Parallel Corpora and Linguistic Resource Creation for Statistical Machine Translation

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science & Engineering

by

Yeka Jayendra Rakesh
200802044
jayendra.rakesh@gmail.com

International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2017

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Parallel Corpora and Linguistic Resource Creation for Statistical Machine Translation" by Yeka Jayendra Rakesh, has been carried out under my supervision and is not submitted elsewhere for a degree.

23/12/17
Date

Dipti Misra
Adviser: Prof. Dipti Misra Sharma

23 Dec, 2017
Date

[illegible]
Co-Adviser: Dr. Radhika Mamidi

To Amma, Nanna, Sravani and Rinky

Acknowledgments

The research conducted as part of Benchmarking of English↔Hindi parallel corpora project has been supported by the funds provided as part of EILMT consortium project by Department of Information Technology (DIT), India. I would like to thank TDIL for distributing the resources from the projects. I thank the students of Banasthali Vidyapeth for their work on validating the EILMT corpora and helping in the extraction of the NCERT corpus. The state-of-art resources utilised for enriching the corpora were provided by various consortia project members.

I thank *Sarvodaya Mandal* and *Gandhi Research Foundation* whole heartedly for granting us access to Gandhi's autobiography and also to various other resources available on their website. A special mention of gratitude to the NCERT authorities and the Ministry of Law and Justice for making the bilingual resources available for free download and free of copyright permissions, of course with fair usage policy. I would also like to mention the contributions of the *Scientific and Technical Hindi* group[1] for their *Parivarthak* open-source scripts, utilising which the font converters have been constructed.

The work done as part of the treebanking project has been supported by the funds provided by NSF. I thank DEITY for providing the financial support for the construction of modules as part of the ILMT consortium. I thank Bharat Ram Ambati for this initial contribution and dedication towards the establishment of pipeline framework. I kindly thank Anil Kumar Singh for permitting us to use the Sanchay toolkit, without which the annotation of the treebank could not have taken place.

I sincerely thank Prasanth Kolachina for his guidance and help during times of self conflict and in making me passionate towards research. I thank my teachers Dr. Sriram Venkatapathy, Prashanth Mannem sir, Dr. Samar Hussain for the knowledge and guidance they have provided me. I thank Dr. Vineet Chaithanya for his guidance and support throughout this work. I thank Dr. Radhika Mamidi for her encouragement and help with the resources.

I thank my friends and fellow researchers Deepak, Puneeth, Vishnu and Nagarjuna for making my research life a memorable experience. I thank my friend Aravindh for always setting an unreachable benchmark in the fields of hard work, self-discipline and thirst for knowledge, which stood like a beacon of light in my moments that lacked motivation. I also thank Mahesh providing the said lack of motivation and helping me cool off the steam. I sincerely thank my guru and friend Sri Kalyan for his out-of-mortal-realm lectures that helped me grow into a better human.

[1] `https://sites.google.com/site/technicalhindi/home/converters`

I sincerely thank my manager and colleague Rama Kiran - *Ramki* for his support and readily granting me sabbatical(twice) from work to finish my thesis. I thank Dr. Radhika Mamidi for her help with the resources and for her valuable feedback in drafting this thesis. I thank my professor Dr. Dipti Misra for dealing with my laziness, patiently and correcting my skewed perspective on many occasions, may it be personal or professional.

I would also like to thank my acquaintances, my seniors, my juniors and many more people whose influence had a lasting effect in shaping my character, my principles and my goals.

I thank the special people in my life - my parents and my sister for their immense love for me, their patience in dealing my absence at home, for the support they provided in my times of need and for always being by my side. Finally, I thank my special lady for being by my side like my family, believing in me when I lost my faith, cautioning me when I had too much of it and for poking, nudging, pushing and shoving me to finish this book.

Note - Across the thesis, ’we’ is used to acknowledge the support and the group effort of the individuals involved, however, to indicate personal opinions and messages, I have used ’I’.

Abstract

Statistical Machine Translation (SMT) is an automated approach to Natural-Language translation, that utilizes statistical models generated from the bilingual parallel corpora. The system is thus fundamentally reliant on the bilingual parallel corpora, to translate text from one language to another. Additionally, in the past several decades, research in SMT has been focused on augmenting syntactic and linguistic information to the statistical models to generate better translations. Thus, construction of the aforementioned linguistic resources like bilingual parallel corpora and corpora enriched with syntactic and linguistic information plays a key role in building better SMT systems.

SMT work for the English-Hindi languages are limited and have been resource deprived, especially for the Hindi language. The prime motivation for the research in this thesis is to bridge the gaps in the level of resources available for Hindi language and the English-Hindi language pair, from an SMT perspective. The initial experiments at building a Hindi-English SMT system with Part-Of-Speech (POS) augmented models yielded unfavourable results, with translations that are no better than the plain statistical models. Root cause analysis revealed fundamental flaws like lack of better bilingual corpora and better syntactic models, which hinders the improvements to SMT system, thus setting the goal of this thesis towards addressing these issues.

Firstly, we start by presenting several parallel corpora for English↔Hindi and talk about their natures and domains. We also discuss briefly a few previous attempts in MT for translation from English to Hindi. The lack of uniformly annotated data makes it difficult to compare these attempts and precisely analyse their strengths and shortcomings. With this in mind, a standard pipeline to provide uniform linguistic annotations to these resources using the state-of-art NLP technologies is presented. The benchmark scores for various English→Hindi SMT systems are constructed using the aforementioned parallel corpora. A total of 1,37,578 sentences were cleaned and MT systems were benchmarked as part of this work.

Subsequently which, this thesis talks about several digital text sources from different domains in English and Hindi languages which were processed with the intention of extracting parallel sentences. We faced several difficulties while extracting text content from different documents, especially the non-linear presentation of text in varying page layouts. We focus on this problem of linearizing the text from these digital sources by proposing a clustering based algorithm to eliminate the noise. The effort undertaken helped us process the raw sources available in different formats, different fonts and varying page layouts to produce 79,422 English and 68,584 Hindi sentences with comparable meanings across both languages.

Lastly, a semi-automated framework is presented to enable speeding up dependency annotation task for corpora. We talk about the slow game of pass-the-parcel in between both the automation and human sides which finally results in raw corpora being transformed into dependency annotated sentences. The methods of automation chosen to overcome the laborious and time-consuming process of corpora annotation are discussed. Along side which the errors and multiple analyses that result through the task of annotation and ways to recover are also discussed in detail. A total of 20,968 dependency annotated Hindi sentences and 7,120 Urdu sentences are created using the mentioned framework.

Overall, the thesis tries to present three different efforts undertaken to quench the thirst for resources in the field of English$\leftrightarrow$Hindi SMT.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One cannot ennunciate enough on the importance of parallel corpora when discussing about Statistical Machine Translation (SMT). The view of SMT itself is heavily blinded without the looking glasses of parallel resources, to an extent that size of parallel corpora becomes a crucial measure of performance of the SMT system. (Kolachina et al., 2012) describes in detail how enhancing the quantity of parallel corpora improves the translational quality of SMT models. For language domain of English-Hindi where the parallel resources are limited in comparision to the likes of English and other European languages ((Koehn, 2005)), undertaking the task of increasing the data is essential. The advent of factored models in SMT (Koehn and Hoang, 2007), shows us a glimpse of how SMT systems can benifitted from syntactic features, but also adds another dimension to the need of resources in SMT. The need for syntactically rich corpora is more pressing now, when we can improve quality of translations by augmenting the translational models with morphological and contextual features. With that perspective, this thesis finds its place rightfully in the context of English-Hindi SMT by improving upon both existing resources and contributing several resources which are essential in building better SMT systems.

1.1 Motivation

Initial experiments with Factored models on Hierarchical Statistical Machine translation set the tone and nature of the thesis. These experiments have greatly influenced the work pursued and further directs the flow this thesis adopts. The experiment aimed to improve translation quality by rescoring the k-best candidate translations of each test sentence, with the aid of POS tagged factored models. This, however, resulted in a BLEU score of *0.3012* for the factored hierarchical model, when compared with baseline hierarchical SMT system with a BLEU score of *0.3058*. The results point out that the performance of the factored model did not improve over baseline phrase based SMT. One can refer to Chapter 3 for a detailed explanation of the experiments and models used, where we discuss the resources utilized, procedures adopted to inject factored models into Hierachical SMT systems and also reasons of failure. Though the short-sightedness to check for the scale of improvisation seems to be the major

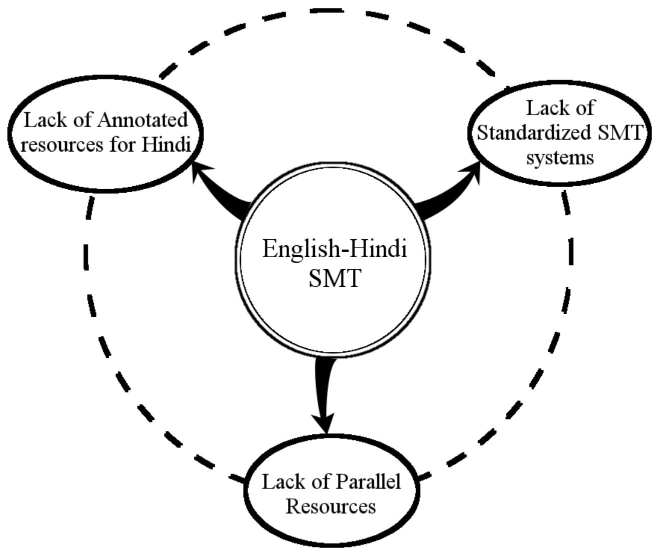

Lack of Annotated resources for Hindi
Lack of Standardized SMT systems
English-Hindi SMT
Lack of Parallel Resources

Figure 1.1: Issues of English-Hindi SMT

short-coming, the experiments made me realise other major issues which undermined improvements to English-Hindi SMT systems, which are

- Lack of standardized MT systems to which experiments can be compared with:
  English-Hindi SMT experiments scattered and lack standardisation to make them comparable with each other. In many cases although results are made public, the resources are not. The resources available are not standardised so as to easily replicate the results.

- Lack of parallel resources:
  European languages boast quite large amounts of parallel corpora (Koehn, 2005) which English-Hindi language pair does not have the privilege of.

- Presence of feature rich parallel corpora is meagre:
  Syntactic Augmented Machine Translation (*SAMT*) grammar models have proved that feature rich corpora can marginally boost the quality of translations. However, the presence of feature rich parallel corpora for English-Hindi is negligible. This lack of resources and tools necessary for feature annotation is significant in Hindi language when compared with English. English boasts a feature rich monolingual corpora and also the tools to build it.

1.2 Contribution

This thesis tries to address the above mentioned deficiencies, by tackling each individual problem, and providing

- Benchmarked SMT systems to aid in comparisions.
- Algorithm to linearize text resources to generate parallel corpora.
- Semi-automated framework for dependency annotation

A paradigm to generate clean and linguistically enriched corpora is propsed using which a total of 1,37,578 parallel sentences belonging to various domains have been extracted. These sentences are cleaned, annotated, standard SMT systems are constructed domain-wise and the results are benchmarked, and the corpora are made available.

While addressing the lack of parallel resources, method to effectively extract linear text sources from varying page layouts was proposed using clustering algorithms. Several potential parallel resources available in both English and Hindi are explored, processed, linearised from multiple page layouts and a total of 79,422 English and 68,584 Hindi unaligned raw sentences have been extracted from them.

A Semi-automated dependency framework was proposed, using which dependency annotated corpora has been created efficiently and cleanly for the languages of Hindi and Urdu. The system churned out a total of 20,968 dependency annotated, morpho-syntactic feature rich sentences in Hindi language and has been extended to Urdu which produced a total of 7,120 sentences.

1.3 Outline

Chapter 2 provides a brief background and related works on the field of Machine Translation. It also provides a brief overview about dependency annotation and its format of representation namely SSF for Indian languages. We also describe few basic Machine Learning algorithms which have been used as part of this thesis.

Chapter 3 describes the initial experiments involved in improvising translations by inducing factored models into Hierarchical SMT system for a Hindi$\rightarrow$English MT system.

Chapter 4 presents the standardized benchmark scores and the performance of different Statistical Machine Translation systems with the corpora created and gathered.

Chapter 5 describes the effort involved in linearisation of text resources from varied page layout structures to extract comparable parallel resources.

In Chapter 6, we describe the Semi-automated framework established for the purpose of creating annotated treebanks, describing the efforts, pitfalls and issues faced during the semi-automated annotation. We also describe the nature of tools utilized and monolingual corpora created.

Chapter 7 concludes the thesis by summarizing the entire work and the possible extensions to this thesis.

Chapter 2

Background work

This section presents the basic background knowledge about important topics and algorithms in the fields of Machine Translation and Machine Learning which aids the reader in better understanding of the work involed as part of this thesis. Firstly we present the basics of machine translation and different machine translation paradigms which helps understand the work in *Chapters 3 & 4*. We, then, proceed to explain few basic clustering algorithms utilized as part of this thesis to linearize raw text sources in *Chapter 5*. Finally, we present SSF format and dependency karaka labels utilized for the task dependency annotation in *Chapter 6*.

2.1 Machine Translation

Machine translation eyes towards building automated intelligent systems which translate a given piece of text automatically from one language to another. Though multiple approaches have been proposed, the following two paradigms stand out more, and have withstood the test of time

- Rule based machine translation $RBMT$
- Statistical machine translation SMT

Rule based machine translation aims at using linguistic information about the source and target languages to generate syntactic rules and semantic analysis of both languages, which aid in translation of a new piece of text. RBMT has a major downfall of being unable to predict translations for text which differs from the syntactic rules defined so far in the translation system. RBMT expects the syntactic structure of the test sentence to be familiar to that of the sentences used during the rule extraction or the training phase. Hence RBMT systems are highly tasking and resource expensive to build.

$$f \rightarrow e \tag{2.1}$$

Figure 2.1: Intended translation

2.1.1 Statistical Machine Translation

Statistical machine translation (SMT) is a machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual text corpora.

Brown et al. (1993) presents the the mathematics behind SMT which is explained in this section in detail. With the aim to translate a sentence in foreign language to a sentence in English, we provide the logic how a SMT system functions

As shown in equation 2.1, given a foreign sentence f, we seek the English sentence e that maximizes $P(e|f)$, the conditional probability that the sentence f translates to e. The maximum of $P(e|f)$ is defined as follows

$$argmax\ P(e|f) \tag{2.2}$$

$$P(e|f) \ = \ \frac{P(e) * P(f|e)}{P(f)} \tag{2.3}$$

$argmax\ P(e_i|f)$ represents the best translation e_i predicted by the SMT system for a foreign language sentence f. Using *Bayes rule* presented in equation 2.3, value of expression 2.4 can be calculated as follows

$$argmax\ P(e_i|f) = argmax\ P(e_i) \ * \ P(f|e_i) \tag{2.4}$$

For the purpose of maximizing the probabilities the constant $P(f)$ is safely ignored.

Using the *Noisy-Channel* model, we try to evaluate the possible translations e_i for a foreign sentence f and their probabilities $P(e_i|f)$. Assume there is a noisy communication channel through which we try to transmit the sentence e through one ends, but it gets corrupted by noise in the channel and ends up as sentence f when it is received in the other end. We wish to reason how the corruption has created sentence f, thereby trying to infer original sentence e from sentence f. Noisy channel model seems to do the opposite of what *equation* 2.1 shows, but the intended direction for the construction of an SMT system remains the same. Noisy channel model is thus used to translate sentence f to sentence e by trying to predict the nature of corruption. Tasked with recovering the most likely e_i, we reason about the following two probabilities based on the equation 2.4

- $P(e)$, probability that one utters sentence e in any independent context.

  This indicates the probability that a sentence is grammatically correct. That is it tries to estimate the probabilty of the words in the sentence essentially appearing in that particular order.

- $P(f|e)$, probability that when given the sentence e, a human translator would translate it to f

  This indicates the probability that words in sentence e translate to words in f. This component does not bother about the order of words appearing in sentence f.

When we observe f and try to come up with the most likely translation e, as per noisy channel model and equation 2.4 every e_i gets the score $P(e_i) * P(f|e_i)$. Here $P(e)$ worries about word order so that $P(f|e)$ doesn't have to. Hence $P(f|e)$ only says whether or not a bag of English words corresponds to a bag of foreign language words.

Now describe the ways we estimate language model probability $P(e)$ and translation model probability $P(f|e)$

2.1.1.0.1 Language model Language model probability $P(e)$ is calculated using n-gram models. n-gram models estimate the probability of a sequence of n words appearing in a given order. Language model relies on the logic that a sentence is assumed to grammatically correct if the components of the sentence are reasonably grammatical and these components coalesce reasonably to form the sentence. The simplest form this assumption would generalize to is the statement - Probability of an entire sentence being grammatically correct is product of probabilities of each pair of words appearing side by side. Thus $P(e)$ for a sentence of length l words is estimated using the equation 2.5.

$$\begin{aligned} P(e) &= P(w_1|start-of-sentence) * P(w_2|w_1) \\ &\quad * P(w_3|w_2) .... \ P(w_l|w_{l-1}) * P(end-of-sentence|w_l) \end{aligned} \tag{2.5}$$

2.1.1.0.2 Translation model Translation model probability $P(f|e)$, is calculated using IBM Model 3 (Brown et al., 1993). Similar to how $P(e)$ is estimated by breaking down into smaller components, translational probability is also estimated by calculating the probability of translation of smaller components. The sentence e is divided into its smallest components, the bag of words e_i which are then used to estimate the cost of translations into foreign words f_i. Cost of translation is also dependent on the cost of alignment of the text. That is the cost of estimating the translation for English to foreign language has the cost of producing a particular number of words, in particular positions and the individual costs of the bag of English words getting converted to foreign words. Equation 2.6 shows how $P(f|e)$ also incorporates alignment costs.

- Foreign words aligned from English words with fertility n
- Foreign words translate from English words with probability t
- English words distort into positions of foreign words with probability d
- Foreign words generated from NULL word with probabilty p

.

Cost of the alignment on the other hand is dependent on the parameters n, t, d, p described as

$$\begin{aligned} P(f|e) &= P(f,a|e) \\ &= P(f_1 f_2 ... f_m, a_1 a_2 ... a_m | e_1 e_2 ... e_l, m) \\ &= \prod n(\phi_i|e_i) * \prod t(f_j|e_{aj}) * \prod d(j|a_j, l, m) * \prod p(f_i|e_{ai}) \end{aligned} \tag{2.6}$$

However, parameter costs can be estimated using alignment probabilities, which in turn can be estimated using the parameter values as shown in equation 2.6. To resolve this conundrum we use Estimation Maximization (EM) to calculate both parameters and word alignments. We initialize the parameters by assigning all probable parameter scenarios equal weightage. Now we compute alignment probabilities for all pair of sentences using which we estimate a new set of revised parameter values. These updated parameter values take into account correlation data in the bilingual corpus and thus are used once again to estimate updated word alignment probabilities. Repeating this procedure ensures that the probabilties of alignment converge and provide the estimates for $P(f|e)$ as required.

Thus using $p(e)$ and $p(f|e)$ we arrive upon $p(e|f)$ for all the translation candidates for the sentence f.

2.2 Hierarchical Phrase based Statistical Machine Translation

Phrase based SMT is able to achieve word re-ordering to a successful level, however, falls back in phrase level re-orderings. (Chiang, 2005) proposes a solution to the problem mentioned through the usage of Hierarchical phrase based rules. Hierarchical rules are based on weighted synchronous grammar models and hence can be represented as

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle \tag{2.7}$$

Where X is a non-terminal, with γ and α are strings of both terminals and non-terminals, with the one-to-one mappings or alignments between the strings captured in $\sim$.
Hierarchical phrases are represented in a synchronous CFG as:

$$X \rightarrow \langle yu\ X_{(1)}\ you\ X_{(2)},\ have\ X_{(2)}\ with\ X_{(1)} \rangle \tag{2.8}$$

$$X \rightarrow \langle X_{(1)}\ de\ X_{(2)},\ the\ X_{(2)}\ that\ X_{(1)} \rangle \tag{2.9}$$

$$X \rightarrow \langle X_{(1)}\ zhiyi,\ one\ of\ X_{(1)} \rangle \tag{2.10}$$

These grammar rules, however, need to join together and terminally form the sentence, for which two glue grammar rules are suggested.

$$X \rightarrow \langle S_{(1)} X_{(1)}, S_{(2)} X_{(2)} \rangle \tag{2.11}$$

$$X \rightarrow \langle X_{(1)}, X_{(2)} \rangle \tag{2.12}$$

$$\begin{aligned}
\langle S_{(1)}, S_{(2)}\rangle &\implies \langle S_{(2)}X_{(3)}, S_{(2)}X_{(3)}\rangle \\
&\implies \langle S_{(4)}X_{(5)}X_{(3)}, S_{(4)}X_{(5)}X_{(3)}\rangle \\
&\implies \langle X_{(4)}X_{(5)}X_{(3)}, X_{(4)}X_{(5)}X_{(3)}\rangle \\
&\implies \langle Aozhou\ X_{(5)}X_{(3)}, Australia\ X_{(5)}X_{(3)}\rangle \\
&\implies \langle Aozhou\ shi\ X_{(3)}, Australia\ is\ X_{(3)}\rangle \\
&\implies \langle Aozhou\ shi\ X_{(7)}\ zhiyi, Australia\ is\ one\ of\ X_{(7)}\rangle \\
&\implies \langle Aozhou\ shi\ X_{(8)}\ de\ X_{(9)}\ zhiyi, Australia\ is\ one\ of\ the\ X_{(9)}\ that\ X_{(8)}\rangle \\
&\implies \langle Aozhou\ shi\ yu\ X_{(1)}\ you\ X_{(2)}\ de\ X_{(9)}\ zhiyi, Australia\ is\ one\ of\ the\ X_{(9)} \\
&\qquad that\ have\ X_{(2)}\ with\ X_{(1)}\rangle
\end{aligned}$$

Figure 2.2: Example partial derivation of synchronous CFG

Figure 2.2 shows an example derivation of a parallel translations from glue grammar rules. This also clearly demonstrates how the phrase-ordering is achieved.
Hierarchical SMT model suggested here, however, deviates from the *Noisy channel* model and instead adopts the *Source channel* model put forth by (Och and Ney, 2002). Thus the training model used changes to a log-linear approach where a collection of features is used to capture the translational information included in each grammar rule. The scoring function $w(D)$ is defined in equation 2.13

$$w(X \to \langle\gamma, \alpha\rangle) = \prod_i \phi_i(X \to \langle\gamma, \alpha\rangle)^{\lambda_i} \tag{2.13}$$

Grammar rules are constructed from the parallel corpora similar to phrase based SMT models. Initially word alignments are generated using GIZA++(Och and Ney, 2003) and extracted bi-directional aligments are used to form initial phrase pairs using which grammar rules are extracted as follows:
Definition 1. $\langle f_i^j, e_{i'}^{j'}\rangle$ is said to be an initial phrase pair iff words align as $f_k \sim e_{k'}$ where $k \in [i, j]$ and $k' \in [i', j']$.
Definition 2. After the above initial phrase pairs, rules are constructed as

- $X \to \langle f_i^j, e_{i'}^{j'}\rangle$ is a rule where $\langle f_i^j, e_{i'}^{j'}\rangle$ is an initial phrase pair.
- $X \to \langle\gamma 1\ X_{(k)}\ \gamma 2, \alpha 1\ X_{(k)}\ \alpha 2\rangle$ is a rule where $X_{(k)}$ is an existing rule formed by the initial phrase pair $\langle f_i^j, e_{i'}^{j'}\rangle$, that is the rule $X_{(k)} \to \langle f_i^j, e_{i'}^{j'}\rangle$.
  $\gamma 1, \gamma 2 \in f\ and\ \alpha 1, \alpha 2 \in e$ are words that do not form a clean initial phrase pair based on Definition 1.

During the decoding phase using CKY parser, the best derivation e for a given f is calculated by the following equation:

$$e(argmax\ w(D)) \tag{2.14}$$

where $w(D)$ is derived using equation 2.13, which is simplified to a log linear model.

2.3 Clustering

Clustering tries to partition a given dataset into discrete sets of data points such that elements in the sets are more similar to each other than elements across different sets.

2.3.1 K-means

K-means is an unsupervised clustering algorithm devised by MacQueen (1967). *K-means* is used for segregating the vector space into k clusters of data points. *K-means* tries to minimize the sum of squared distance between the points in a cluster and the mean of the cluster across all clusters as in equation 2.15.

$$arg_s\ min \sum_{i=1}^{k} \sum_{x \in s_i} |x - \mu|^2 \tag{2.15}$$

The algorithm works in two phases listed as *pseudocode 1*. Initially the centroids of the clusters are randomly initialized to k random data points in the vector space.

After updating the centroids for each cluster group shown in step $UpdateCentroids$, the algorithm starts back from step $Assignment$ and repeats until the clusters converge with no changes in the assignment. The drawback of this algorithm with respect to clustering is that one needs to have prior knowledge about the number of clusters present in the given dataset in order to effectively cluster the data set.

2.3.2 Hierarchical Clustering

In hierarchical clustering the data from the dataset is not partitioned into different clusters but rather the data is represented as a complex tree structure called a dendogram in which the existing data starts as a single cluster running into smaller clusters until individual single data points. Hierarchical clustering follows two approaches

- **Agglomerative** - A bottom-up approach where each data point is clustered together to form the hierarchy

- **Divisive** - A top-down approach where the entire dataset is treated as a single cluster and is broken down until the hierarchy for clusters is established.

Algorithm 1 K-means

```
1: Assignment:
2:     while $i < dataSize$ do
3:         while $j < k$ do
4:             $d = ||X[i] - \mu_j||$
5:             if $d < minC$ then
6:                 $minC \leftarrow d$
7:                 $index \leftarrow j$
8:             end if
9:         end while
10:        $C_{index}.add(x)$
11:    end while
12: Update-centroids:
13:    while $C_i \in C$ do
14:        $\mu_i \leftarrow \frac{(\sum c)}{(\ |C_i|\ )}\ where\ c \in C_i$
15:    end while
16: if $C_i \in C$ does not change then
17:    return $C$
18: else
19:    goto $Assignment$
20: end if
```

Where X is the set of data points, μ_j is the current centroid of the cluster C_j, and C represents the set of clusters.

$minC$ represents the distance between a data point and the closest cluster centroid μ_j

We discuss agglomerative clustering as it is the most usual method to achieve hierarchical clustering.

The algorithm begins with each data point as an individual cluster which are grouped under C, and proceeds with the steps shown in algorithm 2

Algorithm 2 Hierarchical clustering

```
1:  Cluster-closest
2:      $min \leftarrow \infty$
3:      $x \leftarrow 0$
4:      $y \leftarrow 0$
5:      for all $c_i, c_j \in C, i \neq j$ do
6:          if $d(c_i, c_j) < min$ then
7:              $min \leftarrow d(c_i, c_j)$
8:              $x \leftarrow i$
9:              $y \leftarrow j$
10:         end if
11:     end for
12:     $C.remove(c_x)$
13:     $C.remove(c_y)$
14:     $c_{new} \leftarrow c_x + c_y$
15:     $C.add(c_{new})$
16: if $|C| \neq 1$ then
17:     goto $Cluster - closest$
18: end if
```

For the sake of estimating the distance $d(c_i, c_j)$ between two clusters any of the following methods is used.

- **Complete linkage**

$$d(c_i, c_j) = max\{ \ eucDist(p, q) \ : \ p \ \in \ c_i, \ q \ \in \ c_j \ \} \tag{2.16}$$

- **Single linkage**

$$d(c_i, c_j) = min\{ \ eucDist(p, q) \ : \ p \ \in \ c_i, \ q \ \in \ c_j \ \} \tag{2.17}$$

- **Average linkage**

$$d(c_i, c_j) = \frac{1}{|c_i| * |c_j|} \sum_{p \ \in \ c_i} \sum_{q \ \in \ c_j} eucDist(p, q) \tag{2.18}$$

Hierarchical clustering has the advantage of not needing any prior information regarding the number of clusters present in the data.

2.4 SSF

Shakti Standard Format - SSF in short is an annotated data representation format presented in (Bharati et al., 2007). It is based on XML syntax, but deviates from XML standards to facilitate better readability. SSF supports both phrase structured and dependency structured analyses and also facilitates representation of partial analysis of sentences. At text level, SSF has two major segments - header and body. The header contains information regarding the origin, creation, and distribution of the text enclosed in Creative Markup Language(CML) scheme.
The body contains uniquely identified blocks of text each representing a sentence and its analysis, enclosed in a <*Sentence*>XML tag. Every sentence block contains nodes seperated by *newline* and each node contains 4 system properties separated by *tabs*, namely:

- **Address** - a unique value indicating the position of the node relative to chunk boundary
- **Token** - a value indicating any of chunk boundaries ((,)) or a word/numeric/symbol of the sentence
- **Category** - Category of the node which is either a POS tag or a Chunk label
- **Others** - Feature set corresponding to the specific node or corresponding to the chunk. It is absent for the end of chunk boundary -)).

Figure 2.3 shows the representation of sentence *Children are watching some programmes on television in the house* in SSF format.

SSF representation can be classified into two different types, depending on the values the four properties can take. They are:

2.4.1 Inter-Chunk dependency format

The chunk boundary “((" indicates the start of a new chunk and “))" indicates the end of the chunk. For the start of the chunk boundary “((", the Category property is the chunk label and *Other* field contains information regarding the type of dependency relation with its parent and also a unique name formulated from the chunk label. In certain intermediate formats, the *Other* property also contains information about the head of the chunk. The end of chunk boundary "))" does not contain *Category* and *Other* properties. Additional information such as the type of voice in the sentence is also present.
For the remaining nodes, the *Category* property indicates the Part of Speech tag. The *Other* property

```
Address   Token        Category       Attribute-value pairs
  ----------------------------------------------
  1       ((             NP
  1.1     children       NNS     <fs af=child,n,m,p,3,0,,>
          ))
  2       ((             VG
  2.1     are            VBP     <fs af=be,v,m,p,3,0,,>
  2.2     watching       VBG     <fs af='watch,v,m,s,3,0,,' aspect=PROG>
          ))
  3       ((             NP
  3.1     some           DT      <fs af=some,det,m,s,3,0,,>
  3.2     programmes     NNS     <fs af=programme,n,m,p,3,0,,>
          ))
  4       ((             PP
  4.1     on             IN      <fs af=on,p,m,s,3,0,,>
  4.1.1   ((             NP
  4.1.2   television     NN      <fs af=television,n,m,s,3,0,,>
          ))
          ))
  5       ((             PP
  5.1     in             IN      <fs af=in,p,m,s,3,0,,>
  5.2     ((             NP
  5.2.1   the            DT      <fs af=the,det,m,s,3,0,,>
  5.2.2   house          NN      <fs af=house,n,m,s,3,0,,>
          ))
          ))
```

Figure 2.3: Example: Representation of Shakti Standard Format (SSF)

contains several fields of analysis for each corresponding node. The *af* or abbreviated feature set contains the morphological analysis of the node, namely - root word/lemma, coarse POS tag, gender, number, person, TAM (tense-aspect-modality) and Vibhakti values in a comma separated format. The *Other* property also contains a unique node - *name* formulated from the token value and a relative position value - *posn* of the node in the sentence.

2.4.2 Intra-Chunk dependency format - Expanded

In the expanded format, the chunk boundaries are removed and information is added into the *Other* property of the node indicating the name of the chunk they are in. For each node, the *Other* property additionally contains information about dependency relation with the node's parent and also information about the chunk label and the chunk type. Though the expanded format lacks the chunk boundaries, it contains all the information present in the inter-chunk format and more.

This ends the basics of the algorithms and procedures that serve as the basis for the work done as part of this thesis.

Chapter 3

Initial experiments: Factored model Hierarchical machine translation

The Hierarchical phrase based translation was introduced by David Chiang in his paper (Chiang, 2005). Based on which the tool *JOSHUA* is developed (Li et al., 2009). The idea was to use Hierarchical grammar rules for phrase translation rather than phrase based synchronous rules for translation. This facilitates learning the word-ordering information as part of grammar rule instead of separately tackling word and phrase reorderings while generating translations.

$$\text{Eg - } X \;\rightarrow < X_1\ de\ X_2,\ the\ X_2\ that\ X_1 >$$

Joshua's decoder (named *Thrax*) works by translating synchronous hierarchical grammar rules generated by chart parser, then reducing the search space complexity through cube pruning. We will be referring to Joshua's decoder as Thrax decoder from here on.

When it comes to English-Hindi language pair, word and phrase re-ordering is crucial while learning the translational grammar rules. Though Hindi is a free word order language, and is lenient with respect to the ordering quality of translations, it needs to be noted that both Hindi and English differ at a structural level of sentence construction. Hindi has a *SOV* (subject, object, verb) ordering where as English utilises a *SVO* (subject, verb, object). Also Hindi employs postpositions, where as English employs prepositions, resulting in the word order of a sentence to be reversed in the case of both languages. The thought process behind these experiments was the idea that - injecting factored information about POS tags would aid the inherently better system of hierarchical SMT(Chiang, 2005) to generate better word and phrase orderings in translations. For the sake of experiments we considered a Hindi $\rightarrow$ English translational model to induce factored models.

Thrax decoder during the time of experimentation did not support additional feature scores for searching through solution space. Hence we inclined towards a post-translation model to induce factored models into hierarchical based translation system. We proceeded with the experiments using the basic model as follows :

- Lexical sentence will be translated using Thrax decoder to generate k-best lexical translations, ranked based on their cost of translation.
  Parallel corpora is used to train the Hierachical Machine Translation system. The cost of translation for each of k-best candidate translations is tracked.
- Corresponding sequence of POS tags, which will referred as POS sentence, will be translated using Thrax decoder to generate k-best POS translations, ranked based on their cost of translation.
  POS tags generated for both sides of the above parallel corpora are used as a new parallel corpora for training a Hierarchical Machine translation System. The cost of translation for each k-best candidate is calculated.
- The translations will be rescored again based on the combined ranking of both outputs and the k-best translations are re-ordered.

3.1 Data

The Tourism corpus consisting of 11300 sentences has been used for this experiment, from which a subset of 500 sentences have been extracted for testing purposes. Hindi sentences in the parallel corpora are converted from unicode to WX format. The parallel corpora is then lower-cased on the English side and tokenized on both Hindi and English sides.

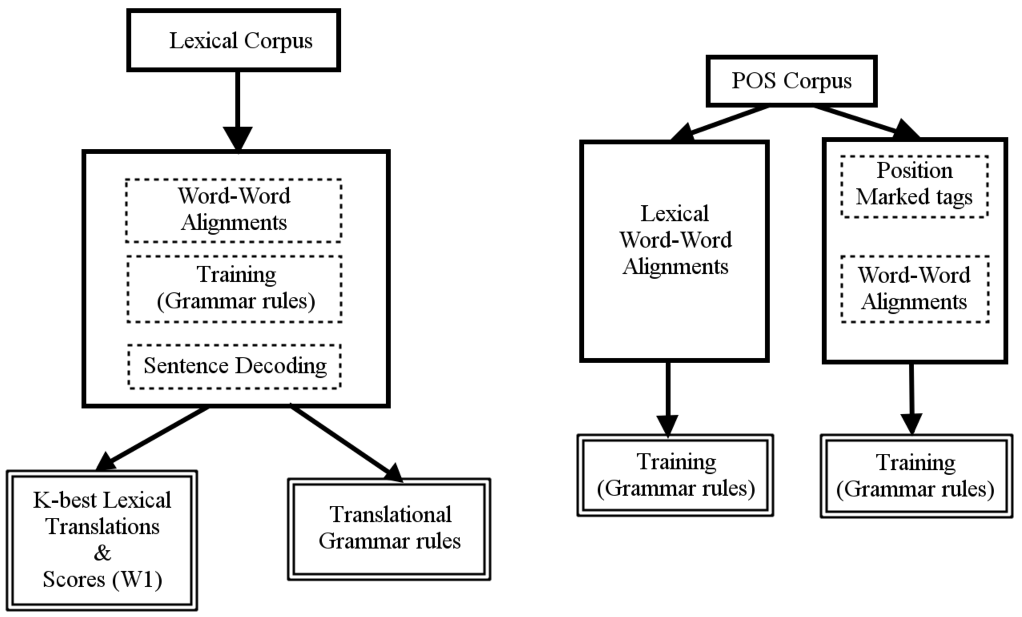
3.2 Implementation

3.2.1 Corpora

Using the basic model mentioned previously, we proceed by building hierarchical machine translation systems for both lexical and POS parallel corpora.
The parallel corpora is POS tagged to extract parallel POS tag sentence corpora corresponding to lexical parallel corpora. CRF tagger (Phan, 2006) is used for generating POS tags for English side. CEL Hindi POS tagger[1] by IIT Kharagpur was used for generating Hindi side POS tags.

Figure 3.1 depicts how the hierarchical SMT system has been built for this experiment. Word to word alignments for the corpora are extracted using giza++(Och and Ney, 2003) system embedded as part of *Moses* Statistical Machine Translation System. Joshua system has been trained on the parallel corpora, namely the lexical corpus and POS tag corpus. Finally, the decoder is run on the lexical corpus over test dataset to yield the base case k-best sentences along with their cost of translations W_1
In the preliminary rounds of the experiment, while trying to generate POS word alignments, we noticed an issue with word alignments. In case of repetition of POS tags say like in

[1]`http://nltr.org/snltr-software/abc.php?file=p24.php`



Lexical grammar generation pipeline POS grammar generation pipeline

Figure 3.1: Grammar generation for Lexical and POS corpora

The big brown fox is quick. (DET JJ JJ NN VB NN)

the alignments of source language (Hindi) converged to the first occurance of POS tag on the target side (English). That is, all noun mappings are mapped to first noun occurance and all verb mappings are mapped to the first verb occurance. This caused several issues while generating hierarchical grammars as they resulted in incorrectly mapped grammar rules for several sentences. Another point to note is that, as the word-word alignments are bi-directional, the resulted alignments are quite sparse with only few words in Hindi side aligned with few words of English side.

To remove this limitation, we experimented with two different solutions

- Use position based tags instead of plain POS tags, that is append the position of the word along with the tag
  Eg - NN_12, JJ_3
- Use the alignments generated for corresponding lexical sentences without any position marking for the POS tags.

Experiments have been repeated with both the paradigms mentioned and we proceed both the paradigms with the following phases sequentially,

```
[[S]] ||| [S,1] [X,2] ||| [S,1] [X,2] ||| 0.4343 0.0000 0.0000
[[S]] ||| [S,1] [X,2] ||| [S,1] [X,2] ||| 0.4343 0.0000 0.0000
[[S]] ||| [S,1] [X,2] ||| [S,1] [X,2] ||| 0.4343 0.0000 0.0000
[[S]] ||| [X,1] ||| [X,1] ||| 0.0000 0.0000 0.0000
[[X]] ||| sitI [X,1] ||| city [X,1] ||| -0.0000 1.2069 0.0669
[[X]] ||| pElesa [X,1] mahala ||| palace [X,1] mahal ||| -0.0000 0.8027 0.7081
[[X]] ||| viSAla saMracanA hEM , ||| magnificent structure , ||| -0.0000 3.4324 2.1757
[[X]] ||| jayapura Sahara ||| jaipur city ||| -0.0000 0.1273 0.0868
[[X]] ||| kI [X,1] ||| [X,1] of ||| 0.1885 0.5953 0.3803
[[X]] ||| [X,1] sAwa [X,2] ||| [X,2] the seven [X,1] ||| 0.1761 0.0334 1.0016
[[X]] ||| [X,1] ko [X,2] Ora ||| [X,1] [X,2] and ||| 0.4260 2.1000 0.0428
[[X]] ||| xIvAra ||| wall ||| 0.0310 0.2121 0.0805
[[X]] ||| GerawA hE ||| is a ||| -0.0000 3.8417 1.5172
[[X]] ||| eka battA ||| occupies one seventh ||| -0.0000 1.1609 1.6198
[[X]] ||| mugala [X,1] hEM . ||| mughal [X,1] . ||| -0.0000 1.5120 0.0654
[[X]] ||| [X,1] sWApawya kalA [X,2] ||| [X,1] [X,2] architecture ||| -0.0000 1.2401 0.2292
[[X]] ||| [X,1] rAjapUwa ||| [X,1] rajput ||| -0.0000 0.0706 0.1313
[[X]] ||| waWA ||| and ||| 0.0090 0.8556 0.0732
[[X]] ||| kA ArScajanaka milana ||| wonderful blend of ||| -0.0000 2.7114 1.4202
```

Figure 3.2: Lexical Translational Grammar rules

```
[S] ||| [S,1] [X,2] ||| [S,1] [X,2]
[S] ||| [S,1] [X,2] ||| [S,1] [X,2]
[S] ||| [S,1] [X,2] ||| [S,1] [X,2]
[S] ||| [X,1] ||| [X,1]
[X] ||| NP_0 [X,1] ||| NN_0 [X,1]
[X] ||| NP_1 [X,1] NP_6 ||| NN_1 [X,1] JJ_5
[X] ||| NP_2 NP_3 NP_4 PU_5 ||| JJ_2 NN_3 SYM_4
[X] ||| NP_7 NP_8 ||| NN_6 NN_7
[X] ||| PP_9 [X,1] ||| [X,1] IN_17
[X] ||| [X,1] NP_12 [X,2] ||| [X,2] DT_12 CD_13 [X,1]
[X] ||| [X,1] PP_14 [X,2] NP_17 ||| [X,1] [X,2] CC_11
[X] ||| NP_13 ||| NN_8
[X] ||| NP_15 NP_16 ||| VBZ_9 DT_10
[X] ||| NC_10 VM_11 ||| VBZ_14 CD_15 NN_16
[X] ||| NP_18 [X,1] VAUX_26 PU_27 ||| JJ_18 [X,1] SYM_25
[X] ||| [X,1] NP_21 NP_22 [X,2] ||| [X,1] [X,2] NN_24
[X] ||| [X,1] NP_20 ||| [X,1] NN_20
[X] ||| NP_19 ||| CC_19
[X] ||| PP_23 NC_24 VM_25 ||| JJ_21 NN_22 IN_23
```

Figure 3.3: POS Grammar rules

3.2.2 Training: Grammar model generation

By using the word-word alignments generated, grammar models are extracted by training the Joshua SMT system over the training data sets of both lexical and POS parallel corpora.

3.2.3 Decoding test data set

Thrax decoder is used to generate k-best translations for the lexical test data set. *Figure 3.1* shows the lexical SMT system that is used to extract the k-best translation candidates with their cost of translationsW_1. The decoder is modified to output the grammar rules that have been used to achieve the translation candidate, along with scores of the individual grammar rules used. *Figure 3.2* shows a sample set of lexical rules extracted during decoding of sentences.
Though running the decoder on POS test set to extract the k-best translations seems like the way forward, we would run into the issue of trying to map the lexical candidate translations with POS candidates.

3.2.4 Weights extraction

The POS tagger is run on the k-best English translations to extract the expected POS candidate translations. To extract the expected cost of translating the Hindi POS sentence to the English POS candidate established now, we utilize the lexical grammar rules generated in *subsection 3.2.3*, and replace the words with respective POS tags. *Figure 3.3* shows sample POS grammar extracted for the sample lexical translations in *figure 3.2*. This yields us the translational grammar rules for POS sentences, which leaves us with the task of extracting the weights corresponding to each grammar rule, from which we can evaluate the total cost of POS translation.
The POS grammar rules, which have been constructed above are cross-referenced with the grammar models generated during training stage in *subsection 3.2.2*. When a rule is found in the grammar model bank, the weights corresponding to the rule are extracted, else the weights are assumed to be a value equivalent to zero for their respective feature functions. In case of feature weights like arity or word penalty where the cost for a given rule is independent of other rules, the weight is inferred from the rule itself. This along with the pre-seeded glue grammar rules and their costs are used to estimate the cost of translation W_2 of POS sentence as follows.

$$W_2 = \sum f_i * w_i \tag{3.1}$$

3.2.5 Sorting and evaluation

The lexical translations are now sorted according to total weights

$$W_f = W_1 + W_2 \tag{3.2}$$

and the best translation e^* is extracted, and BLEU scores are calculated with respect to reference translation.

	Grammar bank size	Hit Percentage
Pure Grammar	262898	76.17%
Position marked	749798	41.86%

Table 3.1: POS Grammar rules

K-best rescoring	Position Marked		Pure		**Best**	
	NIST	BLEU	NIST	BLEU	NIST	BLEU
Moses					6.5118	0.2822
Plain K=50					**6.7458**	**0.3058**
K=5	6.6222	0.2942	6.6398	0.3012	**6.6398**	**0.3012**
K=10	6.6109	0.2922	6.5814	0.2965	6.5814	0.2965
K=50	6.3877	0.2722	6.3722	0.2797	6.3722	0.2797

Table 3.2: BLEU and NIST scores of rescored k-best translations

3.3 Results

The grammar banks created consist of two types,

1. Pure grammar rules
2. Position marked grammar rules

The hit percentages for the extraction of POS grammar rules in *section 3.2.4* stage, in the grammar banks are shown in *table 3.1*.

The grammar bank sizes are as expected due to the fact that position marked POS tags constitute to the diversification of same pure POS tag rule. The hit percentage statistics are also as expected, even accounting for the marginal differences in sizes of Position marked and Pure grammar banks, because the diversity of the key rule that is being searched is also reduced. The following are the statistics for best translation candidates, that is 1-best translations ranked after rescoring the translations with updated costs.

K is also considered a factor influencing the quality of sentences because it is possible that a sentence that may be lexically really bad, say ranking 32^{nd}, but may possess a good aggregate score, accounting for a much better POS score say 4^{th}. To avoid losing good translations by limiting the scope of candidates to rescore, 'K' is also considered as an influential factor in these experiments.

The reasons for choosing the specific k values are none in particular, the upper limit of 50 even though large is chosen so as not to miss out any potential candidates even though their lexical score is small. It is noticeable that the score decrease on increase in value of 'k', because of regarding lower quality translations as potential candidates. *Table 3.2* presents the results of these experiments. The 5-best rescoring system with pure grammar rules has resulted in the best BLEU score of **0.3012** for the Hindi$\rightarrow$ English translation system. However, it did not perform any better than the baseline system with simple hierarchical SMT models yielding a BLEU score of **0.3058**.

3.4 Conclusion

The results do seem disappointing as to the fact that the factored model has not yielded any better output than the base-case. On cross-checking the oracle best translation, that is the translation with the best BLEU score among all the plain k-best candidates is already present at the top of plain lexical translations. Hence for most cases, re-scoring the candidates did push the 'oracle' candidate down in the ranking, thus reducing the BLEU score.
The scope for improvement in translation quality for this corpora is limited not only due to the saturation of oracle best results, but also due to degradation of rankings in POS translations. Tracing back the causes for degradation of POS translation scores we noticed that the lexical candidate translations contain a good deal of untranslated Hindi text. This when coupled with a POS tagger for English during Weight Extraction phase mentioned in *section 3.2.4*, resulted in poorly generated POS candidates.
This may be tackled by trying to induce better lexical translations directly by injecting the decoder with POS features directly, thus possibly leading to better and complete translations. However, due to limited resources pertaining to both lack of English-Hindi parallel resources, possibly without any oracle candidate saturation, and lack of better feature rich models generated from lexical corpora, diminished the scope for further inspection into these experiments.

Digging deeper shows us far more short-comings the field of English$\leftrightarrow$Hindi SMT suffers from, like limited parallel corpora, limited features to bank upon to improve the translation quality, which guide the nature of this thesis towards a resource creation approach.

Chapter 4

Benchmarking of English-Hindi parallel corpora

Machine Translation (MT) has occupied a majority of the spectrum of efforts in NLP in the last couple of decades. Statistical approaches to Machine Translation (SMT) have been gaining more prominence in the recent past. Indian languages are one set for which approaches to SMT have only recently been studied (Ramanathan et al., 2009; Venkatapathy and Bangalore, 2009; Venkatapathy et al., 2010; Arafat et al., 2010). Compared to language pairs for which large amounts of parallel corpora exist, Indian languages fall short in terms of quantity that can be used for SMT. But, parallel corpora resources that may be used by researchers for sake of comparative analysis exist. Hindi, being an Indian language spoken by the majority of the country, has managed to find more sizable resources when compared to other Indian languages. More efforts are ongoing into building large collections of parallel corpora for all Indian languages to help create general purpose SMT systems. Most of these efforts are distributed and result in different corpora sets with variations across texts. Corpora resources created in this manner lack normalization across efforts.

Inspite of lack of parallel corpora for statistical methods, MT from English to Hindi has been the focus of research efforts for close to two decades. Several attempts at English machine translation have been taken up like *NLG human aided MT system*(Rao et al., 1998) a rule based recursive tree-writing MT system, *Mantra* a tree-transfer translational system for translating appointment letters, *MāTra* a human aided machine translation system for English→Hindi, *Anglabhārati* a rule based pattern directed system for translating health campaigns, *Anusaaraka*a transfer-based system consisting of hand-crafted grammatical rules and large bilingual dictionaries ((Bharati et al., 2002) describes all the systems). Apart from these, other attempts at creating general purpose translation systems for translation to Hindi have been made leading to reasonable success. These translation systems use a customized pipeline for carrying out the task of translation, leading to difficulties in comparing their respective approaches to MT. The same is also the case for MT systems created using statistical methods, making reproducibility of results impossible.

What is lacking in these efforts is lack of a standard linguistic analysis benchmark that can be used when evaluating different translation systems. Different translation systems based on the same paradigm may result in significantly different translations due to variation in quality of linguistic analysis provided to these translation systems.

The main contribution of this chapter is a proposal for a standard pipeline for uniform linguistic analysis of parallel corpora to be used across different translation systems. Such a pipeline will provide a framework to create annotated corpora that can be used to compare and analyze different approaches to MT. The goal of this study is to create both annotated corpora resources along with establishing a pipeline for processing parallel texts for English$\rightarrow$Hindi MT.

This chapter is organized as follows: we present different parallel corpora available available for English$\leftrightarrow$Hindi in Section 4.1. We also briefly describe the existing MT systems for translation to Hindi in Section 4.2. We describe the proposed pipeline for linguistic preprocessing of texts in Section 4.3. Finally, we conclude by presenting a few benchmarked models for SMT using resources created through the pipeline in Section 4.4.

4.1 Corpora

In this section, we introduce the different parallel corpora datasets available for English$\leftrightarrow$Hindi. (Bojar et al., 2010) mention three previous datasets for the language pair. One of the first known corpus comes from the EMILLE/CIIL corpus created by a collaboration between Lancaster University and Central Institute of Indian Languages, India through the EMILLE project. The parallel corpora consists of texts in English along with their translations in Hindi, Bengali and three other Indian languages. The corpus contains texts from different domains such as *education*, *health*, *legal texts*. A subset of this parallel corpus was validated and released as part of the ACL (2005) shared task on word-alignment (Mihalcea and Pedersen, 2003).[1] Another corpus that came into use for English$\leftrightarrow$Hindi is the DARPA-TIDES corpus. The corpus was released as part of language contest on SMT in 2002. After manual refinement and cleaning, a subset of this corpus was released for the NLP Tools Contest(Venkatapathy, 2008) on SMT for English$\rightarrow$Hindi.

Apart from these two datasets, efforts to create large-scale multilingual parallel corpora for English, Hindi and several other Indian languages have been part of two projects: English to Indian languages MT (EILMT)[2] and Indian Languages Corpora Initiative (ILCI) [3]. Both the projects (*till date*) have focussed on collecting resources for two particular domains: *tourism* and *health*. In case of the EILMT project, bilingual lexica have been additionally created for both these domains containing domain-specific term translations and multi-word expressions.

On the other hand, the ILCI project provides parallel corpora with parts-of-speech tags created by linguistic annotators. Both the EILMT and ILCI projects are initiatives by the Department of Informa-

[1]We refer to this released dataset as EMILLE-ACL05 corpus in the rest of this Chapter.
[2]`http://www.cdacmumbai.in/e-ilmt`
[3]`http://sanskrit.jnu.ac.in/projects/ilci.jsp?proj=ilci`

tion and Technology (DIT) of India, handled by a consortia of different participating institutions. The distributional effort to create these resources facilitates quick creation of large-scale parallel corpora.

In the rest of this chapter, we refer to the resources created from the EILMT project as Tourism-EILMT, Health-EILMT and ILCI project as Tourism-ILCI and Health-ILCI.

Though the above mentioned projects have led to the creation of parallel corpora in multiple Indian languages, the current work focusses on English↔Hindi portion of these resources to present them along with other existing resources. Table 4.1 shows the statistics of the datasets in their current form.

Corpus	# sents	# En tok	# Hn tok
EMILLE-ACL05	3,556	57,118	70,932
TIDES-ICON08	52,000	12,43,815	13,38,994
Tourism-EILMT	15,198	3,83,992	3,65,163
Health-EILMT	7,484	1,37,396	1,69,039
Tourism-ILCI	25,000	4,25,646	4,23,711
Health-ILCI	25,000	4,22,436	4,40,764
NCERT	9,340	1,73,129	1,98,264
Total	137,578	-	-

Table 4.1: Statistics about English-Hindi parallel corpora

# sents- sent counts in the corpora;

# En tokens- token count in English sentences;

# Hn tokens- token count in Hindi sentences

An additional resource that was created at IIIT Hyderabad is a corpora made up of a small portion of physics text-books taught at the high school level. Individual chapters were extracted from the text books and aligned at the sentence level using an automatic aligner. The automatically aligned corpus was validated and evaluated by a small group of native speakers to prune out erroneous alignments. This corpus is atypical from the other datasets in our work, it is a sample of technical writing. For example, a significant portion of this corpus contains mathematical equations and formulae.

Another notable effort towards creating parallel corpora for Indian languages has been carried out through the use of crowd-sourcing (Post et al., 2012). The resource was created by employing large crowd of cheap translators to translate texts in Indian languages to English. To allow for translation variation, they provide multiple alternate translations for each sentence in Indian language.

At this point, we are also familiar with another recent effort to create large corpora for MT between English↔Hindi (Bojar et al., 2014). The effort resulted in a resource containing about 287,000 translations, 25% of which have been included from the TIDES, Tourism-EILMT and EMILLE-ACL05 corpora. The preliminary version of the corpora released for the shared task on SMT for English$\rightarrow$Hindi was reported [4] to have issues due to quality of sources datasets from which the resource was created.

The work in this Chapter is an independent attempt to improve the quality of existing parallel corpora for English$\leftrightarrow$Hindi and provide uniform linguistic annotations to these resources using start-of-art NLP technologies. We hope that the resources from this attempt will allow for easy and more accurate comparison of the on-going attempts in MT for English$\leftrightarrow$Hindi.

4.2 Machine Translation for English$\rightarrow$Hindi

While approaches for SMT have improved greatly in the recent years, work focussed on using SMT techniques for Indian languages has only begun recently. There has been a surge in recent years on developing general-purpose SMT systems for translating from English to Indian languages (Venkatapathy and Bangalore, 2009; Ramanathan et al., 2008; Ramanathan et al., 2009; Venkatapathy et al., 2010). English to Hindi machine translation, in addition to the lack of large-scale training corpora, also grapples with a number of issues owing to the typological divergence between the two languages.

(Ramanathan et al., 2008) and (Ramanathan et al., 2009) discussed methods to handle the morphological complexity of Indian languages, while translation both to and from Indian languages. (Venkatapathy and Bangalore, 2009) present a context based approach for translating from English to Hindi in the framework on Global Lexical Selection models. Also, (Venkatapathy et al., 2010) proposed a dependency-based SMT system for translation from English to Indian languages. The dependency based framework is best suited for translation between languages with free-word order, another characteristic of a few Indian languages like Hindi, Telugu and Marathi. The framework also allows use of large set of features functions, with a flexible feature design from using discriminative models. There are ongoing efforts into building larger collections of parallel corpus for Indian languages as outlined in Section 4.1 to help in creating general-purpose SMT systems.

On the other hand, long and steady efforts on both research and engineering fronts to develop general-purpose MT systems have been going on for a long time. One of the key features of these systems is that they come from different paradigms in MT as opposed to purely statistical approaches to MT. Transfer-based MT systems separate the task of translation into three steps– analysis of the source sentence, a transfer step followed by a generation module to compose translations in the target language. In the source analysis phase, a sentence in the source language is analyzed using a syntactic parser combined with other modules such as word-sense disambiguation. The role of the transfer component is to translate the words in the source language using a bilingual dictionary and to carry out syntactic transformations to reflect the word order of the target language. Finally, the generation module generates accurate word forms in the target language along with handling agreement phenomena. Typically, both bilingual dictionaries for a specific language pair and transfer grammars used for syntactic transformations are hand-crafted by bilingual experts in both languages.

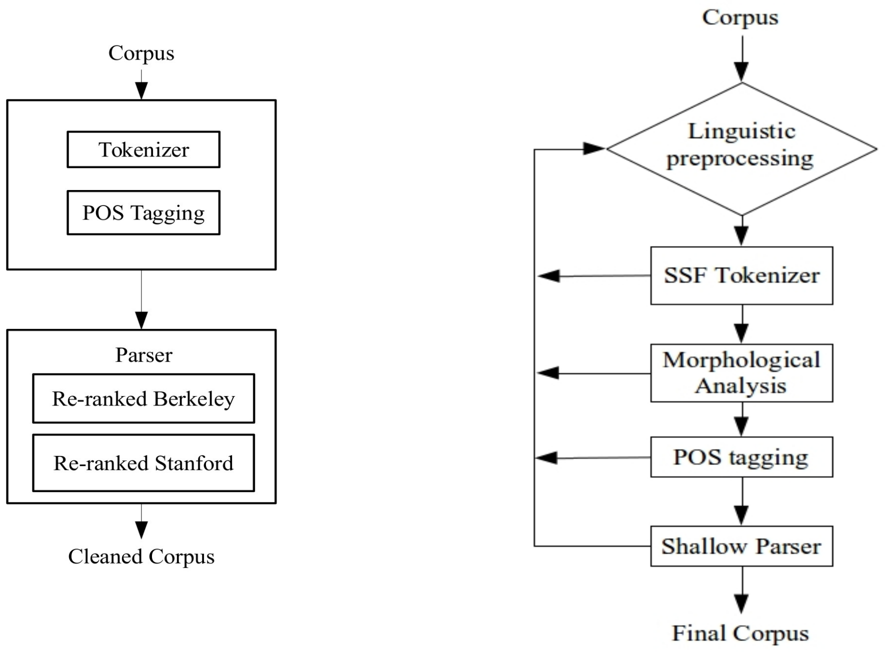
[4] By the authors on `http://ufallab.ms.mff.cuni.cz/~bojar/hindencorp/`

One of the earliest attempts to develop MT systems for English to Indian languages, *Anusaaraka* was created using a transfer-based approach. The system makes use of a combination of multiple state-of-the-art parsers for analyzing the source sentences, and has other components to detect multi-word expressions, generate right inflections in the translation. At the same time, the EILMT project has led to development of both *Shakti*, a transfer-based system and another example-based translation system.

Both statistical and transfer-based make use of different pipelines setup for analyzing texts in English and Hindi. The variations in these pipelines cause difficulty in replicating experiments and accurately comparing the results from different systems. As such, we propose a standard pipeline for linguistic analysis of English and Hindi texts that can be used across different translation systems in the next section.

4.3 Linguistic Preprocessing

At the beginning of this chapter, we mentioned the need for corpora with standard linguistic annotations. In this section, we explain in detail the pipeline that was setup for processing texts in English and Hindi to annotate them with different levels of linguistic analysis. The pipeline is made up of state-of-art tools for syntactic analysis in English and Hindi, set-up in an incremental fashion. Before we describe the pipeline setup, we mention our efforts to clean the datasets discussed in Section 4.1.



Pipeline for English **Pipeline for Hindi**

Figure 4.1: Pipeline for linguistic analysis of English and Hindi texts

4.3.1 Corpora Cleaning

Apart from the EMILLE and TIDES datasets which were released publicly earlier, we noticed several errors in the case of remaining datasets while setting up our pipeline. The cases we observed frequently repeated across the datasets are reported below.

1. **Tokenization errors**: In the ILCI corpus, with manually tagged part-of-speech tags, we noticed several tokens left untagged mostly due to errors in tokenization. Presence of non-uniform delimiter between a word and its part-of-speech tag also caused issue while extracting raw tokens.

   Eg: the\DT person\NN who\WP **has/VBZ** got\VBN

   The presence of different delimiter for **has** is one example from the corpus.

   We also noticed a variation in tokenization across different datasets, for e.g the case of hypenated compound modifiers in English. The difference between "small appliance industry" and "small-appliance industry" is nullified during corpus preparation. All such irregularities were corrected to reflect correct and uniform tokenization across different datasets.

2. **Misalignment**: Some instances in the ILCI datasets were cases of translations being *misaligned* (or *mistranslated*). The topic across translations in these sentences were different. For e.g. the English sentence talks about the human-organ *heart*, its respective translation is focussed on *liver*, an easily detectable error by human verification. However, these errors are difficult to detect automatically. We manually verified the entire corpus and pruned such erroneous sentences out of the datasets.

3. **Incomplete translations**: In some of the corpora, only partial translations of Hindi sentences were noticed on English side. In others, Hindi sentences contained partial translations retaining English text. Also sentences with translator comments and doubts were noticed. Cases like this were identified using heurisitics on sentence length and cross-language length ratio. We chose to prune instances based on the heurisitc given below

   X = avg * 0.3 - diff
   if X < 0 prun sentence
   where 'avg' is average of English and Hindi sentence lengths
   and 'diff' is positive difference in lengths.

   While in some of the above cases the errors were corrected, several sentence pairs were pruned completely.

4. **Word formatting**: As the datasets were created in different environments on different platforms, presence of marking characters like dos-based endmarkers, font conversion residue characters are noticed in several datasets. The presence of these characters causes deviations in MT by marking the token as unseen token. Such cases were automatically corrected and manually verified.

Eg: the person who has got**^M**

The ^M present at the end of sentence makes "got" occuring in this sentence as an unseen token "got^M".
Also multiple forms of same characters create deviation of probabilities. Normalization of variants of quotes, hyphens, mathematical symbols are performed across all the corpora.

Eg: Smart quotes **,** are normalized to ""

4.3.2 Pipeline for Linguistic annotations

Kolachina and Kolachina (2012) conducted an extensive study on benchmarking state-of-art statistical parsers for analyzing English text. The main motivation for their study comes from the need to identify a high quality parser for English that can be used in an English-to-Indian language MT system. The study identifies a reranked variant of the Berkeley parser to perform better over datasets from varied domains. Kolachina (2012) later extended the study to identify a high quality dependency parser by combining multiple parsers to return a consensus analysis for sentences. Following their results, we set up the pipeline for English using the reranked Berkeley parser for producing the syntactic annotation of the sentences.

Figure 4.1 shows the pipeline that we used to annotate the parallel corpora with different levels of linguistic annotations. For texts in English, the tokenizer and part-of-speech tagger were part of Stanford NLP pipeline (Manning, 2011). The part-of-speech tagged text is parsed using reranked variants of Berkeley and Stanford parsers (Kolachina and Kolachina, 2012). For sentences where the reranked Berkeley parser fails, the reranked Stanford parser is used. The syntactic structures are then converted into dependency representations using the same method outlined in Kolachina and Kolachina (2012).

In the case of Hindi texts, the pipeline is made up of independent modules developed for use in the Indian Languages Machine Translation (ILMT) project. Modules used in the Hindi pipeline are morphological analyser, part-of-speech tagger and a shallow parser. The morphological analyzer gives multiple possible analysis for each word in the sentence, which are disambiguated using a pruning module before tagging the sentence with part-of-speech tags. The shallow parser breaks the sentence into *chunks* and assigns to each chunk a *head* word. This essentially reduces the problem of parsing a sentence to parsing these *chunks*, as relations inside a chunk are assigned deterministically based on part-of-speech tags.

One can note that the English sentences are dependency parsed, where as the Hindi sentences are limited to a shallow parsed output only with the chunk-heads marked. This is due to the lack of high accuracy Hindi parsers during the time of experiments. We present the steps taken in this front to tackle this issue and discuss the tools used for creation of a shallow parser in detail in *Chapter 6*

As mentioned previously in Section 4.3 due to the presence of errors, several modules of Hindi and English pipeline experienced hindrances and crashes. In order for the pipeline to proceed forward these errors either had to corrected, normalized or the sentences had to be pruned. We thus augmented the pipeline with a feedback loop (shown in Figure 4.1) allowing us to examine sentences that needed cleaning at the tokenization and formatting levels.

The final output of the pipeline for English contain syntactically annotated sentences with full parses. The Hindi pipeline provides a corpus with morphological analysis, part-of-speech tags and shallow parse information. The head of each chunk is also marked in the sentence. Table 4.2 shows the number of sentences trimmed out of the preprocessing pipeline.

Corpus	# sents
EMILLE-ACL05	0
TIDES-ICON08	0
Tourism-EILMT	6
Health-Merged	486
Tourism-ILCI	52
Health-ILCI	482
NCERT	54
Total	1,080

Table 4.2: Sentences trimmed out of the corpora

# sents- sent removed from the corpora;

4.4 Statistical Machine Translation

In this section, we describe our setup of Statistical Machine Translation (SMT) systems and the relevant experimental details. We use Moses (Koehn et al., 2007), a toolkit for experimenting with different classes of SMT models. In our experiments, we included phrase-based SMT (PBSMT) and hierarchical SMT (Hiero) for translation from English$\rightarrow$Hindi. These classes of models are implemented in the Moses toolkit and thus provide a singular framework for carrying out experiments with different types of SMT models.

In our experiments, we divide the datasets into three partitions for all corpora: training (to extract bilingual information i.e. phrase-tables or synchronous grammars), tuning (to tune parameters of the statistical model) and an evaluation dataset. In the case of the Tourism-EILMT and TIDES dataset, we replicate the partitions provided during the NLP tools contest 2008 (Venkatapathy, 2008) to allow for comparsions with previous results from the shared task. We carried out experiments on EMILLE, Tourism-ILCI, Health-ILCI and NCERT datasets. The statistics of the paritions used in the SMT models are shown in Table 4.3.

Dataset	Training	Tune	Evaluation
EMILLE-ACL05	3,441	25	90
Tourism-ILCI	23,448	750	750
Health-ILCI	23,018	750	750
Tourism-EILMT	14,192	500	500
Health-Merged	30,498	750	750
NCERT	8,286	500	500

Table 4.3: Datasets partition statistics

Dataset	Phrase-Based		Hiero	
	MERT	MIRA	MERT	MIRA
EMILLE-ACL05	43.57	46.73	44.14	46.30
Tourism-ILCI	18.37	18.41	19.73	19.70
Health-ILCI	17.43	17.81	19.39	19.09
Tourism-EILMT	9.04	9.32	6.83	8.25
Health-Merged	17.53	17.66	18.87	19.19
NCERT	17.16	17.28	12.54	12.88

Table 4.4: BLEU scores obtained from different SMT models

The settings used to train Moses models are the same as suggested for baseline models in the WMT shared tasks [5]. We used both the MERT (Och, 2003) and MIRA (Hasler et al., 2011) algorithms to tune parameters of the statistical models. While creating a large language model by combining all the target language texts seemed like a more efficient option, we chose to create the target language model using only the target side of the training corpus. The evaluation of the SMT models were done using BLEU (A. Papineni et al., 2002) which is the widely used MT metric today. Table 4.4 shows the BLEU scores obtained from both PBSMT and Hiero models for all the datasets. We also report the Translation Edit rate scores from the sames in Table 4.5.

The BLEU evaluation scores show that the Hiero models perform significantly between the phrase-based models for a 4 of the 6 datasets. This is expected since hierarchical SMT models are more flexible in reordering translations, a phenomena common in the case of English←Hindi. However, there is a significant drop in the performance of the Hiero models for the Tourism-EILMT and NCERT corpora.

[5] `http://www.statmt.org/wmt11/baseline.html`

Dataset	Phrase-Based		Hiero	
	MERT	MIRA	MERT	MIRA
EMILLE-ACL05	0.4739	0.4709	0.4936	0.4641
Tourism-ILCI	0.6328	0.6354	0.6229	0.6143
Health-ILCI	0.6382	0.6353	0.6215	0.6288
Tourism EILMT	0.8270	0.8177	0.9504	0.9081
Health-Merged	0.6414	0.6349	0.6334	0.6303
NCERT	0.7390	0.7281	0.9123	0.8971

Table 4.5: Translation error rates (TER) obtained from different SMT models

The same pattern is noticed from the Translation edit rate scores for all the datasets. The exceptions in the case of Tourism-EILMT is puzzling given the variation in the behavior when compared with the Tourism-ILCI dataset. Additionally, the low scores might seem puzzling to most as high BLEU scores have been reported in previous attempts to MT for English←Hindi. However, Arafat et al. (2010) previously provided an explanation for these significantly low scores compared to previous results. The variation in the case of NCERT corpora is less puzzling given the small size of the corpus. However, it is interesting to analyze the performance of SMT models on this dataset given the atypical nature of the corpora.

4.5 Conclusion: SAMT grammar experiments

Several experiments utilizing Syntax Augmented Machine translation (SAMT) grammar models have been attempted, which resulted very poor results with baseline BLEU scores of around 4.8 (Tourism ILCI) and 6.11 (EMILLE ACL05). Delving deeper into the reasons we noticed that the language models were too specific even with the amount of corpora provided. Scale of improvements for factored models seems to require presence of even larger quantities of parallel corpora, and (Dungarwal et al., 2014) seems to support this with factored model experiments over large parallel corpora. Also these experiments did not take into consideration reordering features that are crucial for a language pair like English-Hindi where the sentences intrinsically have a varied word order structure.
Due to these reason the benchmark scores for SAMT grammar were dropped, but the corpora in its full entirety with rich linguistic features was released.

Chapter 5

Linearization of parallel text resources with varying page layouts.

5.1 Introduction

Corpora creation has been undertaken by several consortia and groups and research communities for English and Indian language on multiple domains. Attempts pursued by diverse collection of language groups and were always pursued in big numbers as the task is undoubtedly tedious and time consuming. Construction of parallel corpora has always been one of the forefront requirements on NLP community. Numeraous efforts have been made for construction of parallel corpora in English-Hindi language pair like EILMT parallel corpora construction, TIDES, JHU corpora to name a few. (Yeka et al., 2014) lists out a good set of parallel resources in multiple domains. Recent efforts by (Kunchukuttan et al., 2016) (IIT Bombay: English-Hindi corpus) contributed a good amount of parallel corpora did quench the thirst for parallel corpora in English$\leftrightarrow$Hindi languagues. But, the demand for parallel resources is yet to see a down trend.

Though the efforts undertaken in this chapter were initiated with similar intentions to cater to the demand for parallel resources, the effort does fall short in scale to the likeness of (Bojar et al., 2010) and (Kunchukuttan et al., 2016). We, however, present the effort undertaken as part of this thesis in a different light, by tackling the some of the issues encountered during the construction of corpora.

As the chapter progresses we discuss several sources that have been targeted for the purpose of parallel corpora extraction in Section 5.2, then presents the efforts undertaken to convert these sources to formats we can work with in Section 5.3. We proceed to the major issue tackled in this chapter, that is the task of text linearization from varied page layouts in Section 5.5.

5.2 Data Sources

5.2.1 Law-Legal domain

In the law legal domain, during the hunt for openly available data sources in both Hindi and English languages we stumbled upon the Constitution of India containing data in both languages. Constitution of India is available in the public domain through the website of Ministry of Law and Justice (Legislative Department), Govt. of India[1], in both English and Hindi formats for download in PDF formats. The English PDFs utilizes ascii text where as Hindi PDFs are built on *DV_DIVYAE* fonts. Font convertors were applied to convert text to unicode, which are described in detail in *Section 5.4*. During the time of extraction, the Constitution was available divided across different files with total of 22 *PARTs* and 12 *SCHEDULEs*. Files contained pages alternating in English and Hindi texts, with even pages containing Hindi text and odd Pages containing the English text.

5.2.2 NCERT Textbooks

NCERT textbooks for classes 6-12 are publicly available off NCERT website[2], which served as a good source of parallel corpora. Smaller portions of these resources have already been explored by (Yeka et al., 2014), specifically on Physics domain.
Though a wide range of subjects are available for choosing we decided to work with Biology and Psychology text books for classes 11, 12. These files owing to the linear text formats with less equations and formulas were ideal for the linearisation experiments. The chapter-wise PDF files for these subjects are seperately available for both English and Hindi languages. History textbooks starting with class 6 to 12 have also been added to the linearisation experiments. *Table 5.1* presents the data about the raw files subject wise. Biology textbooks contain a good deal of figures and tables inline along with the content,

Subject	Classes	Files from each language
Biology	11-12	42
Psychology	11-12	20
History	6-12	73

Table 5.1: NCERT textbook sources

making diverse page layouts. Linearising these files are addressed in the following sections.

5.2.3 Gandhi's Autobiography

Another good source of parallel data which we stumbled upon were the works of M.K. Gandhi through the website of *Bombay Sarvodaya Mandal and Gandhi Research Foundation*[3]. The works available on the website are free of copyrights and have been accessed with the permission provided by Sarvodaya Mandal. We started with the most famous work of Gandhiji *The Story of My Experiments*

[1]`http://indiacode.nic.in/coiweb/welcome.html`
[2]`http://www.ncert.nic.in/NCERTS/textbook/textbook_hindi.html`
[3]`http://www.mkgandhi.org/ebks/gandhiebooks.htm`

with Truth, his autobiography. Data was available in epub format, with resources available in unicode font for both languages.

5.3 Extraction From PDF

The task of conversion of source documents from PDF to plain text format involved a considerable effort, which will be discussed in detail in this section. Content extraction has been experimented using several tools such as *pdf2text*, online conversion tools, Adobe PDF editor etc. The major issue faced was post content extraction, the text is rendered illegible as the information pertaining to font type and format is lost. Presense of mixed font formats in documents also added to the issue as a single font convertor cannot be applied post extraction.

5.3.1 PDF2HtmlEX Tool

pdf2htmlEX[4] is a tool developed by Lu Wang for the purpose of presenting PDF content directly on the web as HTML documents. *pdf2htmlEX* converts PDF documents to HTML files of the same visual representation. Hence it successfully retains a lot of features useful for post-processing of the extracted content. Features retained include position of the text element i.e. the x,y coordinates, font color, font style indicating boldness, italics of text, font size, font type etc. These features are available in the html content as CSS classes, with the respective stylesheet information presented in the html header.

5.3.2 HTML to text

Extraction of text content from HTML files was achieved using standard java html-parser and CSS script parsers. Manual inspection of the files revealed the CSS class patterns used for identifying page and text entities of a document. The page content is wrapped in a class *pc* and text content with class *t* in a div tag. Each textual entity contains additional CSS class attributes which are used for extraction of features pertaining to each entity. Prior to processing individual text elements, the CSS class contents are parsed from the *head* tag and the Class prefix *vs* value mappings are stored.

Table 5.2 shows the complete notation of features, their class prefixes with example CSS content used in the output of *pdf2htmlex*.

Each individual text element is processed and the class attributes in the *div* tag are mapped to the class mappings extracted and value pertaining to the tag is extracted. The complete feature class with prefix and the respective value of CSS entity is stored in a hash map. For example from the CSS content *x8 : {left: 10.33pts}* the mapping for *x8 - 10.33* is extracted and stored.

Thus the features extracted are used for processing the text content for serialization, paragraph marking and font identification for the purpose of conversions.

[4]Details about the project can be found at `http://coolwanglu.github.io/pdf2htmlEX/`

Feature	Class Prefix	CSS example
font size	fs	font-size: *66*px
font family	ff	font-style: *italic*, font-family: *times new roman*
font colour	fc	#223234 (RGB hex)
x position	x	left: *19.32*pts
y position	y	top: *19.32*pts

Table 5.2: CSS features and classes mapping

5.4 Font conversion

Documents from different sources use different fonts for Hindi content. Font conversion was an essential task to convert the content from third party fonts to unicode standard.

Scientific and technical Hindi website [5] contained many sources and scripts which were used for the conversion of text from fonts like DV_Divya, Walkman_Chanakya and Kruti to unicode. However, these scripts were web-based and were written in Javascript. With little effort these had been migrated to Java for the purpose of automated execution of the font converter scripts.

5.5 Noise removal - Data Serialization

This section describes the different types of noise induced in the text during the process of content extraction and the way it has tackled.
The presence of these noises discussed here caused a considerable amount of effort to be invested in serializing the documents in the runtime of Section 5.3. The noise added to the content extracted is noticed to be majorly of two types.

- Sprinkled noise
- Non-linear text ordering

5.5.1 Sprinkled noise

In the case of NCERT textbooks, all the documents contained a watermark stamped across each page of document in text format, which added to the noise in serialization of text. These text html elements were marked with specific CSS classes using which filtering out the noise was an easy task.

[5] `https://sites.google.com/site/technicalhindi/home/converters`

Presense of shadow texts in the documents which were used for the purpose of highlighting a sub-heading or stylization of captions also interfered with the serialization of text. These duplicate elements are not localized near the actual text in the page layout which adds to the sprinkled noise in the text.

Removing the shadow text is achieved by sorting the text elements based on the x-position and y-position features extracted the in Section 5.3. Every text element in a page is represented as a 2 dimensional vector $t_i \in \mathcal{R}^2$ as $t_i = \{x : i, y : j\}$. The x-position and y-position of a text element can be accessed by $t_i(x)$ and $t_i(y)$ respectively.

The text elements $t_i \in \mathcal{R}^2$, in the given page are sorted, using comparator function $H(t_1, t_2)$, and duplicates i.e. elements with $H(t_1, t_2) = 0$ are eliminated. Function $H(t_1, t_2)$ defined in function 5.1

$$H(t_1, t_2) = \begin{cases} -1 & \text{if } F(t_1(y), t_2(y)) < 0 \\ -1 & \text{if } F(t_1(y), t_2(y)) = 0 \text{ and } F(t_1(x), t_2(x)) < 0 \\ 0 & \text{if } F(t_1(y), t_2(y)) = 0 \text{ and } F(t_1(x), t_2(x)) = 0 \\ 1 & \text{if } F(t_1(y), t_2(y)) > 0 \\ 1 & \text{if } F(t_1(y), t_2(y)) = 0 \text{ and } F(t_1(x), t_2(x)) > 0 \end{cases} \tag{5.1}$$

where $F(p, q)$ is defined as the equation 5.2

$$F(p, q) = \begin{cases} -1 & \text{if} \quad p - q < -t \\ 0 & \text{if} \quad |p - q| \leq t \\ 1 & \text{if} \quad p - q > t \end{cases} \quad \text{where } t \text{ is the threshold value of overlap.} \tag{5.2}$$

After experimenting with the threshold value t, a value of 1.0*pt* chosen for usage in $F(p, q)$ to eliminate the shadow text element which overlap.

5.5.2 Non-linear Text ordering

Variations in page layouts have been noticed due to differences in nature of sources. For example the NCERT text books presented the content in a 2 column page layout as in *figure 5.1* for the domains of Psychology, where as a more complex alternating page layout as in *figure 5.2* is used for the domain of Biology. The page layouts also tend to vary even inside a single document when needing to present an exersice subsection or while presenting a citation. Presence of inline tables and figures also affected the page layout. These variations cause the text to be extracted in a non-linear fashion.

Eliminating the noise pertaining to variations in page layout is a bigger task to tackle than the task of eliminating sprinkled noise. Sorting the text elements in a page using comparator function 5.1 does not yield serial text. For a page-layout shown in Figure 5.4 sorting text elements t_i yields the results in order $p_1 q_1 p_2 q_2 ... p_n q_n$ with elements of P and Q alternating in results, where as the expected order of serialized text is $p_1 p_2 ... p_n q_1 q_2 ... q_n$.

To tackle this problem we use unsupervised clustering to identify the cluster groups in a page. We define cluster groups as a collection of text elements t_i, such that sorting this collection yields linear

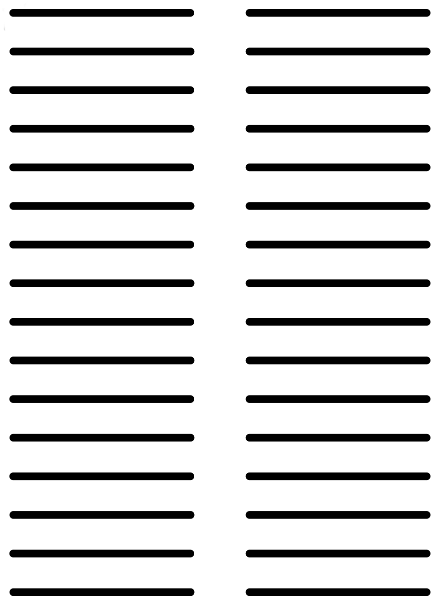


Figure 5.1: Double column page layout

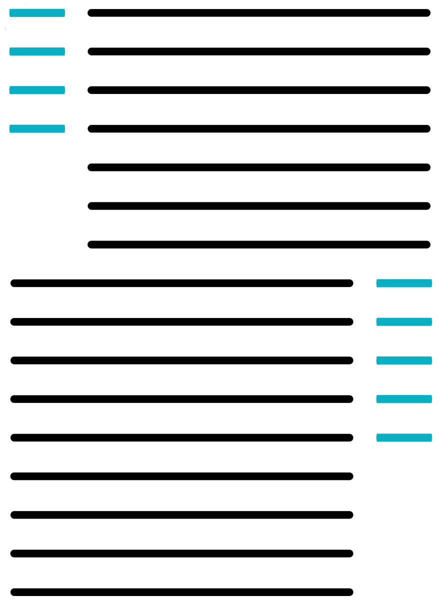


Figure 5.2: Alternating page layout

Figure 5.3: Example Page layouts

readable text. We categorise the elements in a given page layout into different cluster groups, then do an intra-cluster group sort followed by inter-cluster group sort to yield linearised text. K-means, however, has a drawback of needing information pertaining to the number of clusters prior to categorising the solution space. We use a combination of Principal Component Analysis (PCA) and K-Means unsupervised clustering for addressing this. Ding and He (2004) showed the correlation between PCA and K-Means, that the PCA of a solution space is useful in identifying the relaxed solution for the number of clusters in K-Means clustering. This is used to our advantage to initially predict the number of clusters of text elements present in a single page layout. We achieve this by running PCA for all text elements $t_i \in \mathcal{R}^n$ in each page and calculating the Principal Components that represent the solution space. After PCA of text elements in a page is calculated the count (k) of Principal Components that contribute to more than 95% of the variance in the vector space is estimated, which represents the number of clusters present in a page layout.

Using the estimate k for number of means K-Means clustering is run for each page and vector space is clustered to text elements $t_{ij} \in \mathcal{R}^n$, where c_i represents the centroid of the cluster formed by the text elements $t_{i1}, t_{i2}, ...t_{im}$. Thus we arrive up on k clusters and their respective centroids $c_1, c_2, ...c_k$, where $c_i \in \mathcal{R}^n$.

The text elements t_{ij} are now sorted based on the Algorithm 3 and the text is output.

This ensures that the clusters are output in a linear fashion.

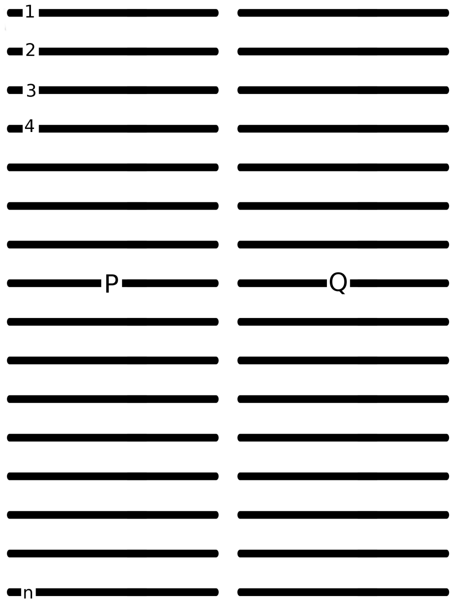


Figure 5.4: Page Layout

It should be noted that the comparator function : 5.1 $H(t_1, t_2)$ was defined for $t_i \in \mathcal{R}^2$ where as sorting of text elements in is over $t_{ij} \in \mathcal{R}^n$. The text elements now are represented using larger dimensional vectors involving additional features of font size, font color, text length and font style extracted in 5.3, but the comparator function $H(t_1, t_2)$ utilizes only x-position and y-position features for sorting.

5.5.3 Cluster 3.0

For the purpose of PCA analysis and K-Means we used Cluster 3.0 (de Hoon et al., 2004) software. Text elements in each page are converted to set of vectors in a tab seperated format as input for Cluster 3.0 and PCA is run. Resulting Principal Components with Eigenvalue contribution and updated co-ordinates are output in different files respectively, which are processed to calculate number of clusters k. K-Means clustering is now run on the input files by passing k as arguments and cluster groups are extracted in multiple output files. More information regarding usage of Cluster 3.0 can be found from their website[6].

[6]http://bonsai.hgc.jp/~mdehoon/software/cluster/

Algorithm 3 K-means Serialization

```
1: Sort-primaryNodes
2:     sort $c_1, c_2, ...c_k$, where $c_i \in \mathcal{R}^n$ using comparator function : 5.1 $H(t_1, t_2)$
3: Sort-clusterGroups
4:     for all $C_i \in clustergroups$ do
5:         sort $t_{i1}, t_{i2}, ...t_{im}$, where $t_{im} \in \mathcal{R}^n$ and $t_{ij} \in C_i$ where $c_i$ is the primaryNode.
6:     end for
7: Print-linearTexts
8:     for all $c_i \in sorted - primary - nodes$ do
9:         for all $t_{ij} \in sorted - C_i$ do
10:            output $t_{ij}$
11:        end for
12:    end for
   Sort-primaryNodes Sort-clusterGroups Print-linearTexts
```

5.6 Corpora Status

The linearised data output by *algorithm 3* is then tokenised on both the languages of the domain. *Table 5.3* presents the current status of the tokenised resources. *Table 5.3* presents the total sentence count, along with the average number of words per sentence in a given domain. Analysis of output text

Domain	File Count	Sentences Extracted		Average Sentence length	
		English	Hindi	English	Hindi
Constitution of India	41	8,468	7,943	17	15
NCERT					
Biology	39	14,479	11,677	16	23
Psychology	18	14,187	10,927	17	24
History	66	30,859	25,158	18	24
Gandhi Auto-biography	1	11,429	12,879	17	15
Total	-	79422	68,584	17	15

Table 5.3: Status of Linearised resources

shows that the text segments still are present sprinkled in nonlinear order occasionally. However the data now is limited to incorrect ordering at cluster level rather than total incorrect ordering. These can now be easily identified and manually corrected with minimal effort. The Constitution of India corpus is being processed to be used as a seed corpus for Judiciary domain SMT system.

We also like to point out several unexplored yet free of copyright resources like works of M.K. Gandhi available through *Sarvodaya Mandal* website, several other subjects from NCERT textbooks like Civics, Economics etc. and a good collection of *Amendment acts* available through Ministry of Law and Justice. These resources show that the numbers presented in *table 5.3*, represent just the tip of the iceberg for an untapped collection of English$\leftrightarrow$Hindi parallel resources.

Chapter 6

Semi-automated annotated treebank construction for Hindi and Urdu

The presence of annotated resources with additional syntactic information has always aided in improving the results of machine translation systems. Syntax Augmented Machine Translation (SAMT) (Zollmann and Venugopal, 2006), Synchronous Tree Substitution Grammar (STSG) (Maletti, 2010) have demonstrated that presence of parsed trees helps the SMT system understand the phrase level reorderings better. However the lack of an accurate dependency parser has been a shortcoming for Hindi SMT experiments. Initial experiments on dependency parsing by (Ambati et al., 2009) and (Ambati et al., 2010), present ways to construct an accurate dependency parser for Hindi. They however, mention the need for larger resource banks for training better and accurate dependency parsers. Narrowing down the train of thought, one can surmise that improving both the quality and quantity of dependency annotated resource banks will pave way for building better parsers, there by aiding in Syntactic SMT.

In this chapter we present the contribution made, to automate an ongoing task (during the time) of manual dependency annotation. Thus boosting the production of annotated Hindi sentences, with an end goal of developing dependency parsers for Hindi.

In the past, many annotated corpora such as Penn Treebank (Marcus et al., 1993) and Prague Treebank (Böhmová et al., 2003) have found prominence by aiding in many NLP related applications. Indian languages inherently contain the property of free word order. This prevents from effectively expressing the analysis in a context free based annotation paradigm. To fully exploit the advantage of free word order in Indian languages, dependency based annotation paradigm has been preferred (Bharati et al., 1995).
After establishing a format like SSF to effectively represent both dependency annotation and context free grammar analysis for Indian languages and creating a fixed set of guidelines for annotation, focus has shifted to creation of annotated corpora. As part of multi-layered, multi representational treebank (Bhatt et al., 2009), the task of dependency annotation for Hindi is taken up at IIIT Hyderabad. Once the initial steps for creation of stable annotation framework for Hindi are completed, the task of creating an annotated corpora for Urdu has also been undertaken. In this chapter, we talk about the automation framework chosen for creating the dependency annotation corpora. We focus on its semi automated nature, where the data interacts with human side and machine side during its flow.

Keeping the focus of the work on the structure and framework of the automation pipeline and its working, we only limit the discussion in this thesis to speak about automation framework introduced to ongoing annotation process and the system errors that are experienced during the functioning of the pipeline. However, one can refer (Agarwal et al., 2012) for a detailed explaination about different linguistic errors encountered because of the nature of modules in the pipeline, and also various methods of correction of errors of linguistic nature.
The chapter is organized as follows. Section 6.1 talks about the sources and the format of raw data. Section 6.2 gives a brief description of the tools used in the pipeline framework. Sections 6.3, 6.4 and 6.5 talk about the annotation procedure, the pipeline framework and the different errors encountered and the steps taken to prevent them. In Section 6.6, we talk about several post-pipeline methods of quality assurance and corrections. Finally, we conclude by presenting the current status of the both Hindi and Urdu treebanks.

6.1 Raw Data

A major portion of the raw data was gathered from FIRE dataset by ISI-Kolkata. This contributes to a major chunk of dependency corpus in the Hindi Treebank, in various subdomains like heritage and pilgrimage. Data corresponding to other domains is gathered from various sources like news paper articles and tourism websites.
In later stages conversation data gathered from short stories written by Nandalal Bharati are added to the corpus. The stories were extracted from the website *http://nandlalbharati.blog.co.in*[1].
The data extracted from the sources then undergo a cleaning phase where the text is cleaned and filtered to ensure that only paragraphs and sentences are present in the text. Content corresponding to page formatting like headings, sub-headings, page numbers, chapter names, author names etc. are filtered out and stored in header section of SSF files as meta-tags. Due to varied sources and different writing formats, the text gathered undergoes necessary font conversions.
Data gathered from these sources not only is of varied fonts but also is in different encoding and formats. UTF-8 and UTF-16 are the most preferred encoding for Hindi and Urdu in text sources. Besides UTF many data sources also use WX format. WX format is an ASCII based transliteration scheme for Indian languages, where characters from Indian languages are mapped with Roman characters based on their phonetic similarities, thus providing a unique representation of Indian Languages in Roman alphabet. The only non-intuitive mapping of characters is done with 'W' and 'X' characters, thus earning the format its name.

[1]The link is currently down

6.2 Tools

The following tools are the major modules used in pipeline:

- **Tokenizer**: Tokenizer was constructed as part of SETU project sanctioned by IIIT Hyderabad. It takes plain text file either in UTF or wx format as input and returns SSF sentences in respective format. It also adds meta-tags to the *header* section of SSF file based on the provided configuration file.

- **Morph**: Morphological analyzer was constructed as part of ILMT Project. It takes an SSF file in WX format as input and returns sentences tagged with morphological analyses in WX format. It outputs multiple possible analyses for each node as it works in a context-free manner.

- **POS**: POS tagger was constructed as part of ILMT Project. It takes an SSF file in UTF format as input and returns POS tagged sentences in UTF format. It takes context into account, hence gives a singular result for each node.

- **Chunker**: Chunker tool was constructed as part of ILMT Project. It takes an SSF file in WX format as input and returns chunked sentences in WX format. It gives a singular prediction for every chunk.

- **Pruner**: Morphological Pruner was constructed as part of ILMT Project. It takes an SSF file in WX format as input and returns sentences tagged with pruned morphological analysis in WX format. It only prunes those multiple analyses for which it is confident above a threshold of probability.

- **Converters**: Converters are constructed as part of ILMT Project. These tools are used as intermediate stages in the pipeline for data conversion from UTF format to WX format and vice versa as required by other modules.

- **Expander**: Expander was constructed by IIIT Hyderabad (Kosaraju et al., 2012) utilizing the Vibhakti Computation and Head computation modules developed as part of ILMT Project. This is a rule based tool that converts SSF sentences in Inter Chunk format to Intra Chunk format. It works on both UTF and WX formats.

- **Sanchay**: Sanchay is an open source platform for working on languages. Sanchay tool provides multiple features like syntactic annotation interface, JAVA SSF API, text editor, UTF to WX and WX to UTF conversions etc. This tool is used by the annotators for annotation and validation of data during different stages of the pipeline framework. It works on both UTF and WX formats (Singh and Ambati, 2010). The tool is available for download from the official Sanchay website[2].

All the ILMT tools are made accessible to the members of ILMT consortia through internal services.

[2]http://sanchay.co.in/

<Sentence id='1'>		
1	raama	unk
2	ne	unk
3	mohana	unk
4	ko	unk
5	niilii	unk
6	kitaaba	unk
7	dii	unk
</Sentence>		

Table 6.1: SSF representation of Tokenized data

6.3 Annotation Process

The dependency annotation in treebank is divided into 2 stages. In the initial stage, the relations among the chunks, namely the Inter-Chunk dependencies are marked, and in the later stage the relations within the chunk, namely Intra-Chunk dependencies are marked.

Annotators are trained in schema of annotation(Begum et al., 2008) and working with the *Sanchay* tool. They are taught to disambiguate between multiple annotations. The dependency guideline sentences (Bharati et al., 2009) are used by the annotators as reference sentences for correctly annotating the sentences.

While the Inter-Chunk dependencies are marked manually by the annotators, the Intra-Chunk dependencies are marked automatically by a rule based tool called *Expander*. This division makes the annotation task more efficient as the human annotators are freed from the tedious work of annotating chunk internal dependencies which can be automatically marked with high degree of accuracy (Kosaraju et al., 2012).

6.4 Pipeline

Figure 6.1 shows the route map of how the raw data passes through different stages to reach the error validation stage. During the entire pipeline, there are some things that are automated and some things that are either validated or annotated manually and there will be constant interactions between both these sides. The automated sections derive their functioning from the idea behind the shallow parser created by ILMT consortia[3].

[3]`http://ltrc.iiit.ac.in/analyzer/hindi/`

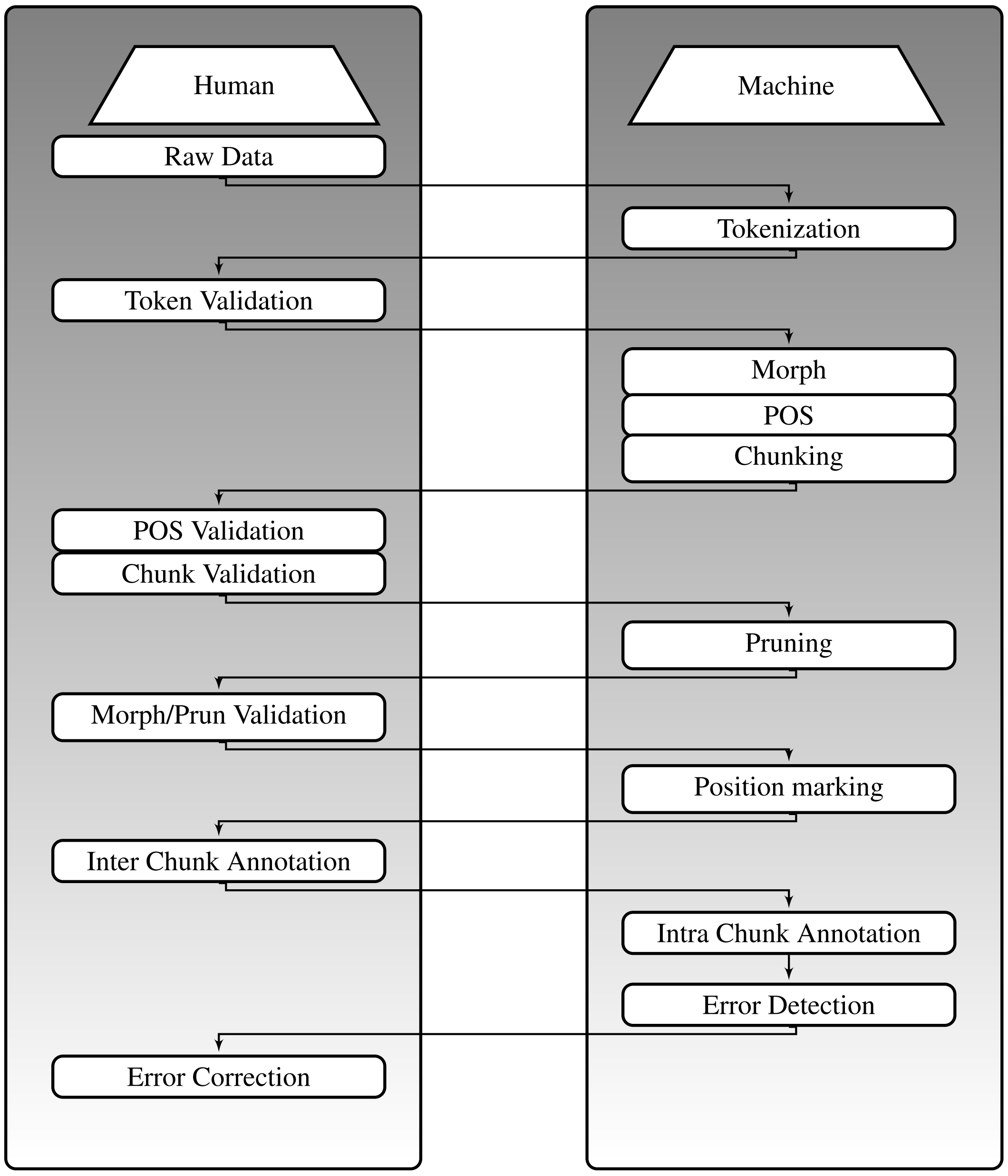


Figure 6.1: Treebanking Annotation Pipeline

```
<Sentence id=’1’>
1          ((        NP
1.1        raama     NNP   <fs af=’raama,n,m,sg,3,d,0,0’>|<fs af=’raama,n,m,pl,3,d,0,0’>|<fs
                           af=’raama,n,m,sg,3,o,0,0’>|<fs af=’raama,n,m,pl,3,o,0,0’>
1.2        ne        PSP   <fs af=’ne,psp,,,,,,’>
           ))
2          ((        NP
2.1        mohana    NNP   <fs af=’mohana,n,m,sg,3,d,0,0’>|<fs af=’mohana,n,m,pl,3,d,0,0’>|<fs
                           af=’mohana,n,m,sg,3,o,0,0’>
2.2        ko        PSP   <fs af=’ko,psp,,,,,,’>
           ))
3          ((        NP
3.1        niilii    NNC   <fs af=’niilii,n,f,sg,3,d,0,0’>|<fs af=’niilii,n,f,sg,3,o,0,0’>
3.2        kitaaba   NN    <fs af=’kitaaba,n,f,sg,3,d,0,0’>|<fs af=’kitaaba,n,f,sg,3,o,0,0’>
           ))
4          ((        VGF
4.1        dii       VM    <fs af=’de,v,f,sg,any,,yaa,yaa’>
           ))
</Sentence>
```

Table 6.2: SSF representation of Morph, POS and Chunked data

6.4.1 Tokenization

Once the cleaning of the raw data is done, the data is passed to the machine side where the tokenizer tool is run. Each of the raw data files may have many paragraphs. The tool initially divides the file into paragraphs based on the space and *new line* marker. Within each paragraph, the sentences are divided by using the sentence end marker. The tokens/words in each sentence are divided by using space. This is a stage where plain text is converted into SSF format. Each sentence in a file will be assigned with a unique numeric ID. Also, for each token in the sentence, a unique integer is assigned as an address, which helps to access the token. The tags <*Sentence* id=>and </*Sentence*>indicate the start and end of the sentence.

Once the tokenizer tool is run, the output is passed to the human side where they validate the output of the tool and do necessary corrections, if any. This validation stage is required because, for some of the tokens like Mr. P. V. Narasimha Rao, the tokenizer tool fails to identify this as a single token, instead it identifies Mr, P, V and Narasimha Rao as 4 different sentences.

An SSF representation of the tokenization output can be seen in *Table 6.1*

6.4.2 Morph, POS and Chunking

Once we are done with the token validation stage, we now move to the machine side where Morph, POS and Chunker tools are run.
For each token in the sentence, the Morph Analyzer tool outputs all the possible morph analyses for that particular token, irrespective of the context. The Morphological information has 8 different categories. They are root, category, gender, number, person, case, tense/aspect, and suffix. All these categories are represented in the feature structure of the token/word together by using a special attribute called *af* or abbreviated features. The field for each attribute is at a fixed position and a comma is used as a separator. The field is left blank for undefined values.
POS tagger assigns a Part of Speech (POS) tag for each token.
The Chunker tool marks the boundaries of the chunks in the sentence and also assigns a chunk tag for each chunk. Chunk boundary is represented using “((" and “))" which indicate the initiation and closure of the chunk respectively. In this stage, the address of the chunk is given as an integer and the address of the token is given as a decimal. The whole part of the decimal indicates the chunk to which the token belongs to and the decimal part of the address indicates the relative position of the token in that particular chunk.
Once the Morph, POS and Chunker tools are run, the data is sent to the annotators for Validation of POS and Chunk information. After the validation of POS and chunk information by an annotator, the same information is cross validated by a different annotator to remove any erroneous cases, if present. This is done to ensure better quality of annotation. The Morph information is not validated here. It is done after the pruning stage.
An SSF representation of the Morph, POS and Chunked output can be seen in *Table 6.2*

6.4.3 Morph Pruning

After the POS and Chunk validation, the data is sent to the Machine side for Pruning. In the pruning stage, the morph analyses for which the value of the category attribute is not in mapping with the POS tag of the token are removed automatically by the tool. After this, the data is sent to the morph annotators for validation. Now, the morph annotators look at the pruned data and finally choose the morph analysis that perfectly fits into that context.
An SSF representation of the Pruner Validated output can be seen in *Table 6.3*

6.4.4 Position Marking

The data is again passed to Machine side to add a new attribute in the feature structure of every token, called *posn* which indicates the position of the word more explicitly in the sentence. The value for the *posn* attribute would be in multiples of 10, in order to facilitate the insertion of new tokens between the existing tokens.
An SSF representation of the Position marked output can be seen in *Table 6.4*

```
<Sentence id='1'>
1       ((        NP
1.1     raama     NNP   <fs af='raama,n,m,sg,3,o,0,0'>
1.2     ne        PSP   <fs af='ne,psp,,,,,,'>
        ))
2       ((        NP
2.1     mohana    NNP   <fs af='mohana,n,m,sg,3,o,0,0'>
2.2     ko        PSP   <fs af='ko,psp,,,,,,'>
        ))
3       ((        NP
3.1     niilii    NNC   <fs af='niilii,adj,f,sg,,d,,'>
3.2     kitaaba   NN    <fs af='kitaaba,n,f,sg,3,d,0,0'>
        ))
4       ((        VGF
4.1     dii       VM    <fs af='de,v,f,sg,any,,yaa,yaa'>
        ))
</Sentence>
```

Table 6.3: SSF representation of Pruned Data

<Sentence id='1'>			
1	((	NP	
1.1	raama	NNP	<fs af='raama,n,m,sg,3,o,0,0' posn='10'>
1.2	ne	PSP	<fs af='ne,psp,,,,,,' posn='20'>
	))		
2	((	NP	
2.1	mohana	NNP	<fs af='mohana,n,m,sg,3,o,0,0' posn='30'>
2.2	ko	PSP	<fs af='ko,psp,,,,,,' posn='40'>
	))		
3	((	NP	
3.1	niilii	NNC	<fs af='niilii,adj,f,sg,,d,,' posn='50'>
3.2	kitaaba	NN	<fs af='kitaaba,n,f,sg,3,d,0,0' posn='60'>
	))		
4	((	VGF	
4.1	dii	VM	<fs af='de,v,f,sg,any,,yaa,yaa' posn='70'>
	))		
</Sentence>			

Table 6.4: SSF representation of Position Run data

<Sentence id=’1’>			
1	((	NP	<fs drel=’k1:VGF’ name=’NP’>
1.1	raama	NNP	<fs af=’raama,n,m,sg,3,o,0,0’ name=’raama’ posn=’10’>
1.2	ne	PSP	<fs af=’ne,psp,,,,,,’ name=’ne’ posn=’20’>
	))		
2	((	NP	<fs drel=’k4:VGF’ name=’NP2’>
2.1	mohana	NNP	<fs af=’mohana,n,m,sg,3,o,0,0’ name=’mohana’ posn=’30’>
2.2	ko	PSP	<fs af=’ko,psp,,,,,,’ name=’ko’ posn=’40’>
	))		
3	((	NP	<fs drel=’k2:VGF’ name=’NP3’>
3.1	niilii	NNC	<fs af=’niilii,adj,f,sg,,d,,’ name=’niilii’ posn=’50’>
3.2	kitaaba	NN	<fs af=’kitaaba,n,f,sg,3,d,0,0’ name=’kitaaba’ posn=’60’>
	))		
4	((	VGF	<fs stype=’declarative’ name=’VGF’ voicetype=’active’>
4.1	dii	VM	<fs af=’de,v,f,sg,any,,yaa,yaa’ name=’dii’ posn=’70’>
	))		
</Sentence>			

Table 6.5: SSF representation of Inter Chunk Dependency data

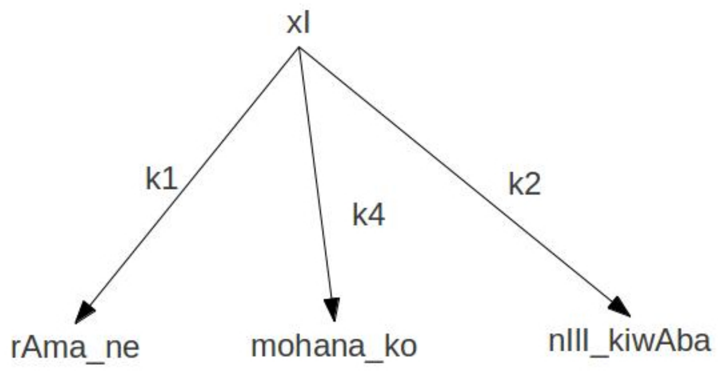


Figure 6.2: Inter-Chunk Dependency Tree

6.4.5 Inter-Chunk Dependency Annotation

After running the position marking tool, the data is passed to the dependency annotators. Inter-chunk dependencies are manually marked by the annotators.
An SSF representation of the Inter-chunk marked output can be seen in *Table 6.5*
A graphical representation of dependency tree constructed after inter-chunk dependency annotation stage can be seen in *Figure 6.2*

6.4.6 Intra-Chunk Dependency Annotation

Once the manual annotation of Inter-Chunk dependencies are done, the Intra chunk dependencies are marked automatically using *Expander* tool. As part of the Expander tool (Kosaraju et al., 2012), the Inter-Chunk data is run through a head-computation module which identifies the head token for each chunk. The dependency relations that were initially present between the chunks at Inter-Chunk level will now be transferred between the head tokens of those chunks. To get the Intra-Chunk relations, a rule template has been created manually. A sample rule is shown in *Table 6.7*.
After running the Expander, the POS and Morph information of each token gets transferred from Inter-Chunk level to Intra-Chunk level without any modification. The bracketing which represent the chunks at Inter-Chunk level is stripped off, but the chunk information is still preserved in the Intra-Chunk format by introducing two additional attributes “ChunkId" and “ChunkType". The unique “name” attribute of the chunk is added as the “ChunkId” value to every token inside the chunk, thus ensuring the “ChunkId” again stays unique. All the chunk members are stamped with “ChunkType" which is either of the two values, “head" or “child" thus indicating the direction of dependency relation inside a chunk. In practice, rather than specifying two different tags and their values, the information is represented using a single tag as “ChunkType=type:ChunkId”.

<Sentence id='1'>

1	raama	NNP	<fs af='raama,n,m,sg,3,o,0_ne,0' drel='k1:dii' vpos='vib_2' name='raama' chunkId='NP' chunkType='head:NP' posn='10'>
2	ne	PSP	<fs af='ne,psp,,,,,,' drel='lwg__psp:raama' chunkType='child:NP' name='ne' posn='20'>
3	mohana	NNP	<fs af='mohana,n,m,sg,3,o,0_ko,0' drel='k4:dii' vpos='vib_2' name='mohana' chunkId='NP2' chunkType='head:NP2' posn='30'>
4	ko	PSP	<fs af='ko,psp,,,,,,' drel='lwg__psp:mohana' chunkType='child:NP2' name='ko' posn='40'>
5	niilii	JJ	<fs af='niilii,adj,f,sg,,d,,' drel='nmod__adj:kitaaba' chunkType='child:NP3' name='niilii' posn='50'>
6	kitaaba	NN	<fs af='kitaaba,n,f,sg,3,d,0,0' drel='k2:dii' name='kitaaba' chunkId='NP3' chunkType='head:NP3' posn='60'>
7	dii	VM	<fs af='de,v,f,sg,any,,yA,yA' stype='declarative' voicetype='active' name='dii' chunkId='VGF' chunkType='head:VGF' posn='70'>

</Sentence>

Table 6.6: SSF representation of Intra Chunk Dependency data

Chunk Name	Parent Constraints	Child Constraints	Contextual Constraints	Dependency Relation
NP	POS==NN	POS==JJ	posn(parent)>posn(child)	nmod__adj

Table 6.7: A sample rule from Expander tool

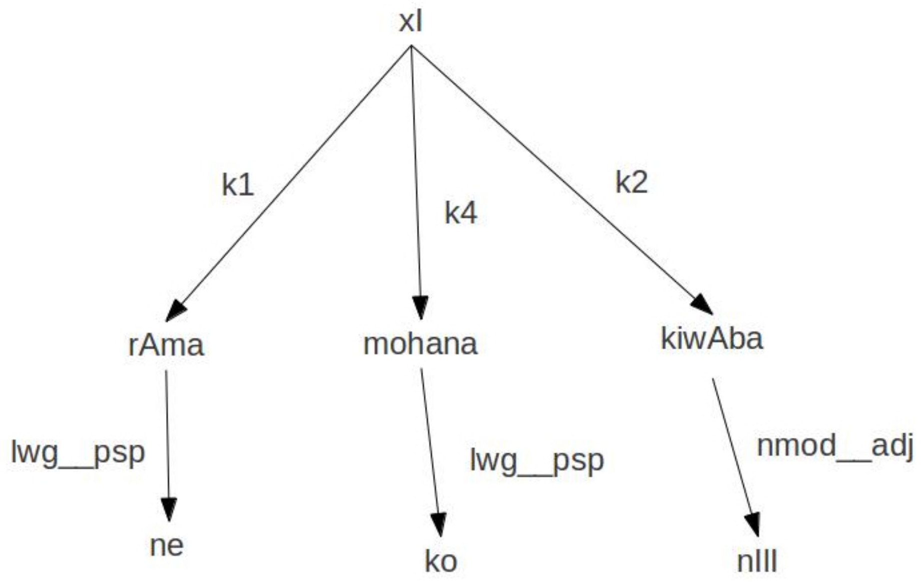


Figure 6.3: Intra-Chunk Dependency Tree

An SSF representation of the Intra-Chunk marked output can be seen in *Table 6.6*

A graphical representation of dependency tree constructed after Intra-Chunk dependency annotation stage can be seen in *Figure 6.3*

6.5 Errors and sanity

Due to limited training data and generic domain nature of the tools, most of the tools fail to predict proper analysis in several cases and result in crashes. This brings forth the need to ensure that these errors do not trickle along different stages of the pipeline and make sure that the crashes that have been detected are immediately localized.
The Morph tool is prone to crashes due to limited entries in morphology table and in many cases the

crashes result in partial sentences with missing nodes or even missing sentences. The chunker tool is also prone to crashes with similar manifestation of errors.
Intra-pipeline sanity tools compare the input and output files after every run of module and check for discrepancies and errors. Only changes that are respective to the module are permitted and rest are reported by the tools. After every module-run, another script called the *difference-generator* checks and ensures that the word and sentence counts remain unchanged before and after every module-run. Every issue noticed is reported to manual validators to either modify corresponding word or sentence; in case of larger errors, the sentence is removed. Thus, intra-pipeline data sanity is ensured.
Another set of tools check for the correctness of annotations and aid in manual cross-validation of the dependencies. As previously mentioned, during the validation stage, the data passes through validations which cover Morph, POS, Chunking and Dependency stages individually. At these stages, automatic error detection tool(Agarwal et al., 2012) is applied to reduce the effort of the manual validators. The Error detection tool extracts the potential error cases from the data and these are manually verified by the annotators and do necessary corrections, wherever required.

6.6 Quality Assurance

Once the data completely goes through the pipeline, it undergoes several stages of quality checking where the complete data is scrutinized automatically for errors and reported errors are corrected manually .
Of the several stages of quality assurance, the initial stages concentrate on ensuring correctness of SSF format of the sentences.

- **Meta-data and CML correctness**: These checks ensure that the header section maintains consistency as per CML scheme. The tools check for the presence of specific meta-tags like source and date of creation.
- **Sentence repetition**: Due to good chance of crashes in the internal modules, sentences are prone to repeat both inside a file and sometimes across the files. These checks ensures that a persistent sentence count is achieved.
- **Feature structure error detection**: These checks look for errors in formats of *Other* property and checks for the validity of the values in *Category field*.
- **Morphological error detection**: These checks ensure the validity of values present in the *af* field of *Other* property. Also, care is taken to ensure that the coarse POS tag and *Category* POS tag agree with each other.

Another set of stages check for the presence of errors in dependency annotation of sentences.

- **Dependency and forest checks**: The sentences are checked for errors in dependency annotations like validity of the type of dependency labels, validity of both parent and child. Also sentences are checked to ensure that a singular dependency tree exists per sentence, without any hanging nodes or *forests*.

- **Cycle Detection**: The sentences are checked for presence of looped dependencies or cycles. The algorithm checks whether all the chunks are connected finally to the root and reports if the root is not reached in fixed number of iterations. Thus presence of cycles is reported for correction.
- **Pattern based dependency checks**: The sentences are checked for presence of specific patterns in the dependency relations for individual verbs. The nodes and their dependencies are cross checked with a list of “Should exist together" and “Should not occur together" and violations are reported to the annotators for corrections.

Thus data is cleaned and quality of the data is ensured before the corpora is locked out as release-ready.

6.7 Status

Hindi Treebank contains data from three domains, general news articles, tourism and heritage, and conversations as represented in short stories, while Urdu Treebank has data only from newspaper articles. Table 6.8 shows the sizes of the two treebanks.

	HTB		**UTB**	
	Sentences	Words	Sentences	Words
News Articles	17,882	395K	7,120	200K
Tourism	1,058	15K	-	-
Conversation	2,028	27K	-	-
Total	20,968	437K	7,120	200K

Table 6.8: Sizes of the two treebanks

The Hindi annotated corpora generated by the automated pipeline has been released at the *Workshop on Machine Translation and Parsing in Indian Languges (MTPIL-2012)* as part of the dependency parsing shared task. The sentences released contained gold standard morphological analyses, POS tags and dependency relations as mentioned previously. **The direct outcome of the shared task has been the development of state of the art dependency parsers for Hindi, submitted as part of the tools contest.** Out of seven participating teams, (Kukkadapu et al.,) and (Jain et al., 2012) stand out as the best parsers for Hindi, there by bringing this research to fruition.
Also, the research under gone to build the pipeline was used in its raw form, utilizing just the automated segments of the pipeline to create a shallow parser. This shallow parser is the one similar to the one built

by ILMT consortia[4] and has been utilized to create Hindi parsed sentences in *Chapter 4*, due to lack of a wholesome dependency parser for Hindi at the time of research.

[4]`http://ltrc.iiit.ac.in/analyzer/hindi/`

Chapter 7

Conclusions

In this work, we discussed the nature this thesis aspires to take on, by highlighting circumstances which guide the research and the issues it tries to tackle. *Figure 1.1* illustrates the different issues that plague English-Hindi SMT, limiting its scope.

We presented different parallel corpora available for English↔Hindi and discussed the nature of these datasets. A standard pipeline has been proposed for processing texts in English and Hindi to annotate them with different levels of linguistic analysis. The main motivation of this work was to create uniformly annotated corpora resources for English↔Hindi SMT. along with baseline statistical machine translation systems that can be used as reference to future work in MT for English-Hindi. These systems are trained using the resources created from the pipeline. The purpose of creating both the annotated datasets and baseline systems is to serve as benchmarks for future translation systems. A total of 1,37,578 annotated parallel sentences belonging to various domains were released.

We also discussed the effort and issues that one faces while trying to generate parallel corpora. The work undertaken highlights the process of generating linear text from multi-column page layout in PDF files. The effort is worth the while as it taps a good set of raw bilingual documents available free of copyright for linguistic purposes. These resources along with the paradigm and tools used are made available with the hope of generating parallel resources. This work contributes a total of 79,422 English and 68,584 Hindi unaligned raw sentences.

A semi-automated pipeline utilized for improving the dependency annotation task, along with the different tools utilised as part of the annotation pipeline was also presented. These tools, alongside the different sanity tasks installed, ensure the quality of annotation work. The pipeline churned out a total of 20,968 Hindi and 7,120 Urdu dependency annotated sentences with morpho-syntactic features. The data produced by the annotation pipeline has been utilised to successfully produce state-of-the-art parsers for Hindi.

From a wider perspective, one can note that this thesis still retains an SMT flavour, by addressing fore-mentioned drawbacks of *Chapter 1* individually by

- Providing a set of standardized Machine Translation systems for English-Hindi language pairs. Cleaning, structuring and benchmarking of available parallel corpora over several SMT standard paradigms.

- Providing a paradigm and a gauge of effort for linearisation of text resources from multiple layouts.
  The task of linearising parallel resources available in varying page design layouts was undertaken, with the intention of generating parallel corpora out of the resources.

- Providing a framework for automation of the dependency annotation, thus facilitating the generation of feature rich resources.
  Aiding in the ongoing effort of dependency annotation, by automating the manual annotation pipeline for Hindi, Urdu languages.

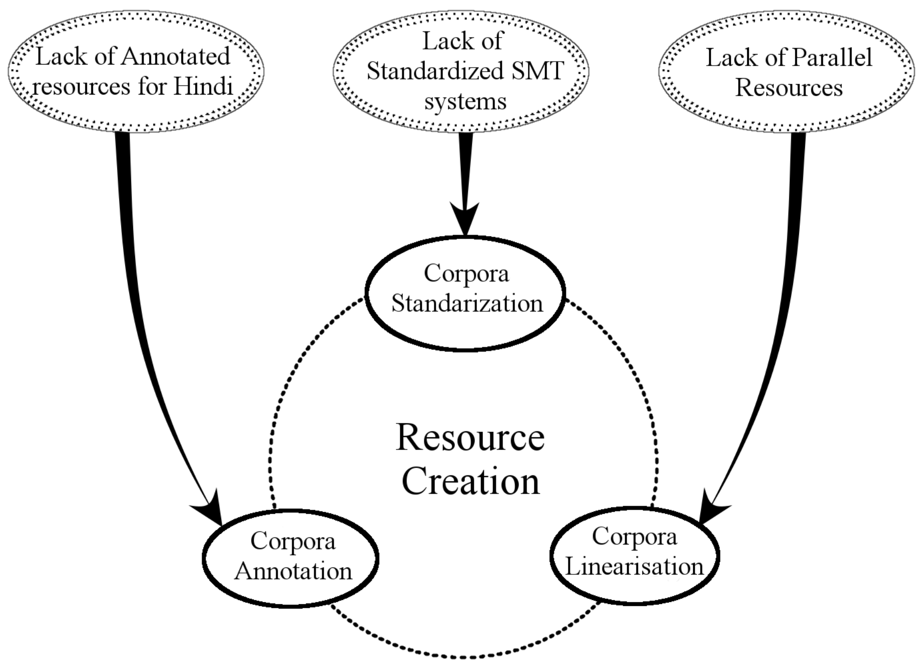

Lack of Annotated resources for Hindi
Lack of Standardized SMT systems
Lack of Parallel Resources
Corpora Standarization
Resource Creation
Corpora Annotation
Corpora Linearisation

Figure 7.1: Addressing issues of English-Hindi SMT

We would like to point out that, while undertaking the task of building parallel corpora for English-Hindi language pair we limit ourselves to the the generation of comparable raw corpora, as this task itself has turned to be quite resource and effort consuming. Thus limiting the scope of this thesis to build linearised raw corpora for both English-Hindi languages.

This thesis has been built on the pain and frustration of several failed experiments. Though the entire work started with the heavy ambitions to create a state of art SMT system for English↔Hindi, this thesis stands as a proud foundation for that cause and limits itself to that level.

Ultimately, I would like to end on a note,

"The climb ahead steep, and summit hidden and cloudy, and with the fading resolve I retire.
I look back in regret, to find the trail I laid, waiting for the next summiteer."

We conclude this thesis with the hope that the contributions made, quench the need of linguistic resources for English↔Hindi SMT, and that keener and younger minds will utilise these resources to build better SMT systems.

Related Publications

- **Jayendra Rakesh Yeka**, Prasanth Kolachina, Dipti Misra Sharma. "Benchmarking of English-Hindi parallel corpora" in *Proceedings of Nineth international conference on Language resources and evaluation* (LREC'2014)
- **Jayendra Rakesh Yeka**, Vishnu Ramagurumurthy, Dipti Misra Sharma. "Semi-automated annotated treebank construction for Hindi and Urdu" in *Second Workshop on Indian Language Data: Resources and Evaluation in LREC'2014* (WILD^{RE} 2014)

Bibliography

A. Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Agarwal, R., Ambati, B. R., and Singh, A. K. (2012). A gui to detect and correct errors in hindi dependency treebank. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 1907–1911, Istanbul, Turkey, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1439.

Ambati, B. R., Gadde, P., and Jindal, K. (2009). Experiments in indian language dependency parsing. In *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing.*, Hyderabad, India.

Ambati, B. R., Hussain, S., Nivre, J., and Sangal, R. (2010). On the role of morphosyntactic features in hindi dependency parsing. In *Proceedings of NAACL-HLT 2010 workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, Los Angeles, CA.

Arafat, A., Kolachina, P., Kolachina, S., Sharma, D. M., and Sangal, R. (2010). Coupling Statistical Machine Translation with Rule-based Transfer and Generation. In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas (AMTA)*.

Begum, R., Husain, S., Dhwaj, A., Sharma, D. M., Bai, L., and Sangal, R. (2008). Dependency annotation scheme for indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Bharati, A., Chaitanya, V., Sangal, R., and Ramakrishnamacharyulu, K. (1995). *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi.

Bharati, A., Sharma, D. M., and Kulkarini, A. P. (2002). Machine translation activities in india: A survey.

Bharati, A., Sangal, R., and Sharma, D. M. (2007). SSF: Shakti standard format guide.

Bharati, A., Sharma, D. M., Husain, S., Bai, L., Begam, R., and Sangal, R. (2009). Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank.

Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D., and Xia, F. (2009). A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189, Suntec, Singapore, August. Association for Computational Linguistics.

Böhmová, A., Hajič, J., Hajičová, E., and Hladká, B. (2003). The prague dependency treebank. In *Treebanks*, pages 103–127. Springer.

Bojar, O., Straňák, P., and Zeman, D. (2010). Data Issues in English-to-Hindi Machine Translation. In Chair), N. C. C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Bojar, O., Diatka, V., Rychlý, P., Straňák, P., Tamchyna, A., and Zeman, D. (2014). Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the Ninth International Language Resources and Evaluation Conference (LREC'14)*, Reykjavik, Iceland, May. ELRA, European Language Resources Association. in prep.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Stroudsburg, PA, USA. Association for Computational Linguistics.

de Hoon, M. J., Imoto, S., Nolan, J., and Miyano, S. (2004). Open source clustering software. *Bioinformatics*, 20(9):1453–1454.

Ding, C. and He, X. (2004). K-means clustering via principal component analysis. pages 225–232. ACM Press.

Dungarwal, P., Chatterjee, R., Mishra, A., Kunchukuttan, A., Shah, R. M., and Bhattacharyya, P. (2014). The iit bombay hindi-english translation system at wmt 2014.

Hasler, E., Haddow, B., and Koehn, P. (2011). Margin Infused Relaxed Algorithm for Moses. In *The Prague Bulletin of Mathematical Linguistics*, volume 96, pages 69–78, September.

Jain, N., Singla, K., Tammewar, A., and Jain, S. (2012). Two-stage approach for hindi dependency parsing using maltparser. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*, pages 163–170, Mumbai, India, December. The COLING 2012 Organizing Committee.

Koehn, P. and Hoang, H. (2007). Factored translation models. In *EMNLP-CoNLL*, pages 868–876.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Kolachina, S. and Kolachina, P. (2012). Parsing Any Domain English text to CoNLL Dependencies. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA).

Kolachina, P., Cancedda, N., Dymetman, M., and Venkatapathy, S. (2012). Prediction of learning curves in machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 22–30, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kolachina, S. (2012). Non-local Features in Syntactic Parsing. Hyderabad, India.

Kosaraju, P., Ambati, B. R., Husain, S., Sharma, D. M., and Sangal, R. (2012). Intra-chunk dependency annotation : Expanding hindi inter-chunk annotated treebank. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 49–56, Jeju, Republic of Korea, July. Association for Computational Linguistics.

Kukkadapu, P., Malladi, D. K., and Dara, A.). Ensembling various dependency parsers: Adopting turbo parser for indian languages. In *24th International Conference on Computational Linguistics*, page 179.

Kunchukuttan, A., Mehta, P., and Bhattacharyya, P. (2016). Indian institute of technology bombay: English-hindi corpus.

Li, Z., Callison-Burch, C., Dyer, C., Khudanpur, S., Schwartz, L., Thornton, W., Weese, J., and Zaidan, O. (2009). Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March. Association for Computational Linguistics.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In Cam, L. M. L. and Neyman, J., editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Maletti, A. (2010). Why synchronous tree substitution grammars? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 876–884. Association for Computational Linguistics.

Manning, C. D. (2011). Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *Computational Linguistics and Intelligent Text Processing - 12th International Conference (CICLing)*, volume 6608 of *Lecture Notes in Computer Science*, pages 171–189. Springer.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Mihalcea, R. and Pedersen, T. (2003). An Evaluation Exercise for Word Alignment. In Mihalcea, R. and Pedersen, T., editors, *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Phan, X.-H. (2006). Crftagger: Crf english pos tagger.

Post, M., Callison-Burch, C., and Osborne, M. (2012). Constructing Parallel Corpora for Six Indian Languages via Crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 401–409, Montréal, Canada, June. Association for Computational Linguistics.

Ramanathan, A., Hegde, J., Shah, R. M., Bhattacharyya, P., and Sasikumar, M. (2008). Simple Syntactic and Morphological Processing Can Help English-Hindi Statistical Machine Translation. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, pages 513–520, Hyderabad, India, January. Asian Federation of Natural Language Processing (AFNLP).

Ramanathan, A., Choudhary, H., Ghosh, A., and Bhattacharyya, P. (2009). Case markers and Morphology: Addressing the crux of the fluency problem in English-Hindi SMT. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint inproceedings on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 800–808, Suntec, Singapore, August. Association for Computational Linguistics.

Rao, D., Bhattacharya, P., and Mamidi, R. (1998). Natural language generation for english to hindi human-aided machine translation.

Singh, A. K. and Ambati, B. R. (2010). An integrated digital tool for accessing language resources. In *LREC*.

Venkatapathy, S. and Bangalore, S. (2009). Discriminative Machine Translation Using Global Lexical Selection. *ACM Transactions on Asian Language Information Processing*, 8(2).

Venkatapathy, S., Sangal, R., Joshi, A., and Gali, K. (2010). A Discriminative Approach for Dependency Based Statistical Machine Translation. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 66–74, Beijing, China, August. Coling 2010 Organizing Committee.

Venkatapathy, S. (2008). NLP Tools Contest-2008: Machine Translation for English to Hindi . In *Proceedings of the International COnference on Natural Language Processing*.

Yeka, J. R., Kolachina, P., and Sharma, D. M. (2014). Benchmarking of english-hindi parallel corpora. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1812–1818, Reykjavik, Iceland, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1095.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 138–141, Stroudsburg, PA, USA. Association for Computational Linguistics.