**Hindi Word Sketches**

# Anil Krishna Eragani1, Varun Kuchibhotla1, Dipti Sharma1, Siva Reddy2, Adam Kilgarriff3

# 1IIIT Hyderabad, India; 2University of Edinburgh, UK; 3Lexical Computing Ltd., UK

# Abstract

Word sketches are one-page automatic, corpus-based summaries of a word’s grammatical and

collocational behaviour. These are widely used for studying a language and in lexicography. Sketch Engine is a leading corpus tool which takes as input a corpus and generates word sketches for the words of that language. It also generates a thesaurus and ‘sketch differences’, which specify similarities and differences between near-synonyms. In this paper, we present the functionalities of Sketch Engine for Hindi. We collected HindiWaC, a web crawled corpus for Hindi with 240 million words. We lemmatized, POS tagged the corpus and then been loaded it into Sketch Engine.

# Introduction

A language *corpus* is simply a collection of texts, so-called when it is used for language research. Corpora can be used for all sorts of purposes: from literature to language learning; from discourse analysis to grammar to language change to sociolinguistic or regional variation; from translation to technology.

Corpora are becoming more and more important, because of computers. On a computer, a corpus can be searched and explored in all sorts of ways. Of course that requires the right app. One leading app for corpus querying is the Sketch Engine (Kilgarriff et al 2004). The Sketch Engine has been in daily use for writing dictionary entries since 2004, first at Oxford University Press, more recently at Cambridge University Press, Collins, Macmillan, and in National Language Institutes for Czech, Dutch, Estonian, Irish, Slovak and Slovene. It is also in use for all the other purposes listed above. On logging in to the Sketch Engine, the user can explore corpora for sixty languages. In many cases the corpora are the largest and best available for the language. For Indian languages, there are the corpora for Bengali, Gujarati, Hindi, Malayalam, Tamil and Telugu. The largest is for Hindi with 240 million words – we would be referring to it as HindiWaC in the rest of the paper.

The function that gives the Sketch Engine its name is the **'word sketch'**, a one-page, automatically-derived summary of a word's grammatical and collocational behaviour, as in Figure 1. Since the images in this paper are screen shots taken from Sketch Engine, translations and gloss have not been provided for the Hindi words in the images.

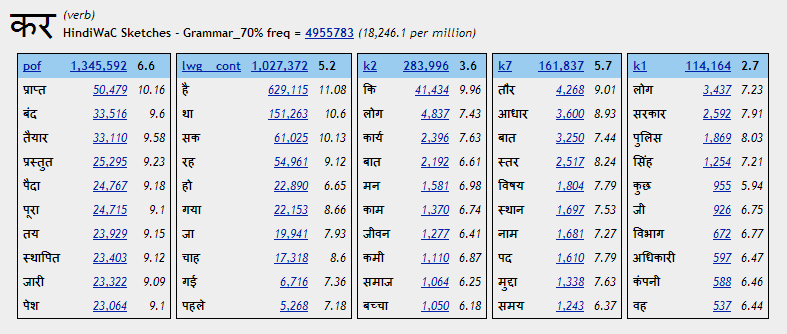


Figure 1: Word sketches for the verb कर (do)

In this paper we first introduce the main functions of the Sketch Engine, with Hindi examples. We then describe how we built and processed HindiWaC, and set it up in the Sketch Engine.

## The Sketch Engine for Hindi

## The Simple Concordance Query function

A Simple concordance query shows the word as it is used in different texts. Figure 2 shows the query box, while Figures 3 shows its output. A Simple search query for a word such कर (do) searches for the lemma as well as the words which have कर (do) as the lemma, so कर (do), किया (did), करने (to do), करते ([they] will do), etc. are all retrieved. Figure 3 shows the first 20 results out of the retrieved ~5 million results.



Figure 2: Simple concordance query



Figure 3: The resulting concordance lines

## The Frequency functions

The Sketch Engine interface provides easy access to tools for visualizing different aspects of the word frequency (see Figure 4). The *Frequency Node forms* function on the left hand menu in Figure 4 shows which of the returned forms are most frequent.

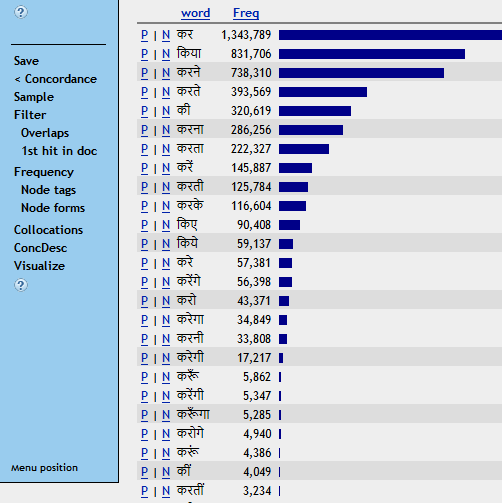


Figure 4: Frequency of word forms of कर (do)

Thus we have immediately discovered that the commonest forms of the lemma कर (do) are कर (do), किया (did) and करने (to do).

The **p/n** links are for positive and negative examples. Clicking on **p** gives a concordance for the word form, while clicking on **n** gives the whole concordance *except* for the word form.

## The Word List function

The Word List function allows the user to make frequency lists of many types (words, lemmas, tags). Figure 5 shows the most frequent words in the corpus. In addition to most frequent words, keywords of any target corpus can be extracted. This is done by comparing frequent words from the target corpus with the frequent words from a general purpose corpus. Figure 6 displays the keywords of a Hindi Election corpus, where this is the target corpus, and the general purpose corpus is the HindiWaC.

Almost every keyword closely relates to the trend of news articles in the 2014 Indian Parliament elections. Since the Hindi Election Corpus is of small size, the frequency-per-million column contains projected values. These are significantly higher than the same words in the HindiWaC since the Election Corpus is domain specific.

Figures 5 & 6: Frequency list of the whole corpus for Words and Keywords extracted automatically from Hindi Election Corpus by comparing it with Hindi Web Corpus

## The Word Sketch and Collocation Concordance functions

The Word Sketch function is invaluable for finding collocations. The word sketches of the word लोग (people) for three dependency relations are shown in Figure 7.

The dependency relations that we use are based on the Paninian framework (Rafiya et al., IJCNLP-2008). Three of the most common dependency relations given by this model are as follows:

* k1: agent and/or doer
* k2: object and/or theme
* k3: instrument

These relations are syntactico-semantic in nature, and differ slightly from the equivalent thematic roles mentioned above. More about how we get the word sketches shown in Figure 7 is explained in Section 3.

In figure 7, the three dependency relations shown are:

* nmod\_adj: noun-modifier\_adjective
* k1\_inv: doer\_inverse
* nmod: noun-modifier

The word sketch function assigns weights to each of the collocates and also to the dependency relations.

Clicking on the number after the collocate gives a concordance of the combination (Figure 8).

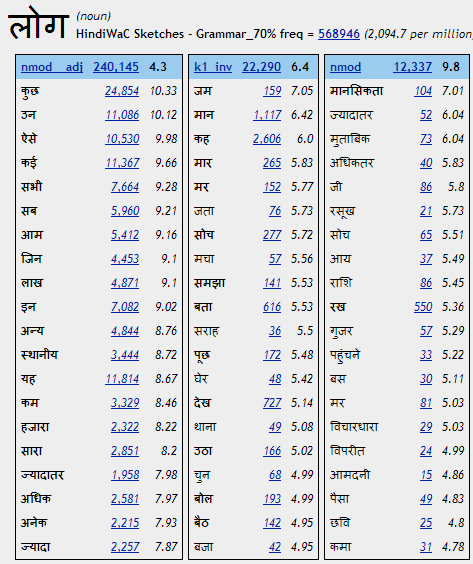
 

Figure 7: Word Sketch results for लोग (people) (left, as also on page 1)   
and Figure 8: Concordance lines for लोग (people) in combination with its gramrel “nmod” (right)

## The Bilingual Word Sketch function

A new function has been added recently to the Word Sketch, which is the Bilingual Word Sketch. This allows the user to see word sketches for two words side by side in different languages. Figure 9 shows a comparison between लाल (red) and *red*. Interestingly, the usage of word red in Hindi and English are very diverse. The only common noun which is modified by *red* in both languages is गुलाब (*rose).*

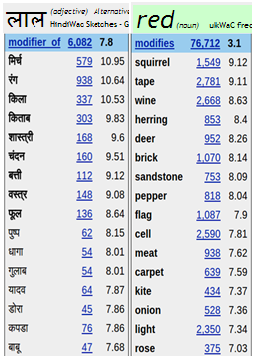


Figure 9: Adjective results of a bilingual word sketch for Hindi लाल (red) and English red

## Distributional Thesaurus and Sketch Diff

The Sketch Engine also offers a distributional thesaurus, where, for the input word, the words 'sharing' most collocates are presented. Figure 10 shows the top entries in similarity to कर (do). The top result is हो (be). Clicking on it takes us to a 'sketch diff', a report that shows the similarities and differences between the two words in Figure 11.

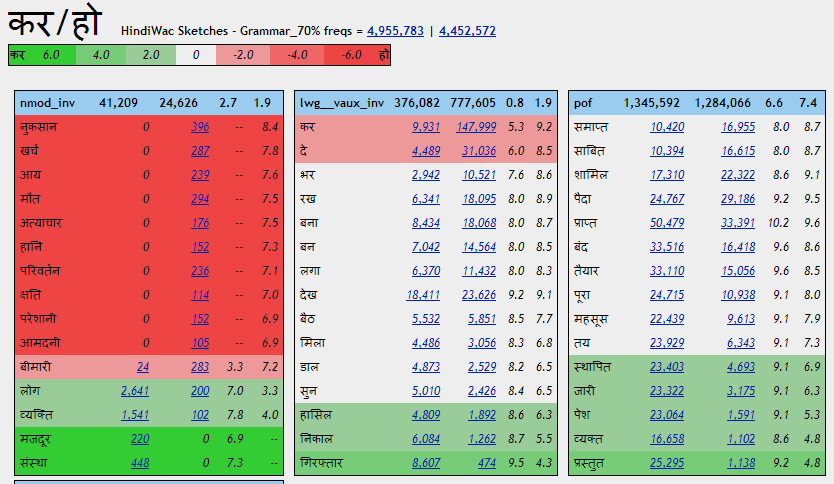
 

Figure 10: Thesaurus search showing entries similar to कर (do) (left)

Figure 11: Sketch Diff comparing collocates of कर (do) and हो (be) (right)

The red results occur most frequently with हो (be), the green ones with कर (do). The ones on white occur equally with both.

## Building and processing HindiWaC and loading it into the Sketch Engine

HindiWaC was built using the Corpus Factory procedure (Kilgarriff et al 2010). A first tranche was built in 2009, with the crawling process repeated and more data added in 2011, and again in 2014. Corpus Factory method can be briefly described as follows: several thousands of target language search queries are generated from Wikipedia, and are submitted to Microsoft Bing search engine. The corresponding hit pages for each query are downloaded. The pages are filtered using a language model. Boilerplate text is removed using body text extraction, deduplication to create clean corpus. We use jusText and Onion tools (Pomik ́alek, 2011) for body-text extraction and deduplication tools respectively.

The text is then tokenized, lemmatized and POS-tagged using the tools downloaded from <http://sivareddy.in/downloads> (Reddy and Sharoff, 2011). The tokenizer found here is installed in the Sketch Engine.

## Sketch Grammar for Hindi:

A sketch grammar is a grammar for the language, based on regular expressions over part-of-speech tags. It underlies the word sketches and is written in the Corpus Query Language (CQL). Sketch grammar is designed particularly to identify head-and-dependent pairs of words (e.g., खा [eat] and राम [Ram]) in specified grammatical relations (here, k1 [doer]), in order that the dependent can be entered into the head's word sketch and vice versa.

Sketch Grammars are popular with lexicographic and corpus linguistics community, and are used to identify collocations of a word with a given grammatical relation (Kilgarriff and Rundell, 2002). We use Sketch Grammar to identify words in syntactic relations in a given sentence. For example, a grammar rule for the relation "**k1**" (**doer**) is *2:"NN" "PSP\:ने" "JJ"? 1:"VM"*, which specifies that if a noun is followed by a PSP and an optional adjective and followed by a verb, then the noun is the kartha/subject of the verb. The head and child are identified by 1: and 2: respectively. Each rule may often be matched by more than one relation creating ambiguity. Yet they tend to capture the most common behaviour in the language.

Writing a full-fledged sketch grammar with high coverage is a difficult task even for language experts, as it would involve capturing all the idiosyncrasies of a language. Even though such hand-written rules tend to be more accurate, the recall of the rules is very low. In this paper, the grammar we use is a collection of POS tag sequences (rules) which are automatically extracted from an annotated Treebank, Hindi Dependency Treebank (HDT-v0.5), which was released for the Coling2012 shared task on dependency parsing (Cite 3). This treebank uses IIIT tagset described in (Bharati et al., 2006). This method gives us a lot of rules based on the syntactic ordering of the words. Though these rules do not have all the lexical cues of a language, the hope is that, when applied on a large-scale web corpus, the correct matches (sketches) of the rules automatically become statistically more frequent, and hence more significant.

From the above mentioned treebank (HDT) we extract dependency grammar rules (i.e. sketch grammar) automatically for each dependency relation, based on the POS tags appearing in between the dependent words (inclusive). For example, from the sentence,

राम(Ram) ने(erg.) कमरे(room) में(inside) आम(mango) खाया(eat), Ram ate [a] mango in [the] room, we extract rules of the type: **(k1[doer], k2 [object], k7[location])**

**k1** - 2:[tag="NNP"] [tag="PSP\:ने"] [tag="NN"] [tag="PSP\:में"] 1:[tag="VM"]

**k2** - 2:[tag="NN"] 1:[tag="VM"]

**k7** - 2:[tag="NN"] [tag="PSP\:में"] [tag="NN"] 1:[tag="VM"]

In the above example, relation names are in bold with one of the corresponding rules for each of them.

We do include a few lexical features associated with the POS tags **PSP** (post-position) and **CC** (conjunction) in order to disambiguate between different dependency relations. For example, in both the relations **k1 (doer)** and **k2 (object)** we have the rules (1) and (2) given below respectively:

बच्चे (NN) ने (PSP) आम (NN) खाया (VM)

(child) (erg.) (mango) (eat.pst) (the child ate a mango)

2:[tag="NN"] [tag="PSP\:ने"] [tag=”NN”] 1:[tag="VM"] ------- (1)

कागज़ (NN) को (PSP) बाहर (NN) फेंका (VM)

(paper) (acc.) (outside) (throw.pst) ([Someone] threw the paper outside)

2:[tag="NN"] [tag="PSP\:को"] [tag=”NN”] 1:[tag="VM"] ------- (2)

In (1), the ergative marker indicates that the noun (NN) is the *doer* of the verb (VM). In (2), the accusative marker indicates that the noun (NN) is the *object* of the verb (VM). Also, (2) is not a complete sentence – the *doer* has not been mentioned, and only the part of the sentence that the rule is applied to is shown.

If in the rules, the PSP tags didn’t contain the lexical features, both the rules would have been the same, and hence both the rules have been applied on both the sentences, making the word sketches erroneous.

By lexicalising the **PSP** POS tag, the rule(s) formed are now less ambiguous, and more accurate.

After extracting all the dependency rules, we apply each rule on the annotated Treebank (HDT), and compute its precision. For example, if the rule “**k7 (location)** - 2:[tag="NN"] [tag="PSP\:में"] [tag="NN"] 1:[tag="VM"]”

is applied on the HDT, we get all the [2:NN, 1:VM] pairs where the rule holds, say **N** pairs. Out of these **N** pairs, if **M** of them are seen correctly with k7 (place) relation in the training data, then the precision of the rule is.

Conditions that need to be satisfied for a rule to be included in the sketch grammar:

* The rule must have a precision of at least 70%. A higher cut-off gives us rules with better precision, but less recall, and it is the reverse for lower cut-off limits.
* The frequency of the tag sequence (N) must be greater than 4, to ensure some amount of statistical significance of the rules.
* The context size – the maximum allowed length of the tag sequence (rule), is set to 7 to limit the number of rules generated.

1. **Error Analysis**

The word sketches may not always be accurate due to the ambiguous nature of rules, and pos tagging errors. For example, when a rule such as

1:[tag=”NN”] [tag=”PSP: ने”] 2:[tag=”VM”]

is applied on sentences which have nouns ending with the honorific जी in the data, जी is likely to be a collocate of the words. This is an error due to POS tagger that we use to tag the HindiWaC. The tagger tags the honorific जी as NN and not RP. These word sketches can be improved further by improving the POS-tagger, or the grammar, for example by involving local word group information. The other related errors are due to UNK pos tag which is generally assigned to words that are unknown to the tagger.

As the length of a rule increases so does its sparsity, i.e. the number of sentences on which the rule can be applied since all the POS tags in the rule have to occur in that particular order. Hindi being a free word order language the possibility of all the POS tags occurring in that order is even less.

1. **Conclusion and Future Work**

Corpora are playing an increasing role in all kinds of language research as well as in language learning, lexicography, translation, literary studies and discourse analysis. The requirements are, firstly, a suitable corpus, and secondly, a corpus query tool. We have presented HindiWaC, a large corpus of Hindi, which has been prepared for use in the Sketch Engine. We have described how we used Hindi Dependency Treebank to develop the grammar underlying the word sketches. And we have shown the core features of the Sketch Engine, as applied to Hindi.

Our current grammar is prone to data sparseness as the length of rule increases. A future direction of this work could be on building compact grammars using regular expressions. Additionally, one could also explore the usefulness of morphological features in grammar rules to make them semantically accurate.

**References**

Kilgarriff, A., Rychly, P., Smrz, P., & Tugwell, D. (2004). The Sketch Engine. *Proc. EURALEX*, pp 105-116. Lorient, France.

Kilgarriff, A., Reddy, S., Pomikálek, J., & Avinesh, P. V. S. (2010, May). A Corpus Factory for Many Languages. In *LREC*.

A Bharathi, P Mannem and D S Misra “Hindi Parsing Shared Task” Coling workshop 2012

Kilgarriff, Adam, and David Tugwell. "Sketching words." *Lexicography and Natural Language Processing: A Festschrift in Honour of B. TS Atkins* (2002): 125-137.

Reddy, Siva, and Serge Sharoff. "Cross language POS taggers (and other tools) for Indian languages: An experiment with Kannada using Telugu resources."*Cross Lingual Information Access* (2011): 11.