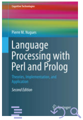
Language Processing with Perl and Prolog

Chapter 5: Counting Words

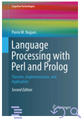
Pierre Nugues

Lund University
Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/



Counting Words and Word Sequences

Words have specific contexts of use.
Pairs of words like *strong* and *tea* or *powerful* and *computer* are not random associations.
Psychological linguistics tells us that it is difficult to make a difference between *writer* and *rider* without context
A listener will discard the improbable *rider of books* and prefer *writer of books*
A language model is the statistical estimate of a word sequence.
Originally developed for speech recognition
The language model component enables to predict the next word given a sequence of previous words: *the writer of books, novels, poetry*, etc. and not *the writer of hooks, nobles, poultry*, . . .



Getting the Words from a Text: Tokenization

Arrange a list of characters:

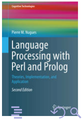
```
[l, i, s, t, ' ', o, f, ' ', c, h, a, r, a, c, t, e, r, s]
```

into words:

```
[list, of, characters]
```

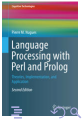
Sometimes tricky:

- Dates: 28/02/96
- Numbers: 9,812.345 (English), 9 812,345 (French and German) 9.812,345 (Old fashioned French)
- Abbreviations: km/h, m.p.h.,
- Acronyms: S.N.C.F.



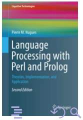
Tokenizing in Perl

```
use utf8;
binmode(STDOUT, ":encoding(UTF-8)");
binmode(STDIN, ":encoding(UTF-8)");

$text = <>;
while ($line = <>) {
  $text .= $line;
}
$text =~ tr/a-zåàâäæçéèêëîïôöœßùûüÿA-ZÅÀÂÄÆÇÉÈÊËÎÏÔÖŒÙÛÜŸ
    ’\-,.?!:;/\n/cs;
$text =~ s/([,.?!:;])/\n$1\n/g;
$text =~ s/\n+/\n/g;
print $text;
```



Improving Tokenization

- The tokenization algorithm is word-based and defines a content
- It does not work on nomenclatures such as Item #N23-SW32A, dates, or numbers
- Instead it is possible to improve it using a boundary-based strategy with spaces (using for instance `\s`) and punctuation
- But punctuation signs like commas, dots, or dashes can also be parts of tokens
- Possible improvements using microgrammars
- At some point, need of a dictionary:
- *Can't* → can n't, *we'll* → we 'll
- *J'aime* → j' aime but *aujourd'hui*



Sentence Segmentation

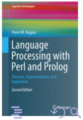
Grefenstette and Tapanainen (1994) used the Brown corpus and experimented increasingly complex rules

Most simple rule: a period corresponds to a sentence boundary: 93.20% correctly segmented

Recognizing numbers:

<table>
<tr><td><code>[0-9]+(\/[0-9]+)+</code></td><td>Fractions, dates</td></tr>
<tr><td><code>([+\-])?[0-9]+(\.)?[0-9]*%</code></td><td>Percent</td></tr>
<tr><td><code>([0-9]+,?)+(\.[0-9]+|[0-9]+)*</code></td><td>Decimal numbers</td></tr>
</table>

93.78% correctly segmented



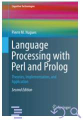
Abbreviations

Common patterns (Grefenstette and Tapanainen 1994):

- single capitals: *A., B., C.,*
- letters and periods: *U.S. i.e. m.p.h.,*
- capital letter followed by a sequence of consonants: *Mr. St. Assn.*

Regex	Correct	Errors	Full stop
`[A-Za-z]\.`	1,327	52	14
`[A-Za-z]\.([A-Za-z0-9]\.)+`	570	0	66
`[A-Z][bcdfghj-np-tvxz]+\.`	1,938	44	26
Totals	3,835	96	106

Correct segmentation increases to 97.66%
With an abbreviation dictionary to 99.07%



N-Grams

The types are the distinct words of a text while the tokens are all the words or symbols.

The phrases from *Nineteen Eighty-Four*

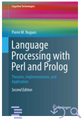
> *War is peace*
> *Freedom is slavery*
> *Ignorance is strength*

have 9 tokens and 7 types.

Unigrams are single words

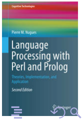
Bigrams are sequences of two words

Trigrams are sequences of three words



Trigrams

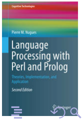
Word	Rank	More likely alternatives
We	9	*The This One Two A Three Please In*
need	7	*are will the would also do*
to	1	
resolve	85	*have know do...*
all	9	*the this these problems...*
of	2	*the*
the	1	
important	657	*document question first...*
issues	14	*thing point to...*
within	74	*to of and in that...*
the	1	
next	2	*company*
two	5	*page exhibit meeting day*
days	5	*weeks years pages months*



Counting Words in Perl: Useful Features

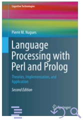
Useful instructions and features: split, sort, and associative arrays (hash tables, dictionaries):

```
@words = split(/\n/, $text);

$wordcount{"a"} = 21;
$wordcount{"And"} = 10;
$wordcount{"the"} = 18;

keys %wordcount
sort array
```



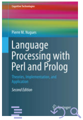
Counting Words in Perl

```
use utf8;
binmode(STDOUT, ":encoding(UTF-8)");
binmode(STDIN, ":encoding(UTF-8)");

$text = <>;
while ($line = <>) {
  $text .= $line;
}
$text =~ tr/a-zåàâäæçéèêëîïôöœßùûüÿA-ZÅÀÂÄÆÇÉÈÊËÎÏÔÖŒÙÛÜŸ
    ’\-,.?!:;/\n/cs;
$text =~ s/([,.?!:;])/\n$1\n/g;
$text =~ s/\n+/\n/g;
@words = split(/\n/, $text);
```



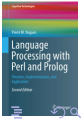
Counting Words in Perl (Cont'd)

```
for ($i = 0; $i <= $#words; $i++) {
  if (!exists($frequency{$words[$i]})) {
    $frequency{$words[$i]} = 1;
  } else {
    $frequency{$words[$i]}++;
  }
}
foreach $word (sort keys %frequency){
  print "$frequency{$word} $word\n";
}
```



Counting Bigrams in Perl

```
@words = split(/\n/, $text);
for ($i = 0; $i < $#words; $i++) {
  $bigrams[$i] = $words[$i] . " " . $words[$i + 1];
}
for ($i = 0; $i < $#words; $i++) {
  if (!exists($frequency_bigrams{$bigrams[$i]})) {
    $frequency_bigrams{$bigrams[$i]} = 1;
  } else {
    $frequency_bigrams{$bigrams[$i]}++;
  }
}
foreach $bigram (sort keys %frequency_bigrams){
  print "$frequency_bigrams{$bigram} $bigram \n";
}
```

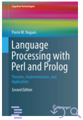


Probabilistic Models of a Word Sequence

$$
\begin{aligned}
P(S) &= P(w_1, ..., w_n), \\
&= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, ..., w_{n-1}), \\
&= \prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1}).
\end{aligned}
$$

The probability $P(\textit{It was a bright cold day in April})$ from *Nineteen Eighty-Four* corresponds to
It to begin the sentence, then *was* knowing that we have *It* before, then *a* knowing that we have *It was* before, and so on until the end of the sentence.

$$
\begin{aligned}
P(S) = &\; P(It) \times P(was|It) \times P(a|It, was) \times P(bright|It, was, a) \times ... \\
&\times P(April|It, was, a, bright, ..., in).
\end{aligned}
$$



Approximations

Bigrams:

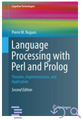
$$P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-1}),$$

Trigrams:

$$P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1}).$$

Using a trigram language model, $P(S)$ is approximated as:

$$\begin{aligned} P(S) \approx & P(It) \times P(was|It) \times P(a|It, was) \times P(bright|was, a) \times ... \\ & \times P(April|day, in). \end{aligned}$$



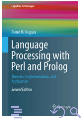
Maximum Likelihood Estimate

Bigrams:

$$P_{MLE}(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{\sum_{w} C(w_{i-1}, w)} = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}.$$

Trigrams:

$$P_{MLE}(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}.$$



Conditional Probabilities

A common mistake in computing the conditional probability $P(w_i|w_{i-1})$ is to use

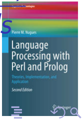
$$\frac{C(w_{i-1}, w_i)}{\#bigrams}.$$

This is not correct. This formula corresponds to $P(w_{i-1}, w_i)$.
The correct estimation is

$$P_{MLE}(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{\sum_w C(w_{i-1}, w)} = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}.$$

Proof:

$$P(w_1, w_2) = P(w_1)P(w_2|w_1) = \frac{C(w_1)}{\#words} \times \frac{C(w_1, w_2)}{C(w_1)} = \frac{C(w_1, w_2)}{\#words}$$



Training the Model

The model is trained on a part of the corpus: the **training set**

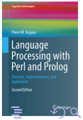
It is tested on a different part: the **test set**

The vocabulary can be derived from the corpus, for instance the 20,000 most frequent words, or from a lexicon

It can be closed or open

A closed vocabulary does not accept any new word

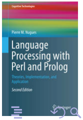
An open vocabulary maps the new words, either in the training or test sets, to a specific symbol, `<UNK>`



Probability of a Sentence: Unigrams

<s> *A good deal of the literature of the past was, indeed, already being transformed in this way* </s>

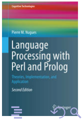
w_i	$C(w_i)$	#words	$P_{MLE}(w_i)$
<s>	7072	–	
a	2482	115212	0.023
good	53	115212	0.00049
deal	5	115212	$4.62\ 10^{-5}$
of	3310	115212	0.031
the	6248	115212	0.058
literature	7	115212	$6.47\ 10^{-5}$
of	3310	115212	0.031
the	6248	115212	0.058
past	99	115212	0.00092
was	2211	115212	0.020
indeed	17	115212	0.00016
already	64	115212	0.00059
being	80	115212	0.00074
transformed	1	115212	$9.25\ 10^{-6}$
in	1759	115212	0.016
this	264	115212	0.0024
way	122	115212	0.0011
</s>	7072	115212	0.065



Probability of a Sentence: Bigrams

<s> A good deal of the literature of the past was, indeed, already being transformed in this way </s>

w_{i-1}, w_i	$C(w_{i-1}, w_i)$	$C(w_{i-1})$	$P_{MLE}(w_i \mid w_{i-1})$
<s> a	133	7072	0.019
a good	14	2482	0.006
good deal	0	53	0.0
deal of	1	5	0.2
of the	742	3310	0.224
the literature	1	6248	0.0002
literature of	3	7	0.429
of the	742	3310	0.224
the past	70	6248	0.011
past was	4	99	0.040
was indeed	0	2211	0.0
indeed already	0	17	0.0
already being	0	64	0.0
being transformed	0	80	0.0
transformed in	0	1	0.0
in this	14	1759	0.008
this way	3	264	0.011
way </s>	18	122	0.148



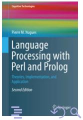
Sparse Data

Given a vocabulary of 20,000 types, the potential number of bigrams is $20,000^2 = 400{,}000{,}000$
With trigrams $20,000^3 = 8{,}000{,}000{,}000{,}000$
Methods:

- Laplace: add one to all counts
- Linear interpolation:

$$P_{DelInterpolation}(w_n|w_{n-2}, w_{n-1}) = \lambda_1 P_{MLE}(w_n|w_{n-2}w_{n-1}) + \lambda_2 P_{MLE}(w_n|w_{n-1}) + \lambda_3 P_{MLE}(w_n),$$

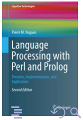
- Good-Turing: The discount factor is variable and depends on the number of times a n-gram has occurred in the corpus.
- Back-off



Laplace's Rule

$$P_{Laplace}(w_{i+1}|w_i) = \frac{C(w_i, w_{i+1}) + 1}{\sum\limits_{w}(C(w_i, w) + 1)} = \frac{C(w_i, w_{i+1}) + 1}{C(w_i) + Card(V)},$$

w_i, w_{i+1}	$C(w_i, w_{i+1}) + 1$	$C(w_i) + Card(V)$	$P_{Lap}(w_{i+1}\|w_i)$
<s> *a*	133 + 1	7072 + 8635	0.0085
a good	14 + 1	2482 + 8635	0.0013
good deal	0 + 1	53 + 8635	0.00012
deal of	1 + 1	5 + 8635	0.00023
of the	742 + 1	3310 + 8635	0.062
the literature	1 + 1	6248 + 8635	0.00013
literature of	3 + 1	7 + 8635	0.00046
of the	742 + 1	3310 + 8635	0.062
the past	70 + 1	6248 + 8635	0.0048
past was	4 + 1	99 + 8635	0.00057
was indeed	0 + 1	2211 + 8635	0.000092
indeed already	0 + 1	17 + 8635	0.00012
already being	0 + 1	64 + 8635	0.00011
being transformed	0 + 1	80 + 8635	0.00011
transformed in	0 + 1	1 + 8635	0.00012
in this	14 + 1	1759 + 8635	0.0014
this way	3 + 1	264 + 8635	0.00045
way </s>	18 + 1	122 + 8635	0.0022



Good–Turing

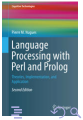
Laplace's rule shifts an enormous mass of probability to very unlikely bigrams. Good–Turing's estimation is more effective

Let's denote N_c the number of n-grams that occurred exactly c times in the corpus.

N_0 is the number of unseen n-grams, N_1 the number of n-grams seen once, N_2 the number of n-grams seen twice The frequency of n-grams occurring c times is re-estimated as:

$$c* = (c+1)\frac{E(N_{c+1})}{E(N_c)},$$

Unseen n-grams: $c* = \frac{N_1}{N_0}$ and N-grams seen once: $c* = \frac{2N_2}{N_1}$.



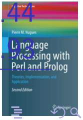
Good-Turing for *Nineteen eighty-four*

Nineteen eighty-four contains 37,365 unique bigrams and 5,820 bigram seen twice.

Its vocabulary of 8,635 words generates $86352^2 = 74{,}563{,}225$ bigrams whose 74,513,701 are unseen.

Unseen bigrams: $\dfrac{37,365}{74,513,701} = 0.0005$. Unique bigrams: $2 \times \dfrac{5820}{37,365} = 0.31$.

Freq. of occ.	N_c	$c*$	Freq. of occ.	N_c	$c*$
0	74,513,701	0.0005	5	719	3.91
1	37,365	0.31	6	468	4.94
2	5,820	1.09	7	330	6.06
3	2,111	2.02	8	250	6.44
4	1,067	3.37	9	179	8.93

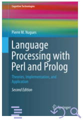


Backoff

If there is no bigram, then use unigrams:

$$P_{\text{Backoff}}(w_i|w_{i-1}) = \begin{cases} P(w_i|w_{i-1}), & \text{if } C(w_{i-1}, w_i) \neq 0, \\ \alpha P(w_i), & \text{otherwise.} \end{cases}$$

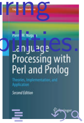
$$P_{\text{Backoff}}(w_i|w_{i-1}) = \begin{cases} P_{\text{MLE}}(w_i|w_{i-1}) = \dfrac{C(w_{i-1}, w_i)}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) \neq 0, \\ P_{\text{MLE}}(w_i) = \dfrac{C(w_i)}{\#\text{words}}, & \text{otherwise.} \end{cases}$$



Backoff: Example

w_{i-1}, w_i	$C(w_{i-1}, w_i)$		$C(w_i)$	$P_{Backoff}(w_i \mid w_{i-1})$
<s>			7072	—
<s> *a*	133		2482	0.019
a good	14		53	0.006
good deal	0	backoff	5	$4.62\ 10^{-5}$
deal of	1		3310	0.2
of the	742		6248	0.224
the literature	1		7	0.00016
literature of	3		3310	0.429
of the	742		6248	0.224
the past	70		99	0.011
past was	4		2211	0.040
was indeed	0	backoff	17	0.00016
indeed already	0	backoff	64	0.00059
already being	0	backoff	80	0.00074
being transformed	0	backoff	1	$9.25\ 10^{-6}$
transformed in	0	backoff	1759	0.016
in this	14		264	0.008
this way	3		122	0.011
way </s>	18		7072	0.148

The figures we obtain are not probabilities. We can use the Good-Turing technique to discount the bigrams and then scale the unigram probabilities. This is the Katz backoff.



Quality of a Language Model

Per word probability of a word sequence: $H(L) = -\frac{1}{n}\log_2 P(w_1, \ldots, w_n)$.
Entropy rate: $H_{rate} = -\frac{1}{n} \sum_{w_1,\ldots,w_n \in L} p(w_1, \ldots, w_n) \log_2 p(w_1, \ldots, w_n)$,

Cross entropy:

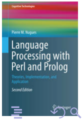
$$H(p,m) = -\frac{1}{n} \sum_{w_1,\ldots,w_n \in L} p(w_1, \ldots, w_n) \log_2 m(w_1, \ldots, w_n).$$

We have:

$$\begin{aligned} H(p,m) &= \lim_{n\to\infty} -\frac{1}{n} \sum_{w_1,\ldots,w_n \in L} p(w_1, \ldots, w_n) \log_2 m(w_1, \ldots, w_n), \\ &= \lim_{n\to\infty} -\frac{1}{n} \log_2 m(w_1, \ldots, w_n). \end{aligned}$$

We compute the cross entropy on the complete word sequence of a test set, governed by p, using a bigram or trigram model, m, from a training set.
Perplexity:

$$PP(p,m) = 2^{H(p,m)}.$$



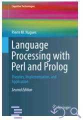
Other Statistical Formulas

- Mutual information (The strength of an association):

$$I(w_i, w_j) = \log_2 \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \approx \log_2 \frac{NC(w_i, w_j)}{C(w_i)C(w_j)}.$$

- T-score (The confidence of an association):

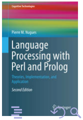
$$\begin{aligned} t(w_i, w_j) &= \frac{mean(P(w_i, w_j)) - mean(P(w_i))mean(P(w_j))}{\sqrt{\sigma^2(P(w_i, w_j)) + \sigma^2(P(w_i)P(w_j))}}, \\ &\approx \frac{C(w_i, w_j) - \frac{1}{N}C(w_i)C(w_j)}{\sqrt{C(w_i, w_j)}}. \end{aligned}$$



T-Scores with Word *set*

Word	Frequency	Bigram *set* + word	*t*-score
up	134,882	5512	67.980
a	1,228,514	7296	35.839
to	1,375,856	7688	33.592
off	52,036	888	23.780
out	12,3831	1252	23.320

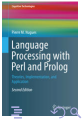
Source: Bank of English



Mutual Information with Word *surgery*

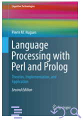
Word	Frequency	Bigram word + *surgery*	Mutual info
arthroscopic	3	3	11.822
pioneeing	3	3	11.822
reconstructive	14	11	11.474
refractive	6	4	11.237
rhinoplasty	5	3	11.085

Source: Bank of English



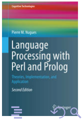
Mutual Information and T-Scores in Perl

```
...

@words = split(/\n/, $text);
for ($i = 0; $i < $#words; $i++) {
  $bigrams[$i] = $words[$i] . " " . $words[$i + 1];
}
for ($i = 0; $i <= $#words; $i++) {
  $frequency{$words[$i]}++;
}
for ($i = 0; $i < $#words; $i++) {
  $frequency_bigrams{$bigrams[$i]}++;
}
```



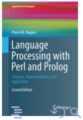
Mutual Information in Perl

```
for ($i = 0; $i < $#words; $i++) {
  $mutual_info{$bigrams[$i]} = log(($#words + 1) *
    $frequency_bigrams{$bigrams[$i]}/
    ($frequency{$words[$i]} * $frequency{$words[$i + 1]}))/
    log(2);
}

foreach $bigram (keys %mutual_info){
  @bigram_array = split(/ /, $bigram);
  print $mutual_info{$bigram}, " ", $bigram, "\t",
    $frequency_bigrams{$bigram}, "\t",
    $frequency{$bigram_array[0]}, "\t",
    $frequency{$bigram_array[1]}, "\n";
}
```



T-Scores in Perl

```
for ($i = 0; $i < $#words; $i++) {
  $t_scores{$bigrams[$i]} = ($frequency_bigrams{$bigrams[$i]}
    -  $frequency{$words[$i]} *
    $frequency{$words[$i + 1]}/($#words + 1))/
    sqrt($frequency_bigrams{$bigrams[$i]});
}

foreach $bigram (keys %t_scores ){
  @bigram_array = split(/ /, $bigram);
  print $t_scores{$bigram}, " ", $bigram, "\t",
    $frequency_bigrams{$bigram}, "\t",
    $frequency{$bigram_array[0]}, "\t",
    $frequency{$bigram_array[1]}, "\n";
}
```

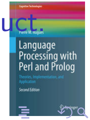


Information Retrieval: The Vector Space Model

The vector space model is a technique to compute the similarity of two documents or to match a document and a query.
The vector space model represents a document in word space:

Documents \Words	w_1	w_2	w_3	...	w_m
D_1	$C(w_1, D_1)$	$C(w_2, D_1)$	$C(w_3, D_1)$	...	$C(w_m, D_1)$
D_2	$C(w_1, D_2)$	$C(w_2, D_2)$	$C(w_3, D_2)$	...	$C(w_m, D_2)$
...					
D_n	$C(w_1, D_1 n)$	$C(w_2, D_n)$	$C(w_3, D_n)$	...	$C(w_m, D_n)$

We compute the similarity of two documents through their dot product.



The Vector Space Model: Example

A collection of two documents D1 and D2 are:

D1: Chrysler plans new investments in Latin America.

D2: Chrysler plans major investments in Mexico.

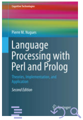
The vectors representing the two documents:

D.	america	chrysler	in	investments	latin	major	mexico	new	plans
1	1	1	1	1	1	0	0	1	1
2	0	1	1	1	0	1	1	0	1

The vector space model represents documents as bags of words (BOW) that do not take the word order into account.

The dot product is $\vec{D}1 \cdot \vec{D}2 = 0+1+1+1+0+0+0+0+1 = 4$

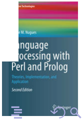
Their cosine is $\frac{\vec{D}1 \cdot \vec{D}2}{||\vec{D}1||.||\vec{D}2||} = \frac{4}{\sqrt{7}.\sqrt{6}} = 0.62$



Giving a Weight

Word clouds give visual weights to words

Image: Courtesy of Jonas Wisbrant



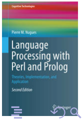
$TF \times IDF$

The frequency alone might be misleading

Document coordinates are in fact $tf \times idf$: Term frequency by inverted document frequency.

Term frequency $tf_{i,j}$: frequency of term j in document i

Inverted document frequency: $idf_j = \log(\frac{N}{n_j})$

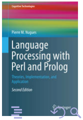


Document Similarity

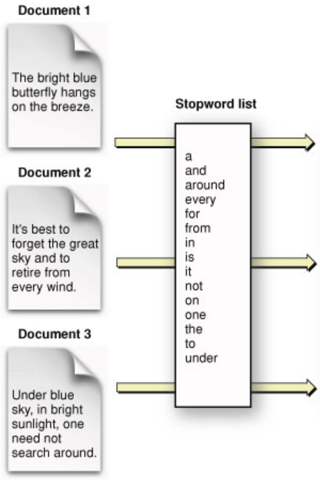
Documents are vectors where coordinates could be the count of each word:
$\vec{d} = (C(w_1), C(w_2), C(w_3), ..., C(w_n))$
The similarity between two documents or a query and a document is given by their cosine:

$$\cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}.$$

Application: Lucene, Wikipedia



Inverted Index (Source Apple)



Inverted index

ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2

http://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/SearchKitConcepts/index.html