further. At present, besides Google, there is a variety of other popular search engines (e.g., Yahoo! [55], MSNSearch [38], Teoma [47],

[13]KDD Cup required that the precision winner to appear in the top-10 F1 values and this is why Id 25 in Table 7 was not the winner. We are currently not aware of this participant's Id in Table 8 as well as his/her method that achieved this precision.

[14]The interested reader may find the details at: `http://www.acm.org/sigs/sigkdd/kdd2005/Kddcup05_Presentation.zip`

etc.) All of the aforementioned search engines answer the users' queries by performing keyword matching. In our approach however, we employ linguistic analysis in order to get a deeper understanding of the textual content in the hope that this will lead to better search results.

There are also companies such as Autonomy [4], Inquira [24], Inxight [25] and iPhrase [26] that aim to improve information retrieval through the use of language analysis. Although these companies employ some type of linguistic processing in one form or another,[15] they mainly focus on enterprise textual collections. Such collections are typically smaller and more homogeneous than the information available on the Web. Furthermore, their user base and information needs are quite different from the general Web population.

A different approach to combining linguistic analysis with the information on the Web is one that aims at creating an *answer-engine* [1, 28]. That is, given a user's query that is given in the form of a question, the engine tries to come up with a few authoritative answers. Examples of such an approach was the first version of Ask.com [3], the START answering system at MIT [46], and BrainBoost [6]. Although such approaches have potential, we believe that in most cases full sentential parsing is necessary in order to provide a truly reliable service. Other issues include inferencing, the need for common-sense knowledge, and identifying outliers, all of which are very tough challenges that remain to be solved.

7 Conclusions and Future Work

We presented Infocious, a Web search engine that employs linguistic analysis techniques and aims at helping users find information more easily by resolving ambiguities in natural language text. In realizing Infocious, we analyzed current natural language technologies (e.g. POS-tagging, concept extraction, etc.) for their benefits and trade-offs in applying them to Web-scale information retrieval. Equally important are the considerations for enabling the user to exploit the power of these technologies intuitively and transparently.

We believe that Infocious is a first step in the promising path of realizing the many benefits NLP can have in improving information retrieval, one of the most important tasks performed on the Web today. In its first incarnation described in this paper, Infocious incorporates only a few of the available NLP technologies, with great opportunities for improvement still left unexplored. It is this challenge that excites and motivates us to further bridge the gap between NLP research and Web searching. Here are some of the challenges we are currently exploring.

Word sense disambiguation: WSD accuracy suffers from the lack of training data. Fortunately, innovative approaches have been proposed to generate them automatically, such as one based on search engines [36]. Since Infocious has amassed large amounts of NLP annotated text, this resource can be used to generate training data for improving WSD models. With reliable word senses Infocious can index directly on word meanings, thus enabling users to search for a specific meaning of polysemous word, such as *living plants* versus *manufacturing plants*.

Full sentential parsing: While time complexity still remains an issue for parsing, the questions of how to represent, index, and query parsed text at the Web scale are largely left unanswered. Nevertheless, the potential benefits for parsing are great, for it can potentially provide for more precise searching, improved text summarization, question answering, machine translation, and others. Finding the best way to bring these benefits to the end user also poses many interesting challenges.

Text classification: More studies are needed to compare different classification algorithms and to better understand the dynamics of categorization errors. For example, examining categorization errors for queries with topical ambiguity, i.e., when Infocious' Categories feature is the most useful to the user, may be more important than aiming for absolute categorization accuracy.

Robustness to disambiguation errors: Even with humans, natural language disambiguation is not perfect. Hence, systems that utilize NLP information need to be robust against errors. We have taken initial steps in Infocious by maintaining probabilities from the NLP disambiguation, but more work is needed to study the impact of NLP errors on search quality, and better ways to cope with them.

Many more possibilities exist for applying our NLP annotated repository to improve other NLP tasks, such as machine translation, text summarization, and question answering. Additionally, we would like to explore the potentials of our NLP technologies to better connect businesses with potential customers. That is, we plan to investigate how Infocious can improve the relevance of advertisements through our better understanding of what users are searching for.

[15] Unfortunately detailed information on the technology of these companies is not publicly available.

References

[1] E. Agichtein, S. Lawrence, and L. Gravano. Learning to find answers to questions on the web. *ACM Trans. Inter. Tech.*, 4(2):129–162, 2004.

[2] Altavista Inc. `http://www.altavista.com`.

[3] Ask Jeeves Inc. http://www.ask.com.

[4] Autonomy Inc. `http://www.autonomy.com`.

[5] R. Baeza-Yates and G. Navarro. *Modern Information Retrieval*. Addison-Wesley, 1999.

[6] Brainboost. `http://www.brainboost.com`.

[7] E. Brill. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Washington, 1994.

[8] G. Chao. *A Probabilistic and Integrative Approach for Accurate Natural Language Disambiguation*. PhD thesis, University of California, Los Angeles, July 2003. `http://www.infocious.com/papers/chao-2003.pdf`.

[9] G. Chao and M. G. Dyer. Maximum entropy models for word sense disambiguation. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, August 2002.

[10] C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal. Web search using automatic classification. In *Proceedings of WWW-96, 6th International Conference on the World Wide Web*, San Jose, US, 1996.

[11] J. Cho. *Crawling the Web: Discovery and Maintenance of a Large-Scale Web Data*. PhD thesis, Stanford University, January 2002.

[12] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the Twenty-sixth International Conference on Very Large Databases (VLDB)*, September 2000.

[13] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proc. of SIGMOD conf.*, May 2000.

[14] J. Cho and A. Ntoulas. Effective change detection using sampling. In *Proceedings of the Twenty-eighth International Conference on Very Large Databases (VLDB)*, August 2002.

[15] N. Cristianini and J. Shawe-Taylor. *An Introduction To Support Vector Machines*. Cambridge University Press, 2000.

[16] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[17] The open directory project. http://www.dmoz.org.

[18] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[19] S. T. Dumais. Latent semantic indexing (LSI) and TREC-2. In *The Second Text Retrieval Conference (TREC-2)*, 1994.

[20] Excite Inc. http://www.excite.com.

[21] Google Incorporated. `http://www.google.com`.

[22] C.-C. Huang, S.-L. Chuang, and L.-F. Chien. Liveclassifier: creating hierarchical text classifiers through web corpora. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*. ACM Press, 2004.

[23] Infocious Incorporated. `http://search.infocious.com`.

[24] Inquira Inc. http://www.inquira.com.

[25] Inxight Inc. http://www.inxight.com.

[26] iPhrase Inc. http://www.iphrase.com.

[27] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, N.J., 2000.

[28] B. Katz, J. Lin, D. Loreto, W. Hildebrandt, M. Bilotti, S. Felshin, A. Fernandes, G. Marton, and F. Mora. Integrating web-based and corpus-based techniques for question answering. In *Twelfth Text REtrieval Conference (TREC 2003)*, 2003.

[29] T. Kudoh and Y. Matsumoto. Use of support vector learning for chunk identification. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, 2000.

[30] C. Li, J.-R. Wen, and H. Li. Text classification using stochastic keyword generation. In *Twentieth International Conference on Machine Learning (ICML)*, pages 464–471, 2003.

[31] X. Li and D. Roth. Exploring evidence for shallow parsing. In W. Daelemans and R. Zajac, editors, *Proceedings of CoNLL-2001*, pages 38–44, Toulouse, France, 2001.

[32] Lycos Inc. http://www.lycos.com.

[33] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[34] M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[35] O. A. McBryan. GENVL and WWWW: Tools for taming the web. In *First International Conference on the World Wide Web*, CERN, Geneva, Switzerland, May 1994.

[36] R. Mihalcea. Bootstrapping large sense tagged corpora. In *Proceedings of the 3rd International Conference on Language Resources and Evaluations (LREC)*, Las Palmas, Spain, May 2002.

[37] G. Miller, C. Leacock, and R. Tengi. A semantic concordance. In *Proceedings of ARPA Human Language Technology, Princeton*, 1993.

[38] MSNSearch. http://www.msnsearch.com.

[39] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[40] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, 2006.

[41] A. Ntoulas, P. Zerfos, and J. Cho. Downloading textual hidden web content through keyword queries. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, 2005.

[42] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Database Group, Computer Science Department, Stanford University, November 1999. http://dbpubs.stanford.edu/pub/1999-66.

[43] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing*, pages 133–142, 1996.

[44] S. Robertson and K. Spärck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–46, 1976.

[45] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, first edition, 1983.

[46] START natural language question answering system. `http://www.ai.mit.edu/projects/infolab/`.

[47] Teoma. http://www.teoma.com.

[48] S. M. Thede and M. P. Harper. A second-order hidden markov model for part-of-speech tagging. In *Proceedings of ACL-99*, 1999.

[49] E. F. Tjong Kim Sang. Text chunking by system combination. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, 2000.

[50] E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon, Portugal, 2000.

[51] H. van Halteren. Chunking with wpdv models. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, 2000.

[52] A. J. Viterbi. Error bounds for convolutional codes and an asymtotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–267, 1967.

[53] D. D. Wackerly, W. Mendenhall, and R. L. Scheaffer. *Mathematical Statistics With Applications*. PWS Publishing, 1997.

[54] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kauffman Publishing, San Francisco, 2nd edition, 1999.

[55] Yahoo! Inc. http://www.yahoo.com.