IMPROVEMENT IN MACHINE TRANSLATION FROM ENGLISH TO PUNJABI BY IDENTIFYING THE MORPHEME BOUNDARIES

[1]Simran Kaur Jolly, [2]Dr.Rashmi Agrawal
[1]Assistant Professor/Research Scholar, [2]Professor
FCA,
[1]MRIIRS, FARIDABAD, INDIA
FCA,
[2]MRIIRS, FARIDABAD, INDIA

Abstract: We discuss about language distinct and an unsupervised approach for Morphological Analysis. The algorithm is based on probability that uses the distance, frequency and length of the strings. In future it would solve problems of large corpora and agglutinative languages as well. We perform the algorithm on English data as well as Punjabi data and get the results as follows, as the number of morphemes recognized are more in English than in Punjabi language due to the fluctuations in random behavior showing smaller segmentations in small data sizes. There will be always a room for change ahead as the language grows.

***Index Terms* – Morphology, Gatekeeper Grammar, Gensim**

I. INTRODUCTION

In this section we will give an insight of inflectional morphology(M.Kasthuri,2015)- a module of morphology that tells how words can be broken down into different parts, so that there is systematic variation of words in order to show their gender, tense, number and status. e.g.: go can have many forms like going, goes, went and gone.

So, a Morphology framework has three main modules: a lexeme, operation, gatekeeper grammar.

A lexeme stands for associated forms of the lexicon.

Operation stands for a function that maps one word form to another word form.

Gatekeeper grammar that basically assesses the correctness of the affixes (base form).

Each of the lexicon has morphological operations associated with it that are homogeneous i.e. it must be applied in one form to produce another form.

Base cell	→	Inflected form: operation
1 Sg	→	2 Sg
3 Pl	→	1 Pl

For instance we can see that the word 'sleep' in English is translated to 'nind' in Punjabi.
Using Punjabi morphology we can generate many inflectional forms of the word.
Word = Root = ਨੀਂਦ
(Word Class: Noun; Gender: Feminine; Number: Singular; Case: Direct ;)
(Word Class: Noun; Gender: Feminine; Number: Singular; Case: Oblique ;)
And these are the word forms generated from morphs:
Word = ਨੀਂਦ (to sleep)
(Word Class: Noun; Gender: Feminine; Number: Singular; Case: Direct ;)
(Word Class: Noun; Gender: Feminine; Number: Singular; Case: Oblique ;)

Word = ਨੀਂਦੋਂ
(Word Class: Noun; Gender: Feminine; Number: Singular; Case: Ablative ;)

Word = ਨੀਂਦੇ
(Word Class: Noun; Gender: Feminine; Number: Singular; Case: Locative ;)

Word = ਨੀਂਦਾਂ
(Word Class: Noun; Gender: Feminine; Number: Plural; Case: Direct ;)
(Word Class: Noun; Gender: Feminine; Number: Plural; Case: Oblique ;)
So each morphological operation can have many different changes 1Sg->1Pl. The most important component of the sub lexicon is the gatekeeper grammar that assigns a positive or a negative weight to lexicon.

Derivation & Inflection: It is a type of morphological sub-process that derives a new word from base word. E.g. parking the root word is park. We can attach more than one affix in the base word. E.g. Mis-happen-ings.

Compounding: It is a process of joining two nouns or adjectives or prepositional forms.

E.g.:
Earphones: noun +noun
Blackboard: adjective +noun
Daydream: noun +noun
So the problem we will be discussing in morphology will be nominalization of compounds that takes input as noun and gives output as noun.
Two Step Morphology: It is a process (Vishal Goyal, 2008) which has two levels in morphology: V-N, N-V. Consider a word increase, if it is a verb it means "become or make greater in size, amount, intensity or degree" and if it is a noun it means "an instance of growing or making greater".

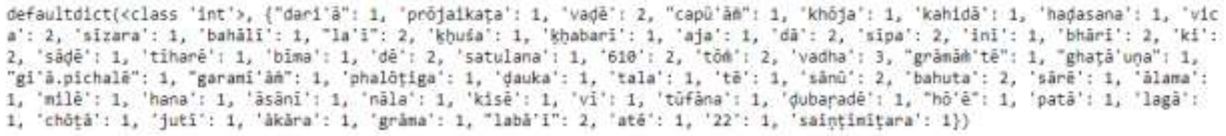
Level 1
V-N (A+ ity)
Level 2
N-V (V+ er)

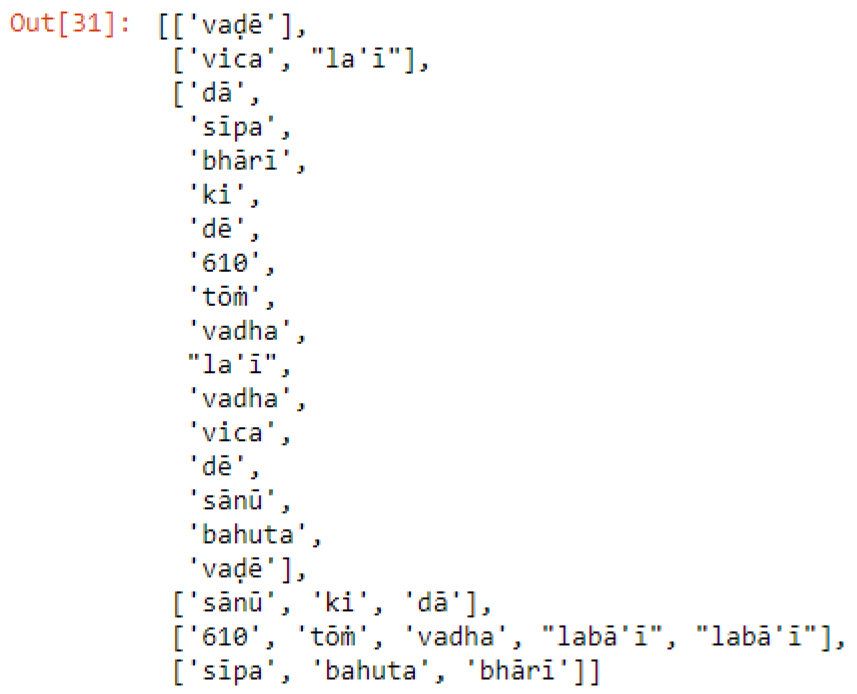
So we treat fast, manage in one class and beautiful, stupid in another class.
If beautiful is a noun then it enters second state and the agentive rule doesn't apply here on the word. If beautiful is a noun then it enters second state and the agentive rule doesn't apply here on the word and it enters second stage of morphology. Fast is taken as noun and enters level 2 and becomes faster. Hence internal structure of the word is important in morphology which helps in language modelling and semantic representations. In order to show internal representations i.e. word embedding's in our language we use genism (python library) which helps ahead in analogy and reasoning. So after some fixed epochs we get vectors of words.

II. MORPHOLOGICAL TRANSFORMATION USING GENSIM

The implemented approach used for inflectional morphology is deployed by genism module [W1] in python library that extracts the corpora and perform tokenization on it using the function genism. So in order to reduce the complexities in our inflected language we use morph syntactic transformations (2 level morphology) to remove lexical ambiguities as shown below:
Raw corpus (transliterated to Punjabi):
"The River Project calls the discovery of the large oyster ",
"Good news for oyster restoration in the Hudson",
"Today's oyster was so heavy that it maxed out our triple beam balance clocking in at over 610 grams",
"We found a lot of large oysters last summer at the bottom of the floating dock",
"Easily dwarfs any of the oysters we found it's the size of a small shoe",
"Weighing in at over 610 grams and measuring 22 cm in length",
"The oyster was too heavy"

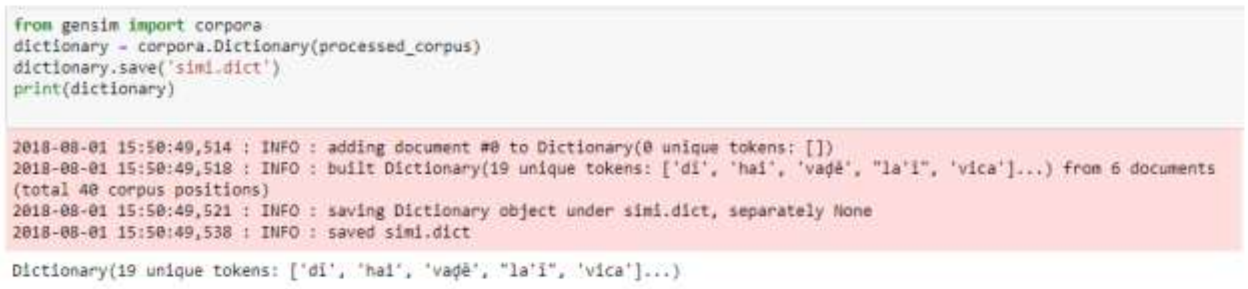
"Dari'ā prōjaikaṭa vaḍē capū'āṁ dī khōja kahidā hai",
"haḍasana vica sīzara dī bahālī la'ī ḵẖuśa ḵẖabarī",
"aja dā sīpa inī bhārī sī ki iha sāḍē tīharē bīma dē satulana nū 610 tōṁ vadha grāmāṁ'tē ghaṭā'uṇa la'ī vadha gi'ā."
"Pichalē garamī'āṁ vica phalōṭiga ḍauka dē tala tē sānū bahuta sārē vaḍē ālama milē hana",
"āsānī nāla kisē vī tūfāna nū ḍubaṛadē hō'ē sānū patā lagā ki iha ika chōṭā jutī dā ākāra hai",
"610 grāma tōṁ vadha dī labā'ī atē 22 saiṇṭīmīṭara labā'ī",
"sīpa bahuta bhārī sī"
We are having a corpus of 9 strings above so we need to create a dictionary from the corpus or a lexicon that can be used ahead.


```
defaultdict(<class 'int'>, {"dari'ā": 1, 'prōjaikaṭa': 1, 'vaḍē': 2, "capū'āṁ": 1, 'khōja': 1, 'kahidā': 1, 'haḍasana': 1, 'vic
a': 2, 'sīzara': 1, 'bahālī': 1, "la'ī": 2, 'ḵẖuśa': 1, 'ḵẖabarī': 1, 'aja': 1, 'dā': 2, 'sīpa': 2, 'inī': 1, 'bhārī': 2, 'ki':
2, 'sāḍē': 1, 'tīharē': 1, 'bīma': 1, 'dē': 2, 'satulana': 1, '610': 2, 'tōṁ': 2, 'vadha': 3, "grāmāṁ'tē": 1, "ghaṭā'uṇa": 1,
"gi'ā.pichalē": 1, "garamī'āṁ": 1, 'phalōṭiga': 1, 'ḍauka': 1, 'tala': 1, 'tē': 1, 'sānū': 2, 'bahuta': 2, 'sārē': 1, 'ālama':
1, 'milē': 1, 'hana': 1, 'āsānī': 1, 'nāla': 1, 'kisē': 1, 'vī': 1, 'tūfāna': 1, 'ḍubaṛadē': 1, "hō'ē": 1, 'patā': 1, 'lagā':
1, 'chōṭā': 1, 'jutī': 1, 'ākāra': 1, 'grāma': 1, "labā'ī": 2, 'atē': 1, '22': 1, 'saiṇṭīmīṭara': 1})
```
Figure 1: Tokenization



```
Out[31]: [['vaḍē'],
          ['vica', "la'ī"],
          ['dā',
           'sīpa',
           'bhārī',
           'ki',
           'dē',
           '610',
           'tōṁ',
           'vadha',
           "la'ī",
           'vadha',
           'vica',
           'dē',
           'sānū',
           'bahuta',
           'vaḍē'],
          ['sānū', 'ki', 'dā'],
          ['610', 'tōṁ', 'vadha', "labā'ī", "labā'ī"],
          ['sīpa', 'bahuta', 'bhārī']]
```

Figure 2: Processed Corpus

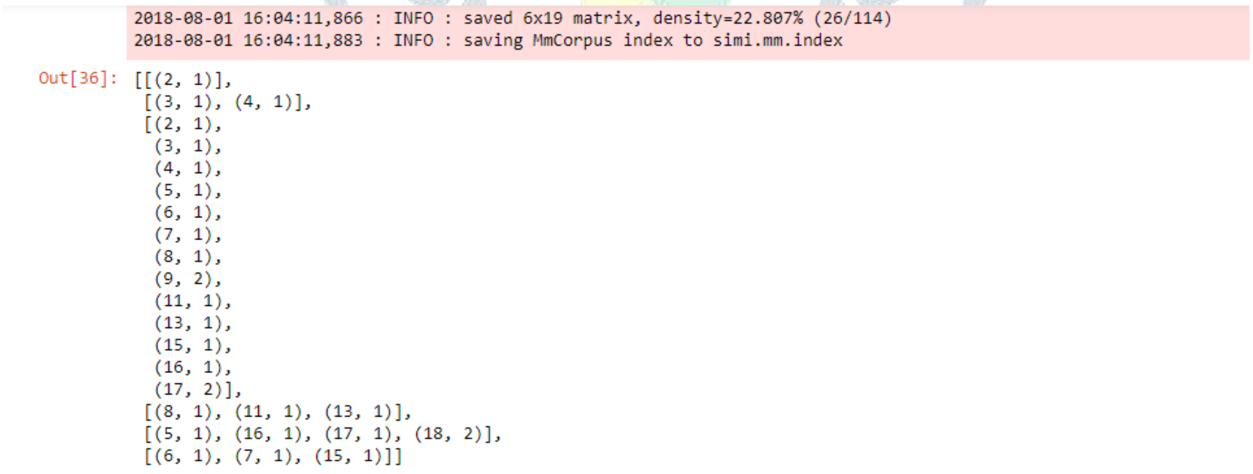


```
from gensim import corpora
dictionary = corpora.Dictionary(processed_corpus)
dictionary.save('simi.dict')
print(dictionary)

2018-08-01 15:50:49,514 : INFO : adding document #0 to Dictionary(0 unique tokens: [])
2018-08-01 15:50:49,518 : INFO : built Dictionary(19 unique tokens: ['dī', 'hai', 'vaḍē', "la'ī", 'vica']...) from 6 documents
(total 40 corpus positions)
2018-08-01 15:50:49,521 : INFO : saving Dictionary object under simi.dict, separately None
2018-08-01 15:50:49,538 : INFO : saved simi.dict

Dictionary(19 unique tokens: ['dī', 'hai', 'vaḍē', "la'ī", 'vica']...)
```

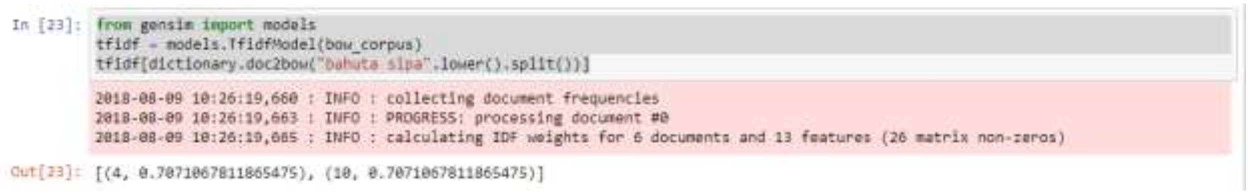
Figure 3: Dictionary



```
          2018-08-01 16:04:11,866 : INFO : saved 6x19 matrix, density=22.807% (26/114)
          2018-08-01 16:04:11,883 : INFO : saving MmCorpus index to simi.mm.index

Out[36]: [[(2, 1)],
          [(3, 1), (4, 1)],
          [(2, 1),
           (3, 1),
           (4, 1),
           (5, 1),
           (6, 1),
           (7, 1),
           (8, 1),
           (9, 2),
           (11, 1),
           (13, 1),
           (15, 1),
           (16, 1),
           (17, 2)],
          [(8, 1), (11, 1), (13, 1)],
          [(5, 1), (16, 1), (17, 1), (18, 2)],
          [(6, 1), (7, 1), (15, 1)]]
```

Figure 4: Matrix Generation



```
In [23]: from gensim import models
         tfidf = models.TfidfModel(bow_corpus)
         tfidf[dictionary.doc2bow("bahuta sīpa".lower().split())]

         2018-08-09 10:26:19,660 : INFO : collecting document frequencies
         2018-08-09 10:26:19,663 : INFO : PROGRESS: processing document #0
         2018-08-09 10:26:19,665 : INFO : calculating IDF weights for 6 documents and 13 features (26 matrix non-zeros)

Out[23]: [(4, 0.7071067811865475), (10, 0.7071067811865475)]
```

Figure 5: TF IDF Calculator

In the above given illustrations we split the given number of sentences into its constituent tokens and then place in one semantic space that are closely related to each other. So the challenges faces while morphology is polysemy and synonymy. And goal is to find a model that idle for terms and documents and morphology is the core component of natural language processing while translation that bring words to its root forms bringing wide coverage and precision in this paper we are dealing with corpora that is in form of sentences and showing all the word segmentation using genism that can be further used for topic modelling as well.

III. PROPOSED MODEL

The model that we will be following is a Lemmatization Model with a dictionary or a lexicon, so we will be using a lemmatizer to improve the corpus stream method given by (Goldsmith, 2001) and view the contents inside our corpus which will be using a two factor matrix model which is best for lemmatization in source language.

Algorithm:

```
1. Let D be the documents having set of possible words (Z) and the class vocabulary D = (original word, $T_{n1,} T_{n2...} T_{nj)}$
2. For each $W_{ip,i.e.(1..n),p(1..n)}$
3. $W_{ip\,=}$ Translated word in dictionary
      Else
   $W_{ip}$
4. Here W-D(words are mapped to dictionary)
5. Mapping depends on ambiguity
   $D_{i\,=\,m}$ $(w_{i,}w_c^{d})$,c<=i<=d $w_c^d$ is the word sequence from c to d and $w_i$ is the single word. (1)
```

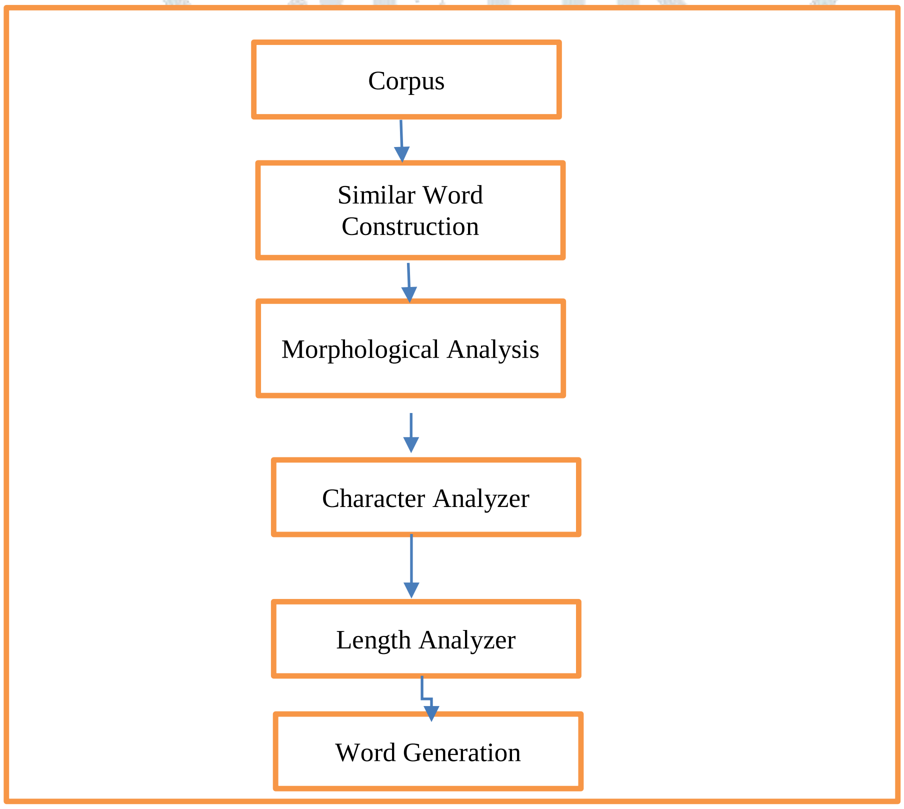


Figure 6: Morphological Analysis

1. Corpus: let W denote the input string from w1…wk, wi is character in position I, 1<i<k, So corpus is electronic data present online that is used for statistical analysis and text alignment.
2. Similar Word Construction: then the next step is constructing similar words dictionary using the function default dict in gensim. Read the stop words and check its match with w, if k exists in the stop word remove it otherwise go to next step. If length of the word<3 then it's a stem word else go to next step. Let Ɣ denote filtered strings S from β whose first three character = k. then,
   $B_{i\,=}$ $C_{I1,}$ CI2, CI3, CIJ
   $B_{i\,=}$ $X_{1,}$ X2, CI4, CIJ
   Morphological Analysis: then if there are matching words in both the strings we remove the stem words from second string.

3. Character Analyzer and Length Analyzer: Then compute ED and LCS between α, w and Ɣ where $Ɣ_{p}$ = ($Ɣ_{1}$, $Ɣ_{2...}$ $Ɣ_{n}$) where Ɣ<= cardinality of B. here ED stands for levenshtein distance or the editing distance between two strings that has to be minimum. Length of $Ɣ_{p}$ = mp, then LCS is the lowest common subsequence where ED p+ LCSp = K(if 2 words of are present then minimum length is chosen and if more than one value is present then common characters between two are chosen through dynamic programming).
4. The last step is calculating the frequencies assigned to each lemma in the corpus
   :$(\sum_{i=1}^{nu} fui)! \quad / \prod_{i=1}^{nu} fui)^{-1} = (\frac{N!}{\prod_{i=1}^{nu} fui})^{-1}$ sum of the product of the factorial of frequencies. (2)

IV. EVALUATION OF MORPHS

To test the quality of morphs generated is correct or not depends on length of the morphs and semantic structure as well (low ambiguity to other morph) e.g. hoped is stemmed to: hope + d, hop + ed

1. Divide the corpus into words using appropriate tokenization algorithm.
2. Divide the segmented corpus into 2 parts of equal size: Training Dictionary and Test Dictionary.
3. Align the training segmented words with their equivalent morpheme labels: girls have morpheme label PL GEN (plural female).
4. Now calculate probability for morph/morpheme over the whole training data
   :$\frac{1}{N}\sum_{I=1}^{N} Pi(\frac{morpheme}{morph})$ (3)

The English data that we take consists of Weather report corpus from which stop words are removed using genism library in python that uses defaultdict function to find the frequently occurring keywords. Morph syntactic analysis of words is done using gensim that extracts the keywords with their frequencies (no of time they appear in a corpus). So we can see as the dataset increases the keywords also increases. So morpheme length, frequency distribution are important for our test data.

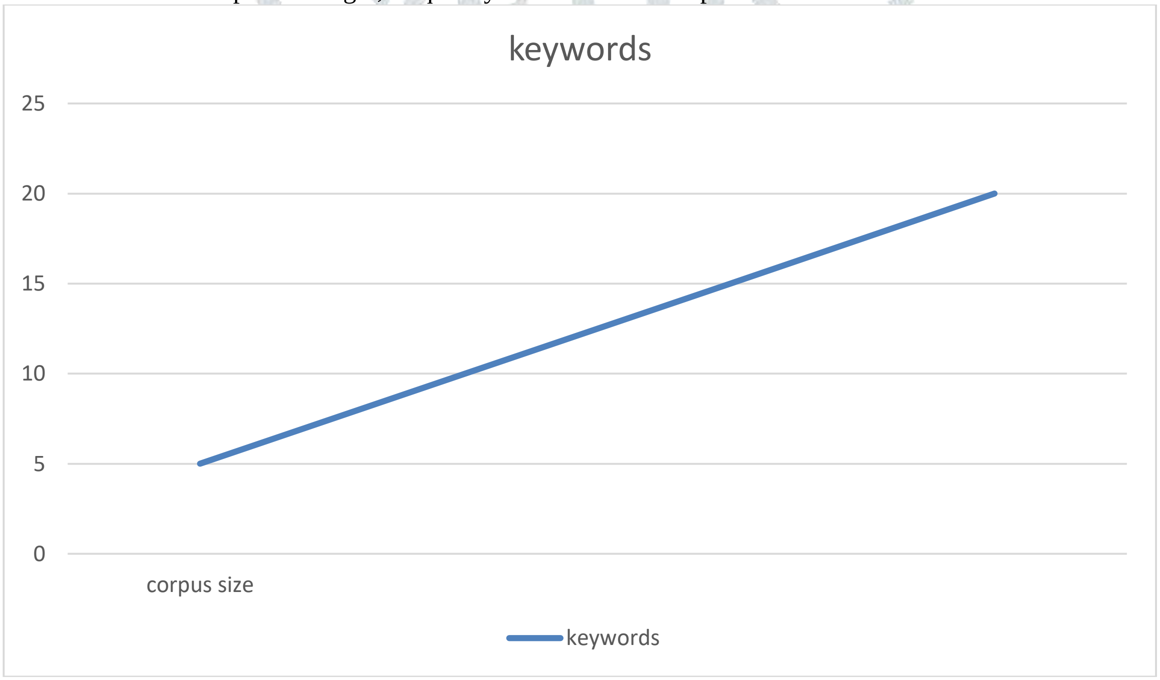


Figure 7: corpus Analysis

References

[1] Hindi Morphological Analyser and Generator(Vishal Goyal , Gurpreet Singh Lehal, lecturer Dept of CS, Punjabi university Patiala, professor and hod) (2008) ieee (first international conference on emerging trends in engineering and technology).

[2] M.Kasthuri, Dr.S.Britto Ramesh Kumar, "A Framework for Language Independent Stemmer using Dynamic Programming", International Journal of applied Engineering Research, issn0973-4562 vol 10, pp 390000-39004, Number.18, 2015.

[3] Vishal Gupta, "Hindi Rule Based Stemmer for Nouns", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 1, January 2014. . (Journal or magazine citation).

[4] Mayfield James and McNamee Paul, "Single N-gram stemming", Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, 415-416, 2003. (Conference Proceedings)

[5] Goldsmith, J. (2001). Unsupervised learning of Morphology of a natural language. Computational Linguistics, 27(2):153-198.

[6] M Zafar and A Masood, "Interactive English to Urdu machine translation using example-based approach," in International Journal of Computer Science and Engineering, vol. 1, no. 3, pp. 275-282, 2009

[7] Sitender and S Bawa, "Survey of Indian machine translation systems," in International Journal of Computer Science and Technology, vol. 3, Issue 1, Jan–Mar 2012.

[8] F J Och, "Proceedings of the 2007 joint conference on empirical methods in natural language learning," in Prague, Association for Computational Linguistics, pp. 858-867, June 2007, [Online] Available: http://www.translate.google.com, http://translate.google.com/about/intl/en_ALL/.

[9] Staab, S., and Studer, R. Handbook on Ontologies, International handbooks on Information Systems,2nd ed., London, New York: Springer Dordrecht Heidelberg, 2009.

[10] Web References

a. [W1] https://en.wikipedia.org/wiki/Word_order
b. [W2] https://en.wikipedia.org/wiki/Subject-object-verb
c. [W3] https://en.wikipedia.org/wiki/Machine_translation
d. [W4] https://en.wikipedia.org/wiki/Punjabi_language4