Facilitating Multi-Lingual Sense Annotation: Human Mediated Lemmatizer

Pushpak Bhattacharyya IIT Bombay pushpakbh@gmail.com
Ankit Bahuguna IIT Bombay / TU Munich ankitbahuguna@outlook.com
Lavita Talukdar IIT Bombay lavita.talukdar@gmail.com
Bornali Phukan IIT Bombay bornali31phukan@gmail.com

Abstract

Sense marked corpora is essential for supervised word sense disambiguation (WSD). The marked sense ids come from wordnets. However, words in corpora appear in morphed forms, while wordnets store lemma. This situation calls for accurate lemmatizers. *The lemma is the gateway to the wordnet*. However, the problem is that for many languages, lemmatizers do not exist, and this problem is not easy to solve, since rule based lemmatizers take time and require highly skilled linguists.Satistical stemmers on the other hand do not return legitimate lemma.

We present here a novel scheme for creating accurate lemmatizers quickly. These lemmatizers are *human mediated*. The key idea is that a trie is created out of the vocabulary of the language. The lemmatizing process consists in navigating the trie, trying to find a match between the input word and an entry in the trie. At the point of first mismatch, the yield of the subtree rooted at the partially matched node is output as the list of possible lemma. If the correct lemma does not appear in the list- as noted by a human lexicographer- backtracking is initiated. This can output more possibilities. A ranking function filters and orders the output list of lemma.

We have evaluated the performance of this human mediated lemmatizer for eighteen Indian Languages and five European languages. We have compared accuracy values against well known lemmatizers/stemmers like Morpha, Morfessor and Snowball stemmers, and observed superior performance in all cases. Our work shows a way of speedily creating human assisted accurate lemmatizers, thereby removing a difficult roadblock in many NLP tasks, e.g., sense annotation.

1 Introduction

Supervised WSD- the ruling paradigm for high accuracy sense determination- requires sense marked corpus in large quantity. Sense annotation is a difficult job, requiring linguistic expertise, knowledge of the topic and domain, and most importantly a fine sense of word meanings.

Most often the sense annotation task is accomplished by using a *Sense Marker Tool*, like the one described in Chatterjee *et. al.* (2010). This particular tool, equipped with an easy to use GUI facilitates the task of manually marking each word with the correct sense of the word, as available in the wordnet of the language. The tool has the wordnet sense repository resident in its memory. It displays the senses of word for the human annotator to choose from. The mentioned tool is extensively used by a number of language groups in India to produce high quality sense marked corpus. Figure 1 shows the Sense Marker Tool, where a user is marking the sense of the word “banks” in English.

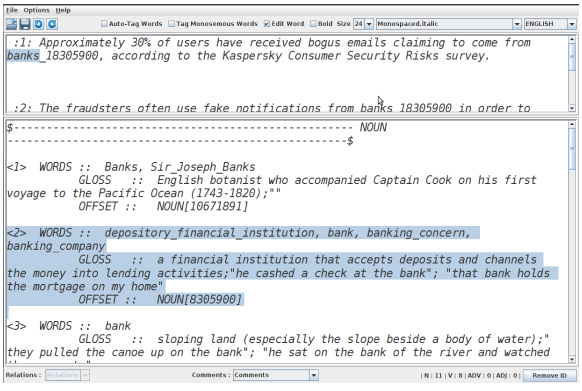

Figure 1: Sense Marking Tool - manually marking correct sense for English word “banks”

Now it is obvious that if the wordnet is to be accessed for the senses to be displayed, the lemma of the words must be available. The lemma is the gateway to the wordnet. Lemmatization is an im-

portant activity in many NLP applications. We stress at this point that our focus of attention is lemmatization and not stemming. As a process, stemming aims to reduce a set of words into a cannonical form which may or may not be a *dictionary word* of the language. Lemmatization, on the other hand, always produces a legal root word of the language. To give an example, a stemmer can give rise to "ladi" from "ladies". But a lemmatizer will have to produce "lady". And if the senses of 'ladies' has to be found, the lemma 'lady' is required.

There are three basic approaches to lemmatization, *viz.*, affix removal (rule based systems), statistical (supervised/unsupervised) and hybrid (rule based + statistical). Developing a rule based system is an uphill task, requiring a number of language experts and enormous amount of time. Purely statistical systems fail exploit linguistic features, and produce non-dictionary lexemes. A hybrid system utilizes both the above mentioned approaches.

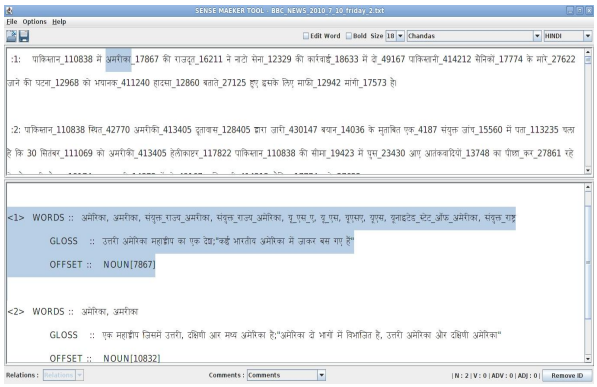

Figure 2: Sense Marking Tool - Input Language: Hindi

In this paper, we discuss an alternative approach to lemmatization which is quick, takes user help and is exact. The key idea is that a trie is created out of the vocabulary of the language. The lemmatizing process consists in navigating the trie, trying to find a match between the input word and an entry in the trie. *At the point of first mismatch- i.e., maximum prefix matching, the yield of the subtree rooted at the partially matched node is output as the list of possible lemma.* If the correct lemma does not appear in the list- as noted by a human lexicographer- a backtracking is initiated. This can output more possibilities, since the yield of a node at a higher level of the trie is output. A ranking function filters and orders the output list of lemma.

Our ultimate goal is to integrate the human mediated lemmatizer with the *Sense Marking Tool* (Figure 2).Currently, the tool supports the following 9 languages: English, Hindi, Marathi, Tamil, Telugu, Kannada, Malayalam, Bengali and Punjabi.

For Example, In Assamese:

Inflected Word Form: ঘৰলৈ (ghoroloi ~ to home)

Root Word: ঘৰ (ghor ~ home)

We give an example to give a feel for how our lemmatiser works. When a linguist tries to mark the correct sense of the inflected Assamese word as shown in Figure 3 ('gharalai' meaning 'in the house'), in the current scenario no output is displayed, but with our lemmatizer integrated, it will show the possible lemma of that word.

(ঘৰ ঘৰৰ ঘৰঘৰ ঘৰ-ঘৰ ঘৰচকা ঘৰহীন ঘৰুৱা)

Here, the correct lemma is shown at the top of the list, and the lemma can pull out the senses from the wordnet for the linguist to tag with.

The remainder of this paper is organized as follows. We describe related work and background in section 2. Section 3 explains the core of our human mediated lemmatiser. Implementation details are in Section 4. Experiments and results are discussed in Section 5. Comparison with existing stemmers/lemmatizers are in Section 6. Section 7 concludes the paper and points to future directions.

2 Related Work and Background

Lovins described the first stemmer (Lovins, 1968), which was developed specifically for IR/NLP applications. His approach consisted of the use of a manually developed list of 294 suffixes, each linked to 29 conditions, plus 35 transformation rules. For an input word, the suffix with an appropriate condition is checked and removed. Porter developed the Porter stemming algorithm (Porter, 1980) which became the most widely used stemming algorithm for English language. Later, he developed stemmers that covered Romance (French, Italian, Portuguese and Spanish), Germanic (Dutch and German) and Scandinavian languages (Danish, Norwegian and Swedish), as

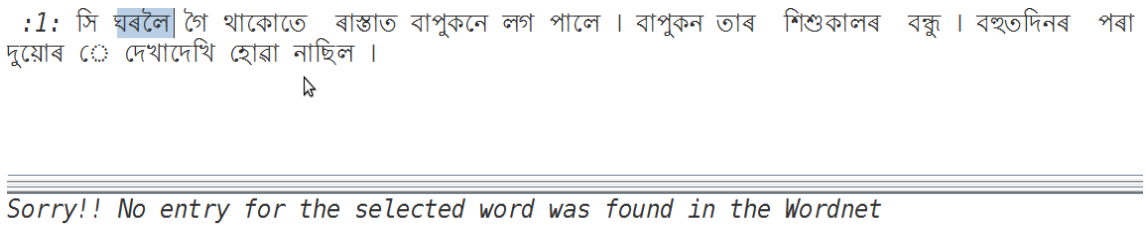

Figure 3: Tool shows no output for inflected Assamese word "ghoroloi" (highlighted) meaning *to home*.

well as Finnish and Russian (Porter, 2006). These stemmers were described in a very high level language known as Snowball[1].

A number of statistical approaches have been developed for stemming. Notable works include: Goldsmith's unsupervised algorithm for learning morphology of a language based on the Minimum Description Length (MDL) framework (Goldsmith, 2001; 2006). Creutz uses probabilistic maximum a posteriori (MAP) formulation for unsupervised morpheme segmentation (Creutz, 2005; 2007).

A few approaches are based on the application of Hidden Markov models (Melucci et al., 2003). In this technique, each word is considered to be composed of two parts "prefix" and "suffix". Here, HMM states are divided into two disjoint sets: *Prefix state* which generates the first part of the word and *Suffix state* which generates the last part of the word, if the word has a suffix. After a complete and trained HMM is available for a language, stemming can be performed directly.

Plisson proposed the most accepted rule based approach for lemmatization (Plisson et al., 2008). It is based on the word endings, where suffixes are removed or added to get the normalized word form. In another work, a method to automatically develop lemmatization rules to generate the lemma from the full form of a word was discussed (Jongejan et al., 2009). The lemmatizer was trained on Danish, Dutch, English, German, Greek, Icelandic, Norwegian, Polish, Slovene and Swedish full form-lemma pairs respectively.

Kimmo (Karttunen et al., 1983) is a two level morphological analyser containing a large set of morphophonemic rules. The work started in 1980 and the first implementation n LIST was available 3 years later.

Tarek El-Shishtawy proposed the first non-statistical Arabic lemmatizer algorithm (El-Shishtawy et al., 2012). He makes use of different Arabic language knowledge resources to generate accurate lemma form and its relevant features that support IR purposes and a maximum accuracy of 94.8% is reported.

OMA is a Turkish Morphological Analyzer which gives all possible analyses for a given word with the help of finite state technology. Two-level morphology is used to build the lexicon for a language (Ozturkmenoglu et al., 2012).

Grzegorz Chrupala (Chrupala et al., 2006) presented a simple data-driven context-sensitive approach to lemmatizating word forms. Shortest Edit Script (SES) between reversed input and output strings is computed to achieve this task. An SES describes the transformations that have to be applied to the input string (word form) in order to convert it to the output string (lemma).

As for lemmatizers for Indian languages, the earliest work by Ramanathan and Rao (2003) used manually sorted suffix list and performed longest match stripping for building a Hindi stemmer. Majumdar et. al (2007) developed YASS: Yet Another Suffix Stripper. Here conflation was viewed as a clustering problem with a-priory unknown number of clusters. They suggested several distance measures rewarding long matching prefixes and penalizing early mismatches. In a recent work related to Affix Stacking languages like Marathi, (Dabre et al., 2012) Finite State Machine (FSM) is used to develop a Marathi morphological Analyzer. In another approach, A Hindi Lemmatizer is proposed, where suffixes are stripped according to various rules and necessary addition of character(s) is done to get a proper root form (Paul et al., 2013). GRALE is a graph based lemmatizer for Bengali comprising two steps (Loponen et al., 2013). In the first, step it extracts the set of frequent suffixes and in the second step, a human manually identifies the case suffixes. Words are often considered as node and edge from node **u** to **v** exist if only **v** can be generated from **u** by addi-

[1]See `http://snowball.tartarus.org/`

tion of a suffix.

Unlike the above mentioned rule based and statistical approaches, our human mediated lemmatizer uses the properties of a "trie" data structure which allows retrieving *possible* lemma of a given inflected word, with human help at critical steps.

3 Our Approach to lemmatization

The scope of our work is suffix based morphology. We do not consider infix and prefix morphology. We first setup the data structure 'Trie' (Cormen et al., 2001) using the words in the wordnet of the language. Next, we match, byte by byte, the input word form and wordnet words. The output is all wordnet words which have the maximum prefix match with the input word. This is the "direct" variant of our lemmatizer.

The second or "backtrack" variant prints the results 'n' levels previous to the maximum matched prefix obtained in the 'direct' variant. The value of 'n' is usr controlled. A ranking function then decides the final output displayed to the user.

In Figure 5, a sample trie diagram is shown consisting of Hindi words given at Figure 4. The words are stored starting from a node subsequent to the root node, in a character by character unicode byte order.

List of words
1. कमरबन्द (kamarband ~ drawstring)
2. कमरा (kamara ~ room)
3. कमरी (kamari ~ small blanket)
4. कमल (kamal ~ Lotus)
5. लड़ (lad ~ fibril)
6. लड़कपन (ladakpan ~ childhood)
7. लड़का (ladka ~ boy)
8. लड़की (ladki ~ girl)
9. लड़ना (ladna ~ fight)

Figure 4: List of sample words.

At each level, we insert the characters of the word in an alphabetical order pertaining to unicode standard for different languages from left to right.

We illustrate the search in the trie with the example of the inflected word "लड़कियाँ" (ladkiyan, *i.e.,* girls). Our lemmatizer gives the following output:

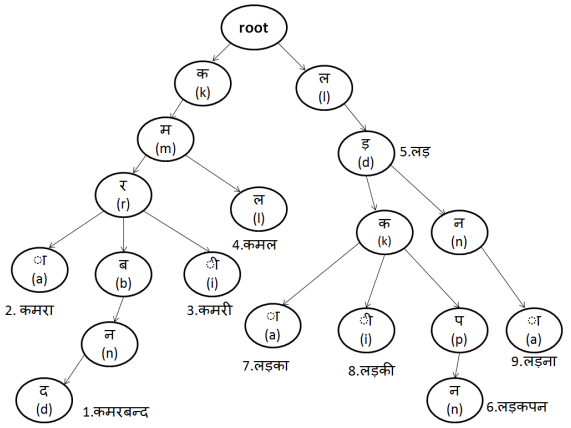

Figure 5: A simple trie storing Hindi words (kamarband, kamara, kamari, kamal, lad, ladakpan, ladka, ladki and ladna)

(ल लड़ लड़का लड़की लड़कपन लड़कोरी लड़कौरी)

From this result set, a trained lexicographer can easily pick the root word as "लड़की" (ladki, *i.e.*, girl). It is the user controlled backtracking which is novel and very useful in our lemmatizer. We elaborate on backtracking below.

3.1 Backtracking

We explain backtracking using the sample words in Figure 6 and the trie as shown in Figure 7.

List of words
1.असणे (asane ~ hold)
2.असली (asali ~ real)
3. आज (aaj ~ today)

Figure 6: List of sample words : Backtracking.

We take the example of असलेले (*aslele*) which is an inflected form of the Marathi word असणे (*asane*). Here, when we call the first iterative procedure without backtracking, the word असली (*asali*) is given as output. Although, being a valid wordnet word, it is not the correct root form of the word असलेले. Hence, we perform a backtrack from असल (*asal*) to अस (*asa*) thereby getting असणे (*asane*) as one of the outputs.

An example involving two levels of backtracking is that of a Marathi word कापसाला (*kApsAlA*), which is an inflected form of the root word कापुस, (*kApusa*) *i.e.*, cotton). Here are the results from our lemmatizer:

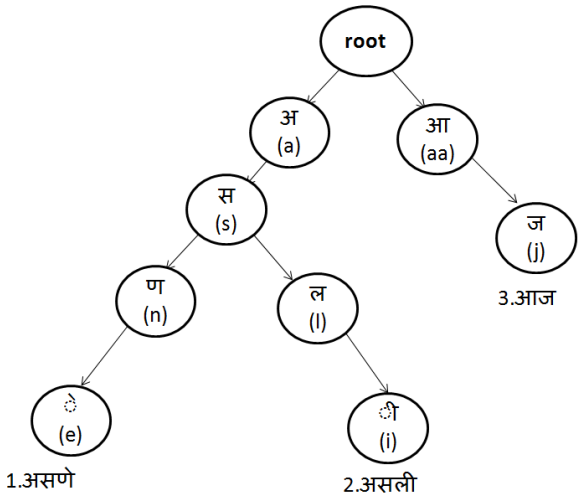

Figure 7: Search in Trie - Backtracking in case of Marathi word "असलेले" (*aslele*)

Basic:
(क का काप कापसाचा)

1 - level backtrack:
(क का काप कापसी कापसाचा)

2 - level backtrack:
(क का काप कापड कापणे
कापरा कापरे कापसी कापूर **कापूस**
कापडेपट कापणावळ कापराचा
कापलेला कापसाचा कापाकापी)

Thus, we can find the root word कापुस at 10th position in the result list of two level backtrack.

3.2 Ranking Lemmatizer Results

Our lemmatizer by default prints out all the possible root forms given a queried word. We have employed a heuristic to filter the results and hence minimize the size of the result set:

1. Only those output matches are accepted whose length is smaller than or equal to the length of the queried word. This is based on an assumption that the root word length shall not be greater than the length of its equivalent inflected word form (we agree that this is not universally true; hence the word 'heuristic' for the pruning strategy).

2. The filtered results are sorted on the basis of the length (in an ascending order) which in most cases displays the root word earlier in a given set of words.

For example, in case of the Marathi word "असलेले" (*aslele*) the ranking function receives a number of words as input, after a first level backtrack is performed (as shown in Figure 8). The function then applies the ranking heuristic based on length and then sorts them in their ascending order. Thus, the final lemmatizer output is generated with the root word असणे (*asane*) in first position.

Input to Rank Function:
असंख्य||असंगती||असंतुलित||असंतुलित_बल||असंतुष्ट||असंतुष्टता||असंतोषी||असंदिग्ध||असंबद्ध||असंबद्धता||असंभव||असंभवनीय||असंभाव्य||असंभाव्यता||असंमती||असंयत||असंयम||असंयमित||असंयमी||असंशायात्मक||असणे||असतेपण||असत्य||असत्यता||असनसियान||असन्माननीय||असफल_होणे||असभ्य||असभ्यपणा||असभ्यपणे

LEMMATIZER OUTPUT:

(**असणे** असंभव असंयत असंयम असत्य असभ्य असंख्य असंगती असंमती असंयमी असतेपण असंतोषी असंबद्ध असंयमित असत्यता)

Figure 8: Ranking of results - For Marathi word "असलेले" (*aslele*)

4 Implementation

We have developed an on-line interface (Figure 9) and a downloadable Java based executable jar which can be used to test our implementation. The interface allows input from 18 different Indian languages (Hindi, Assamese, Bengali, Bodo, Gujarati, Kannada, Kashmiri, Konkani, Malayalam, Manipuri, Marathi, Nepali, Sanskrit, Tamil, Telugu, Punjabi, Urdu and Odiya) and 5 European languages (Italian, Danish, Hungarian, French and English) and they are linked to their respective lexical databases. For easy typing, a virtual keyboard is also provided. The first iteration of stemming is performed by clicking "Find" and backtracking is performed by clicking "Backtrack". The backtrack utility allows us to backtrack upto 8 levels and the level of backtrack is displayed in a field. This was implemented, since there are several inflected words in our test data which required a backtrack of more than one level to get the root word. The interface also has a facility to *upload* a text document related to a specific language. We can then download the results in an output file containing possible stems of all the words present in the input document. The human annotator can thus choose the correct lemma from the list of possible stems associated with each word.

5 Experiments and Results

We performed several experiments to evaluate the performance of the lemmatizer. The basis of our

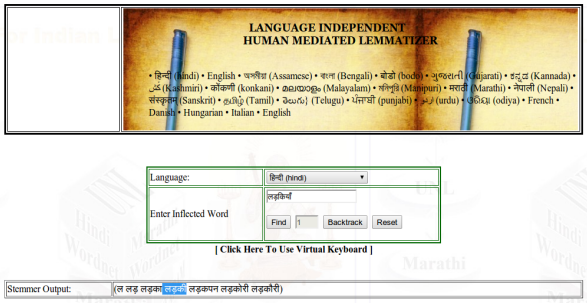

Figure 9: Online Interface - Language Independent Human Mediated Lemmatizer

evaluation was that for every inflected input word, a set of output root forms will be generated by the lemmatizer. **Even if one of the result in the top 10 from this set matches the root in the gold standard, then we consider our result to be correct.** Following this approach the accuracy of inflected nouns undergoing stemming is very high. Although, due to readjustment in verbs, for the first iteration without backtracking, our lemmatizer gives a fairly low accuracy. This can be further improved through backtracking, when we traversed the trie, one level up at a time. The first iteration of stemming gives result among the best five outputs and the backtracking approach gives among the best ten outputs. Interestingly, for inflected words in Italian our lemmatizer performs better when we include the results of one level backtrack as compared to a non-backtrack variant. The results for various languages are shown in Table 1 based on the lemmatizer's default variant, *i.e.*, without using backtrack feature.

5.1 Preparation of Gold Data

We prepared the gold data to perform evaluation for Hindi and Marathi languages, by using the domain specific sense marked corpus[2] which contains inflected words along with their root word forms. This corpus was created by trained lexicographers. Similarly, we had a sense marked corpus for Bengali, Assamese, Punjabi and Konkani.

For Dravidian languages like Malayalam and Kannada and for European languages like French and Italian, we had to perfrom manual evaluation as the sense marked data was unavailable. We did this, with a list of inflected words provided by native speakers and found remarkable precision in result.

[2]See `http://www.cfilt.iitb.ac.in/wsd/annotated_corpus/`

Language	Corpus Type	Total Words	Precision Value
Hindi	Health	8626	89.268
Hindi	Tourism	16076	87.953
Bengali	Health	11627	93.249
Bengali	Tourism	11305	93.199
Assamese	General	3740	96.791
Punjabi	Tourism	6130	98.347
Marathi	Health	11510	87.655
Marathi	Tourism	13176	85.620
Konkani	Tourism	12388	75.721
Malayalam[*]	General	135	100.00
Kannada[*]	General	39	84.615
Italian[*]	General	42	88.095 #
French[*]	General	50	94.00

Table 1: Precision calculation for output produced by our lemmatizer based on the first variant, *i.e.*, without using backtrack feature. [*] denotes manual evaluation and # denotes one level backtracking.

5.2 Analysis

Errors are due to the following:

1. Agglutination in Marathi and Dravidian languages.

   Marathi and Dravidian languages like Kannada and Malayalam show the process of agglutination[3]. It is a process where a complex word is formed by combining morphemes each of which have a distinct grammatical or semantic meaning. Such words do not produce correct results in the first go, and we need to back track the trie upto a certain level to get the correct root form.

2. Suppletion.

   The term suppletion[4] implies that a gap in the paradigm was filled by a form "supplied" by a different paradigm. For example, the root word "go", has an irregular past tense form "went". For these irregular words the output of the lemmatizer is incorrect, as the root words are stored in their regular forms in the trie.

We have reported the precision scores (Table 1) for all the languages (except Italian) on the "direct" variant of our lemmatizer, *i.e.*, without using

[3]See `http://en.wikipedia.org/wiki/Agglutination`

[4]See `https://en.wikipedia.org/wiki/Suppletion`

Corpus Name	Human Mediated Lemmatizer	Morpha	Snowball	Morfessor
English_General	89.20	90.17	53.125	79.16
Hindi_General	90.83	NA	NA	26.14
Marathi_General	96.51	NA	NA	37.26

Table 2: Comparative Evaluation (precision values) of Human Mediated Lemmatizer [Without using Backtracking] against other classic stemming systems like Morpha, Snowball and Morfessor

backtrack feature. It is clear from our examples in Marathi viz. "असलेले" (*aslele*) and the scores reported in Italian that *precision improves when backtracking is used*.

6 Comparative evaluation with existing lemmatizers/stemmer

We compared performance of our system against three most commonly used lemmatizers/stemmers, *viz.* Morpha (Guido et al., 2001), Snowball [5] and Morfessor (Creutz, 2005; 2007). The results are shown in Table 2. Our lemmatizer works better than Morfessor for Hindi (up by 64%) and Marathi (up by 59%). For English, our lemmatizer outperforms Snowball by almost 36% and Morfessor by almost 10%. Although, as an exception, Morpha lemmatizer works better by about 1% in this case.

Snowball and Morpha lemmatizers are not available for Indian Languages and thus the results are marked with 'NA'. Morfessor being statistical in nature does not capture all the linguistic and morphological phenomenons associated with Indian languages and Snowball and Morpha are strictly rule based in nature and do not use any linguistic resource to validate the output. We have, of course, compared our lemmatizer with in-house rule based Hindi and Marathi morph analyzers. The Marathi Morphological Analyzer (Dabre et al., 2012) has an accuracy of 72.18%, with a usability of 94.33%, where as the in-house Hindi Morphological Analyzer[6] accuracy is close to 100% with average usability around 96.5% (Table 3). Here, usability is the percentage of total number of words analyzed out of total words in corpus.

However, these morph analysers have taken years to build with many false starts and false hits and misses. Compared to this, our human mediated lemmatizer was constructed in a few weeks' time. Once it was realized that we need lemmatizers for accessing senses of words, the system was built in no time. It is the preparation of gold data, generation of accuracy values and comparison with existing systems that took time.

	Nouns	Verbs
Total words in test Corpus	14475	13160
Correctly analyzed words	13453	13044
Unidentified words	1022	116

Table 3: Hindi Morphological Analyzer - Results

7 Conclusion and future work

We gave an approach for developing light weight and quick-to-create human mediated lemmatizer. We applied the lemmatizer to 18 different Indian languages and 5 European languages. Our approach uses the longest prefix match functionality of a trie data structure. Without using any manually created rule list or statistical measure, we were able to find lemma of the input word within a ranked list of 5-15 outputs. The human annotator can thus choose from a small set of results and proceed with sense marking, thereby greatly helping the overall task of Machine Translation and Word Sense Disambiguation. We also confirm the fact that the combination of man and machine can identify the root to near 100 percent accuracy.

In future, we want to further prune of output list, making the ranking much more intelligent. Integration of the human mediated lemmatizer to all languages' sense marking task needs to be completed. Also we want to expand our work to include languages from different linguistic families.

References

Arindam Chatterjee, Salil Rajeev Joshi, Mitesh M. Khapra and Pushpak Bhattacharyya 2010. *Introduction to Tools for IndoWordnet and Word Sense Disambiguation*, The 3rd IndoWordnet Workshop, Eighth International Conference on Natural Lan-

[5] See http://snowball.tartarus.org/index.php

[6] See http://www.cfilt.iitb.ac.in/~ankitb/ma/

guage Processing (ICON 2010), IIT Kharagpur, India.

Grzegorz Chrupala 2006. *Simple data-driven context-sensitive lemmatization* , Chrupaa, Grzegorz (2006) Simple data-driven context-sensitive lemmatization. In: SEPLN 2006, 13-15 September 2006, Zaragoza, Spain.

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest and Charles E. Leiserson 2001. *Introduction to Algorithms*, 2nd Edition, ISBN:0070131511, McGraw-Hill Higher Education.

Mathis Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0.*, Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

Mathis Creutz and Krista Lagus. 2007. *Unsupervised models for morpheme segmentation and morphology learning*, Association for Computing Machinery Transactions on Speech and Language Processing, 4(1):1-34

Raj Dabre, Archana Amberkar and Pushpak Bhattacharyya 2012. *Morphology Analyser for Affix Stacking Languages: a case study in Marathi*, COLING 2012, Mumbai, India, 10-14 Dec, 2012.

Tarek El-Shishtawy and Fatma El-Ghannam 2012. *An Accurate Arabic Root-Based Lemmatizer for Information Retrieval Purposes*, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 3, January 2012 ISSN (Online): 1694-0814.

John A. Goldsmith 2001. *Unsupervised Learning of the morphology of a Natural Language*, Computational Linguistics, 27(2): 153-198

John A. Goldsmith 2006. *An algorithm for the unsupervised Learning of morphology*, Natural Language Engineering, 12(4): 353-371

Bart Jongejan and Hercules Dalianis 2009. *Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike*, Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 145 - 153, Suntec, Singapore, 2-7 August 2009

Lauri Karttunen 1983. *KIMMO: A General Morphological Processor* , Texas Linguistic Forum, 22 (1983), 163-186.

Aki Loponen, Jiaul H. Paik and Kalervo Jarvelin 2013. *UTA Stemming and Lemmatization Experiments in the FIRE Bengali Ad Hoc Task* , Multilingual Information Access in South Asian Languages Lecture Notes in Computer Science Volume 7536, 2013, pp 258-268

J.B. Lovins 1968. *Development of a stemming algorithm*, Mechanical Translations and Computational Linguistics Vol.11 Nos 1 and 2, pp. 22-31.

Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. *YASS: Yet another suffix stripper*, Association for Computing Machinery Transactions on Information Systems, 25(4):18-38.

Prasenjit Majumder, Mandar Mitra and Kalyankumar Datta 2007. *Statistical vs Rule-Based Stemming for Monolingual French Retrieval*, Evaluation of Multilingual and Multi-modal Information Retrieval, Lecture Notes in Computer Science vol. 4370, ISBN 978-3-540-74998-1, Springer, Berlin, Heidelberg.

James Mayfield and Paul McNamee 2003. *Single N-gram Stemming*, SIGIR '03, Toronto, Canada.

Massimo Melucci and Nicola Orio 2003. *A novel method of Stemmer Generation Based on Hidden Markov Models*, CIKM '03, New Orleans, Louisiana, USA.

Guido Minnen,John Carroll and Darren Pearce. 2001. *Applied morphological processing of English*, Natural Language Engineering, 7(3). 207-223.

Okan Ozturkmenoglu and Adil Alpkocak 2012. *Comparison of different lemmatization approaches for information retrieval on Turkish text collection* , Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on.

Plisson Joël, Lavrac Nada, Mladenic Dunja 2008. *A rule based approach to word lemmatization*, Proceedings of 7th International Multi-conference Information Society, IS 2004, Institute Jozef Stefen, Ljubljana, pp.83-86, 2008

Snigdha Paul, Nisheeth Joshi and Iti Mathur 2013. *Development of a Hindi Lemmatizer*, CoRR, DBLP:journals/corr/abs/1305.6211 2013

M.F. Porter 1980. *An algorithm for suffix stripping*, Program; 14, 130-137

M.F. Porter 2006. *Stemming algorithms for various European languages*, Available at [URL] `http://snowball.tartarus.org/texts/stemmersoverview.html` As seen on May 16, 2013

Ananthakrishnan Ramanathan, and Durgesh D Rao 2003. *A Lightweight Stemmer for Hindi.* , Workshop on Computational Linguistics for South-Asian Languages, EACL