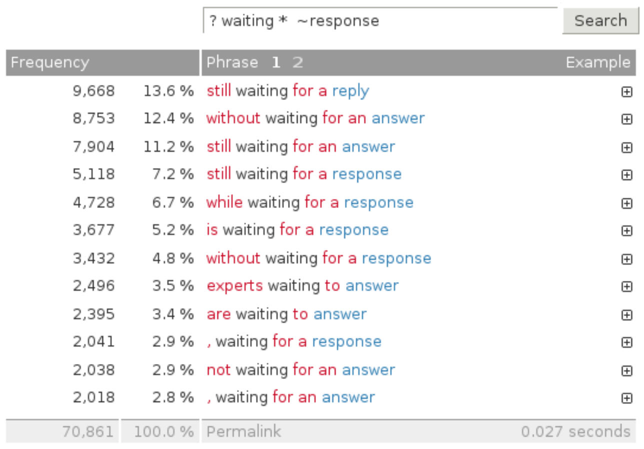
The NETSPEAK WORDGRAPH: Visualizing Keywords in Context

Patrick Riehmann Henning Gruendl
Bernd Froehlich
Virtual Reality Systems Group [*]

Martin Potthast Martin Trenkmann
Benno Stein
Web Technology and Information Systems Group [*]

Bauhaus-Universität Weimar



? waiting * ~response [Search]

Frequency		Phrase 1 2	Example
9,668	13.6 %	still waiting for a reply	⊞
8,753	12.4 %	without waiting for an answer	⊞
7,904	11.2 %	still waiting for an answer	⊞
5,118	7.2 %	still waiting for a response	⊞
4,728	6.7 %	while waiting for a response	⊞
3,677	5.2 %	is waiting for a response	⊞
3,432	4.8 %	without waiting for a response	⊞
2,496	3.5 %	experts waiting to answer	⊞
2,395	3.4 %	are waiting to answer	⊞
2,041	2.9 %	, waiting for a response	⊞
2,038	2.9 %	not waiting for an answer	⊞
2,018	2.8 %	, waiting for an answer	⊞
70,861	100.0 %	Permalink	0.027 seconds

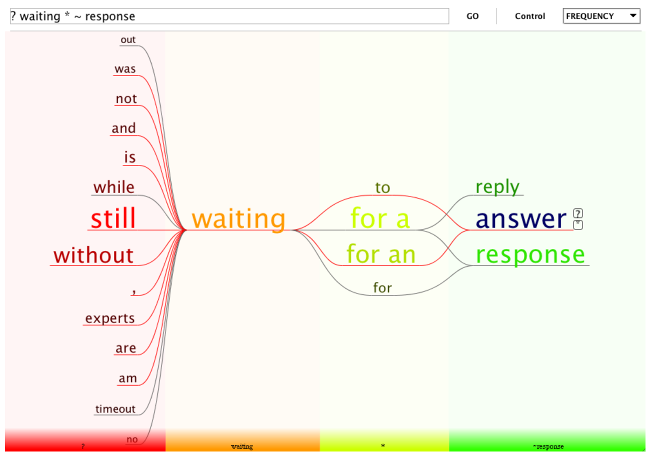


Figure 1: The textual Web interface versus the WORDGRAPH visualization of the NETSPEAK writing assistant.

ABSTRACT

NETSPEAK helps writers in choosing words while writing a text. It checks for the commonness of phrases and allows for the retrieval of alternatives by means of wildcard queries. To support such queries, we implement a scalable retrieval engine, which returns high-quality results within milliseconds using a probabilistic retrieval strategy. The results are displayed as WORDGRAPH visualization or as a textual list. The graphical interface provides an effective means for interactive exploration of search results using filter techniques, query expansion and navigation. Our observations indicate that, of three investigated retrieval tasks, the textual interface is sufficient for the phrase verification task, wherein both views support context-sensitive word choice, and the WORDGRAPH best supports the exploration of a phrase's context or the underlying corpus. The preferred view for context-sensitive word choice seems to depend on query complexity (i.e. the number of wildcards in a query).

Keywords: Information visualization, visual queries, text visualization, information retrieval, Web *n*-grams, wildcard search.

Index Terms: H.5.2 [User Interfaces]: —

[*]E-Mail: <first name>.<last name>@uni-weimar.de

1 INTRODUCTION

NETSPEAK is a tool for context-sensitive word choice. It allows its users to check the commonness of a phrase and to express uncertainties in choosing words by formulating queries that contain wildcards. The search results are provided as a classic textual list as well as a legible and interactive graph visualization—the WORDGRAPH (see Figure 1). The graph layers follow the structure of a query, showing one layer for every literal word and wildcard, which are filled dynamically with the results obtained from NETSPEAK's retrieval engine. Search results visualized by the WORDGRAPH can be interactively explored, refined, and expanded by means of filter techniques and navigation. Queries are processed by a scalable retrieval engine, which returns high-quality results within milliseconds using a probabilistic retrieval strategy. It indexes a corpus of more than 3 billion word n-grams up to a length of $n = 5$ words along with their occurrence frequencies in a large portion of the Web.

NETSPEAK's intended audience is people who have doubts about how certain phrases are commonly formed. In particular, second language speakers face difficulties in this respect, since their innate sense of language—their sprachgefühl—is often not sufficiently developed. They ask themselves how others would formulate a particular phrase; a piece of information that is generally hard to come by. NETSPEAK implements a statistical solution by contrasting alternative phrases based on their absolute and relative occurrence frequency. Our working hypothesis is that choosing more common phrases over uncommon ones may improve readability, comprehensibility, and writing style. Obviously, this is not true in general, but as non-native speakers we found the suggestions of NETSPEAK immensely helpful in all our daily writing tasks.

A variety of visualizations of relations among words, phrases or collocations (also called "keywords in context") have appeared in recent years, such as the WORDTREE, PHRASENETS, Google Scribe, and AWKCHECKER, to name only a few. The WORDTREE [17] employs suffix trees to index text and to visualize the tree starting from the query word(s). PHRASENETS encode subject-predicate-object triplets in a directed graph, which are mined from a text by specifying the predicate and considering subject and object as wildcards (e.g. `? loves ?`). Google Scribe [6] assists authors with writing by suggesting the next word in a phrase—very similar to AWKCHECKER [12]. Both systems are (likely to be) based on language model theory.

The WORDGRAPH visualization and the NETSPEAK engine can be considered as a generalization and a combination of the aforementioned approaches: the WORDGRAPH has more complex layout constraints than the tree layout of the WORDTREE. PHRASENETS visualize only triplets with a fixed predicate and the force-based layout limits the phrase legibility. Both tools focus on corpus exploration instead of being interactive word choice tools. AWKCHECKER and Google Scribe do not employ visualizations at all, as they are not intended for ad hoc wildcard queries.

Thus, our contributions are threefold. First, we present the WORDGRAPH, a dynamic graph visualization for interactive exploration of search results for complex keywords-in-context queries. Secondly, we introduce our new and scalable NETSPEAK retrieval engine that operates efficiently on a huge corpus of text from the Web. Lastly, we perform a pilot user study comparing WORDGRAPH visualization and the textual Web interface, analyze query logs of the NETSPEAK service and investigate typical retrieval tasks related to choosing words.

2 RELATED WORK

Besides the aforementioned systems, several others keyword-in-context tools exist or are being developed. Viégas and Wattenberg present the Web Seer prototype[1], which allows one to contrast query suggestions from the Google Web search engine for two queries. The visualization encompasses two trees whose roots represent one of the queries each, while the children represent the suggestions obtained from Google. Shared suggestions are unified, thus visualizing tree similarity, while edge thickness and node positions tell something about how often a suggested query has been posed. Paley's Textarc [11] visualizes the sentences of a text centrifugal along an ellipse shape. Frequent words of the text are depicted inside the ellipse. The legibility of individual phrases is limited with this approach. C. Harrison [7] has generated static word graphs from small portions of the Google *n*-gram corpus as showcase examples. However, no means is provided to generate these visualizations on demand, and it also lacks interaction so that it can only be viewed as is. Collins *et al.* [5] visualize the text produced by automatic machine translation tools in the form of lattice graphs in order to support translators. Uncertainties of the tools in choosing the right translation for a word are represented by alternative paths in the lattice graph, where the commonness of an alternative, as determined by the language model underlying the machine translator, is encoded by size and shade of the nodes and their edges. Here, however, no manual wildcard queries are possible. Although our system displays a graph instead of a tree, the Degree-of-Interest Trees (DOITrees) by Heer and Card [8] and the SpaceTree by Plaisant *et al.* [13] provide some convenient patterns for the navigation of different levels of detail, supported by animated transitions in huge tree structures.

Corpora of *n*-grams are frequently used in natural language processing and information retrieval for training purposes [10]. Search engines comparable to ours include WEBCORP, WEBASCORPUS, PHRASESINENGLISH, and LSE.[2] They target researchers of linguistics whose primary concern is the study of language use. By contrast, our search engine also targets the average writer, whose information needs and prior knowledge differ from those of a linguist. An informal comparison suggests that NETSPEAK outperforms these tools both in terms of retrieval speed and the extent of the indexed language resources. Cafarella *et al.* [4] propose a retrieval engine that supports linguistic queries; a comparison with our approach is still missing.

[1] http://hint.fm/projects/seer

3 NETSPEAK'S INTERFACES

The NETSPEAK text interface provides a straightforward way to search for phrases.[3] It is designed to conform to current and traditional best practices of Web interfaces for search engines, with an emphasis on simplicity and minimalism (Figure 1, left). It utilizes a query language that is defined by the grammar shown in Table 1.

Table 1: EBNF grammar of the NETSPEAK query language.

query	=	$\{$ word | wildcard $\}_1^5$
word	=	([" ′ "] (letter { alpha })) | " , "
letter	=	" a " | ... | " z " | " A " | ... | " Z "
alpha	=	letter | " 0 " | ... | " 9 "
wildcard	=	" ? " | " ⋆ " | synonyms | multiset
synonyms	=	" ~ " word
multiset	=	" { " word { word } " } "

A query is a sequence of literal words and wildcard operators, wherein the literal words must occur in the expression sought after, while the wildcard operators allow to specify uncertainties. Currently four operators are supported: the question mark, which matches exactly one word; the asterisk, which matches any sequence of words; the tilde sign in front of a word, which matches any of the word's synonyms; and the multiset operator, which matches any ordering of the enumerated words. The interface displays the search results for the given query as a ranked list of phrases, ordered by decreasing occurrence of absolute and relative frequencies. This way, the user can find confidence in choosing a particular phrase by judging both its absolute and relative frequencies. For example, a phrase may have a low relative frequency but a high absolute frequency, or vice versa, which in both cases indicates that the phrase is not the worst of all choices. Furthermore, the Web interface offers example sentences for each phrase, which are retrieved on demand when clicking on the plus sign next to a phrase. This allows users who are still in doubt to get an idea of the larger context of a phrase.

The WORDGRAPH visualizes the resulting *n*-grams of a query in a layered graph (Figure 1, right) and offers interactions with the result set. The nodes of the graph correspond to the words of the *n*-grams, and an edge represents the connection between two subsequent words of an *n*-gram. Consequently, each *n*-gram of a result set is represented as a path through the graph. The layers of the graph are arranged in vertical columns to facilitate reading. Every column corresponds to one element of the query, which can be a literal word or a wildcard character, as defined in Table 1. Multiple occurrences of the same word in a column are merged into a single node.

The graph can be drawn in two ways, a *split path view* and a *condensed path view* (Figure 2). The *split path view* displays the *n*-gram paths of a result set independently—similar to the text view—and reveals the overall complexity of the result set. The *condensed path view* merges shared subpaths of different *n*-grams, which is a compact abstraction of the result set, where individual *n*-grams are

[2] www.webcorp.org.uk, www.webascorpus.org, www.phrasesinenglish.org, and [14].

[3] NETSPEAK's text interface is accessible at www.netspeak.cc.

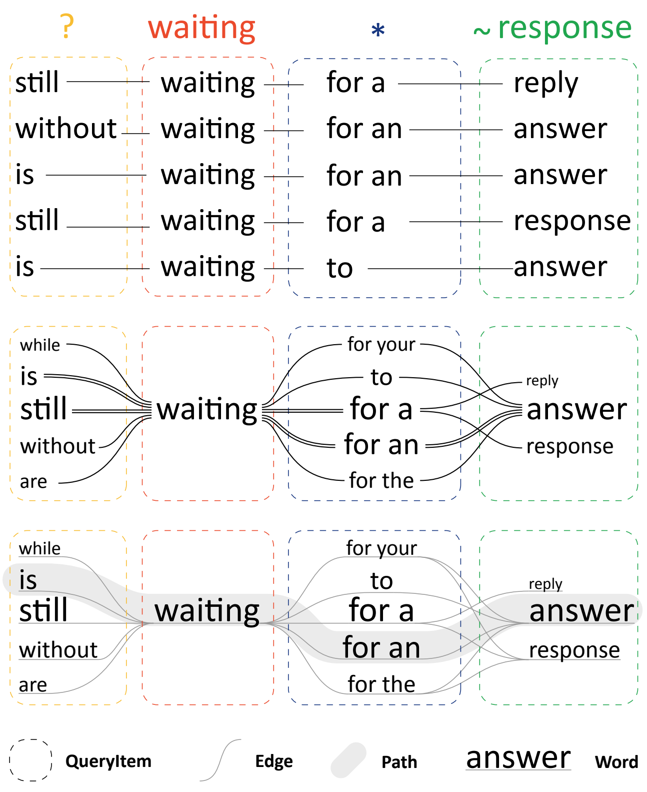


Figure 2: The query `? waiting * ~response` combines one word and three wildcards; the search results are shown. The WORDGRAPH visualizes n-grams as paths through the graph, merging multiple occurrences of a word within a column. Every path can be drawn individually (split path view, middle) or shared subpaths can be merged (condensed path view, bottom). The latter view increases the result capacity (i.e. a path for `is waiting for a reply` is there although this 5-gram is not part of the result set.)

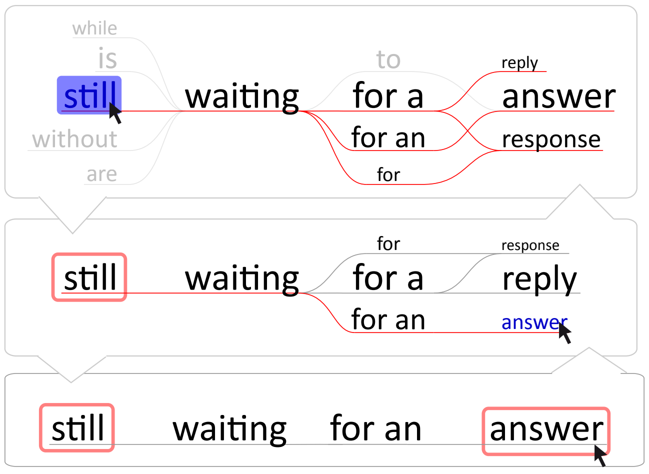


Figure 3: The WORDGRAPH offers several filter operations: (1) Hovering the mouse above a node highlights all n-gram paths passing through the node. (2) Selecting a node deemphasizes all paths of n-grams that do not contain the selected word. Multi-selection is supported. (3) The subgraph filter hides elements that do not belong to selected paths.

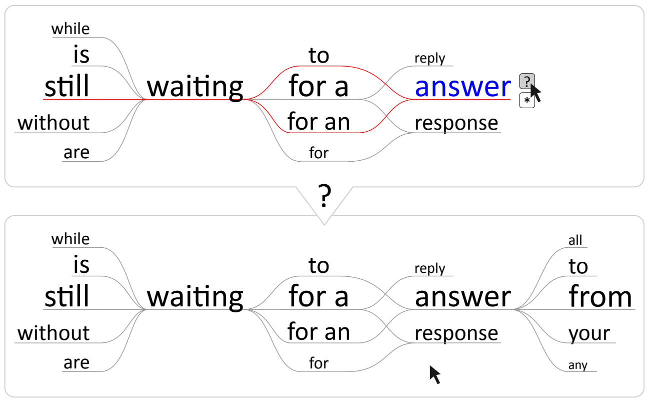


Figure 4: Query expansion: by clicking on the wildcard icon next to the word `answer`, a set of queries is generated for all n-grams whose paths pass through this word's node, complemented by the respective wildcard. The n-grams retrieved with these queries are integrated into WORDGRAPH which results in a new column.

no longer directly visible. While merging the paths significantly enhances overall legibility of large graphs, it also brings about the problem of increased result capacity (i.e. more paths can be created than are actually supported by the result set). However, the condensed path view is the preferred view in practice, which is why we have developed several interaction techniques to counter this problem, and, to allow users to explore the result set.

4 WORDGRAPH INTERACTIONS

While the text interface of NETSPEAK offers no interaction beyond the retrieval of example sentences for a particular n-gram, the WORDGRAPH provides various means for exploring the search results, including filter techniques, query expansion, and support for navigation.

The filter operations allow users to reveal the paths passing through a certain node, to emphasize certain paths, and to select a subgraph by specifying a set of nodes (Figure 3). The filter operations are orthogonal, as in they can be applied repeatedly in an arbitrary order. Users may also switch between the condensed path view and the split path view at any time. Transitions between different views are animated to facilitate the understanding of the relationships between the different layouts.

Since the basis of our retrieval engine is the Google n-gram corpus, only n-grams up to a length of $n = 5$ words can be retrieved. By means of our query expansion technique we can address this limitation and allow the retrieval of longer phrases based on those already displayed in WORDGRAPH (Figure 4). By clicking on the expansion icons shown next to a word while hovering over it, new queries are constructed for all n-grams whose paths go through the word's node, using up to four preceding words and appending the respective wildcard `?` or `*`. The union of the result sets of all these queries is then integrated into the existing graph structure, adding new columns as needed. Every expansion entails $O(k^3)$ new queries, where k denotes an upper bound on the number of incoming edges of a word in the WORDGRAPH with k typically being between 4 and 10. The n-grams formed in this way, where $n > 5$ may be incorrect or meaningless; yet, sensible results have been observed in many cases.

The query expansion technique produces word graphs that contain more columns than fit on the screen. To allow for navigation, we implemented horizontal panning and scrolling support. An overview bar at the bottom of the screen (Figure 5) helps the user to control the horizontal panning and allows to immediately jump to a specific column, which scrolls automatically into the center of the

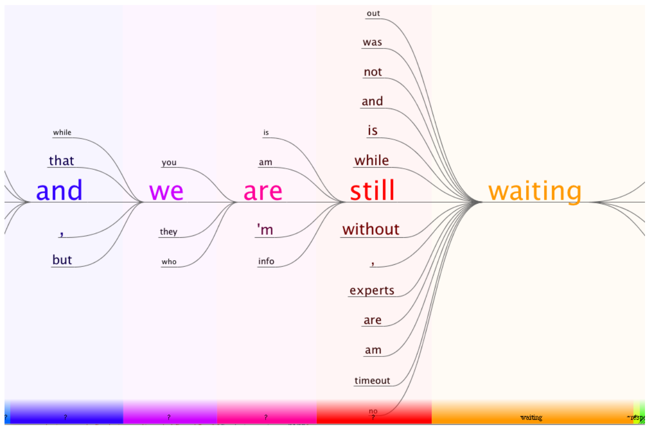


Figure 5: Navigation: The overview bar at the bottom of the screen shows the columns of WordGraph. Selecting a column (collapsed or not) moves that column into the center using an animated transition. Also, the whole graph can be moved horizontally while columns are collapsed and expanded as necessary.

screen. Columns, which do not fit on the screen appear collapsed on the overview bar. Vertical navigation is also provided within columns in case the vertical space is too small to display all words. Currently, words outside the vertical scope are clipped while vertical scrolling makes them visible. At first, the implementation of an explicit focus and context technique was envisioned, but edges of the graph pointing towards clipped words are a sufficient hint at information outside the visible area.

5 WordGraph Layout Details

This section explains the important design decisions for the layout and rendering in WordGraph. Central concern is the legibility of phrases, and hence the placement of words in subsequent columns is essential. The layout also needs to reflect properties of individual words (e.g. font, size, color and opacity), properties of edges (e.g. path, color and width) and attributes of *n*-grams (e.g. absolute and relative occurrence frequency). The layout process consists of four steps: (1) horizontal partitioning of available screen space into columns; (2) vertical ordering within these column; (3) exact placement of words and (4) drawing of edges between (underscoring) words.

5.1 Screen Partitioning and Word Placement

The initial layout of WordGraph evolves from the submitted query. The longest *n*-gram returned determines the number of necessary columns. The width of each column considers font sizes, word lengths and additional padding, as shown in Figure 6. Within a column, each word is horizontally centered, except for the first and last column respectively.

The vertical arrangement can be done in two ways: one strategy is the *top spread ordering* (Figure 7, bottom), which is similar to the text view. The second strategy is the *center spread ordering*, which places words in a column with decreasing font size, starting from the center and alternating the placement between above and below (Figure 7, top). The latter strategy is preferred since it places the most important query result in the middle of the screen and facilitates the tracing of alternative phrases without introducing large inter-column skips.

For the vertical word placement within a column we experimented with two possible layouts, shown in Figure 6: the *maximal word spreading* uses the entire vertical and horizontal space of a column for equally distributing the words; it is independently applied for each column. The alternative *grid-based word placement*

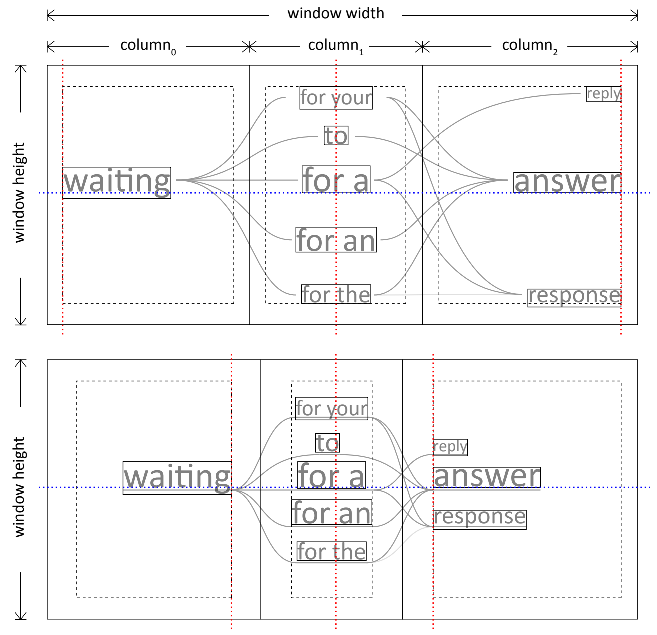


Figure 6: Column layout and word placement in a column. Maximal word spreading (top). Grid-based word placement of all columns (bottom).

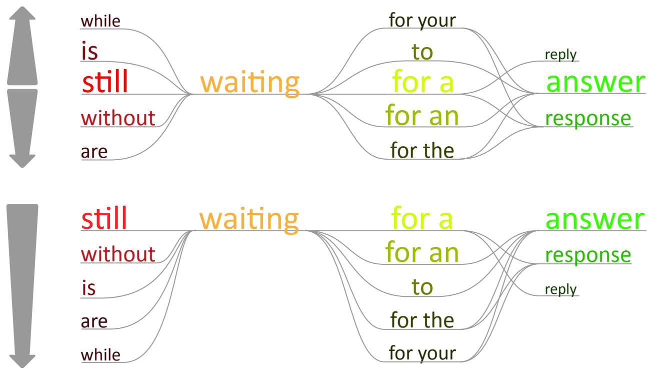


Figure 7: Arranging words in a column according to their frequency: The *center spread ordering* (top) and the *top spread ordering* (bottom). The relative occurrence frequency of a word in a column is mapped to its font size, color and brightness.

is more compact and uses a grid to place the words. The defined cell height for all columns depends on the font size of the most frequent word of all columns. In every column the algorithm starts from the center and places the words above and below, aligned to the defined cell height (Figure 6, bottom), which minimizes the vertical spread from the center. In horizontal direction the algorithm is more flexible: in the first and the last column the words are aligned to the inner padding, while in the other columns they are centered. We found that the grid-based vertical partitioning of all columns along with a minimal spread from the center (Figure 6, bottom) facilitates the readability of the phrase fragments since it resembles a printed page.

5.2 Edge Drawing

As previously mentioned in section 3, the edges of WordGraph can be rendered in two different ways (Figure 2). The condensed path view draws a direct representation of the graph with at most only a single edge between words. The split path view shows all n-grams contained in WORDGRAPH by drawing all the edges of the n-grams into the graph. Each edge is defined by a cubic Bézier curve. The start point and end point are located at defined locations (ports) on the source word and the target word. The tangents at these points are always horizontal to allow for a smooth transition from a straight line through the word into the edge.

The *condensed path view* places the port for connecting edges at either end of the baseline of the word. The baseline of the word itself is drawn, such that the line passes below the word to the other port and further on into an outgoing edge (Figure 8.1). We found that drawing a continuous line below the words, which connects incoming and outgoing edges significantly contributes to the readability of phrase fragments. Moreover, interrupting the curves by words is recognized rather as a set of single words without meaning than as a coherent phrase.

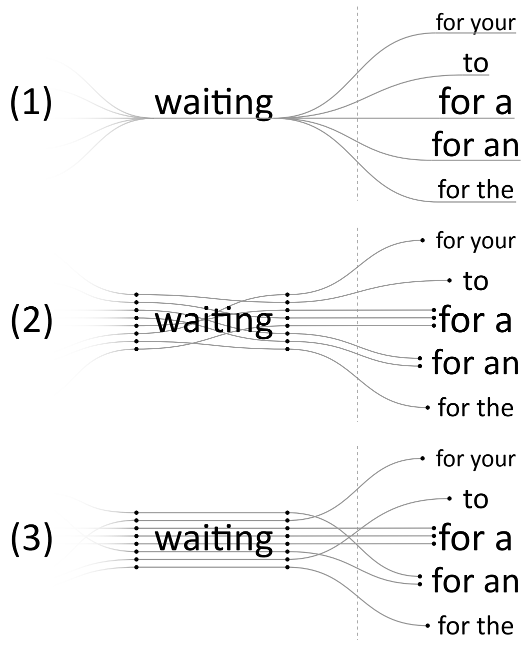


Figure 8: Possibilities for edge drawing. The edge ports are marked as small dots. (1) *Condensed path view:* all paths between two words are drawn as a single edge and the incoming and outgoing edges are connected by a line passing below the word to improve readability. (2) *Split path view:* each path is drawn independently. Crossings occur in the background of the words. (3) *Split path view:* each path is drawn independently. Crossings occur behind the words.

The *split path view* shows all paths defined by the n-grams from the search result set at once. The paths are also drawn in the background of the words such that tracing of an individual path across multiple columns is fully supported. Figure 8.2 and 8.3 show two different ways of vertically arranging the incoming and outgoing edges of a word. Figure 8.2 attaches the incoming and outgoing edges of a path to ports at the same vertical position and avoids crossings behind the word, but introduces additional crossing outside the word. Alternatively, incoming and outgoing edges on both sides are attached to appropriate ports depending on their starting position (Figure 8.3). In this case edge crossings occur behind the word, which was generally preferred particularly in combination with the available interaction techniques.

5.3 Layout Guidelines

Based on our experience with alternative implementations of the graphical NETSPEAK interface we derived the following list of layout guidelines:

- Prefer *center spread ordering* over *top spread ordering*.
- Use a vertical grid to align words across different columns.
- A minimal vertical word placement starting from the center is preferred.
- Underlining emphasizes the connectivity of a collocation and improves legibility significantly.
- Under the split path view, drawing edge crossings behind words (instead of in the background of the words) appears less tangled.
- Animated transitions are essential for filtering operations, query exploration and navigation.

6 NETSPEAK'S RETRIEVAL ENGINE

The three main building blocks of NETSPEAK's retrieval engine are (1) an index of frequent n-grams on the Web, (2) a query language to formulate n-gram patterns (introduced in Section 3, Table 1), and (3) a probabilistic top-k retrieval strategy which finds n-grams that match a given query and which allows to trade recall for time.[4]

6.1 Web Language Index

To provide relevant suggestions, a wide cross-section of written text on the Web is required which is why we resort to the Google n-gram corpus "Web 1T 5-gram Version 1" [3], which consists of 42 GB of phrases up to a length of $n = 5$ words along with their occurrence frequency on the Web as of 2006. This corpus has been compiled from approximately 1 trillion words extracted from the English portion of the Web, totaling in more than 3 billion n-grams. We applied two post-processing steps to the corpus: case reduction and vocabulary filtering. For the latter, a white list vocabulary V was compiled and only these n-grams whose words appear in V were retained. V consists of the words found in the Wiktionary and various other dictionaries, complemented by words from the 1-gram portion of the Google corpus whose occurrence frequency is above 11 000. After post-processing, the size of the corpus has been reduced by about 46%. In NETSPEAK the n-gram corpus is implemented as an inverted index, μ, which maps each word $w \in V$ onto a postlist π_w. For this purpose we employ a minimal perfect hash function based on the CHD algorithm [2]. π_w is a list of tuples $\langle \hat{d}, f(d) \rangle$, where $\hat{d}$ refers to an n-gram d on the hard disk that contains w, and where $f(d)$ is the occurrence frequency of d reported in the n-gram corpus. A tuple also stores information about w's position.

6.2 Probabilistic Retrieval Strategy

Given the n-gram index μ and a query q, the task is to retrieve all n-grams D_q from μ that match q according to the semantics defined by NETSPEAK's query language. This is achieved within two steps: (1) computation of the intersection postlist $\pi_q = \bigcap_{w \in q} \pi_w$ and (2) filtering of π_q with a pattern matcher that is compiled at run-time from the regular expression defined by q. Retrieving D_q poses no algorithmic challenge unless retrieval time is considered. Postlists often consist of up to millions of entries, which is the case for stop words. If a query contains only stop words, the retrieval for D_q may take tens of seconds or even up to a minute, depending on the size of the indexed corpus. From a user's perspective this is clearly unacceptable. In cases where a query also contains a rare word w', it is often more effective to apply the pattern matcher directly to $\pi_{w'}$, which is possible since $\pi_q \subseteq \pi_w$ holds for all $w \in q$. But altogether this and similar strategies do not solve the problem: the frequency distribution of the words used in queries will resemble that of written text, simply because of the NETSPEAK use case.

[4]The results of this section have been described in a poster paper [16].

In contrast, Web search engines typically deal with queries consisting of combinations of (comparatively infrequent) topic words.

To allow for an adjustable retrieval time at the cost of recall,[5] we have devised a probabilistic retrieval strategy, which incorporates rank-awareness within the postlists. Hence, our strategy is a special kind of a top-k query processing technique [1, 9]. The strategy requires an offline pre-processing of μ, so that (1) each postlist is sorted in order of decreasing occurrence frequencies and (2) each postlist is enriched by quantile entries κ, which divide the word-specific frequency distribution into portions of equal magnitude. Based on a pre-processed μ, the retrieval algorithm described above is adapted to analyze postlists only up to a predefined quantile. Consequently, the portion of a postlist whose frequencies belong to the long tail of the distribution is pruned from the search.

An important property of our search strategy is what we call *rank monotonicity*: given a pre-processed index μ and a query q, the search strategy will always retrieve n-grams in decreasing order of relevance, independently of κ. This follows directly from the postlist sorting and the intersection operation. An n-gram that is relevant for a query q is not considered if it is beyond the κ-quantile in some $\pi_w, w \in q$. The probability for this depends, among other things, on the co-occurrence probability between q's words. This fact opens up new possibilities for further research in order to increase the recall (e.g. by adjusting κ in a query-specific manner). Such options, however, have yet to be explored.

7 Evaluation Results and Discussion

With its textual interface, Netspeak has been publicly available since 2008, while the WordGraph interface is still in development to be released to the public. In this section we provide implementation details, and evaluate Netspeak's components: we report on experiments to assess the retrieval quality of our query processing strategy, conduct a query log analysis as well as a pilot user study, and conclude with a discussion of use cases for the WordGraph interface.

7.1 Implementations Details

The communication between Netspeak's interfaces and its retrieval engine is implemented via Ajax, using the lightweight data-interchange format JSON (JavaScript Object Notation). The retrieval engine is written in C/C++ and is deployed at our site, accessible through a servlet container. The textual Web interface is implemented using the Google Web Toolkit and it is deployed on the Google App Engine. The visualization client is as a stand-alone application written in Java, deployed at our site. The Java scene graph project Scenario is used to manage and display graphical 2D-elements. Scenario provides convenient methods to handle different kinds of animations (see [15]).

7.2 Retrieval Quality Evaluation

To evaluate the retrieval quality of our query processing strategy, we report on an experiment in which the average recall is measured for a set of queries Q, $|Q| = 55\,702$, with respect to different pruning quantiles. The queries originate from the query logs of Netspeak. We distinguish between macro-averaged recall and micro-averaged recall:

$$rec_{\text{macro}}(\mu, q) \quad = \quad \frac{|D_q \cap D_q^*|}{|D_q^*|}$$

[5]Recall is one of the most prominent retrieval performance measures in information retrieval. Given a query q, the recall quantifies whether everything that ought to be retrieved for q is actually retrieved by a retrieval engine. Its counterpart is called precision, which quantifies whether everything that is retrieved for q actually ought to be retrieved. The precision of our retrieval engine is always perfect by construction.

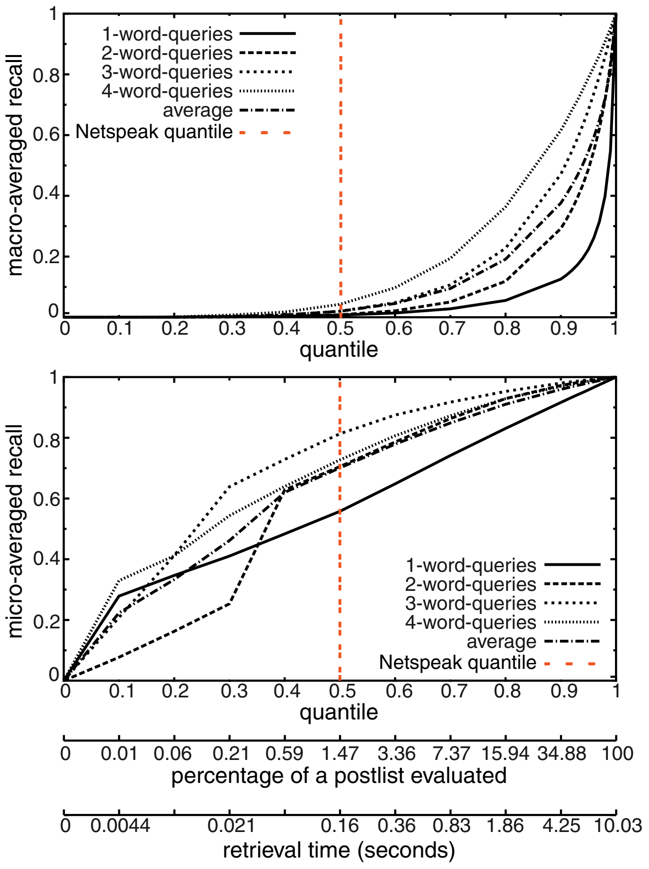


Figure 9: Macro-averaged recall (top) and micro-averaged recall (bottom) over quantiles. The additional axes indicate how much of a postlist is evaluated and the required processing time.

$$rec_{\text{micro}}(\mu, q) \quad = \quad \frac{\sum_{\langle d, f(d)\rangle \in (\pi_q \cap \pi_q^*)} f(d)}{\sum_{\langle d, f(d)\rangle \in \pi_q^*} f(d)}$$

As described above, D_q and π_q are the results retrieved from μ for query q under a top-k strategy, while D_q^* and π_q^* are the results if the postlists of μ are evaluated completely. While rec_{macro} considers only the result list lengths, rec_{micro} attributes more weight to n-grams with high occurrence frequencies, since they are more relevant to the user. Figure 9 shows the obtained results for different query sizes. The macro-averaged recall differs significantly from the micro-averaged recall, which indicates that most of the relevant n-grams are retrieved using our strategy. The current Netspeak quantile of $\kappa = 0.5$ marks our chosen trade-off between recall and retrieval time. At quantile 0.5 only 1.47% of a postlist is evaluated on average, which translates into a retrieval speedup of factor 68. The average retrieval time at this quantile seems to leave much room in terms of user patience to evaluate more of a postlist; however, it does not include the time to generate and ship the results page. Short queries are more difficult to answer because the size of the expected result set is much larger on average than that of a long query. From an evaluation standpoint the micro-averaged view appears to be more expressive.

7.3 Query Log Analysis

The analysis of the Netspeak's query logs reveals interesting patterns of user behavior. The average query length is 3.15 tokens (words or wildcards), and since more than 50% of the queries contain exactly 3 tokens, the major share of wildcard use occurs in them, too. The distribution of wildcard use is dominated by the asterisk wildcard (*, 61%), which represents an arbitrary number of words, whereas the simpler question mark wildcard (?) is only used in 20% of the queries. The remainder of the queries either contain the tilde wildcard (~) for synonym search or none at all. Interestingly, the fraction of queries that do not contain any wildcards is almost 20%, so that in turn, an average of 80% of the queries do. Queries without wildcards supposedly only check for the existence or the commonness of a phrase.

An in-depth analysis reveals that most users interact with NETSPEAK in sessions (i.e. by posing a series of queries within a certain time frame). Only 18% of the queries belong to single-query sessions. We have identified two different session types:

(1) *Bunch of Queries.* In this case, a session consists of unrelated queries, where none of the queries have words or wildcards in common with previous or succeeding queries. It appears as if users first write a large chunk of text and then check those phrases about which they are uncertain.

(2) *Query Refinement Session.* In this case, a session consists of related queries, where queries following each other are very likely to have words and wildcards in common. It appears as if a user is working on a particular phrase, searching for alternatives. This session type is most common.

The average number of queries per session is 5.6 and the average duration of a session is about 6.5 minutes. A few sessions took very long indeed, lasting more than half an hour. In some of the refinement sessions the users obviously struggled with a certain phrase and permanently exchanged words and wildcards over a long period of time. Long refinement sessions are sometimes interrupted by unrelated queries and then continued later on.

7.4 User Feedback

We performed an initial pilot study to assess the usability of our system. In particular, we were interested in the following aspects:

- Is the WORDGRAPH interface intuitive?
- How does the WORDGRAPH interface compare to the textual interface?
- Which of the two interfaces would the participants like to see in the Web service?

Ten non-native English speakers with higher English education (mostly postgraduate students) participated in this study. Some of them were already familiar with the textual Web interface. The study comprised of ten selected test queries from the NETSPEAK query log files that were selected for variety. After a brief introduction of the WORDGRAPH interface, the participants were asked to enter the test queries, select the most suitable results and to answer the questionnaire. In addition, the participants were asked about desired improvements with respect to interaction and layout.

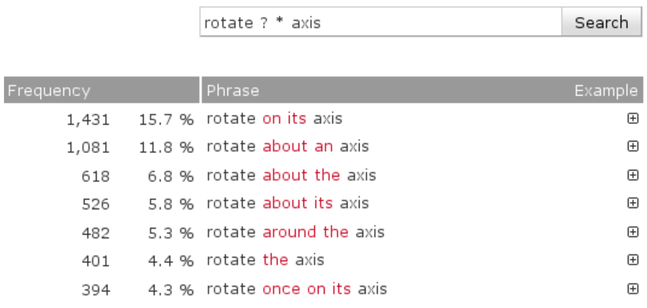
All participants would like to see WORDGRAPH being provided as an additional interface for NETSPEAK; 50% even considered WORDGRAPH as a substitute for the textual Web interface. They assessed WORDGRAPH as "very intuitive" with an average of 5.3 on a scale from 1 to 6. This coincides with the answers to the question about the most useful WORDGRAPH features: a fluent result filtering was mentioned by 80%, starting from an overview with the most important information was mentioned by 60%, 50% emphasized the possibility of exploring the response set in detail by succeeding or alternating applications of subgraph filtering. Finally, two participants mentioned the improved legibility of the word sequences within the graph by following the edges through the nodes: "It gives the impression of reading from a sheet of lined paper." These results are encouraging, and we plan to conduct an extensive user study in the future that covers more aspects of WORDGRAPH and the textual Web interface, in particular with regard to the use cases outlined below.

7.5 Use Cases and Experiences

Based on our query log analysis, the session types identified and the pilot study, we identified three practical retrieval tasks related to word choice, which have an increasing level of difficulty:

(1) *Phrase Verification.* The most basic retrieval task is to check whether a given phrase is commonly used. As mentioned above, almost 20% of all queries come without wildcards. For this task, the textual interface is fully sufficient.

(2) *Context-Sensitive Word Choice.* In this retrieval task a writer is uncertain about what alternative for a word in a given phrase is a good choice, or whether there are in fact any alternatives. This task pertains particularly to second-language speakers who often translate words using a dictionary—the exact translation of many words depends on context. In this respect, NETSPEAK serves as a context-sensitive thesaurus. Choosing the correct adverbs and prepositions is also a common problem.



Frequency		Phrase	Example
1,431	15.7 %	rotate on its axis	⊞
1,081	11.8 %	rotate about an axis	⊞
618	6.8 %	rotate about the axis	⊞
526	5.8 %	rotate about its axis	⊞
482	5.3 %	rotate around the axis	⊞
401	4.4 %	rotate the axis	⊞
394	4.3 %	rotate once on its axis	⊞

Figure 10: Word choice with NETSPEAK's Web interface (excerpt).

The query language of NETSPEAK is powerful in that it allows to specify rather complex patterns of *n*-grams to be retrieved. A user who inserts more than one wildcard into a query is less confident about how to write a certain phrase and seeks to generalize the query in order to cover more of the possible alternatives. This, in turn, yields a longer list of results in the textual interface, which may be difficult to overview and which may not always reveal the true picture about which words to choose. Figure 10 shows an example, where `about` appears in three of the *n*-grams, which indicates that this word should most likely follow `rotate`. The textual Web interface, however, obscures this fact and the user is forced to scan the entire result list several times to grasp the true relationships. By contrast, the WORDGRAPH visualization for the same query as above provides an overview at a glance (see Figure 11). It reveals the bimodal frequency distribution for the words retrieved for both wildcards. In this respect, our pilot user study indicates that there might be a relation between interface preference and the number of wildcards contained in a query.

(3) *Exploration.* This retrieval task is about writers who want to explore the typical context of a phrase by looking at what comes before, after or in between the phrase's words. With the NETSPEAK's textual interface, this task is limited to exploring a context of up to four words around a query that comprises, say, only one word surrounded by asterisks. Only by means of additional queries, a user may get a broader view of a phrase's context, having to keep in mind the results of all

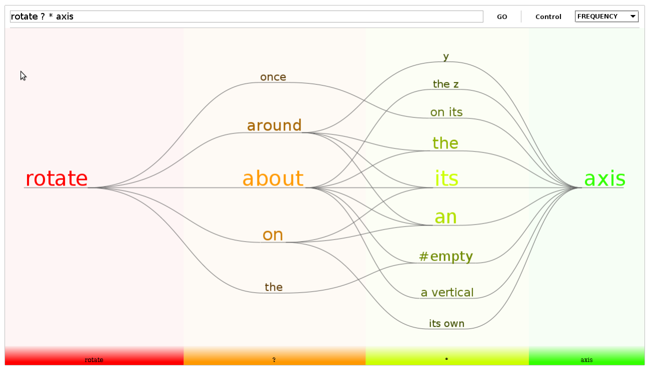


Figure 11: Word choice with the NETSPEAK WORDGRAPH.

previous queries. With the WORDGRAPH interface, this task is supported without further ado by means of the query expansion technique (Figure 4). The results of additional queries, which can be posed interactively, are integrated seamlessly into an existing graph so that users can construct a full picture of a phrase's context. This capability of WORDGRAPH is particularly useful for expert users, including linguists who investigate the characteristics of language use in a given corpus.

Remarks. While writing a text, such as a scientific paper, users often switch back and forth between different retrieval tasks. Phrase verification is the least observed task, which is documented by NETSPEAK's query logs; 80% of the queries comprise wildcards. There are two common types of queries: queries asking for the most suitable word in a given context, and queries asking for the typical context of a particular word or, more precisely, which common collocations a particular word has. Thus it is context sensitivity that is most relevant to the users, which is difficult to express with other commonly available tools. With the textual Web interface, one typically looks at the top results and ignores the rest—similar to the use of a Web search engine. With WORDGRAPH, one explores the results more thoroughly and discovers relationships between words that are not apparent in the textual interface. While the latter often forces a user to formulate a sequence of similar queries, the former provides an effective means for implicit query specification, using filter techniques, query expansion and navigation.

8 CONCLUSIONS AND FUTURE WORK

NETSPEAK answers complex word sequence queries that are formulated in an expressive query language. The system is designed for efficiency and allows for real-time querying of a 42 GB text data base. The result set is explored via a textual Web interface or the graphical WORDGRAPH interface. Our analysis shows that the textual interface is sufficient for phrase verification and the comparison of related sentences. The WORDGRAPH interface allows an interactive exploration of the result set and is superior for word choice problems on complex queries. The layout of WORDGRAPH focuses on facilitating legibility, which is achieved by using center spread ordering, grid-based word placement and underscoring edges. Participants of our pilot user study describe WORDGRAPH as very intuitive and appreciate the possibility of graph-based filtering during explorative analyses.

We see NETSPEAK as a great educational tool for improving the knowledge of a second language. Additional smart operators for the query language such as antonym wildcards or semantic constraints (e.g. person names, places, dates and times) and support for further languages besides English would broaden the scope of NETSPEAK. An extension towards domain-specific corpora can help inexperienced authors to become familiar with the appropriate expressions and writing style in a specific field. However, if the domain becomes too small, such as using a corpus based simply on visualization papers, all the papers and talks from non-native speakers might end up using the same kind of Viz speak.

REFERENCES

[1] H. Bast, D. Majumdar, R. Schenkel, M. Theobald, and G. Weikum. IO-top-k: index-access optimized top-k query processing. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 475–486. VLDB Endowment, 2006.

[2] D. Belazzougui, F. Botelho, and M. Dietzfelbinger. Hash, displace, and compress. In *ESA '09: Proceedings of the 17th European Symposium on Algorithms*, pages 682–693, Springer Berlin / Heidelberg, 2009. Springer.

[3] T. Brants and A. Franz. Web 1T 5-gram Version 1. Linguistic Data Consortium LDC2006T13, Philadelphia, 2006.

[4] M. J. Cafarella and O. Etzioni. A search engine for natural language applications. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 442–452, New York, NY, USA, 2005. ACM.

[5] C. Collins, M. S. T. Carpendale, and G. Penn. Visualization of uncertainty in lattices to support decision-making. In *EuroVis*, pages 51–58, 2007.

[6] GoogleLabs. Google scribe [online]. Available: http://scribe.googlelabs.com/. [Accessed: September 22, 2010].

[7] C. Harrsion. Web trigrams [online]. Available: http://www.chrisharrison.net/projects/visualization.html. [Accessed: March 22, 2010].

[8] J. Heer and S. K. Card. Doitrees revisited: scalable, space-constrained visualization of hierarchical data. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 421–424, New York, NY, USA, 2004. ACM.

[9] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):1–58, 2008.

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

[11] W. B. Paley. Textarc: Showing word frequency and distribution in text. Poster Infovis 2002. Available: http://www.textarc.org/appearances/InfoVis02/InfoVis02_TextArc.pdf, 2002. [Accessed: September 22, 2010].

[12] T. Park, E. Lank, P. Poupart, and M. Terry. Is the sky pure today? awkchecker: an assistive tool for detecting and correcting collocation errors. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 121–130, New York, NY, USA, 2008. ACM.

[13] C. Plaisant, J. Grosjean, and B. B. Bederson. Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 57, Washington, DC, USA, 2002. IEEE Computer Society.

[14] P. Resnik and A. Elkiss. The linguist's search engine: an overview. In *ACL '05: Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 33–36, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[15] Scenario. Projekt scene graph [online]. Available: https://scenegraph.dev.java.net/. [Accessed: August 30, 2010].

[16] B. Stein, M. Potthast, and M. Trenkmann. Retrieving Customary Web Language to Assist Writers. In C. G. et al., editor, *Advances in Information Retrieval, Proceedings of the 32nd European Conference on Information Retrieval, ECIR 2010, category poster*, volume 5993 of *Lecture Notes in Computer Science*, pages 631–635, Heidelberg, 2010. Springer.

[17] M. Wattenberg and F. B. Viégas. The word tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1221–1228, 2008.