Large Web Corpora for Indian Languages

Adam Kilgarriff , Girish Duvuru

Lexical Computing Ltd,
Brighton, UK

Abstract. For many languages there are no large, general-language corpora available. Until the web, all but the richest institutions could do little but shake their heads in dismay as corpus-building was long, slow and expensive. But with the advent of the Web it can be highly automated and thereby fast and inexpensive. In this demo we describe the 'corpus factory' method we use for collecting large web corpora for Indian and other languages. We have recently collected corpora for Hindi, Telugu, Kannada, Urdu, Gujarati, Tamil, Malayalam and Bengali. We also describe the Sketch Engine, a corpus tool that offer lots of language analysis function, and CQL, the advanced query language used by this system.

Keywords: Corpus, Indian Languages, Sketch Engine, CQL

1 Introduction

For the major world languages, large corpora are publicly available. But for most other languages, they are not, especially for Indian Languages. Corpus collection used to be long, slow and expensive - but then came the internet: texts, in vast number, are now available by mouse-click. The prospects of web as corpus were first explored in the late 1990s by Resnik (1999) and early 2000s by Jones and Ghani (2000). Grefenstette and Nioche (2000) showed just how much data was available. Kilgarriff and Grefenstette (2003) present an overview of work up to that date, including Keller and Lapata (2003) which establishes the validity of web corpora by comparing models of human response times for collocations drawn from web frequencies with models drawn from traditional corpus frequencies, and showing that they compared well. Sharoff (2006) has prepared web corpora, typically of around 100 million words, for ten major world languages, primarily for use in teaching translation. Here we collected large corpora for many Indian languages with the process described by Kilgarriff and Reddy (2010). The corpus thus collected is loaded into the Sketch Engine, a tool that offers a number of language-analysis functions. It accepts advanced queries in CQL format (Christ 1994) which great flexibility.

2 Corpus Collection of Indian Languages Brief Description

Our method described by Kilgarriff and Reddy (2010), piggybacks on the work of the commercial search engines (Google, Bing) and is based on the BootCaT method (Baroni and Bernardini, 2004). Search engines crawl and index the Web, identify text-rich pages and address character-encoding issues (though they do this with mixed success, as we see below).

Steps involved in corpora collection are

```
1. Gather a `seed word' list of several hundred midfrequency words of
   the language
2. Repeat several thousand times (until the corpus is large enough):
   a. Randomly select three (typically) of these words to create a
      query
   b. Send the query to a commercial search engine (we have used
      Google, Yahoo and Bing) which returns a 'search hits' page.
   c. Retrieve pages identified in the search hits page. Store them.
3. Clean the text, to remove navigation bars, advertisements and other
   recurring material
```

```
4. Remove duplicates
5. Tokenise, and, where tools are available, lemmatise and POS tag
6. Load into a corpus query tool.
```

For each language, we need seed words to start the process. Wikipedia (Wiki) is a huge knowledge resource built by collective effort with articles from many domains. The whole dataset can be downloaded. These are used as seed words. For each language, a Wiki corpus is extracted from a Wiki dump of the language (a Wiki dump is a single large XML file containing all the articles of the Wikipedia). We used a slightly modified version of the Wikipedia2Text tool[1] to extract plain text. These text files are tokenized to get frequency lists. The first 1000 words are used as stop words and the next 5000, as seed words. (We make use of stop words in cleaning: we reject texts where less than 25% of tokens are stop words.) The seed words are used to make queries to search engines. We use tuples of seed words, typcially 3, 4, or 5: we slect the tuple size to optimise the process acocridn ghte web size for the language. We exploit some features of search engines such as the ability to specify language in queries. This is useful where languages share script and vocabuary.

Table 1. Indian languages currently available (December 2010).

Language	No of Tokens
Hindi	31,355,212
Telugu	4,697,932
Malayalam	21,193,984
Kannada	12,764,312
Bengali	13,719,158
Urdu	16,845,136
Gujarati	22,201,247
Tamil	32,861,569

2.1 Reasons for small size compared to other languages with fewer speakers

For European and East Asian languages, similar methods have readily provided corpora of over 100m words. For Indian languages, we have noted that the web is relatively small given the number of speakers. We suspect this is because the dominant language of education in India is English, coupled with the confusing variety of encodings which are possible for Indian languages: most Indian web users know enough English to use the web in English, and find this easier, as they will not miss pages in the wrong encoding. (For the same reasons, web authors often choose to write in English.) As web use penetrates further, and as encodings standards are more widely adopted, we would expect this to change over the next few years.

For example, for Hindi, we found 31m words but could not find appreciably more. It turns out that there are nine different encodings in use apart from UTF-8. We prepared queries in all nine encodings and then used them to collect corpora, but there was no significant improvement in corpus size. This supports the idea that India's dominant languages on the web is English.

Other problems we faced are for languages that share script (Devanagari) and some vocabulary set with Hindi, like Marathi and Nepali. If we use search engine language filters we find very few documents. Without the language filter we get far more documents but can no longer be sure whether they belong to that language or Hindi. Punjabi can be written in three scripts: Gurumuki, Shanmuki and Devanagari. The Wikipedia dataset is in Gurmuki but there is not much data on the web for this script. So, for a variety of reasons, corpora are small than those for languages from other parts of world.

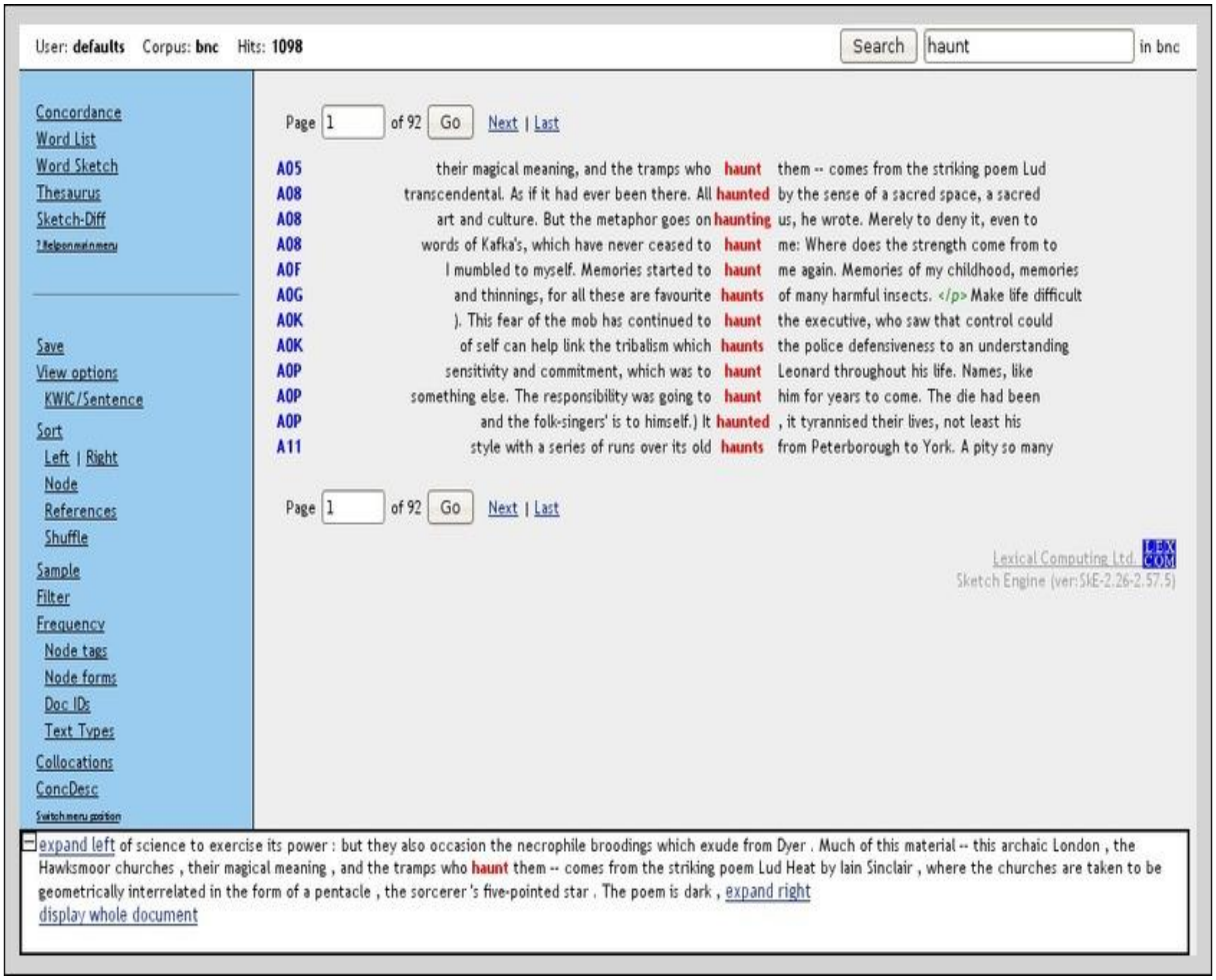
[1] http://evanjones.ca/software/wikipedia2text.html

3 Sketch Engine Demo and functions

Once we have created a corpus, we load it into the Sketch Engine corpus query system (Kilgarriff et al 2004) and make it available through the web service at http://www.sketchengine.co.uk. (Sign up for a free trial; all the corpora listed above, and more as the months proceed, will be available for you to explore.)

The Sketch Engine is a web-based Corpus Query System, which takes as its input a corpus of any language with an appropriate level of linguistic mark-up and offers a number of language-analysis functions like Concordance, word sketches, distributional thesaurus and sketch difference. We need lemmatised and POS tagged corpus, along with grammatical relations for exploiting the whole functionality of Sketch Engine for a given language. Without these tools (which have not yet been applied for Indian languages) we still have concordances, and functionality such as frequency lists and collocate lists. For demo purpose of other functionality we will use English.

A concordance is a display of all occurrences from the corpus for a given query. This system accepts simple queries as well as complex queries in CQL.

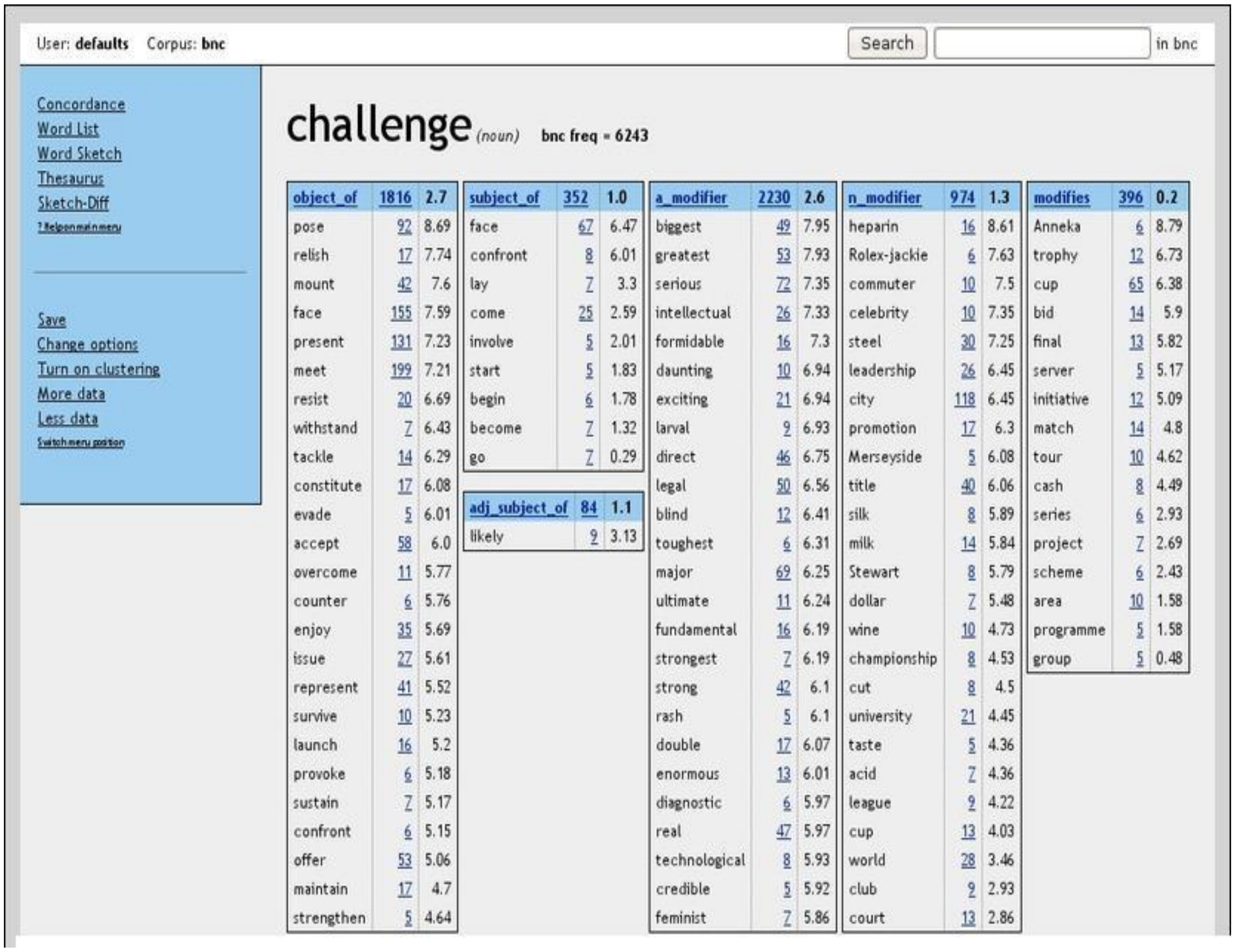


A05	their magical meaning, and the tramps who	haunt	them -- comes from the striking poem Lud
A08	transcendental. As if it had ever been there. All	haunted	by the sense of a sacred space, a sacred
A08	art and culture. But the metaphor goes on	haunting	us, he wrote. Merely to deny it, even to
A08	words of Kafka's, which have never ceased to	haunt	me: Where does the strength come from to
A0F	I mumbled to myself. Memories started to	haunt	me again. Memories of my childhood, memories
A0G	and thinnings, for all these are favourite	haunts	of many harmful insects. </p> Make life difficult
A0K	). This fear of the mob has continued to	haunt	the executive, who saw that control could
A0K	of self can help link the tribalism which	haunts	the police defensiveness to an understanding
A0P	sensitivity and commitment, which was to	haunt	Leonard throughout his life. Names, like
A0P	something else. The responsibility was going to	haunt	him for years to come. The die had been
A0P	and the folk-singers' is to himself.) It	haunted	, it tyrannised their lives, not least his
A11	style with a series of runs over its old	haunts	from Peterborough to York. A pity so many

expand left of science to exercise its power : but they also occasion the necrophile broodings which exude from Dyer . Much of this material -- this archaic London , the Hawksmoor churches , their magical meaning , and the tramps who **haunt** them -- comes from the striking poem Lud Heat by Iain Sinclair , where the churches are taken to be geometrically interrelated in the form of a pentacle , the sorcerer 's five-pointed star . The poem is dark , expand right
display whole document

Fig. 1. Concordance for the English lemma *haunt*.

A Word Sketch is a corpus-based summary of a word's grammatical and collocational behaviour.



challenge *(noun)* bnc freq = 6243

object_of	1816	2.7
pose	92	8.69
relish	17	7.74
mount	42	7.6
face	155	7.59
present	131	7.23
meet	199	7.21
resist	20	6.69
withstand	7	6.43
tackle	14	6.29
constitute	17	6.08
evade	5	6.01
accept	58	6.0
overcome	11	5.77
counter	6	5.76
enjoy	35	5.69
issue	27	5.61
represent	41	5.52
survive	10	5.23
launch	16	5.2
provoke	6	5.18
sustain	7	5.17
confront	6	5.15
offer	53	5.06
maintain	17	4.7
strengthen	5	4.64

subject_of	352	1.0
face	67	6.47
confront	8	6.01
lay	7	3.3
come	25	2.59
involve	5	2.01
start	5	1.83
begin	6	1.78
become	7	1.32
go	7	0.29

adj_subject_of	84	1.1
likely	9	3.13

a_modifier	2230	2.6
biggest	49	7.95
greatest	53	7.93
serious	72	7.35
intellectual	26	7.33
formidable	16	7.3
daunting	10	6.94
exciting	21	6.94
larval	9	6.93
direct	46	6.75
legal	50	6.56
blind	12	6.41
toughest	6	6.31
major	69	6.25
ultimate	11	6.24
fundamental	16	6.19
strongest	7	6.19
strong	42	6.1
rash	5	6.1
double	17	6.07
enormous	13	6.01
diagnostic	6	5.97
real	47	5.97
technological	8	5.93
credible	5	5.92
feminist	7	5.86

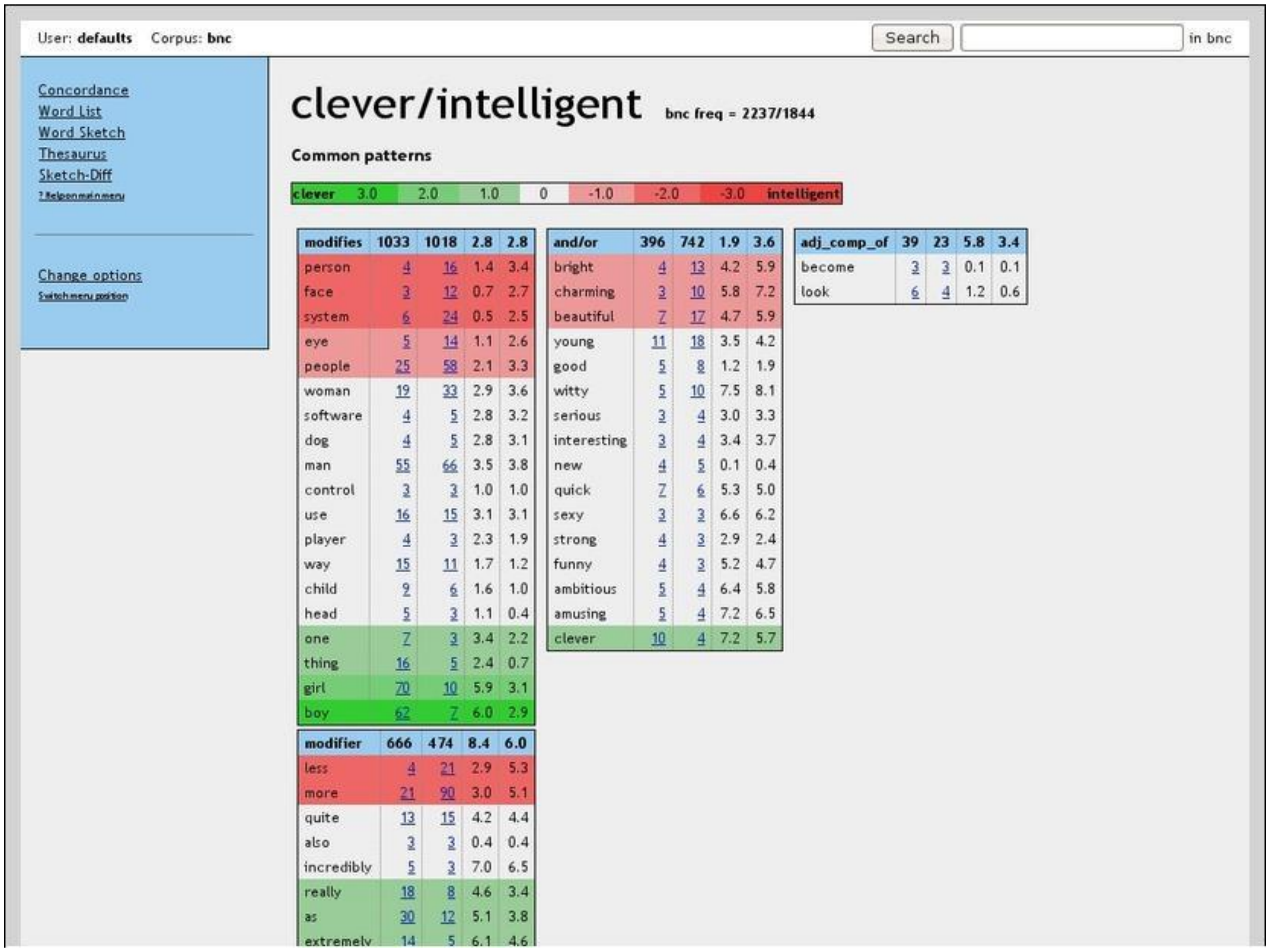
n_modifier	974	1.3
heparin	16	8.61
Rolex-jackie	6	7.63
commuter	10	7.5
celebrity	10	7.35
steel	30	7.25
leadership	26	6.45
city	118	6.45
promotion	17	6.3
Merseyside	5	6.08
title	40	6.06
silk	8	5.89
milk	14	5.84
Stewart	8	5.79
dollar	7	5.48
wine	10	4.73
championship	8	4.53
cut	8	4.5
university	21	4.45
taste	5	4.36
acid	7	4.36
league	9	4.22
cup	13	4.03
world	28	3.46
club	9	2.93
court	13	2.86

modifies	396	0.2
Anneka	6	8.79
trophy	12	6.73
cup	65	6.38
bid	14	5.9
final	13	5.82
server	5	5.17
initiative	12	5.09
match	14	4.8
tour	10	4.62
cash	8	4.49
series	6	2.93
project	7	2.69
scheme	6	2.43
area	10	1.58
programme	5	1.58
group	5	0.48

Fig. 2. Each column show the words that typically combine with *challenge* in a particular grammatical relation (or "gramrel"). Thus "object_of" lists - in order of statistical significance rather than raw frequency - the verbs that most typically occupy the verb slot in cases where *challenge* is the object of a verb.

On the basis of this data the Sketch Engine generates a "distributional thesaurus". A distributional thesaurus is an automatically produced "thesaurus" which finds words that tend to occur in similar contexts, with the same collocates, as the target word.

A 'Sketch Difference' is a neat way of comparing two similar words: it shows those patterns and combinations that the two items have in common, and also those patterns and combinations that are more typical of, or unique to, one word rather than the other.


User: **defaults** Corpus: **bnc**

clever/intelligent **bnc freq = 2237/1844**

Common patterns

clever 3.0	2.0	1.0	0	-1.0	-2.0	-3.0 intelligent

modifies	1033	1018	2.8	2.8
person	4	16	1.4	3.4
face	3	12	0.7	2.7
system	6	24	0.5	2.5
eye	5	14	1.1	2.6
people	25	58	2.1	3.3
woman	19	33	2.9	3.6
software	4	5	2.8	3.2
dog	4	5	2.8	3.1
man	55	66	3.5	3.8
control	3	3	1.0	1.0
use	16	15	3.1	3.1
player	4	3	2.3	1.9
way	15	11	1.7	1.2
child	9	6	1.6	1.0
head	5	3	1.1	0.4
one	7	3	3.4	2.2
thing	16	5	2.4	0.7
girl	70	10	5.9	3.1
boy	62	7	6.0	2.9

and/or	396	742	1.9	3.6
bright	4	13	4.2	5.9
charming	3	10	5.8	7.2
beautiful	7	17	4.7	5.9
young	11	18	3.5	4.2
good	5	8	1.2	1.9
witty	5	10	7.5	8.1
serious	3	4	3.0	3.3
interesting	3	4	3.4	3.7
new	4	5	0.1	0.4
quick	7	6	5.3	5.0
sexy	3	3	6.6	6.2
strong	4	3	2.9	2.4
funny	4	3	5.2	4.7
ambitious	5	4	6.4	5.8
amusing	5	4	7.2	6.5
clever	10	4	7.2	5.7

adj_comp_of	39	23	5.8	3.4
become	3	3	0.1	0.1
look	6	4	1.2	0.6

modifier	666	474	8.4	6.0
less	4	21	2.9	5.3
more	21	90	3.0	5.1
quite	13	15	4.2	4.4
also	3	3	0.4	0.4
incredibly	5	3	7.0	6.5
really	18	8	4.6	3.4
as	30	12	5.1	3.8
extremely	14	5	6.1	4.6

Fig. 3. Suppose you want to compare *clever* and *intelligent*. In the thesaurus entry for *clever*, *intelligent* comes top of the list: it is statistically the most similar word in terms of shared contexts of occurrence. Click on *intelligent* and you are taken to a new screen which shows both of their collocates, colour-coded to show whether they occur more only with *intelligent* (strong red), more with *intelligent* (pale red), equally with both (neutral), more with *clever* (pale green) or only with *clever* (strong green).

4. Introduction to CQL

Corpus Query Language (CQL) was developed at the Corpora and Lexicons group, IMS, University of Stuttgart in the early 1990s (Christ 1994). The underlying representation for the corpus is as a sequence of tokens – words or punctuation – and each token can have a number of attributes associated with it, typically 'word' (the word as it appears in the text), 'lemma' (its lemma), and 'tag' (its part-of-speech tag). The input formalism for the Sketch Engine makes this clear: sometimes called 'vertical' text, it has one token on each line, with the attributes associated with the token on the same line, separated by tabs, thus:

```
The    the    DT
Boys   boy    NNS
Are    be     VBP
Comingcome    VVG
.      .      SENT
```

In CQL a simple query is an ‘attribute expression’ with syntax [attribute=”query word”]. The usual attributes that Sketch Engine uses are for example `[word="clever"], [tag="NN"].` A query can consists of a regular expression over *attribute expressions*. For example `[word="confus.*"].`

We often want a wild-card word: any single word, it doesn't matter which. We use the "match any token" operator `[]` (similar to the dot for "match any character" in regular expressions over strings). The query

```
    [word="confus.*"] [] [word="by"]
```

finds all sequences of a word beginning with **confus**, followed by any word, followed by **by**. The match-any operator must not be the first expression in a query. We search for exactly two words between **confus.*** and **by** with

```
    [word="confus.*"] []{2} [word="by"]
```

We can use part-of-speech tags to make grammatical queries, for example we can search for s sequence of an adjective, a noun, a conjunction and another noun with

```
    [tag="JJ.*"] [tag="N.*"] [word="and|or"] [tag="N.*"]
```

CQL is used to write a ‘Sketch Grammar’ for the language. A Sketch Grammar defines a set of grammatical relations such as ‘subject’ ‘object’, ‘modifier’, and is the additional input required in order that the Sketch Engine can prepare word sketches, thesaurus and sketch differences.

For a full introduction to CQL and Sketch Grammars see the Advanced User Manual in the Sketch Engine Help pages.

6. Future Plans

For the Indian languages where we already have corpora, the next step is to lemmatise and part-of-speech tag. For a number of the languages, taggers developed under the Indian Government’s TDIL (Technology Development for Indian Languages) program are available and we plan to use those. For other languages we shall explore further to seek out a lemmatiser and POS-tagger.

Once the corpus is lemmatized and POS-tagged, the next stage is to write a Sketch Grammar. For this we need an individual who is both a computational linguist and a native or near-native speaker of the language. For Hindi and Telugu, we have this expertise within our team. For the other languages we are currently looking for collaborators to work with us on assessing lemmatiser and POS-tagger output quality, and for Sketch Grammar development.

We are developing corpora for Indian languages including Punjabi, Marathi and Nepali. For these languages, where Wikipedia doesn’t have enough data or where other problems were encountered, we need some base corpus for determining seed words. Once we find the resource we would be able to complement them with our ever improving Sketch Engine services to provide world class resources for Indian languages on a par with European and East Asian ones.

References

Baroni, Marco, and Silvia Bernardini. 2004. Bootcat: Bootstrapping corpora and terms from the web. In Proc. LREC,

Baroni, Marco, and Adam Kilgarriff. 2006. Large linguistically-processed web corpora for multiple languages. In Proc. EACL

Christ, O: "A modular and flexible architecture for an integrated corpus query system". COMPLEX'94, Budapest, 1994.

Grefenstette, Gregory, and Julien Nioche. 2000. Estimation of English and non-English language use on the WWW. In Proc. RIAO, Paris.

Jones, Rosie, and Rayid Ghani. 2000. Automatically building a corpus for a minority language from the web. In Proceedings of the Student Workshop of the 38th Annual Meeting of the Association for Computational Linguistics, pages 29-36.

Keller, Frank, and Mirella Lapata. 2003. Using the Web to Obtain Frequencies for Unseen Bigrams. Computational Linguistics 29(3): 459-484.

Kilgarriff, Adam, and Gregory Grefenstette. 2003. Introduction to a Special Issue on Web as Corpus. Computational Linguistics 29 (3).

Kilgarriff, Adam, Pavel Rychly, Pavel Smrz, David Tugwell 2004. The Sketch Engine. Proc. EURALEX. Lorient, France

Kilgarriff, Adam, Siva Reddy, Jan Pomikalek, Avinesh PVS. 2010. A Corpus Factory for Many Languages, Proc. Language Resource and Evaluation Conference, Malta, May.

Resnik, Philip. 1999. Mining the web for bilingual text. In Proc. 37th Meeting of ACL, pages 527-534, Maryland, June.

Sharoff, S. 2006. Creating general-purpose corpora using automated search engine queries. In WaCky! Working papers on the Web as Corpus. Gedit.

Pomikálek, Jan, and Pavel Rychlý. 2008. Detecting co-derivative documents in large text collections. LREC, Marrakech, Morocco.