Using Web N-Grams to Help Second-Language Speakers

Martin Potthast   Martin Trenkmann   Benno Stein

Bauhaus-Universität Weimar

www.webis.de

Introduction

Introduction

Writing in a foreign language is difficult.

Problems include

- Spelling
- Grammar
- Translation
- Word Choice
- Writing Style

Tools include

- Spell checkers.
- Grammar checkers.
- Dictionaries, (machine translation).
- Thesauri.
- Style checkers.

Anything missing?

Introduction

What about text commonness?

Introduction

What about text commonness?

Correctness vs. Commonness

We present NETSPEAK, a tool

- to assist with word choice, and
- to check phrase commonness.

NETSPEAK implements wildcard queries on top of a Web n-gram index.

Netspeak *The Writing Assistant*

looks fine ? me — Search

Frequency		Phrase	Example
19,103	93.8 %	looks fine to me	⊞
810	4.0 %	looks fine for me	⊞
353	1.7 %	looks fine with me	⊞
107	0.5 %	looks fine by me	⊞
20,373	100.0 %		0.186 seconds

Home Learn more Plugin API Terms About Permalink

http://www.netspeak.cc

Wildcard N-Gram Retrieval

Wildcard N-Gram Retrieval

Given a set of n-grams, $n \leq 5$, and their frequencies.

A query q defines a pattern as a sequence of n-grams and wildcards.

A wildcard may be substituted for a defined subset of the n-grams.

Given a query q, retrieve all n-grams that match q.

Wildcard N-Gram Retrieval

Given a set of n-grams, $n \leq 5$, and their frequencies.

A query q defines a pattern as a sequence of n-grams and wildcards.

A wildcard may be substituted for a defined subset of the n-grams.

Given a query q, retrieve all n-grams that match q.

Straightforward solution:

- Construct a keyword index for the n-grams.
- Retrieve all n-grams that contain all of q's words.
- Compile a pattern matcher from q and filter the retrieved n-grams.

Improvements:

- Exploit information encoded in queries and n-grams, and that n is small.
- Exploit closed retrieval settings, e.g., the n-gram set is constant.
- Trade wildcard expressiveness and retrieval recall for time.
- Exploit information about the application domain.

Wildcard N-Gram Retrieval

`use the same ?`

- Only 4-grams can match.
- First word `use`, second word `the`, third word `same`.

Our index stores information about n-gram length and word position in the pre-image of the index lookup function.

`prefer * over`

- 2- to 5-grams can match.
- First word `prefer`, and last word `over`.

Variable-length queries are sub-divided into fixed-length queries:
`prefer over`; `prefer ? over`; `prefer ?? over`; `prefer ??? over`

More search heuristics are described in [Stein *et al.*, ECIR'2010]

Netspeak *The Writing Assistant*

looks fine ? me | Search

Frequency		Phrase	Example
19,103	93.8 %	looks fine to me	⊞
810	4.0 %	looks fine for me	⊞
353	1.7 %	looks fine with me	⊞
107	0.5 %	looks fine by me	⊞
20,373	100.0 %		0.186 seconds

Home Learn more Plugin API Terms About Permalink

http://www.netspeak.cc

Wildcard N-Gram Retrieval

Wildcard N-Gram Retrieval

Given a set G of n-grams, $n \leq 5$, and their frequencies $f : G \to \mathbf{N}$.

A query q defines a pattern as a sequence of n-grams and wildcards.

A wildcard may be substituted for every n-gram from a defined subset of G.

Given a query q, retrieve all n-grams R from G that match q.

Wildcard N-Gram Retrieval

Given a set G of n-grams, $n \leq 5$, and their frequencies $f : G \rightarrow \mathbf{N}$.

A query q defines a pattern as a sequence of n-grams and wildcards.

A wildcard may be substituted for every n-gram from a defined subset of G.

Given a query q, retrieve all n-grams R from G that match q.

Straightforward solution:

- Construct an inverted index $\mu : V \rightarrow \mathcal{P}(G)$, where V is G's vocabulary.
- Retrieve all n-grams $R = \bigcap_{w \in q} \mu(w)$ that contain all of q's words $w \in V$.
- Compile a pattern matcher from q and filter R.

Improvements:

- Exploit information encoded in queries and n-grams, and that n is small.
- Exploit closed retrieval settings, e.g., if G is constant.
- Trade wildcard expressiveness and retrieval recall for time.
- Exploit information about the application domain.

Wildcard N-Gram Retrieval

NETSPEAK's approach:

- Construct an inverted index $\mu : V \times \underbrace{\{1,\ldots,5\}}_{\text{n-gram length}} \times \underbrace{\{1,\ldots,5\}}_{\text{word position}} \rightarrow \mathcal{P}(G)$

- Sort $\mu(w,i,j)$ in descending order of f, where $w \in V$ and $i, j \in \{1,\ldots,5\}$.

Wildcard N-Gram Retrieval

NETSPEAK's approach:

- Construct an inverted index $\mu : V \times \underbrace{\{1,\ldots,5\}}_{\text{n-gram length}} \times \underbrace{\{1,\ldots,5\}}_{\text{word position}} \to \mathcal{P}(G)$

- Sort $\mu(w,i,j)$ in descending order of f, where $w \in V$ and $i,j \in \{1,\ldots,5\}$.

- Subdivide q into $\{q_1,\ldots,q_m\}$ so that $R_q = \bigcup_{i=1}^{m} R_{q_i}$, and each q_i matches only n-grams with a fixed length. Process the sub-queries in parallel.

- Retrieve all n-grams $R_{q_i} = \bigcap_{w \in q_i} \mu(w, |q_i|, q_{i|w})$, where $q_{i|w}$ is w's position in q_i.

Wildcard N-Gram Retrieval

NETSPEAK's approach:

- Construct an inverted index $\mu : V \times \underbrace{\{1,\ldots,5\}}_{\text{n-gram length}} \times \underbrace{\{1,\ldots,5\}}_{\text{word position}} \rightarrow \mathcal{P}(G)$

- Sort $\mu(w,i,j)$ in descending order of f, where $w \in V$ and $i,j \in \{1,\ldots,5\}$.

- Subdivide q into $\{q_1,\ldots,q_m\}$ so that $R_q = \bigcup_{i=1}^{m} R_{q_i}$, and each q_i matches only n-grams with a fixed length. Process the sub-queries in parallel.

- Retrieve all n-grams $R_{q_i} = \bigcap_{w \in q_i} \mu(w, |q_i|, q_{i|w})$, where $q_{i|w}$ is w's position in q_i.

- Start to process each $\mu(w,i,j)$ at entry k, with $f(\mu(w,i,j)_k) \leq \min_{g \in q}(f(g))$.

- Stop to process each $\mu(w,i,j)$ at entry

$$\begin{cases} |\mu(w,i,j)| & \text{if the postlist is smaller than a page, or} \\ l_1 & \text{if a pre-specified amount of results have been retrieved, or} \\ l_2 & \text{if } \sum_{i'=0}^{l_2} f(\mu(w,i,j)_{i'}) \text{ covers } \kappa\% \text{ of the frequency distribution.} \end{cases}$$