Shallow NLP techniques for Internet Search

Alex Penev **Raymond Wong**

National ICT Australia and School of Computer Science and Engineering,
University of New South Wales, Sydney, NSW 2052, Australia
{alexpenev,wong}@cse.unsw.edu.au

Abstract

Information Retrieval (IR) is a major component in many of our daily activities, with perhaps its most prominent role manifested in search engines. Today's most advanced engines use the keyword-based ("bag of words") paradigm, which concedes some inherent disadvantages. We believe that natural language (NL) is a more user-oriented, context-preservative and intuitive mechanism for web search.

In this paper, we explore shallow NLP techniques to support a range of NL queries over an existing keyword-based engine. We present JASE, a web application enveloping the Google search engine, which performs web searches by decomposing input NL queries and generating new queries that are more suitable for the search engine. By using some of Google's syntactic operators and filters, it creates "clever" queries to improve precision.

A preliminary evaluation was conducted to test JASE's accuracy, and results have been encouraging. We conclude that the NL model has potential to not only rival the keyword-based paradigm, but substantially surpass it.

Keywords: Information Retrieval, Natural Language Processing, Google.

1 Introduction

At present, the holy grail of IR is embodied in the World Wide Web—an ever-growing source of self-updating information that is easy to access yet difficult to discover.

Today's engines use the keyword-based paradigm, by implicitly connecting given keywords with boolean operators (and, or, not). This model concedes certain inherent disadvantages that are becoming increasingly evident as the web continues to expand—context is lost once keywords are isolated and treated on an individual basis, and many words carry double-meanings. Together, these deficiencies result in larger recall which is filled with noise, frustrating the user.

We believe that natural (or *everyday*) language is the ideal mechanism for information discovery—it is user-oriented, because it is intuitive and requires no training. It allows users to express a query in the way that it is rationalized and constructed in their mind, while both providing a context and helping to disambiguate word senses. But NL queries such as "*german or austrian composers born in the 1600s*" and "*native animals in australia, but not marsupials*" show that there is room for improvement in today's engines. More elaborate queries such as "*700ml Johnnie Walker Red Label, in Sydney for under $30*" cannot be answered at all, even when appropriate documents are indexed by the engine.

Through practice and tribulation, users learn to mentally cull their queries into a set of only the few "most important" components, before sending a request to an engine. This structural rearrangement, coupled with search engines' treatment of keywords as individuals, could be impeding their accuracy.

Furthermore, to become proficient at using a certain engine, users must learn its special operators. These differ between each engine and render the search mechanism to be unnatural, due to the introduction of foreign modifiers into the query. Our evaluation survey indicates that average Google users are largely unaware of its operators and filters, and rarely use them in practice.

There has been few significant advances in Internet search for half a decade, and the shortcomings of the keyword-based paradigm are likely to be globally costing millions of hours each year in labor for wading through voluminous results. Meanwhile, the Internet continues to grow and permeate our way of life, and search results become larger and potentially noisier. Therefore, we believe that NL will play a principal role in web and media search in the near future, because it is more intuitive and provides more information than the current model.

In this paper, we explore shallow NLP techniques to support NL queries over Google. We evaluate the performance and accuracy of JASE, an application enveloping Google, which decomposes NL queries to form Google-friendly queries and reranks the retrieved results. We define a categorical classification of searchable entities and highlight how they can be used in conjunction with Google's advanced operators and filters. We propose heuristics that can be used for the reranking step. Finally, we assess our system's performance for a set of keyword and NL queries.

The remainder of this paper is organized as follows; §2 provides an overview of search engines, and defines our problem domain. §3 outlines the algorithmically-disparate phases and data structures of our system (explored in greater detail in §4 and §5). We conduct a comparison of the accuracy of our system against Google and "average Google users" for a mix of keyword and NL queries in §6. §7 covers related work, and §8 concludes this paper.

2 Background

As JASE is a wrapper for a search engine, it is vital to understand how search engines work. This allows

us to determine which subprocesses are to be implemented by JASE and which are delegated to Google.

2.1 Search Engines

A search engine is an online program which, for a given query, retrieves references to web documents that match it. In theory, a search engine has four components:

document processor indexes new documents. Indices are a mapping between words and what documents they appear in. Most engines are spider-based, so a crawl of the web for new documents and the updating of the index is automated.

query processor inspects a user's query and translates it into something internally meaningful.

matching function uses the above internally meaningful representation to extract documents from the index.

ranking scheme positions the more-relevant documents on top, using some relevance measure.

Users communicate with the query processor, which is the only visible component. It carries out several tasks, usually (but not limited to):

- tokenizing of the query to remove invalid characters, and to recognize meta-keywords or special syntactic operators.
- removal of stopwords; words which are too common and rarely help in the search (e.g. the, a, of, to, which).
- stemming; a process designed to improve the performance of IR systems, involving normalizing semantically similar words to their root forms (e.g. produce, produced, producer, producers, produces and producing map to *produc-*).
- assigning a weight to each keyword/keyphrase, to aid with ranking(Salton & Buckley 1988).

After results are retrieved by the matching function, they are ranked by relevance based on some ranking measure and set of heuristics (called the ranking scheme). Often taken into account are:

term frequency how many times keywords appear in the document(Luhn 1957).

inverted document frequency a value which aims to determine how important a term is in discriminating a document from others(Salton 1989, Jones 1972).

semantic proximity words synonymous to a given keyword may be matched, boosting the score of the document.

term position keywords appearing in the title or heading (rather than the body) should contribute more to a document's weight.

term proximity a document in which the query terms are close together is considered more relevant than one in which they are far apart.

cluster distance how far apart groupings of matched terms are.

percentage of query terms matched.

In our case, JASE implements the query processor and the ranking scheme, while Google provides the document processor and matching function. Of course, Google utilizes its own query processor and ranking scheme for any query that it answers, and therefore JASE's results will be heavily influenced by Google's own relevance measure. We exercise some indirect control over these components, since JASE's query processor is invoked before and JASE's ranking scheme is invoked after Google's.

2.2 JASE

The initial design decisions for our system were that JASE will be a web application into which a user enters free text (preferably NL) in a query box, presses a button, and views the corresponding search results. It should implement the query processor and ranking scheme components of a search engine, so that it can influence what was being sent to Google, and influence what was being relayed back to the user.

Behind the scenes, JASE would invoke Google via an API[1]. Google is not designed to handle NL queries, so JASE would have to manipulate the input to make it Google-friendly. It would also take advantage of the syntactic operators and filters to try and improve precision. JASE would display the same information that Google displays—a title, URL and snippet for each matched document. We had decided not to collect any further information regarding documents, such as a downloading of the actual source.

2.3 Google's API

Google (the company) provides many online services, most important of which is Google (the search engine). This engine is useful for JASE because it:

- advertises a free API (over WSDL/SOAP), allowing it to be remotely queried from many programming languages and environments.
- provides the title, URL and snippet of matched documents, which serve as a *synopsis*.
- has powerful operators and filters: +, -, ~, OR, intitle:, inurl:, site:, filetype:, numerical ranges, timestamps, related:, link:, and some limited wildcard matching.
- performs stemming.
- is case-insensitive.
- ignores most punctuation.
- uses sophisticated link- and structure-based analysis to determine the importance of documents on both global (e.g. PageRank(Page, Brin, Motwani & Winograd 1998)) and per-query scales (e.g. anchor text, keyword proximity, and whether or not keywords appear emphasized with markup in the document).

Some limitations of the API are that a maximum of 10 terms can be sent per query, that languages other than English are poorly supported, and that only 10 results can be retrieved per query. Case-insensitivity is a pro because it allows users to be sloppy, but also a con because acronyms and proper nouns will sometimes match unrelated documents. The API is still in the beta stage, and its functionality may be altered at any time.

These points affect the methods and messages that JASE can use to communicate with Google. Obviously, we cannot, for example, perform web search

[1] http://www.google.com/apis

using regular expressions, because the engine does not offer such functionality.

2.4 Problem Domain

One of the challenges of applying NLP over English is that the language allows many syntactic variations of sentences. Semantically identical questions can usually be worded in more ways than one: *how old are you?*, *what is your age?*, *which birthday did you celebrate this year?*. Questions with logical constraints can be permuted even further; for example, with composer as subject, and logical constraints on a date of birth and nationality, we can ask for "Which composers were born in Germany or Austria between 1600 and 1700?", or "What are the names of some German or Austrian composers born in the 1600s?", or "In the 17th century, what famous composers were born in either Germany or Austria?", and many others. The wording of the query will differ between people, but all of these will have somethings in common—a composer, a nationality German/Austrian, and a year of birth. These are the entities that we wish to extract, whereby two variations on the same query will yield the same decomposition.

It is important that we restrict the types of queries we want to handle, as the general query domain is too overwhelming. We focus on a subset of all possible query structures, to roughly satisfy the grammar:

- `P = ( K | Y | M )`
  where `K` = some keywords that form a phrase
  `Y` = year phrase
  `M` = money phrase
- `PS = P ( connective? P )+`
  Two or more disjoint runs of phrases. The connectives will mostly be conjunctions, prepositions or adverbs.
- `( P | PS ) negation+ P`
  Things that the user does not want, e.g. "*[American presidents] [except] [Bush]*". Negations are usually negative coordinating/correlative conjunctions or adverbs.
- `( P | PS ) ''in C'' (P | PS )?`
  where *C* is some location, e.g. "*[composers born] [in Germany]*". In general, words that follow *in* will rarely be locations, so we must explicitly provide a list of permissible matches. We can use such matches to focus queries towards certain domains, such as *.de*, for the given example.

Some typical examples of the atoms include:
`K` = *jujitsu*
`Y` = *1920*, *500 BC*, "*during the 90s*"
`M` = *$60*, "*cheaper than $10*", "*between $5 and $10*"
`PS` = "*[endangered animals] in [australia]*", "*[composers born] [during the 1600s]*".

The above is not intended as a formal grammar, but as a guide to visualize possible queries. JASE still perform a best-effort search, irregardless of whether a query satisfies the above grammar.

3 System Overview

Our system is active at the beginning and end of the search session and we follow a typical Service Oriented Architecture, with a consumer (JASE) and provider (Google). Consequently, we are able to partition the functionality in two disparate phases.

Phase One JASE acts as the query processor. The search query is parsed and decomposed. Decomposition involves tokenizing the text to discover structural objects, such as words, numbers and punctuation. These must be stored as some internally-meaningful representation, which we call the **SearchTerms**. The contents of this container are the seed for generating new queries.

Phase Two JASE acts as the ranking scheme. In Phase One, variants of the input query are submitted using the API calls, and each retrieves up to 10 results. This leaves us with many "best" ranked documents, as each result set has its own top match (note that some result sets are likely to overlap). We must perform a reranking on the results as a whole, based on some relevance measure. We rerank by assigning a score to each document and sorting. Our heuristics for calculating a document's score are described in §3.2.

3.1 SearchTerms, a link between Phases

At the beginning of Phase One, useful information is extracted from the query, grouped and then classified in the SearchTerms container. In Phase Two, retrieved documents are compared against these SearchTerms to receive a score. This container is the connection between the phases, which are otherwise independent. The container is composed of several sets of "searchable entities", which we define as:

input query as a unit phrase.

primary keyphrases extracted keywords and keyphrases. Phrases are the most specific and discriminatory part of queries, thus adjacent keywords should be grouped as a phrase wherever possible. A phrase may contain a singleton word, if it happens to be bounded at both ends by non-keywords. This set is never empty.

secondary keywords primary keyphrases are broken down into their individual atomic keywords, each becoming a secondary keyword. This set may be empty, since it will not contain keywords that are already primary.

tertiary words any remainder terms from the original query, which have not been categorized as primary or secondary are inserted here. This set will largely consist of stopwords.

synonyms/hyponyms/meronyms for singleton primary/secondary words come from an external source, such as WordNet[2].

- synonyms are words with corresponding meaning, e.g. alcohol/liquor.
- hyponyms are more-specific words, e.g. dog/poodle.
- meronyms are parts of a larger whole, e.g. dashboard/car.

exceptions are undesirable matches; things the user does not want.

Two additional pieces of information are also recorded. The first is a list of numerical upper and lower bound tuples, used to apply numerical range matching. As a proof-of-concept, JASE supports dates and prices, but other ranges are possible to detect and match. The second datum that we record is a domain restriction, in order to restrict some queries to a specific domain. At present, we handle mappings

[2] http://www.cogsci.princeton.edu/%7ewn/

from the ISO 3166[3] list, with a few obvious adjustments (e.g. removal of *.us*).

3.2 Weighting

The SearchTerms sets are assigned base weights, representative of the desirability of their inclusion. Each document in a result set is assigned a score, which is derived by comparisons against the SearchTerms. When grading a document, we consider only the title, URL and snippet—no extra information about documents is obtained. Using only the title/URL/snippet allows us to evaluate JASE's performance and accuracy using information which is made visible by the search engine. Digging deeper and downloading the source of documents may lead to more accurate scoring, but is outside our scope.

As matching a phrase should be more desirable than matching only one word of that phrase, one would expect phrases to contribute more than words. The location of the match (title/URL/snippet) also affects its contribution. Table 1 defines the base weights for the SearchTerms. These values are subject to tweaking as there is no correct answer, but they seem to work well in practice. Certain matches may appear in more than one set, in which case the higher weight is used. We also use stemming, casemismatch and term frequency (see §5.1).

Entity	title	URL	snippet
Input query	3.0	3.0	1.2
Primary	1.0	1.0	0.4
Secondary	0.5	0.5	0.2
Synonym	0.3	0.3	0.12
Hyponym	0.3	0.3	0.12
Meronym	0.3	0.3	0.12
Tertiary	0.2	0.2	0.08
Exception	-10	-10	-4

Table 1: Base weights of SearchTerms entities

4 Phase One

This phase involves the parsing and tokenizing of the input query, to build the SearchTerms container. Extracting the most sensible keyphrases and keywords from the user's NL query is critical, as Google is a keyword-based engine and its results will heavily fluctuate depending on what terms are chosen. Our strategy is to create "clever" Google-specific queries, which contain several of:

keywords are important and discriminatory words. Keywords can be directly sent to Google.

keyphrases are sequences of keywords, where order is important. JASE detects them by looking for phrasal boundaries. Keyphrases must be quoted to be recognized by Google, e.g. "*vampire bats*".

exceptions are terms the user does not want, often explicitly stated. Both word and phrase exceptions must be preceded with a minus, e.g. "*pets -dogs*".

domain restriction concentrates a search on a particular domain, e.g. "*national park site:nz*".

synonymy refers to terms which are related to a keyword, but they need not be specified by the user. We are only interested in three categories: synonyms, hyponyms and meronyms. Others categories exist (such as hyper/anto/pertai/holonyms), but are not useful for this task. Google will look for some synonyms if a word is preceded by a tilde; for example, the search "*~movie -movie*" matches *video*, *film*, *dvd*, *mpeg*, *cinema*, *soundtrack* and *trailer*. JASE does not perform query expansion using synonymy, but it does look for them when scoring a document.

numerical ranges place a constraint for lower and upper bound matches. Ranges of the form *lo..*[*hi*] are supported by the engine, and some units are also recognized (e.g. "*beethoven symphony 8..*", "$50..60" for price and "100..200 kg" for weight).

One of the advantages of automating web search is the ability to fire off many different queries and selecting only the best results. JASE emits between two and twenty new queries for a given search, depending on the complexity of the query, and how many of the SearchTerms sets are utilized. Some previous work (e.g. (Kwok, Etzioni & Weld 2001, Agichtein, Lawrence & Gravano 2001)) has empirically shown that such an immediate increase in recall, despite its overhead, is a very effective. This strategy, however, raises a few issues. Different queries must be sent each time, prompting the need for a mechanism to formulate slightly variant queries, using the SearchTerms as a seed. Each query will also have its own top result, so an equitable reranking mechanism is needed. Finally, some documents are likely to be returned by several queries, therefore must be clustered as one. JASE addresses each of these points.

4.1 Detecting Keywords and Keyphrases

Known techniques for locating phrases in written text tend to use vector-space weighting algorithms, naïve Bayesian classifiers, inverse document frequency (IDF) tables, lexical chains, or other statistical means. However, these are all intended to be applied to whole documents, and are trained on a specific corpus. In contrast, we are dealing with a single line of input, ranging from one to maybe fifteen words. Such confined input makes it difficult to use statistical models, especially since many phrases will contain proper nouns and not be found in any corpora. From these, we feel that an IDF table is the only suitable approach.

Another NLP technique is to deduce the parts of speech using a Part Of Speech tagger. These can be rule-based(Brill 1992, Brill 1995) and follow patterns, unigram or n-gram based, or Hidden Markov Model based(Charniak 1994, Charniak 1997, Collins 1996) and follow probability. The tagger can be used to detect phrases by collecting disjoint runs of nouns and adjectives. Many taggers exist, but JASE does not use one. Instead, JASE guesses the location of phrasal boundaries by splitting on stopwords, using our own custom 99-word list (a hybrid of Google's and Snowball's[4], with additions). Since stopwords are those with a poor IDF, this strategy emulates the use of an IDF table to some degree.

This means that the "naturality" of the language used is important for adequately deducing phrases. On par with previous NL systems that we have played with, it is not difficult to construct complicated but unnatural-sounding queries to confuse JASE. For our evaluation (§6), we used sensibly-worded queries. Our shallow NLP approach works well for many of the TREC queries, and using deeper NLP will only serve to improve accuracy.

[3] http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

[4] http://www.snowball.tartarus.org

4.2 Detecting a domain restriction and numerical ranges

JASE pattern matches the user's query for countries. If preceded by *in*, *in the*, *from* or *from the*, it will focus approximately half of the generated queries towards that country's domain. Most country domains have some web pages written in English, so it not unreasonable to carry out such searches.

JASE also pattern matches year and monetary phrases, to demonstrate how numerical ranges can be extracted. It handles many cases, best illustrated by some examples (Table 2). Date matching is capped at year 9999. If a lower or upper bound of a monetary range is not specified, JASE assumes 10% for the lower bound and a string of nines to one more significant figure for the upper bound.

Phrase	Range
Rock stars in the 60s	1960..1969
Rock stars in their 60s	60..69
Rock stars before the 60s	0..1959
Wars before 1066	0..1065
Wars during 1066	1066..1066
Roman emperors before 20 BC	21..99
A Kodak camera cheaper than $200	$20..200
A Kodak camera, over $60000	$60000..999999
$49.99 Playstation controller	N/A
$49 Playstation controller	$49..50
Playstation controller between $49 and 60	$49..60

Table 2: Some examples of handled ranges

5 Phase Two

This phase deals with the reranking of all query results, using relevance heuristics against the SearchTerms constructed in Phase One.

All documents are assigned a numerical score (or weight) based on the relevance measure. The documents are sorted, and only the top 20 are displayed to the user. As mentioned previously, JASE only uses the synopsis of a document (title/URL/snippet) to weigh a document.

Two morphological processes are carried out before documents are inspected—folding case and stemming. The original synopsis is used for case comparison; case-insensitive match is performed on the lower-case version, and stem matching is performed on the stemmed version. As Google uses stemming, it is possible to encounter a partial match in the document synopsis, which needs to be rewarded. We use a Porter stemmer(Porter 1980).

5.1 Relevance Measure and Heuristics

Upon meeting a document in the result set, its synopsis scanned for entities within the SearchTerms sets. As described in §3.2, certain sets are more important than others and contribute different base weights, and the location of a match also affects the contribution. It is furthermore influenced by the nature of the match:

- if a term is indirectly matched via stemming, its contribution is penalized by 25%.
- if case agrees, its contribution is boosted by either 25% if it appears in the title/URL, or 10% otherwise.
- if a term is matched via some numerical range, its contribution is penalized by 25%.
- if every primary and secondary term is matched at least once, the overall document score is boosted by 50%.

These figures are subjective and there is no correct answer, but they work well in practice.

5.2 Overall Score of a Document

A naïve algorithm to determine the total score for a document would be to sum all individual contributions for all SearchTerms entities. If an entity is matched, it contributes its base weight less penalties plus bonuses, and if it is not matched, it contributes nil. This approach suffers the problem that repetitive matching of a single entity, while matching few or even none of the others, results in a score that unfairly represents the document. This weakness becomes more evident with small result sets like JASE's, as they are more volatile to fluctuations in rating. JASE cannot get an "averaging out" effect without retrieving far larger results sets, and the snippet only partially communicates the content of a document.

Clearly, such a biased boost due to multiple matching of a single entity is inappropriate. But at the same time, we do not want to ignore recurring matches entirely, because it is desirable for every match to contribute "something". To deflate the volatility of recurrent matches, we use a recessive geometric sum to calculate the score of a document:

$$score(doc) \quad = \quad \frac{1}{G} \quad + \quad \sum_{e=e_1}^{e_n} \sum_{i=1}^{j} \frac{\omega_i}{2^{i-1}}$$

where $\{e_1..e_n\}$ are the n searchable entities in the SearchTerms. The set of weights $\{w_1..w_j\}$ represent the contributions by the j occurrences of entity e matched in *doc*'s synopsis. This set of weights is sorted in descending order to maximize the overall result.

Such a summation guarantees that every occurrence of a matched entity e contributes to the document, but no entity can contribute more than twice its highest individual match to the overall document score (recall $\sum_{i=1}^{\infty} \frac{1}{2^{i-1}} = 2$). This satisfies our above desiderata that superfluous entity frequency should not bias a document "too much", but that every occurrence of any entity should contribute.

Furthermore, the reciprocal of Google's suggested rank, G, is augmented to a document score. This bonus contributes as much as 1.0 point for Google's topmost result, and is worth the same as a matching of a primary keyphrase in the title. Because this bonus diminishes for lower ranks in Google's list, it is a means of discriminating between documents that receive similar scores against the SearchTerms, but appear in different ranks in the original results.

5.3 Reranking Documents

The previous section describes how a score is assigned to a document. However, some queries may overlap and documents may be met more than once. We do not wish to display the same document several times over, so the scores for a certain document must be accumulated. Possibilities include a running total, an average, or a recessive geometric sum like the previous section. We use the geometric sum because it is a middle ground between the running total and the average—two methods which work well in most cases, but spectacularly fail for some scenarios.

Once documents are reordered, the top 20 are displayed to the user.

6 Evaluation

This section serves as a summary of our system's accuracy and performance. All results in this section were carried out during Oct 2004.

6.1 Accuracy

An evaluation survey was prepared, and answered by 8 volunteers. All were fluent in English, and all confirmed that they use Google at least once per week, with more than half using it daily. Participants were of mixed age (18–49), mixed gender and mixed nationalities (USA, Australia, UK, France and Switzerland). One participant had a computer science background. We feel that this sample represents "average" Google users to a reasonable degree.

Participants were first asked about their searching prowess, shown in Table 3.

Question	Yes	No
Do you know of operators: +, -, “”?	75%	25%
Do you use them?	38%	73%
Do you know of filters: site, inurl, intitle?	-	100%
Do you know of indirect matchers: ~, x..y?	-	100%

Table 3: Supplementary questions answered by participants

While most knew of the plus, minus and quotes operators—used for inclusion, exclusion and phrases—only half of those admitted to actually using them in practice. A small proportion were familiar with the *site:* filter only, but none of the other operators.

Our survey contained 14 query topics in roughly increasing complexity, which are listed in Table 4. Some queries consisted of only keywords, while others were written in NL. We opted to avoid relying on the TREC test set, because many web documents explicitly quote the TREC queries in the context of TREC, yet are unrelated to the topic at hand. Only q6–q9 are TREC queries.

Id	Query
q1	microsoft
q2	belgian comic strip characters
q3	endangered animals in australia
q4	what happened at the final in the 2002 world cup?
q5	German or Austrian composers born in the 1600s
q6	what is the treatment for alzheimer's?
q7	how much sugar does Cuba export and which countries import it?
q8	the consequences of implantation of silicone gel breast devices
q9	what diseases have hair loss as a symptom?
q10	important discoveries in medicine during the 1600s
q11	A used Toyota Camry 1998 model, in Sydney between $5000 and $10000
q12	700ml Johnnie Walker Red Label in Sydney for under $30
q13	Famous people born on May 1 between 1900 and 1950
q14	I want to do Artificial Intelligence in the best university in Australia

Table 4: Base weights of SearchTerms sets

A scale from 1 to 5 was defined (1 being poor, 5 being excellent). For each query, the top ten Google[5] results were provided. Participants were asked to rate the result set based on their impressions and opinions in regard to accuracy. Participants were then asked to perform their own search using the search engine, and could rewrite the query in any way using any tactics they wished. Their goal was to find relevant documents that satisfied the query. They then proceeded to rate their own results. Finally, participants rated JASE's results for the original query. In all cases, participants were encouraged to view the actual documents retrieved by visiting the hyperlinks.

JASE received favorable ratings, outperforming Google for some queries, while being approximately equal for the remainder. This result was not unexpected—for simple keyword queries, JASE displays almost exactly what Google does, but for more complex queries, its Google-optimized queries and reranking appeared to improve accuracy.

But as one of the aims of this work was to show that NL queries can be used to improve the precision of web searches, we were more interested in how JASE would fare against the participants themselves. Figure 1 shows how each of Google, the participants and JASE performed for each query. A 95% confidence interval is provided, using the t-distribution.

Google's Accuracy
This form of web search represents "unskilled" searching. Participants rated the Google's raw search results for each verbatim query. Simple keyword-only queries received high scores, but ratings gradually fell as queries became more complex and involved NL. This behavior was expected.

Participants' Accuracy
This form of web search represents "semi-skilled" searching. Participants used the given query to create their own new query based on their experience and knowledge of the search engine, and evaluated the results. Observed tactics included phrasal search, domain restrictions, and query expansion. As expected, participants were able to slightly outperform the "unskilled" search.

JASE's Accuracy
This form of web search represents "skilled" searching. Participants rated JASE's search results for each verbatim query. JASE kept up with both unskilled and semi-skilled searches for the simpler queries—which were mostly keyword-based to test the accuracy of JASE's reranking—but maintained a lead for the second half of queries, which were written in NL and highlighted JASE's advantage of dispatching multiple queries and usage of an engine's filters and operators.

One observation is that JASE's confidence interval does not overlap the others for some of the NL queries. According to the t-test, this is a statistically significant result. The queries that caused this (q5, q10, q12 and q13) highlight JASE's advantage of using Google's filters and operators (the ones used here were “”, +, ~ and numerical range). These queries shared in common a need to match text which was implied, but not explicitly stated. Most other queries did not have such implications, and relied on direct keyword-matching. JASE received high ratings there too, most likely due to its increase in recall by retrieving results from multiple queries, and subsequent reranking of results. This conclusion is consistent with previous works (see §7).

The results indicate that JASE outperformed the users themselves for this query set, which involved queries of various difficulty. This suggests that NL queries may be used to improve the accuracy of web search, through shallow NLP systems such as JASE.

[5] http://www.google.com

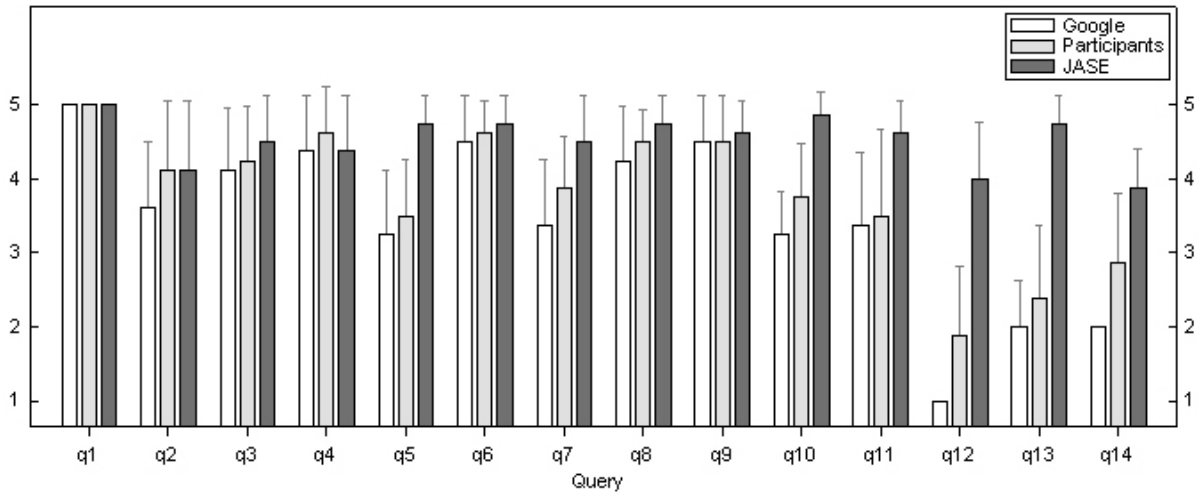

Figure 1: Mean rating per query, with 95% CIs

While expert users may be able to match or surpass the precision of such a system, the system should be beneficial to average users, who are unfamiliar with the esoteric art of accurate web search.

6.2 Performance

As a synchronous wrapper around Google, JASE is inherently slower than Google. There are three separate time periods involved from the moment a user inputs their query for processing until the moment the results are displayed:

GT Google time, consumed by the search engine in answering our queries. It is conveniently reported by the API methods.

CT Communication time (`round trip time - GT`). Timing begins with the invocation and return from the SOAP library routines.

JT JASE time, time during which JASE's code is active, including bootstrapping of any libraries and classes.

The time periods are mutually exclusive, and the total time taken is `TT = JT + CT + GT`. Depending on the complexity of the input, JASE emits up to about 20 queries in total.

Our empirical tests indicated that `JT` $\approx$ `0.11 TT`, `GT` $\approx$ `0.21 TT` and `CT` $\approx$ `0.68 TT`. The biggest performance cost, `CT`, represents networking and communication. The tests suggested that our prototype was not inefficient, as the bulk (89%) of loading time was consumed externally. Our prototype took just under 20 seconds to load for the more complicated queries, because we submitted queries serially. This value may be greatly decreased by issuing queries in parallel, but speed was not our goal.

Participants in our evaluation were asked their opinions on search response times. All agreed that an extra 5 seconds wait on top of Google's average response time (which is between 0–0.5s) in order to produce more accurate results is admissible. In fact, 75% agreed that even 20s was admissible. Such an answer hints at web users' desideration for a system such as JASE.

7 Related Work

Many NLP search systems have been made to date. In particular, a form of NL search called Question-Answering Systems have been well-explored (e.g. (Katz 1991, Kwok et al. 2001, Prager, Brown, Coden & Radev 2000, Ravichandran & Hovy 2002)).

QAS accept wh-questions (who, when, where, what, why) and return a definitive answer. QAS are related to search engines because they retrieve information from a source based on a query. Unlike search engines, QAS provide an answer, rather than a list of top "hits". To do this, QAS need to have at least some idea of what the user is searching for. As such, QAS usually extract knowledge from the query itself—is the user asking for a person, a date, a location, an object, or what? On the other hand, search engines use whatever input they have been given with minimal restrictions on format and structure. QAS have much stronger restrictions on the structure of the input, in order to make it possible to determine what the user is looking for.

START(Katz 1991) was the first online QAS, and focused entirely on geography and MIT-specific knowledge. It used subject-*relation*-object tuples to extract the subject, relation and object from a given query, and then performed a pattern match for the tuple in its knowledge base (KB). The knowledge base was built from a similar process of detecting tuples from scanned documents. START's KB was highly-edited, and non-scalable, and the system could not provide an answer to a query if it failed to match the tuple. Future work(Katz & Lin 2000) found ways to automate the expansion of the KB, but the process was impractically slow.

Search engines, on the other hand, provide references to documents, irregardless if they answer the user's questions or not, and rarely try to "understand" the query.

Ionaut(Abney, Collins & Singhal 2000) was an interactive NL search engine, and used a local cache of documents for its KB with a small coverage. Its most interesting feature was to list related hyperlinks to a given query to branch into different queries, as a means of an iterative search. From experience with it, we feel that its accuracy was lacking, but the interactivity feature was its best asset.

Limited KB coverage is a hurdle that can be overcome. In 1993, MURAX(Kupiec 1993) used boolean searching over an online encyclopaedia, by formulating queries based on the phrasal content of the input wh-question. Noun-phrase hypotheses were extracted from the retrieved results, and new queries were independently made to confirm the hypotheses. Its accuracy was poor, but the concepts of multiple queries and formulating different queries inspired some future

works.

Tritus[6] was an NL search engine(Agichtein et al. 2001) that could use either Google or Altavista as its KB. It handled simple wh-questions that matched specific templates, but used the engines' syntactic operators to improve precision. The authors performed comparisons between different engines, and argued that Google was superior to both Jeeves and Altavista, and that Tritus' Google-optimized queries outperformed raw Google. We consider their testing to be inconclusive, as it is unfair to pipe direct NL queries to a keyword-based engine. An evaluation of this type must involve the users of the engine themselves, who know better than to submit NL queries; the users' employment of the engine should be the real competitor. One further difference between Tritus and JASE is that our work attempts to handle a broader range of input (by not using templates), and that JASE received no training.

Also in 2001, MULDER(Kwok et al. 2001) tried to scale QA to the Web, using Google as back-end. Like MURAX and Tritus, it generated multiple new queries from the input, to increase recall. Query generation was achieved by rearranging the input wh-question to match the potential phrasing of its answer—when asked "what is the capital of Sudan", it would look for documents containing "the capital of Sudan is". MULDER worked on the assumption that the Web is host to more truths than falsities, therefore wh-questions could be answered by collating the results and clustering them. The largest cluster was considered as the correct answer, due to the original assumption. MULDER used deep NLP, but imposed structural limitations on its input queries. We have not had the opportunity to test it, as it has not been available for several years. We believe that MULDER would have performed well for trivia questions because they are frequently cited online, but that it would have had difficulty in answering more elaborate queries such as those in the introduction, for which a rearrangement of the query is unlikely to appear in any online documents.

AskJeeves[7] is advertised as a QAS, but its menu-driven dialogue is more inherent to search engines. It allows searching for both keywords and wh-questions. To answer a NL question, the text must match one its question templates; otherwise, web results are retrieved from Teoma[8]. If a template is matched, AskJeeves provides links to authoritative sites which are known to answer that question. This strategy requires human editors to map the templates to the authoritative sites, and does not scale well. It takes little effort to formulate a NL wh-query which fails to match a template, yet is competently answered by a raw Google search. (Kwok et al. 2001) empirically argues that AskJeeves is limited and awkward to use, and performs poorer than Google.

Intermezzo(Flank 1998) used NLP techniques to retrieve images from an image database based on NL queries, achieving a precision of almost 90%. The content of each image was identified via captions, which were manually written. One of Intermezzo's interesting features was using WordNet to match related terms in the caption to increase the score of an image. This strategy proved effective as the images show physical objects, which have large collections of applicable hypo/hypernyms and mero/holonyms. Such a strategy of boosting term weights using WordNet's synsets is less effective for web queries over large document collections, since matching hyper- and holonyms is less appropriate.

[6] http://tritus.cs.columbia.edu

[7] http://www.ask.com

[8] http://www.teoma.com

Keyword-extraction has a long history and is a component in most IR fields. Several recent approaches to deducing the keyphrases in a piece of text exist (Turney 1999, Turney 2000, Munoz 1996, Frank, Paynter, Witten, Gutwin & Nevill-Manning 1999). However, these methods are intended to extract phrases from entire documents by employing holistic statistical models, while we are interested with extracting useful words and phrases from a single-sentence query. Hence, the algorithms from such works do not apply.

JASE shares the ideas of many of these previous works, such as using an NL wrapper around a boolean data source, and the submission of multiple queries. Because it is not a QAS, it is fundamentally different to MURAX and MULDER in that it imposes less restrictions on input, but therefore cannot use the query structure to its advantage. Tritus was a hybrid, retrieving hyperlinks like a search engine, but handling wh-questions and using the structure of the input to its advantage like a QAS.

Our work aims to be a search engine, but without being able to significantly rely on the structure of the query. The work presented in this paper is most closely related to MULDER and Tritus.

8 Conclusions

This paper has outlined some simple strategies to support NL queries over a keyword-based engine (Google). We have presented an evaluation of a search engine wrapper, JASE, that handles both keyword and NL queries. We parse, tokenize and extract searchable entities from the query, and categorize them into weighted sets. We dispatch multiple queries, and then use the sets against our ranking heuristics to weigh and rerank the retrieved results. Although only shallow NLP techniques were used, they seem work well for many cases, as indicated by our evaluation. Our evaluation survey furthermore revealed that our participants were all willing to sacrifice a large amount of performance in lieu of accuracy, therefore the submission of multiple queries is a justifiable strategy, and may be incorporated in current engines.

Future expansions that we are exploring include deep NLP—a tagger(Brill 1992, Brill 1995) coupled with an IDF table will greatly improve phrase boundary detection. Geographical locations can be detected using a gazetteer, allowing domain restrictions to be used more liberally. Finally, collecting multiple snippets for the top few documents should help improve reranking. This is our preferred way of expanding a document's summary (as opposed to downloading the entire document from the Google Cache), and is achieved by using a combination of *site:*, *allinurl:* and *allintitle:*, coupled with an extra keyword to variate the snippet.

The process of parsing and tokenizing the input to detect important searchable entities is obvious, as these are tasks that the human mind perform when presented with a query topic. But strategies such as submitting multiple queries and reranking of large result sets are only fit for computers. Our evaluation survey revealed that average Google users seldom use its operators and filters, which could be adding noise to their searches and costing them time. It therefore appears beneficial to provide a transparent system that utilizes the power of such strategies behind the scenes, rather than educate everyone to become an expert user. Such a system could accept NL as input, because it is the most intuitive "query language" and provides more information to the engine than the current paradigm.

Our experiments show that the users were able to make use of their own experience of the engine, their awareness of its idiosyncrasies and/or some trial and error to formulate a better query than a given NL topic, and slightly improve on precision. Yet they still favored JASE's results in many cases, which emphasizes an automaton's advantage of redundant and monotonous computation, and use of an engine's syntactic operators and filters.

We believe our results indicate that NL search systems such as JASE can have practical use in society, and that the NL paradigm can be used to improve the precision of web search.

References

Abney, S., Collins, M. & Singhal, A. (2000), Answer extraction, *in* 'Proceedings of the Sixth Applied Natural Language Processing Conference', Morgan Kaufmann, pp. 296–301.

Agichtein, E., Lawrence, S. & Gravano, L. (2001), Learning search engine specific query transformations for question answering, *in* 'World Wide Web', pp. 169–178.

Brill, E. (1992), A simple rule-based part of speech tagger, *in* 'Proceedings of the Third Conference on Applied Natural Language Processing'.

Brill, E. (1995), 'Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging', *Computational Linguistics* **21**(5), 543–565.

Charniak, E. (1994), Statistical language learning, *in* 'Language and Computers 12', The MIT Press.

Charniak, E. (1997), 'Statistical techniques for natural language parsing', *AI Magazine* **18**(4), 33–44.

Collins, M. J. (1996), A new statistical parser based on bigram lexical dependencies, *in* 'Proceedings of the 34th conference on Association for Computational Linguistics', Morgan Kaufmann, pp. 184–191.

Flank, S. (1998), A layered approach to nlp-based information retrieval, *in* 'Proceedings of the 36th ACL and 17th COLING', Morgan Kaufmann, pp. 397–403.

Frank, E., Paynter, G., Witten, I., Gutwin, C. & Nevill-Manning, C. (1999), Domain-specific keyphrase extraction, *in* 'Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence', Morgan Kaufmann, pp. 668–673.

Jones, K. S. (1972), 'A statistical interpretation of term specificity and its application to retrieval', *Journal of Documentation* **28**(1), 11–21.

Katz, B. (1991), 'Text processing with the start natural language system', *Text, ConText, and HyperText: writing with and for the computer* pp. 55–76.

Katz, B. & Lin, J. (2000), Rextor: A system for generating relations from natural language, *in* 'Proceedings of the ACL 2000 Workshop on Natural Language Processing and Information Retrieval'.

Kupiec, J. (1993), Murax: a robust linguistic approach for question answering using an on-line encyclopedia, *in* 'Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval', ACM Press, pp. 181–190.

Kwok, C., Etzioni, O. & Weld, D. (2001), Scaling question answering to the web, *in* 'World Wide Web', pp. 150–161.

Luhn, H. P. (1957), 'A statistical approach to mechanized encoding and searching of literary information', IBM Journal of Research and Development, 4(4), 600-605.

Munoz, A. (1996), Compound key word generation from document databases using a hierarchical clustering ART model. IDA, Amsterdam.

Page, L., Brin, S., Motwani, R. & Winograd, T. (1998), 'The pagerank citation ranking: Bringing order to the web', Stanford Digital Library Technologies Project.

Porter, M. (1980), An algorithm for suffix stripping, *in* 'Program', Vol. 14, pp. 130–137.

Prager, J., Brown, E., Coden, A. & Radev, D. (2000), Question-answering by predictive annotation, *in* 'Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', ACM Press, pp. 184–191.

Ravichandran, D. & Hovy, E. (2002), Learning surface text patterns for a question answering system, *in* 'Association for Computational Linguistics Conference'.

Salton, G. (1989), *Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley Longman Publishing Co., Inc.

Salton, G. & Buckley, C. (1988), 'Term-weighting approaches in automatic text retrieval', *Information Processing and Management* **24**(5), 513–523.

Turney, P. (1999), 'Learning to extract keyphrases from text', Technical Report ERB-1057, National Research Council, Institute for Information Technology.

Turney, P. (2000), 'Learning algorithms for keyphrase extraction', *Information Retrieval* **2**(4), 303–336.

8.1 Appendix: Sample

Figure 2 is a screenshot of our prototype for q13, with the corresponding Google results in Figure 3. Both images date to the time of our evaluation (Oct 2004).

In June 2005, we revisited the top Google hits for this query. The top 10 hits were different to before, perhaps due to PageRank fluctuations and new documents being introduced. JASE's results were, in turn, equally affected. We inspected each of Google's top 20 hits and decided that 3 were relevant, but only one of them appeared in the top 10. JASE managed to extract 8 relevant documents, of which 6 were in its top 10. We were able to identify four times as many "famous people" (to answer the query) from JASE's top 10 hits than from Google's top 20 hits.

Figure 2: JASE's top results for q13

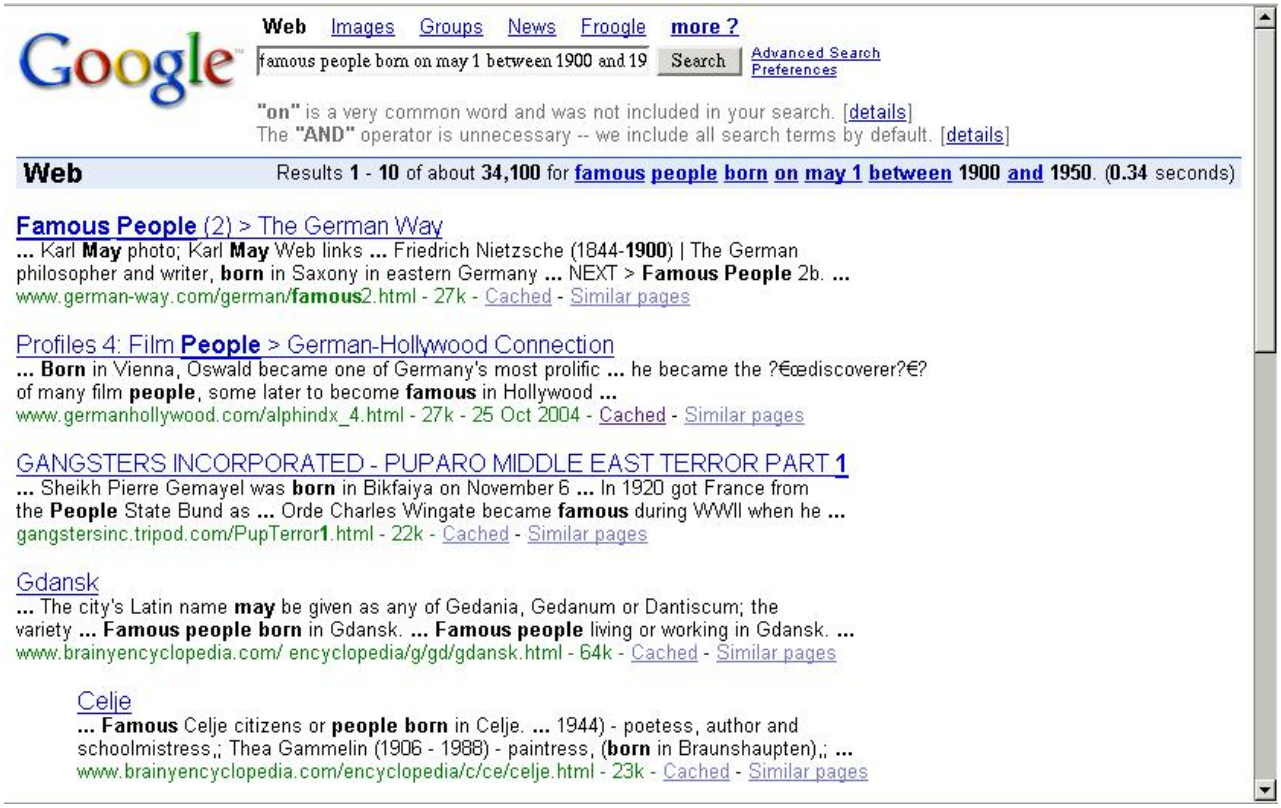

Figure 3: Google's top results for q13