Chapter IR:III

III. Text Transformation

- Text Statistics
- Parsing Documents
- Information Extraction
- Link Analysis

Parsing Documents

Retrieval Unit

The atomic unit of retrieval of a search engine is typically a document.

Relation between documents and files:

- One file, one document.
  Most commonly compiled in a text file is considered a document. Examples: web page, PDF, Word file.
- One file, many documents.
  Examples: archive files, email threads and attachments, Sammelbände.
- Many files, one document.
  Examples: web-based slide decks, paginated web pages, e.g., forum threads.

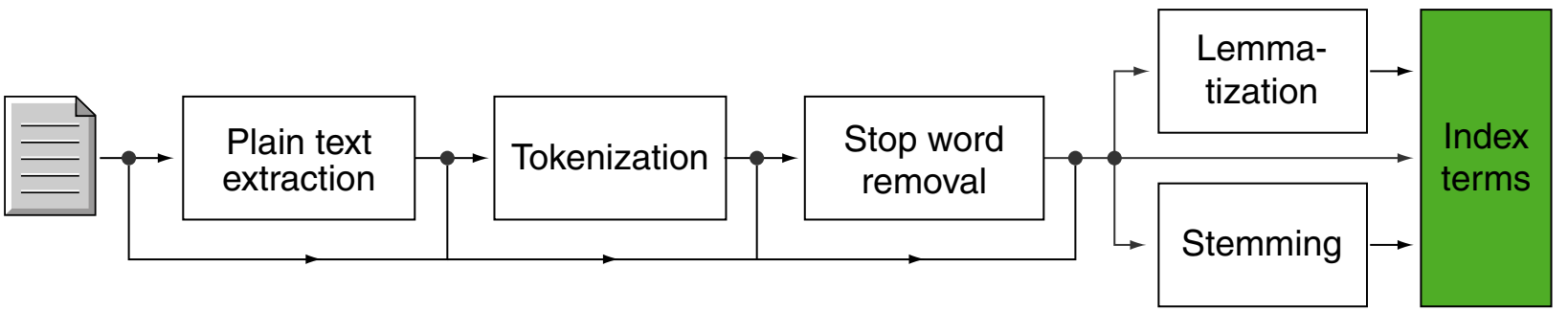
Dependent on the search domain, a retrieval unit may be defined different from what is commonly considered a document:

- One document, many units.
  Examples: comments, reviews, discussion posts, arguments, chapters, sentences, words, etc.

Parsing Documents

Index Term

Documents and queries are preprocessed into sets of normalized index terms.



The primary goal of preprocessing is to unify the vocabularies of documents and queries. Each preprocessing step is a heuristic to increase the likelihood of semantic matches at the expense of spurious matches.

A secondary goal of preprocessing is to create supplemental index terms to improve retrieval performance, e.g., for documents that do not posses many of their own.

Parsing Documents

Document Structure and Markup

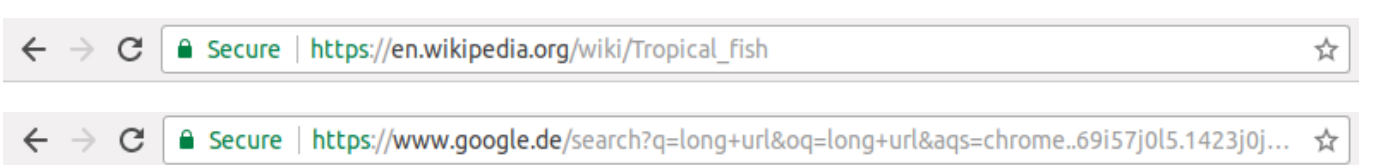
The most common document format for web search engines is HTML. Non-HTML documents are converted to HTML documents for a unified processing pipeline.

Index terms are obtained from URLs and HTML markup.

The address bar of a browser is an important interface for users. Web page authors keep URLs short to facilitate easy recollection and fast typing if a web page has importance in itself.

- Hostnames name the publisher of a web page, paths what it is about.
- Words appearing in the visible parts of a URL are most important.
- Paths may also inform about hierarchical structure of a website.
- Separating words from noise in URLs is an important task.

Examples:



Parsing Documents

Document Structure and Markup

Secure | https://en.wikipedia.org/wiki/Tropical_fish

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Article Talk | Read Edit View history | Search Wikipedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item

Tropical fish

From Wikipedia, the free encyclopedia

Tropical fish are generally those fish found in aquatic tropical environments around the world, including both freshwater and saltwater species. Fishkeepers often keep tropical fish in freshwater and saltwater aquariums.

Contents [hide]

Many tropical fish are coral reef fish.

Aquarium fish [edit]

Tropical fish is a term commonly used to refer to fish that are kept in heated aquariums. Freshwater tropical fish are more commonly kept than saltwater tropical fish due to the common availability of fresh water sources, such as tap water, whereas salt water is not commonly available and has to be recreated by using fresh water with sea salt additions. Salt

Parsing Documents

Document Structure and Markup

```
<html>
<head>
<meta name="keywords" content="Tropical fish, Airstone, Albinism, Algae eater,
Aquarium, Aquarium fish feeder, Aquarium furniture, Aquascaping, Bath treatment
(fishkeeping),Berlin Method, Biotope" />
...
<title>Tropical fish - Wikipedia, the free encyclopedia</title>
</head>
<body>
...
<h1 class="firstHeading">Tropical fish</h1>
...
<p><b>Tropical fish</b> include <a href="/wiki/Fish" title="Fish">fish</a> found in <a
href="/wiki/Tropics" title="Tropics">tropical</a> environments around the world,
including both <a href="/wiki/Fresh_water" title="Fresh water">freshwater</a> and <a
href="/wiki/Sea_water" title="Sea water">salt water</a> species. <a
href="/wiki/Fishkeeping" title="Fishkeeping">Fishkeepers</a> often use the term
<i>tropical fish</i> to refer only those requiring fresh water, with saltwater tropical fish
referred to as <i><a href="/wiki/List_of_marine_aquarium_fish_species" title="List of
marine aquarium fish species">marine fish</a></i>.</p>
<p>Tropical fish are popular <a href="/wiki/Aquarium" title="Aquarium">aquarium</a>
fish , due to their often bright coloration. In freshwater fish, this coloration typically
derives from <a href="/wiki/Iridescence" title="Iridescence">iridescence</a>, while salt
water fish are generally <a href="/wiki/Pigment" title="Pigment">pigmented</a>.</p>
...
</body>
</html>
```

- User-centered meta data: title, emphasized text, anchor text
- Machine readable meta data: Dublin Core [dublincore.org], geotagging [Wikipedia], Schema.org [schema.org]
- Never trust user input: strict limits apply to what is recognized and indexed.

Parsing Documents

Tokenization

Tokenization turns a sequence of characters into a sequence of tokens.

Example:

```
Friends, Romans, Countrymen, lend me your ears;
```

`Friends` `,` `Romans` `,` `Countrymen` `,` `lend` `me` `your` `ears` `;`

Terminology: (simplified)

- A token is a character sequence forming a useful semantic unit.
- A term is a normalized token.

Heuristics: (too basic for practical use)

- Any character sequence separated by white space characters.
- Any alphanumeric sequence of characters of length > 3, separated by a space or punctuation mark. (Used in early TREC evaluations.)

Remarks:

- An important philosophical concept is the type-token distinction. Here, a token is a specific instance of a word (i.e., its specific written form), and a type refers to its underlying concept as a whole. This is comparable to the distinction between class and object in object-oriented computer programming. For example, the sentence “A rose is a rose is a rose.” comprises nine token instances but only four types, namely “a”, “rose”, “is”, and “.”. [Wikipedia]
- Tokenization is strongly language-dependent. English is already among the easiest languages to be tokenized, and there are still many problems to be solved. In Chinese, for example, words are not separated by a specific character, rendering the process of determining word boundaries much more difficult.

Parsing Documents

Tokenization Problems

- Token length

  Short words may be important. Examples: `xp`, `ma`, `pm`, `ben e king`, `el paso`, `master p`, `gm`, `j lo`, `world war II`.

- Capitalization

  Capitalization may carry distinctions between word semantics. Examples: `Bush` vs. `bush`, `Apple` vs. `apple`. Text is still typically lower-cased, since search engine users do not bother with capitalization when typing queries.

- Apostrophe, Hyphen, other special characters
  - Apostrophes can be a part of a word, a part of a possessive, or just a mistake: `it's`, `o'donnell`, `can't`, `don't`, `80's`, `men's`, `master's degree`, `shriner's`
  - Hyphens may be part of a word, a separator, and some words refer to the same concept with or without hyphen: `winston-salem`, `e-bay`, `wal-mart`, `active-x`, `cd-rom`, `t-shirts`.
  - Special characters may form part of words, especially in technology-related text: `M*A*S*H`, `I.B.M.`, `Ph.D.`, `C++`, `C#`, ` `, `http://www.example.com`.

Parsing Documents

Tokenization Problems (continued)

- Numbers
  Numbers form token of their own, and may contain punctuation as well: `6.5`, `1e+010`.
- Tokens with spaces: named entities, dates
  `San Francisco`, `(800) 234-2333`, `Mar 11, 1983`.
- Compound tokens
  German: `Computerlinguistik` for computer linguistics.

Tokenization Approaches

- Rule-based
- Machine learning-based
- ➜ Multiple tokenization
  There is no reason why a text should have just one possible token sequence. Using paradigmatically different tokenization approaches can yield a variety of tokens for the same text, all of which can be used simultaneously for indexing.

Parsing Documents

Stopping (Token Removal)

Stopping refers to the removal of tokens from a token sequence that are not useful for a search engine.

- Frequent tokens (collection-specific)
  Example: `Wikipedia` when indexing Wikipedia.
- Function word tokens (language-dependent)
  `the`, `of`, `and`, etc; strong overlap with frequent tokens.
  Counterexample: `to be or not to be` would be completely lost.
- Punctuation-only tokens
  Counterexample: `;-)`
- Number-only tokens
- Short tokens

Stopping reduces index space, improves query processing speed, and slightly improves effectiveness. The latter, however, is typically handled by more sophisticated retrieval models. If space is no concern, index everything.

Stop word lists are typically customized to the search domain.

Parsing Documents

Token Normalization

Application of heuristic rules to each token in an attempt to unify them.

- Lower-casing
- Removal of special characters
  Example: `U.S.A` → `USA`.
- Removal of diacritical marks
  Example: `café` → `cafe`.
- Stemming
- Lemmatization

Parsing Documents

Stemming

Mapping of a word token to its word stem by removal of inflection.

Inflections:

- noun declination (grammatical case, numerus, gender)
- verb conjugation (grammatical person, numerus, tense, mode, . . .)
- adjective and adverb comparison

A word stem forms the part of a word which remains after removing inflections (e.g., affixes).

Example:

connect	connects
	connected
	connecting
	connection

Parsing Documents

Stemming: Principles [Frakes 1992]

1. Table lookup:
   Given a word stem, store its inflections in a hash table. Problem: completeness.

2. Affix elimination:
   Rule-based algorithms to identify prefixes and suffixes. Given their efficiency and intuitive workings, these are most commonly used.

3. Character n-grams:
   Usage of 4-grams or 5-grams from tokens as stems. Basic heuristic for English: use the first 4 characters as stem.

4. Successor variety:
   Exploits knowledge about structural linguistics to identify morpheme boundaries. The character sequences of tokens are added to a trie data structure; the outdegrees of inner nodes are analyzed to find suitable stems. Problem: difficult to operationalize.

Parsing Documents

Stemming: Affix Elimination

Principle: “iterative longest match stemming”

1. Removal of the longest possible matches based on a set of rules.
2. Repetition of Step 1 until no rule can be applied, anymore.
3. Recoding to address irregularities captured by the rules.

Parsing Documents

Stemming: Affix Elimination

Principle: “iterative longest match stemming”

1. Removal of the longest possible matches based on a set of rules.
2. Repetition of Step 1 until no rule can be applied, anymore.
3. Recoding to address irregularities captured by the rules.

Notation:

- $\mathtt{c}$ denotes a consonant, $\mathtt{C}$ a non-empty sequence of consonants.
  $\mathtt{v}$ denotes a vowel, $\mathtt{V}$ a non-empty sequence of vowels.
  - ➜ Every word is a defined by $[\mathtt{C}](\mathtt{VC})^{\mathtt{m}}[\mathtt{V}]$
- Consonant: Letter that is not a vowel.
- Vowel: Letters A, E, I, O, and U as well as Y after a consonant.
  Example: In TOY the Y is a consonant, in LOVELY the Y is a vowel.

Parsing Documents

Stemming: Porter Stemmer

Concepts:

- 9 rule sets, each consisting of 1-20 rules
- the rules of each group are sorted, to be applied top to bottom
- only one rule per set can be applied
- Rules are defined as follows: `<Premise> S1` $\longrightarrow$ `S2`

Parsing Documents

Stemming: Porter Stemmer

Concepts:

- 9 rule sets, each consisting of 1-20 rules
- the rules of each group are sorted, to be applied top to bottom
- only one rule per set can be applied
- Rules are defined as follows: `<Premise> S1` $\longrightarrow$ `S2`

Semantics:

If a character sequence ends with `S1` and if the subsequence ahead of `S1` (= word stem) fulfills the `<Premise>`, replace `S1` by `S2`

Premises:

`(m>x)`	Number of vowel-consonant-sequences is larger than `x`.
`(*S)`	Word stem ends with `S`.
`(*v*)`	Word stem contains a vowel.
`(*o)`	Word stem ends with `cvc`, where the second consonant `c` $\notin$ `{W, X, Y}`.
`(*d)`	Word stem ends with two identical consonants.

Parsing Documents

Stemming: Porter Stemmer

Selection of rules:

Rule set	Premise	Suffix	Replacement	Example
1a	Null	sses	ss	caresses → caress
1a	Null	ies	i	ponies → poni
1b	(m>0)	eed	ee	feed → fee agreed → agree
1b	(*v*)	ed	ε	plastered → plaster bled → bled
1b	(*v*)	ing	ε	motoring → motor sing → sing
1c	(*v*)	y	i	happy → happi sky → sky
2	(m>0)	biliti	ble	sensibiliti → sensible

Parsing Documents

Stemming: Porter Stemmer

Original text:

```
A relevant document will describe marketing strategies carried out by
U.S. companies for their agricultural chemicals, report predictions for
market share of such chemicals, or report market statistics for
agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer,
predicted sales, market share, stimulate demand, price cut, volume of
sales.
```

Porter stemmer (stop words removed):

```
relevant document describ market strategi carri compani agricultur chemic
report predict market share chemic report market statist agrochem
pesticid herbicid fungicid insecticid fertil predict sale market share
stimul demand price cut volum sale
```

Parsing Documents

Stemming: Porter Stemmer

Weaknesses of the algorithm:

- difficult to modify:
  The effects of changes are hardly predictable.
- tends of overgeneralize:
  `policy/police university/universe organization/organ`
- does not capture clear generalizations:
  `European/Europe matrices/matrix machine/machinery`
- generates word stems that are difficult to be interpreted:
  `iteration/iter general/gener`

Parsing Documents

Stemming: Krovetz Stemmer

The Krovetz stemmer combines a dictionary-based approach with rules. The dictionary captures well-known cases, whereas the rules capture words not known at the time of dictionary creation.

1. Word looked up in dictionary
2. If present, replaced with word stem
3. If not present, word is checked for removable inflection suffixes
4. After removal, dictionary is checked again
5. If still not present, different suffixes are tried

Observations:

- Captures irregular cases such as `is, be, was`.
- Produces words not stems (more readable, similar to lemmatization)
- Comparable effectiveness to Porter stemmer
- Lower false positive rate, somewhat higher false negative rate

Parsing Documents

Stemming: Stemmer Comparison

Original text:

```
A relevant document will describe marketing strategies carried out by
U.S. companies for their agricultural chemicals, report predictions for
market share of such chemicals, or report market statistics for
agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer,
predicted sales, market share, stimulate demand, price cut, volume of
sales.
```

Porter stemmer (stop words removed):

```
relevant document describ market strategi carri compani agricultur chemic
report predict market share chemic report market statist agrochem
pesticid herbicid fungicid insecticid fertil predict sale market share
stimul demand price cut volum sale
```

Krovetz stemmer (stop words removed):

```
relevant document describe marketing strategy carry company agriculture
chemical report prediction market share chemical report market statistic
agrochemic pesticide herbicide fungicide insecticide fertilizer predict
sale market share stimulate demand price cut volume sale
```

Parsing Documents

Stemming: Character n-grams [McNamee et al. 2004] [McNamee et al. 2008]

A substring of length n from a longer string is called a character n-gram. A string of length $m \geq n$ has at most $(m - n) + 1$ distinct character n-grams.

Example: `A relevant document ...`

- 1-grams: `A, r, e, l, e, v, a, n, t, d, o, c, u, m, e, n, t`
- 2-grams: `A, re, el, le, ev, va, an, nt, do, oc, cu, um, me, en, nt`
- 3-grams: `A, rel, ele, lev, eva, van, ant, doc, ocu, cum, ume, men, ent`
- 4-grams: `A, rele, elev, leva, evan, vant, docu, ocum, cume, umen, ment`
- 5-grams: `A, relev, eleva, levan, evant, docum, ocume, cumen, ument`

Parsing Documents

Stemming: Character n-grams [McNamee et al. 2004] [McNamee et al. 2008]

A substring of length n from a longer string is called a character n-gram. A string of length $m \geq n$ has at most $(m - n) + 1$ distinct character n-grams.

Example: `A relevant document ...`

- 1-grams: `A, r, e, l, e, v, a, n, t, d, o, c, u, m, e, n, t`
- 2-grams: `A, re, el, le, ev, va, an, nt, do, oc, cu, um, me, en, nt`
- 3-grams: `A, rel, ele, lev, eva, van, ant, doc, ocu, cum, ume, men, ent`
- 4-grams: `A, rele, elev, leva, evan, vant, docu, ocum, cume, umen, ment`
- 5-grams: `A, relev, eleva, levan, evant, docum, ocume, cumen, ument`

Use all character n-grams for $n = 4$ or $n = 5$ as pseudo-stems of a word.

Observations:

- Language-independent; good retrieval performance for many languages.
- Well-developed stemmers yield better performance (e.g., for English).
- Large overhead in terms of vocabulary size and index size.

Parsing Documents

Phrases (Multi-Token Index Terms)

Queries frequently contain token sequences that are likely to occur consecutively in documents indexed. These word sequences are called phrases.

Examples are names of people, places, products, organizations, etc.

Using phrases as additional index terms may allow for much more efficient and effective query processing. Tokenization therefore also includes phrase extraction.

There are basically three approaches to deal with phrases:

- Positional indexing
  Inclusion of token positions in documents in the index. This allows for checking whether query tokens appear consecutively (or in close proximity) at query processing time.
- (Token) n-grams
  Extraction of all token sequences of length n (e.g., $n \leq 5$), as additional index terms.
- Noun phrase extraction
  Extraction of linguistically defined phrases with a head noun.

Remarks:

- In everyday speech, a phrase may be any group of words, often carrying a special idiomatic meaning; in this sense it is roughly synonymous with expression. In linguistic analysis, a phrase is a group of words (or possibly a single word) that functions as a constituent in the syntax of a sentence, a single unit within a grammatical hierarchy. [Wikipedia]

Parsing Documents

Phrases: N-Grams

A subsequence of length n from a longer token sequence is called a token n-gram. A token sequence of length $m \geq n$ has at most $(m - n) + 1$ distinct token n-grams.

Example:

- 1-grams: `The quick brown fox jumps over the lazy dog`
- 2-grams: `The quick, quick brown, brown fox, fox jumps, jumps over, over the, the lazy, lazy dog`
- 3-grams: `The quick brown, quick brown fox, brown fox jumps, fox jumps over, jumps over the, over the lazy, the lazy dog`

Observations:

- n-grams for $n \geq 1$ combined fit Zipf's law better than just words. [Williams 2015]
- Heap's law does not apply to $n > 1$; other models are required. [Silva 2016]
- Index size grows linearly with n.
- For $n > 1$, n-gram frequency reveals phrases in common use.
- Stop word n-grams speed up query processing for stop word only queries.

Parsing Documents

Phrases: N-Grams

Google: "All Our N-Grams are Belong to You"

- Web search engines index n-grams for $n > 1$
- Google Web 1T 5-gram Version 1: [LDC 2006]

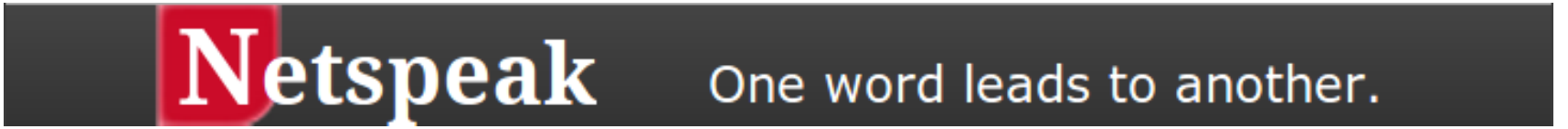
Tokens	1,024,908,267,229
Sentences	95,119,665,584
Unigrams	13,588,391
Bigrams	314,843,401
Trigrams	977,069,902
Fourgrams	1,313,818,354
Fivegrams	1,176,470,663

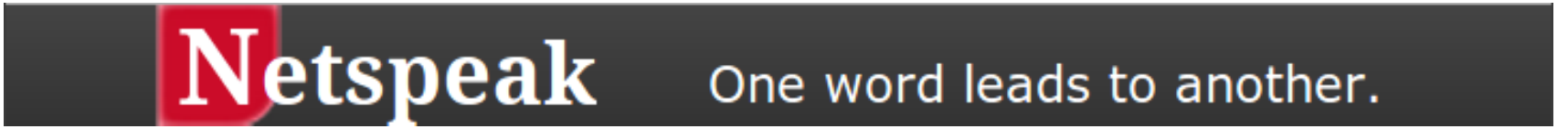
- Most frequent 3-gram on the English web: `all rights reserved`.
- In general, stop word only n-grams do not dominate on the web.
- n-grams with frequency below 40 for $n > 1$ (200 for $n = 1$) are not included.
- Primary use cases for n-gram frequency datasets is language modeling, i.e., training $(n-1)$-order Markov model to predict the next word in a sequence.

Remarks:

- “All your base are belong to us” is a popular Internet meme based on a broken English (“Engrish”) phrase found in the opening cutscene of the 1992 Mega Drive port of the 1989 arcade video game Zero Wing. The quote comes from the European release of the game, featuring poor English translations of the original Japanese version. [Wikipedia]

- Another n-gram collection of interest is the Google Books n-grams collection. It covers the majority of books scanned by the Google Books project in various languages. Besides total frequencies, it also comprises frequencies per year, as per the publication date of all books included. Alongside the data, Google also released a basic search interface to study language use over time, the Google Books Ngram Viewer. See also XKCD.





see ... works		
see how it works	127,000	22.0%
see if it works	100,000	17.3%
see what works	44,000	7.8%
see how this works	31,000	5.4%
see if that works	28,000	4.9%
see how technorati works	23,000	4.0%
see how that works	22,000	3.9%
see if this works	17,000	3.0%
see more works	16,000	2.9%
see if it really works	15,000	2.7%

Netspeak

One word leads to another.

see ... works

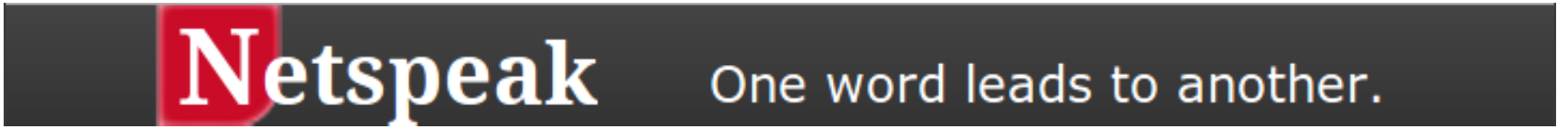
see how it works	127,000	22.0%	+
see if it works	100,000	17.3%	+
see what works	44,000	7.8%	+
see how this works	31,000	5.4%	−

I usually kill plants but we will **see how this works** out.

Ok lets **see how this works**. by guzzmondo. Nov. 23, 2013, 5:16 p.m.. Made with OpenWatch. The Eyes of the World. OpenWatch is a global investigatory network ...

[resolved] Can I **see how this works**? (12 posts). Paul Martin Member Posted 1 year ago #. I've been looking for a solution to be able to change my website's urls ...

with a key to move the bolt but difficult for someone without a key to move it. In the next section, we'll **see how this works** in a basic cylinder lock. Launch Video.



and knows #much

and knows a lot	3,500	65.2%	+
and knows a great deal	690	12.6%	+
and knows much	630	11.5%	+
and knows lots	380	7.1%	+
and knows a good deal	100	1.9%	+
and knows practically	53	1.0%	+
and knows very much	45	0.8%	+