C. Gurrin et al (Eds.): Advances in Information Retrieval
Proceedings of the 32nd European Conference on Information Retrieval, ECIR 2010
Milton Keynes, UK, pp. 631-635, ©Springer 2010.

Retrieving Customary Web Language to Assist Writers

Benno Stein, Martin Potthast, and Martin Trenkmann

Bauhaus-Universität Weimar, Germany
<first name>.<last name>@uni-weimar.de

Abstract This paper introduces NETSPEAK, a Web service which assists writers in finding adequate expressions. To provide statistically relevant suggestions, the service indexes more than 1.8 billion n-grams, $n \leq 5$, along with their occurrence frequencies on the Web. If in doubt about a wording, a user can specify a query that has wildcards inserted at those positions where she feels uncertain.
Queries define patterns for which a ranked list of matching n-grams along with usage examples are retrieved. The ranking reflects the occurrence frequencies of the n-grams and informs about both absolute and relative usage. Given this choice of customary wordings, one can easily select the most appropriate. Especially second-language speakers can learn about style conventions and language usage. To guarantee response times within milliseconds we have developed an index that considers occurrence probabilities, allowing for a biased sampling during retrieval. Our analysis shows that the extreme speedup obtained with this strategy (factor 68) comes without significant loss in retrieval quality.

1 Introduction

Writers who are in doubt about a certain expression often ask themselves: *"What wording would others use in a similar context?"* This question can be answered statistically when given a huge corpus of written text from which matching examples can be retrieved. In this paper we introduce NETSPEAK, which indexes a large portion of the Web, presumably the most comprehensive text corpus today.[1]

Related Work Computer-aided writing has a long history dating back to the very beginning of personal computing, and so has research on this topic. This is why we can give only a brief overview. The main topics in writing assistance include spell checking, grammar checking, choosing words, style checking, discourse organization, and text structuring. Note that spell checkers and, to a certain extent, grammar checkers are currently the only technologies that reached a level of maturity to be shipped large-scale.

Search engines comparable to ours are for instance WEBCORP, WEBASCORPUS, PHRASESINENGLISH, and LSE.[2]All of them target exclusively researchers of linguistics. By contrast, our search engine targets the average writer, whose information needs and prior knowledge differs from those of a linguist. Moreover, NETSPEAK outperforms existing tools in terms of both retrieval speed and the extent of the indexed language resources. In [5] the authors propose an index data structure that supports linguistic queries; a comparison with our approach is still missing.

Corpora of n-grams are frequently used in natural language processing and information retrieval for training purposes [7], e.g., for natural language generation, language

[1] NETSPEAK is available at http://www.netspeak.cc.
[2] See http://www.webcorp.org.uk, http://webascorpus.org, http://phrasesinenglish.org, and [8].

modeling, and automatic translation. In particular, there is research on automatic translation within a language in order to correct writing errors [4]. We want to point out that our research is not directed at a complete writing automation since we expect a semi-automatic, interactive writing aid to be more promising in the foreseeable future.

2 NETSPEAK Building Blocks

The three main building blocks of NETSPEAK are (*i*) an index of frequent n-grams on the Web, (*ii*) a query language to formulate n-gram patterns, and (*iii*) a probabilistic top-k retrieval strategy which finds n-grams that match a given query and which allows to trade recall for time. The results are returned in a Web interface or as XML document.

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 [3]. This corpus is currently the largest of its kind; it has been compiled from approximately 1 trillion words extracted from the English portion of the Web, and for each n-gram in the corpus its occurrence frequency is given. Columns 2 and 3 of Table 1 give a detailed overview of the corpus. We applied two post-processing steps to the corpus at our site: 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, as well as of these words from the 1-gram portion of the Google corpus whose occurrence frequency is above 11 000. See Table 1, Columns 4 and 5, for the size reductions after each post-processing step with respect to the original corpus.

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 as well as other information omitted here for simplicity.

Query Language The query language of NETSPEAK is defined by the grammar shown in Table 2. A query is a sequence of literal words and wildcard operators, where 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. Of course other

Table 1. The Google n-grams before and after post-processing.

Corpus	Original Corpus		Case	Vocabulary
Subset	# n-grams	Size	Reduction	Filtering
1-gram	13 588 391	177.0 MB	81.34 %	3.75 %
2-gram	314 843 401	5.0 GB	75.12 %	43.26 %
3-gram	977 069 902	19.0 GB	83.24 %	48.65 %
4-gram	1 313 818 354	30.5 GB	90.27 %	49.54 %
5-gram	1 176 470 663	32.1 GB	94.13 %	47.16 %
Σ	3 354 253 200	77.9 GB	88.37 %	54.20 %

Table 2. EBNF grammar of the query language.

Production Rule		
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 } ” } ”

sensible operators are conceivable, which is part of our work in progress: constraints on particular parts of speech, person names, places, dates, and times.

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 above. This is achieved within two steps: (*i*) computation of the intersection postlist $\pi_q = \bigcap_{w \in q} \pi_w$, and (*ii*) filtering of π_q with a pattern matcher that is compiled at runtime from the regular expression defined by q. Reaching perfect precision and recall is no algorithmic challenge unless retrieval time is considered. Note in this respect that the length of a postlist often amounts up to millions of entries, which is for instance the case for stop words. If a query contains only stop words, the retrieval time for D_q may take tens of seconds up to a minute, depending on the size of the indexed corpus. From a user 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 don't 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. Note that Web search engines typically get queries with (comparatively infrequent) topic words.

To allow for an adjustable retrieval time at the cost of recall we have devised a probabilistic retrieval strategy, which incorporates rank-awareness within the postlists. Our strategy hence is a special kind of a top-k query processing technique [1, 6]. The strategy requires an offline pre-processing of μ, so that (*i*) each postlist is sorted in order of decreasing occurrence frequencies, and (*ii*) 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. As a consequence, the portion of a postlist whose frequencies belong to the long tail of the distribution is pruned from the search. Note that the retrieval precision remains unaffected by this.

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 raise the recall, e.g., by adjusting κ in a query-specific manner.

3 Evaluation

To evaluate the retrieval quality of our query processing strategy, we report here 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; the service is in public use since 2008. We distinguish between macro-averaged recall and micro-averaged recall:

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

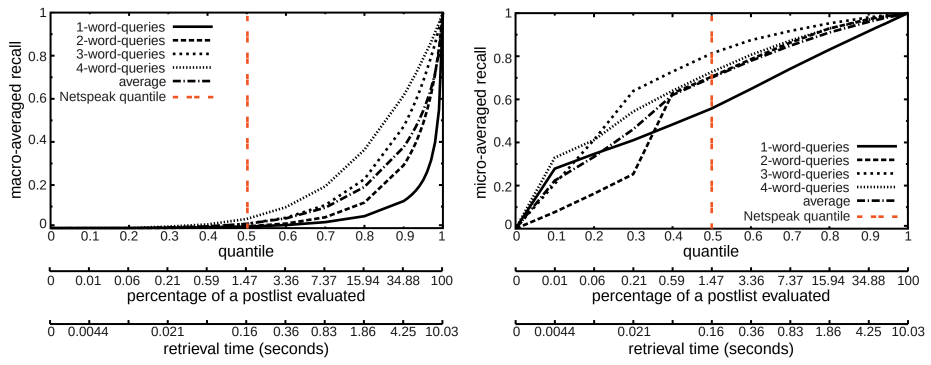

Figure 1. Macro-averaged recall (left) and micro-averaged recall (right) over quantiles. The additional axes indicate how much of a postlist is evaluated and the required processing time.

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} allots more weight to n-grams with high occurrence frequencies, since they are more relevant to the user. Figure 1 shows the obtained results for different query sizes.

Discussion and Conclusion The macro-averaged recall differs significantly from the micro-averaged recall, which indicates that most of the relevant n-grams are retrieved with our strategy. The current NETSPEAK quantile of $\kappa = 0.5$ marks the best 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 result 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. Altogether, our retrieval strategy makes NETSPEAK a fast and reliable writing assistant.

Bibliography

[1] H. Bast, D. Majumdar, R. Schenkel, M. Theobald, and G. Weikum. IO-Top-k: Index-access Optimized Top-k Query Processing. *Proc. of VLDB'06*.

[2] D. Belazzougui, F.C. Botelho, and M. Dietzfelbinger. Hash, Displace, and Compress. *Proc. of ESA'09*.

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

[4] C. Brockett, W.B. Dolan, and M. Gamon. Correcting ESL Errors Using Phrasal SMT Techniques. *Proc. of ACL'06*.

[5] M.J. Cafarella and O. Etzioni. A Search Engine for Natural Language Applications. *Proc. of WWW'05*.

[6] 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.

[7] C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT, 1999.

[8] P. Resnik and A. Elkiss. The Linguist's Search Engine: An Overview. In *Proc. of ACL'05*.