Statistical Machine Translation Using Word Lattices

By:

Alok Nimrani (ID No. : 11CEUOG054)

**A research project submitted
in
partial fulfillment of the requirements
for the degree of**

**BACHELOR OF TECHNOLOGY
in
Computer Engineering**

Internal Guide
*Dr. C. K. Bhensdadia,
Head,
Department of Computer Engineering,
Dharmsinh Desai University*

External Guide
*Dr. Pushpak Bhattacharyya,
Vijay and Sita Vashee Chair Professor,
Department of Computer Science and Engineering,
Indian Institute of Technology Bombay*

**Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University**

April 2015

CERTIFICATE

This is to certify that the research project titled

Statistical Machine Translation Using Word Lattices

is the bonafide work of

Alok Nimrani (ID No.: 11CEUOG054)

carried out in the partial fulfillment of the degree of Bachelor of Technology in Computer Engineering at Dharmsinh Desai University in the academic session December 2014 to April 2015.

Dr. C. K. Bhensdadia, Head, Department of Computer Engineering, Dharmsinh Desai University
(Internal Guide)

Dr. Pushpak Bhattacharyya, Vijay and Sita Vashee Chair Professor, Department of Computer Science and Engineering, Indian Institute of Technology Bombay
(External Guide)

Dr. C. K. Bhensdadia, Department of Computer Engineering, Dharmsinh Desai University
(Head of Department)

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

April 2015

Acknowledgement

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

It gives me an immense pleasure submitting this report towards the partial fulfillment of my academics. Success in any mission cannot be achieved single heartedly. It is the team effort that sails the ship to shore.

Firstly, I am grateful to my external guide Dr. Pushpak Bhattacharyya for allowing me to do my internship under him at IIT Bombay and also for providing the guidance and the inspiration to strive for perfection. I am also thankful to Anoop Kunchukuttan and Girishkumar Ponkiya, pursuing their Ph.D under Dr. Pushpak, for guiding me at every step and sharing their knowledge and experiences.

I also express my sincere thanks to my internal guide and the Head of the Department, Dr. C. K. Bhensdadia**,** for giving me such a life time opportunity to carry out my internship in such a prestigious institute and also for his insightful support, kind co-operation and enthusiastic encouragement.

I would also like to take this opportunity to say thank you to the other members associated with CFILT (Center for Indian Language Technology) at IIT-B who were an integral part of my internship.

Abstract

Word Lattice is a directed acyclic graph used to provide multiple inputs to the decoder in order to resolve the ambiguity. In case of morphologically rich languages, the translation is shown to improve when carried out at lemma level by simplifying the rich morphology and representing word forms as stem + morphemes. But morphology is often ambiguous and so the optimal morphological segmentation of a word may not be known. The work reported here revolves around the investigations made to study the impact of word lattice input on the results of Statistical Machine Translation. Experiments were performed for two Indian language pairs – Gujarati (Gu) and Hindi (Hi) and Malayalam (Ml) and Hindi – where the source languages Gu and Ml are morphologically rich. These experiments, on analysis, reveal that lattice inputs don't produce the best translation when compared with the translation carried out using only the statistically best segmentation of the words.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Natural Language Processing (NLP)

Language, ability to speak & write and communicate is one of the most fundamental aspects of human behaviour. As the study of human-languages developed, the concept of communicating with non-human devices was investigated. Language is a hallmark of intelligence, and endowing computers with the ability to analyze and generate language - a field of research known as Natural Language Processing (NLP) - has been the dream of Artificial Intelligence (AI).

Natural Language Processing is the task of analyzing and generating by computers, languages that humans speak, read and write [1]. Natural language understanding – a subtopic of NLP – is sometimes referred to as an 'AI-Complete' problem, because natural-language recognition seems to require extensive knowledge about the outside world that humans possess and also the ability to manipulate it [16]. NLP has significant overlap with the field of computational linguistics, and is often considered a sub-field of artificial intelligence.

NLP is concerned with questions involving three dimensions: language, algorithm and problem as indicated in Figure 1. The language axis represents different natural languages and linguistics. The problem axis mentions different NLP tasks like morphology, part of speech tagging etc. The algorithm axis depicts mechanisms like HMM (Hidden Markov Model), MEMM (maximum entropy Markov model), CRF etc. for solving problems.

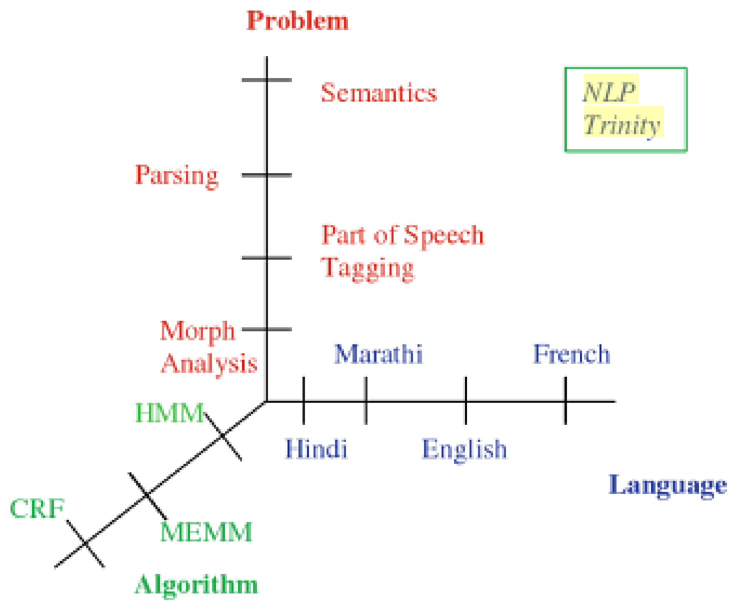


Figure 1: NLP Trinity

1.1.1 Challenges in NLP

A major challenge in NLP is ambiguity. Ambiguity in language is what makes NLP challenging. Few of the other challenges include:

a. Complexity of sentence structure

b. Sentence boundary detection (example "… in the US. Govt. …")
c. Ambiguity in words (polysemy problem)
d. Syntactic ambiguity (ambiguity in syntax representation structures)
e. Meaning is context sensitive:
    - depends on the location (example: play <song> upstairs)
    - depends on the people present (example: how far is it? (miles/km))
f. Recognizing named entities
g. Humour, sarcasm, spelling mistakes

1.1.2 Layers of NLP

NLP consists of seven layers:

a. Phonology and Phonetics (processing of sound)
b. Morphology (processing of word forms)
c. Lexicon/Lexical Analysis (Storage of words and associated knowledge)
d. Parsing/Syntax Analysis (Processing of structure)
e. Semantics (Processing of meaning)
f. Pragmatics (Processing of user intention, modeling etc.)
g. Discourse (Processing of connected text)

Challenges associated with each layer:

- **Phonology and Phonetics**: Utterances are processed here. Apart from many challenges due to noise, two common problems are homophony and word boundary recognition.
    - Homophony arises when two words sound the same, though their meanings are widely different, example - bank (embankment of a water body) and bank (an institution where financial transactions are held). Near homophony is more common and causes difficulty, especially in rapid speech, example - fox and folks.
    - Similarly, word boundary detection is a challenge in case of rapid speech, example - आजायेंगे - aajaayenge (आज आयेंगे (aaj aayenge) – will come today or आ जायेंगे (aa jaayenge) – will come) - The string can be broken in two ways as shown above with two completely different meanings.

- **Morphology**: Words form from the root words or lexemes through the processes of inflexion, derivation, etc [1]. Languages differ in their morphological richness. Dravidian languages are morphologically rich languages. Chinese and English are examples of relatively simpler morphology. The main ambiguity at the level of morphology arises from the choices available in breaking the word into stem and suffix i.e. segmenting the words into morphemes. For example, the word 'unfoldable' can be segmented in two ways: unfold + able or un + foldable [2].

- **Lexicon/Lexical Analysis**: Words are stored in the lexicon with a variety of information that facilitates the further stages of NLP, like question answering, information extraction etc. For example, the word dog might be stored with information like:

  POS (Noun)

Semantic Tag (Animate, 4-legged)
Morphology (takes ‘s’ in plural)

Words typically have multiple meanings even in the same part of speech. Dog, for example, means both - an animal and a very detestable person.

In case of word sense ambiguity, two situations are distinguished - homography and polysemy. Homography like homophony results from foreign word borrowing. Two words are homographic if they are spelt the same, though their meanings are different. Bank is an example of homography. Polysemy arises when a word has multiple meanings, example - falling of a tree and falling of a kingdom.

Word sense or lexical disambiguation refers to the identification of the meaning of an ambiguous word from clues in the context. For example, in ‘I will withdraw some money from the bank’ the most likely sense of bank is the financial institution sense.

- **Parsing/Syntax Analysis**: Parsing or syntactic processing refers to uncovering the hierarchical structure behind a linear sequence of words. For example, the noun phrase (NP) flight from Mumbai to Delhi via Jaipur on Air India has the following structure:

```
[NP 4
        [NP 3
                [NP 2
                        [NP 1 [NN flight]]
                                [PP 1 [P from][NP [NNP Mumbai]]]
                        ]
                        [PP 2 [P to] [NP [NNP Delhi]]]
                ]
                [PP 3 [P via][NP [NNP Jaipur]]]
        ]
        [PP 4 [P on][NP [NNP Air-India]]]
]
```

The above is called a bracketed structure. The above structure shows that flight is a noun (NN) which is modified by the attached preposition phrase (PP 1) from Mumbai to form NP 1. NP 1 is modified by the attached PP 2 to Delhi to form NP 2. NP 2 is modified by the attached PP 3 via Nagpur to form NP 3 which in turn is modified by the attached PP 4 on Air-India to form NP 4. Such bracketed structures are created by a Grammar of the language. In the above example, PP-->P NP is a grammar rule expressing the fact that a preposition phrase is composed of a preposition and a noun phrase.

Now, parsing too faces the challenge of ambiguity called structural ambiguity. Structural ambiguity is of two kinds: scope ambiguity and attachment ambiguity.

◦ Scope Ambiguity:

  Example: Old men and women were taken to safe locations.
  The scope of the adjective 'old' is ambiguous. That is, is the structure (old men and women) or ((old men) and women)?

- ◦ Attachment Ambiguity: Attachment ambiguity arises from uncertainty of attaching a phrase or clause to a part of a sentence. Here are some examples:

  I saw the boy with a telescope.

  It is not clear who has the telescope, I or the boy? In the former case, the preposition phrase with a telescope attaches with the verb 'saw' with the instrumental case. In the latter, the PP attaches to 'the boy' as a modifier. PP-attachment is a classical problem in NLP [1]. The general problem can be stated as follows:

  Given the structure:
  V-NP1-P-NP2
  Where does NP2 attach, V or NP1?

  The problem is attempted to be solved by both rule based and machine learning based approaches. In the former, the properties of V, head (NP1) and head (NP2) are used to formulate rules for attachment. For example, in 'I saw the boy with a pony tail', the PP attaches to the boy, since pony tail does not possess the property of instrumentality and saw with pony tail does not make sense. Such properties - called selectional preferences come from lexical knowledge bases like wordnet and verbnet [1]. Formulating such rules for deciding attachment is human labour intensive, and ensuring correctness and completeness of the rule base is also difficult.

  The alternative approach to solving the attachment problem is machine learning (ML) based. Here one creates annotated corpora of the form:

    See the boy with pony tail: V
    See the tiger with telescope: N

  and then try to teach a machine the conditions for the two kinds of attachment. It must be clear that ambiguity of attachment arises from the dual role of prepositions, and assigning case to nouns with respect to a verb and for modifying a noun phrase. In the example, with can assign instrument case to telescope or specify a particular boy having a telescope.

- **Semantics**: After word forms and structure have been detected, sentence processing devotes itself to meaning extraction. While the meaning of *meaning* is debatable, there is a general agreement that at the stage of semantic processing, the sentence needs to be represented in one of the unambiguous forms like semantic nets. Semantics extraction faces all the challenges arising out of ambiguities of semantic roles or relations. For example:

  Visiting aunts can be trying

  Here, are the aunts visitors (agent role) or are they being visited (object role)?

- **Pragmatics**: This is one of the hardest problems in NLP and has seen very little progress. The problem involves processing user intention, sentiments, belief world, etc. - all of which are highly complex tasks. The following humorous exchange illustrates the nature of the problem:

Tourist (checking out of the hotel): Waiter, go upstairs to my room and see if my sandals are there; do not be late; I have to catch the train in 15 minutes.
Waiter (running upstairs and coming back panting): Yes sir, they are there.

Clearly, the waiter is falling short of the expectation of the tourist, since he does not understand the pragmatics of the situation. But 'are my sandals there' is an ambiguous question if user intent and the situation specificity are considered.

- **Discourse**: This is the task of processing connected sentences. Example:

  Sentence-1: John was coming dejected from the school
  (who is John: most likely a student?)

  Sentence-2: He could not control the class
  (who is John now? Most likely the teacher?)

  Sentence-3: Teacher should not have made him responsible
  (who is John now? Most likely a student again, albeit a special student- the monitor?)

  Sentence-4: After all he is just a janitor
  (all previous hypotheses are thrown away!)

1.1.3 Applications of NLP

- Speech recognition
- Machine translation
- Information retrieval
- Automatic summarization
- Text-to-speech

1.2 Introduction to Machine Translation

Machine Translation (MT) is the field of Natural Language Processing (NLP) and Artificial Intelligence (AI) which focuses on automatic translation of a sentence from one language to another. Nowadays, there is abundant information available on the web published in a variety of languages. MT plays an important role in making this information available to all people in their own language. The content is summarized from the book: Statistical Machine Translation by Philip Koehn (2009)

1.2.1 What is Machine Translation?

Machine translation, sometimes referred to by the abbreviation MT is a sub-field of computational linguistics that investigates the use of software to translate text or speech from one natural language to another. On a basic level, MT performs simple substitution of words in one natural language for words in another, but that alone usually cannot produce a good translation of a text because recognition of whole phrases and their closest counterparts in the target language is needed.

1.2.2 History of Machine Translation

The history of Machine Translation is one of the greatest hopes and disappointments. Efforts for building efficient MT system had started in the days of Second World War to decode the German Language codes. Few of the terminologies set in those days for Machine Translation are still valid today. For example, we still talk about "decoding" a foreign language.

Machine Translation activities that were going on with great enthusiasm stopped due to ALPAC Report in 1966. ALPAC is Automatic Language Processing Advisory Committee that studied the then progress of translation process. The report suggested that the cost of translation is much higher than actual human translation. Thus, committee suggested that funding should rather be diverted towards linguistic research and improvement of human translators. This stopped development in the area of Machine Translation for almost a decade.

Despite lack of research efforts after ALPAC Report, commercial translation systems like Meteo were developed in 1976. SYSTRAN was founded in 1968. It gained commercial importance in 1976. In 1980s, Logos and METAL came into existence. During 1990s, advent of desktop computer systems helped in the development of computer aided translation systems for human translators. Interlingua based systems were of primary focus in 1980s and 1990s. These systems use Interlingua, knowledge representation independent of any language, for translation. Example of such systems is CATALYST by CMU. These systems faced a major problem of representing meanings in formal ways of Interlingua.

Due to difficulties in formalizing activities of translation, another approach of learning translations from past translations came forward. These translation systems are broadly referred to as Example Based Translation Systems (EBMT). In late 1980s, success of statistical approaches in speech recognition motivated the idea of statistical machine translation. IBM Research Lab came forward with statistical machine translation. Although, the statistical machine translation was introduced in 1990s, it gathered focus only around the year 2000. This was due to increased interest of funding agencies in statistical approaches. Events of September 11, 2001 also drove the interest in automatic translation of foreign languages, especially Arabic. Widespread use of very powerful computers with huge data storage and large amount of digital text available acted as catalyst in emergence of statistical machine translation.

1.3 Outline of the Report

The report starts with introduction to Natural Language Processing (NLP) and introduction to Machine Translation (MT). Chapter 2 describes MT in detail – its applications, the approaches to MT namely Rule Based MT, Statistical MT (SMT) and Example based MT. SMT is further described in detail along with the stages involved in SMT. The different evaluation metrics have also been explained. The chapter then focuses on Moses decoder – the introduction to Moses, the comparison between Moses and its predecessor Pharaoh and performing experiments using Moses.

Chapter 3 describes about Morphology – what is morphology, its history, and its various models. It also describes morphemes (smallest grammatical unit in a language), the sparse data problem in case of morphologically rich languages and the ways for simplifying rich morphology. Chapter 4 describes unsupervised approach to word segmentation used in simplifying rich morphology. The chapter describes why unsupervised approach is used and also describes Morfessor tool which performs unsupervised segmentation, its features and the algorithms that it uses.

Chapter 5 discusses about Word Lattices – what a word lattice is, how it is useful, the representation of a lattice for decoding and the command line options to indicate a lattice input.

Chapter 6 describes the experiments performed – the experimental setup, the results and evaluation of those results. Chapter 7 describes the summary of this report and the conclusions made from the experiments. It also discusses the future work to be carried out on word lattice and in the end Chapter 8 specifies the references.

Chapter 2

Machine Translation

2.1 Applications of Machine Translation

The fundamental goal of machine translation is FAHQMT, which stands for fully-automatic high-quality machine translation. Due to complexities involved in languages and translation techniques, this goal has been reached only for domain specific applications. Despite their inherent limitations in providing fully automatic, high quality machine translation, MT programs are used around the world. Few of the applications are:

- With the recent focus on terrorism, the military sources in the United States are interested in translation and processing of languages such as Arabic.

- The notable rise of social networking has led to the use of machine translation software in utilities such as Facebook, Skype, etc - allowing users speaking different languages to communicate with each other.

- The quality of machine translation has now been improved to such levels that its application in online collaboration and in the medical field are being investigated.

- Machine translation applications have also been released for most mobile devices, including mobile telephones; pocket PCs, PDAs, etc. Due to their portability, such instruments have come to be designated as mobile translation tools enabling mobile business networking.

- Instead of using MT system as a Post-editing approach, human translators are more comfortable with using it as a tool which assists in the task of translation. This leads to creation of interactive environment for translators.

2.2 Approaches to Machine Translation

The human translation process may be described as:

1. Decoding the meaning of the source text; and
2. Re-encoding this meaning in the target language.

To decode the meaning of the source text in its entirety, the translator must interpret and analyze all the features of the text, a process that requires in-depth knowledge of the grammar, semantics, etc., of the source language. The translator needs the same in-depth knowledge to re-encode the meaning in the target language. Here lies the challenge in machine translation: how to program a computer that will "understand" a text as a person does, and that will "create" a new text in the target language that "sounds" as if it has been written by a person.

This problem may be approached in a number of ways, through the evolution of which accuracy has improved.

2.2.1 Rule Based MT

Machine translation can use a method based on linguistic rules, which means that words will be translated in a linguistic way – the most suitable words of the target language will replace the ones in the source language. The rule-based machine translation paradigm includes transfer-based machine translation, interlingual machine translation and dictionary-based machine translation paradigms. Generally, rule-based methods parse a text, usually creating an intermediary, symbolic representation, from which the text in the target language is generated. According to the nature of the intermediary representation, an approach is described as interlingual machine translation or transfer based machine translation. This type of translation is used mostly in the creation of dictionaries.

The Vauquois triangle was used in the linguistic rule-based era of machine translation to describe the complexity/sophistication of approaches to machine translation, and also the evolution of those approaches. The first approach used was a direct lexical conversion between languages. According to the model, each step up the triangle required greater effort in source language analysis and target language generation, but reduced the effort involved in conversion between languages. The pinnacle and ideal of the field is a complete analysis of each sentence into an "interlingua" - a schema capable of representing all meaning expressible in any language in language-independent form. The currently best performing SMT systems are crawling at the bottom.

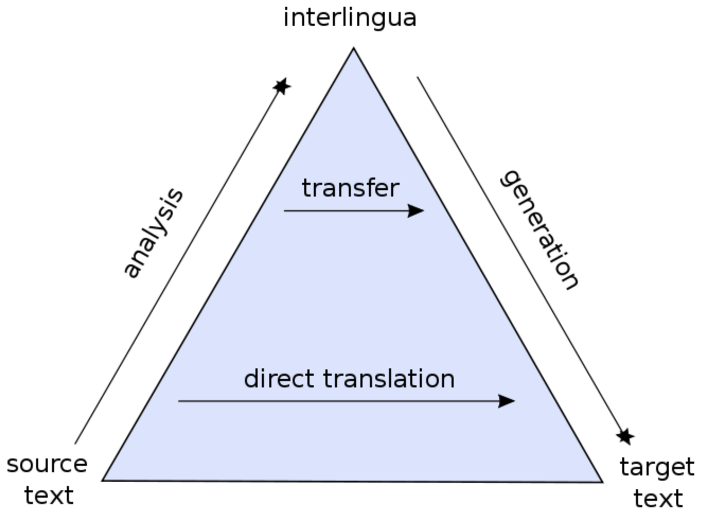


Figure 2: Vauquois Triangle

Unlike other methods, RBMT involves more information about the linguistics of the source and target languages, using the morphological and syntactic rules and semantic analysis of both languages. The basic approach involves linking the structure of the input sentence with the structure of the output sentence using a parser and an analyzer for the source language, a generator for the target language, and a transfer lexicon for the actual translation. RBMT's biggest downfall is that everything must be done explicitly: orthographical variation and erroneous input must be made part of the source language analyzer in order to cope with it, and lexical selection rules must be written for all instances of ambiguity. RBMT is also referred to as Direct MT and it is based on the idea of 'analysis'.

- **Interlingual MT**: In this approach, the source language is transformed into an interlingual language, i.e. a "language neutral" representation that is independent of any language. The target language is then generated out of the Interlingua. One of the major advantages of this system is that the Interlingua becomes more valuable as the number of target languages it can be turned into increases. The obvious disadvantage is that the definition of an Interlingua is difficult and maybe even impossible for a wider domain.

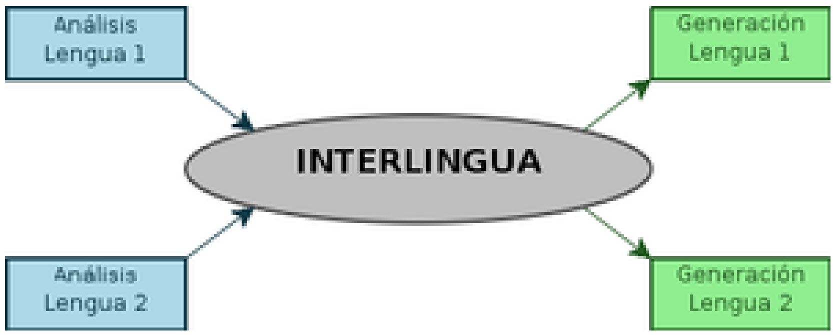


Figure 3: Interlingual MT

- **Transfer Based MT**: In the transfer approach the source language is transformed into an abstract, less language-specific representation. Linguistic rules which are specific to the language pair then transform the source language representation into an abstract target language representation and from this the target sentence is generated.

In dictionary based translation, dictionary entries are used.

2.2.2 Statistical MT

Statistical machine translation tries to generate translations using statistical methods based on analysis of bilingual text corpora. It is based on 'alignment'. The first statistical machine translation software was CANDIDE from IBM. Google Translate and similar statistical translation programs work by detecting patterns in hundreds of millions of documents that have previously been translated by humans and making intelligent guesses based on the findings. Generally, the more human-translated documents available in a given language, the more likely it is that the translation will be of good quality. SMT's biggest downfall includes it being dependent upon huge amounts of parallel texts, its problems with morphologically rich languages (especially with translating *into* such languages), and its inability to correct singleton errors.

2.2.3 Example Based MT

Example-based machine translation (EBMT) approach was proposed by Makoto Nagao in 1984. Example-based machine translation is based on the idea of 'analogy'. In this approach, the corpus that is used is one that contains texts that have already been translated. Given a sentence that is to be translated, sentences from this corpus are selected that contain similar sub-sentential components. The similar sentences are then used to translate the sub-sentential components of the original sentence into the target language, and these phrases are put together to form a complete translation.

2.3 Statistical Machine Translation (SMT)

SMT makes use of statistical models. By making use of statistical models, SMT systems create many alternatives, called as hypothesis. Each of these hypotheses is scored and best scoring hypothesis is chosen as the resultant translation. The process of statistical machine translation can be best explained from the below figure:

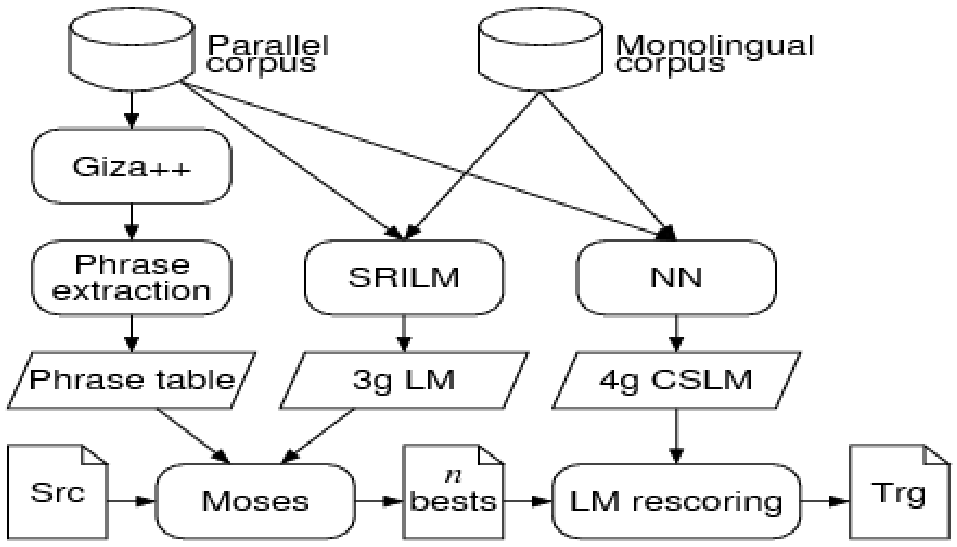


Figure 4: SMT Process

2.3.1 Benefits of SMT

- Better use of resources
    - There is a great deal of natural language in machine-readable format.
    - Generally, SMT systems are not tailored to any specific pair of languages.
    - Rule-based translation systems require the manual development of linguistic rules, which can be costly, and which often do not generalize to other languages.
- More natural translations
    - Rule-based translation systems are likely to result in literal translation. While it appears that SMT should avoid this problem and result in natural translations, this is counterbalanced by the fact that using statistical matching to translate rather than a dictionary/grammar rules approach can often result in text that include apparently nonsensical and obvious errors.

2.3.2 Shortcomings of SMT

- Corpus creation can be costly for users with limited resources.
- The results are unexpected. Superficial fluency can be deceiving.
- Statistical machine translation does not work well between languages that have significantly different word orders (e.g. Japanese and European languages).
- The benefits are overemphasized for European languages.
- For new domains or uncommon language pairs, extensive parallel corpora are often hard to come by.

2.3.3 Types of SMT

2.3.3.1 Word Based Translation

In word-based translation, the fundamental unit of translation is a word in some natural language. Typically, the number of words in translated sentences is different, because of compound words, morphology and idioms. The ratio of the lengths of sequences of translated words is called fertility, which tells how many foreign words each native word produces. Word based translation can't translate between languages with different fertility. An example of word based translation is GIZA++. Due to less efficiency and difficulty in handling different fertility, word based translation is not used nowadays.

2.3.3.2 Phrase Based Translation

Phrase based statistical machine translation approach was proposed due to limitations in word based approaches. Phrase-based translation, where blocks of one or more words-instead of individual words-are modeled as translation units, has been a very popular approach (Och, 1999; Marcu and Wong, 2002; Koehn et al., 2003; Och and Ney, 2004; Kumar et al.. 2006). Models based on translation of words do not perform well, since words are not the best candidates for the task of translation. The word based models often break in the cases where one source word is translated into multiple target language words, or vice versa. Here, the aim is to reduce the restrictions of word-based translation by translating whole sequences of words, where the lengths may differ. The sequences of words are called blocks or phrases, but typically are not linguistic phrases (such as verb phrase, noun phrase, etc), but phrasemes (a multi-word expression) found using statistical methods from corpora. It has been shown that restricting the phrases to linguistic phrases decreases the quality of translation. It uses GIZA++ for alignment purposes. The input foreign language is segmented into phrases. Each phrase is translated into multiple target (English) phrases and then the phrase with the highest probability is taken.

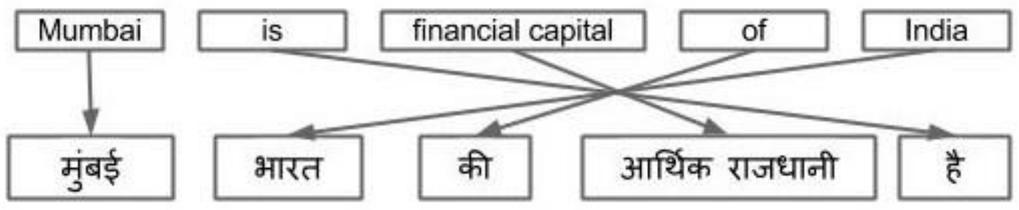


Figure 5: Phrase Based Translation

2.3.3.3 Syntax Based Translation

Syntax-based translation is based on the idea of translating syntactic units, rather than single words or strings of words (as in phrase-based MT), i.e. (partial) parse tress of sentences/utterances. Example of this approach is synchronous context free grammar (SynCFG). It models the reordering of clauses that occurs when translating a sentence by correspondences between phrase-structure rules in the source and target languages. Rules in a SynCFG are superficially similar to CFG rules, except that they specify the structure of two phrases at the same time; one in the source language (the language being translated) and one in the target language.

2.3.4 Challenges

Problems that statistical machine translations have to deal with include:

- **Sentence Alignment:** In parallel corpora, single sentences in one language can be found translated into several sentences in the other and vice versa. Since sentences are not always mapped one-to-one, sentence alignment methods are needed. The parameters of statistical translation models are typically estimated from sentence-aligned parallel corpora. Proper alignments can result in better translations.
- **Different word orders**: Word orders in languages differ. Some classification can be done by naming the typical order of subject (S), verb (V) and object (O) in a sentence and one can talk, for instance, of SVO or OSV languages. There are also additional differences in word orders, for instance, where modifiers for nouns are located for example prefix or suffix, or where the same words are used as a question or a statement. For SMT, the machine translator can only manage small sequences of words i.e. either prefix or suffix and rarely both, and word order has to be thought of by the program designer. Attempts at solutions have included re-ordering models, where a distribution of location changes for each item of translation is guessed from aligned bi-text. Different location changes can be ranked with the help of the language model and the best can be selected.
- **Out of vocabulary words**: SMT systems typically store different word forms as separate symbols without any relation to each other and word forms or phrases that were not in the training data cannot be translated. This might be because of the lack of training data, changes in the human domain where the system is used, or differences in morphology.

2.3.5 Stages involved in SMT

Statistical Machine Translation is a three staged process involving: (a) training – to train the system to enable it for translating input sentences, (b) tuning – to fine tune the parameters used during training to improve the translation quality, and (c) decoding – to translate the given input sentence into the target sentence.

(a) **Training the system**: In order to enable the system for performing translations, first step is to make the system learn from the already translated sentences i.e. to train the system using the provided corpus. Training a phrase model requires a large parallel, sentence aligned corpus. This corpus is a set of sentence-level translations, in the same order, in the source and target language. The amount of training data needed for phrase-based SMT system ranges from tens of thousands to millions of sentences—in general, the more the better.

The training process creates phrase table which represents a one-to-one mapping between source and target phrases along with the probabilities. For each phrase, the phrase table indicates five probabilities: inverse phrase translation probability, inverse lexical weighting, direct phrase translation, direct lexical weighting and phrase penalty. For the phrase table creation, first the word alignment is established using GIZA++ toolkit and then the phrases consistent with this alignment are extracted. If the sentence length is more, there will be more phrases. One way to reduce the number of phrases is to restrict the sentence length. This can be achieved by pre-processing the corpus which includes tokenization – to prune additional spaces between words and cleaning – to restrict the sentence length. Once the system is modeled, a configuration file is created which states the default feature weights to be considered while decoding.

(b) **Tuning the system**: The key to good translation performance is having a good phrase translation table. During training, Moses assigns the scores to the translation hypothesis and the feature functions. These features are the probabilities of language model, translation model and reordering model. Tuning is the process of finding the optimal weights for this hypothesis which maximize the translation performance. The most important is the tuning of the model parameters. The probability cost that is assigned to a translation is a product of probability costs of four models:

- phrase translation table,
- language model,
- reordering model, and
- word penalty.

Each of these models contributes information over one aspect of the characteristics of a good translation:

- The phrase translation table ensures that the target phrases and the source phrases are good translations of each other.
- The language model ensures that the output is fluent.
- The distortion model allows for reordering of the input sentence, but at a cost: The more the reordering, the more expensive is the translation.
- The word penalty ensures that the translations do not get too long or too short.

Each of the components can be given a weight that sets its importance. Mathematically, the cost of translation is:

p(e|f) = ph(f|e)^weight_ph * LM(e)^weight_lm * D(e,f)^weight_d * W(e)^weight_w

The probability p(e|f) of the English translation e given the foreign input f is broken up into four models, phrase translation ph(f|e), language model LM(e), distortion model D(e,f), and word penalty W(e) = exp(length(e)). Each of the four models is weighted by a weight. Setting these weights to the right values can improve translation quality. And for this reason, tuning is performed as the final step in creating a translation system. The training set used for parameter tuning is called the **tuning set** and it is kept separate from the original training data for the estimation of translation and language models. In the initial iteration, the default weights are taken which are then changed heuristically in each iteration.

(c) **Decoding:** Once the system is tuned, decoding can be performed. The goal of decoding is to find the translation with the best score. Decoding is a NP-complete problem. Before translating an input sentence, first the translation table is look up to determine the applicable translation options. During decoding, the partial translations are stored as hypotheses which are then expanded by deciding on the next phrase translation and in the end, the hypothesis with minimum cost is chosen.

A decoder used popularly for translating the input sentences to target language is Moses decoder which is further explained in the next section.

Once decoding is complete, the final step is evaluation of the translated output. The most natural measure of the quality of a translation is to ask human judges to assess the adequacy (preservation of meaning) and fluency (includes grammatical correctness) of machine translation output by reporting how similar it is to a reference translation. But collecting human judgements is time consuming and expensive. Due to this, automatic evaluation metrics have been proposed for machine translation. The goal of these metrics is to emulate human judgments of translation quality. Typically, they compare the output of machine translation systems against human translations. Although, these metrics do not correlate perfectly with human judgements, they provide a fast and cheaper alternative. In NLP, precision and recall are common metrics. In case of machine translation, these metrics are based on word matches. While precision only considers the number of words that match, recall computes how many of the words that a system should generate are correct. However, the interest lies in both precision and recall.

Apart from this, several other evaluation metrics are:

- **Word Error Rate**: It takes word order into account.

- **BLUE (BiLingual Evaluation Understudy):** The currently most popular automatic evaluation metric is the BLEU metric. BLEU counts the number of n-gram matches between the hypothesis and the reference translation. By counting matches of more than just single words, it is sensitive to local orderings of words in the hypothesis, and should provide a better measure of translation fluency than a simple unigram count. Given the n-gram matches, n-gram precision is computed, i.e., the ratio of correct n-grams of a certain order *n* in relation to the total number of generated n-grams of that order. Because this n-gram count is a measure of precision and is scaled by the number of words in the hypothesis, the metric needs some measure of relative length so that it does not overly reward short system output. To this end, BLEU contains a brevity penalty that reduces the score for output that is shorter than the reference. This penalty is calculated over the full evaluation set, not at the sentence level.

- **METEOR (Metric for Evaluation of Translation using Explicit ORdering)**: One perceived flaw of BLEU is that it gives no credit to near matches i.e. use of synonyms. If a word closely related to reference word is there, BLUE doesn't consider the closeness. One way to credit near matches is to reduce the words to their stems and then evaluate. Other way is to consider the synonyms while evaluating. This is where METEOR is used. METEOR incorporates the use of stemming and synonyms by first matching the surface forms of the words, and then backing off to stems and finally semantic classes. However, though METEOR performs better for a single sentence, BLUE is generally used when a large number of sentences are to be evaluated.

  In this report, we have used BLUE score as our evaluation metric.

2.4 Moses Decoder

2.4.1 Introduction to Moses

Moses is an implementation of the statistical (or data-driven) approach to machine translation

(MT) [17]. This is the dominant approach in the field at the moment, and is employed by the online translation systems deployed by the likes of Google and Microsoft. In statistical machine translation (SMT), translation systems are trained on large quantities of parallel data (from which the systems learn how to translate small segments), as well as even larger quantities of monolingual data (from which the systems learn what the target language should look like). Parallel data is a collection of sentences in two different languages, which is sentence-aligned, in that each sentence in one language is matched with its corresponding translated sentence in the other language. It is also known as a bitext.

The training process in Moses takes in the parallel data and uses co-occurrences of words and segments (known as phrases) to infer translation correspondences between the two languages of interest. In phrase-based machine translation, these correspondences are simply between continuous sequences of words, whereas in hierarchical phrase-based machine translation or syntax-based translation, more structure is added to the correspondences. The extra structure used in these types of systems may or may not be derived from a linguistic analysis of the parallel data. Moses also implements extension of phrase-based machine translation know as factored translation which enables extra linguistic information such as part-of-speech, tense, etc to be added to phrase-based systems.

The training process is handled by the training pipeline component of Moses. There are two main components in Moses: the **training pipeline** and the **decoder**. There are also a variety of contributed tools and utilities. The training pipeline is really a collection of tools (mainly written in Perl, with some in C++) which take the raw data (parallel and monolingual) and turn it into a machine translation model. The decoder is a single C++ application which, given a trained machine translation model and a source sentence, will translate the source sentence into the target language.

The training pipeline is an implementation of various stages involved in producing a translation system from translation data. On the other hand, the job of the Moses decoder is to find the highest scoring sentence in the target language (according to the translation model) corresponding to a given source sentence. It is also possible for the decoder to output a ranked list of the translation candidates, and also to supply various types of information about how it came to its decision (for instance the phrase-phrase correspondences that it used).

Apart from providing an open-source toolkit for phrase based SMT, Moses also integrates confusion network and word lattice decoding, which allows the translation of ambiguous input. This enables, for instance, the tighter integration of speech recognition and machine translation. Instead of passing along the one-best output of the recognizer, a network of different word choices may be examined by the machine translation system.

2.4.2 Comparison between Moses and Pharaoh

Moses is a phrase based, open source decoder for SMT. The advances in phrase based translation and the limitations of word based translation motivated Moses. Prior to Moses, Pharaoh (a beam search decoder for phrase based SMT) was used. Just like the Pharaoh decoder, Moses uses a log-linear model, which combines several knowledge sources in translation decisions. Moses differs from Pharaoh by its representation of each input word as a factor as opposed to the word surface form only [3]. Factors can include additional information such as part-of-speech, class, morphology and allow the phrase-based model to incorporate richer linguistic information. Because of richer linguistic information in case of Moses, it achieves slightly higher performance on average than its predecessor Pharaoh. It also features all other capabilities of Pharaoh Decoder (Koehn 2004).

Moreover, Pharaoh is a closed-source toolkit for translation. The source code was not available and so the researchers were unable to work in their way and had to alter the input/output. This is where Moses got importance as the source code is openly available and changes can be made to it. It is also extensible with many LM implementations integrated.

2.4.3 Performing Experiments using Moses Decoder

Steps to be performed for a simple system training and decoding:

(The installation steps as well as commands to perform various actions can be followed on http://www.statmt.org/moses/?n=Moses.Baseline or http://www.statmt.org/moses_steps.html)

- Tools required:
  - GIZA++ for obtaining word alignments from parallel sentences which are used to extract phrase-phrase translations.
  - SRILM/IRSTLM for language model creation. Moses by default comes with KenLM. Language model is a statistical model built using monolingual data in the target language and used by the decoder to try to ensure the fluency of the output.

- As part of pre-processing the parallel corpora for preparing the data to be used while training the translation system, the bilingual text is:
  - Tokenized: This means that spaces have to be inserted between (e.g.) words and punctuation and multiple spaces are pruned to single space.
  - True-cased: The initial words in each sentence are converted to their most probable casing. This helps reduce data sparseness. This is not needed for Indian languages.
  - Cleaned: Long sentences and empty sentences are removed as they can cause problems with the training pipeline, and obviously mis-aligned sentences are removed. Generally, the sentence length is kept max 80 as beyond it noise can result in poor alignment.

- In order to ensure fluency of output, the language model is trained with the target monolingual corpus before training the translation system. Appropriate n-gram order value is also specified. Language models are concerned only with n-grams in the data, so sentence length doesn't impact training times as it does in GIZA++.

- The translation system is then trained by obtaining the word-alignments (using GIZA++), performing phrase extraction and scoring; creating lexicalized reordering tables and Moses configuration file.

- The next step is to tune the system. The tuning process goes through a default number of iterations. The weights become more and more accurate over the iterations starting with the default weights in the first iteration. The translation quality improves with each iteration.

- Once tuning is complete, the system is ready and can be used for translating source sentences into target sentences by running the Moses executable file and providing the log file, with -f option, which is the configuration file moses.ini. Different options such as -t for tracing the decoding can be provided based on the requirement.

- If during decoding, the system runs out of space or is slow then it is recommended to binarize the phrase tables as by doing so only the pairs needed for each sentence are read. It also speeds up loading of phrase tables. Binarizing the phrase tables encodes them.

- The final step is evaluation. Various metrics such as BLUE (Bilingual Evaluation Understudy), METEOR (Metric for Evaluation of Translation using Explicit ORdering) can be used to evaluate the quality of the translated sentence.

Chapter 3

Morphology

3.1 Introduction to Morphology

Morphology is the part of linguistics that deals with the study of words, their internal structure and partially their meanings. It refers to identification of a word stem from a full word form and also determines the affixes and their context. The process of providing grammatical information of a word given its suffix is termed morphological analysis.

English has very limited morphological variation of words. Nouns may be changed due to their count (singular vs. plural). Verbs are changed due to their tense (as in walk, walked, walking). Morphology creates various challenges for machine translation. One way to divide up words in a language is to take the view that they fall into two categories: **content words** that carry meaning by referring to real and abstract objects, actions, and properties (called nouns, verbs, adjectives, adverbs) and **function words** that inform us about the relationships between the content words on the other. The relationship between content words is also encoded in word order and morphology.

While some languages express the relationships between words mostly with function words, others use morphological variation of words. The more morphological variation a language exhibits, the larger its vocabulary of surface forms. One solution would be simply to acquire large amounts of training data, so that all surface forms occur frequently enough. However, the reality of limited training data puts morphologically richer languages at a disadvantage. Moreover, translating into morphologically rich languages often requires additional information for choosing the correct word form that may not be readily available in the input language.

3.1.1 History of Morphology

The history of morphological analysis dates back to the ancient Indian linguist Panini who formulated the 3,959 rules of Sanskrit morphology by using a constituency grammar. The Greco-Roman grammatical tradition also engaged in morphological analysis.

3.1.2 Models of Morphology

There are three principal approaches to morphology:

- Morpheme-based morphology, which makes use of an item and arrangement approach.
- Lexeme-based morphology, which normally makes use of an item and process approach.
- Word-based morphology, which normally makes use of a word and paradigm approach.

3.1.2.1 Morpheme-based Morphology

In morpheme-based morphology, word forms are analyzed as arrangements of morphemes. A morpheme is defined as the minimal meaningful unit of a language. In a word such as *independently*, the morphemes are said to be *in-*, *depend*, *-ent*, and *ly*; *depend* is the root and the other morphemes are, in this case, derivational affixes. In words such as *dogs*, *dog* is the root and the *-s* is an inflectional morpheme. In its simplest and most naive form, this way of analyzing word forms, called "item-and-arrangement", treats words as if they were made of morphemes put after each other.

3.1.2.2 Lexeme-based Morphology

Lexeme-based morphology usually takes what is called an "item-and-process" approach. Instead of analyzing a word form as a set of morphemes arranged in sequence, a word form is said to be the result of applying rules that alter a word-form or stem in order to produce a new one. An inflectional rule takes a stem, changes it as is required by the rule, and outputs a word form; a derivational rule takes a stem, changes it as per its own requirements, and outputs a derived stem.

3.1.2.3 Word-based Morphology

Word-based morphology is (usually) a word and paradigm approach. This theory takes paradigms as a central notion. Instead of stating rules to combine morphemes into word forms, or to generate word forms from stems, word-based morphology states generalizations that hold between the forms of inflectional paradigms. The major point behind this approach is that many such generalizations are hard to state with either of the other approaches.

3.2 Morphemes

In linguistics, a morpheme is the smallest grammatical unit in a language. In other words, it is the smallest meaningful unit of a language. A morpheme is not identical to a word, and the principal difference between the two is that a morpheme may or may not stand alone, whereas a word, by definition, is freestanding. When it stands by itself, it is considered a root because it has a meaning of its own (e.g. the morpheme *cat*) and when it depends on another morpheme to express an idea, it is an affix because it has a grammatical function (e.g. the *–s* in *cats* to specify that it is plural). Every word comprises one or more morphemes. The more combinations a morpheme is found in, the more productive it is said to be.

3.2.1 Classification of Morphemes

Every morpheme can be classified as either free or bound. These categories are mutually exclusive, and as such, a given morpheme will belong to exactly one of them.

- Free morphemes can function independently as words (e.g. *town*, *dog*) and can appear with other lexemes (e.g. *town hall, doghouse*).
- Bound morphemes appear only as parts of words, always in conjunction with a root and sometimes with other bound morphemes. For example, *un-* appears only accompanied by other morphemes to form a word. Most bound morphemes in English are affixes that are attached to word stem, particularly prefixes and suffixes, examples of suffixes are: tion, ation, ible, ing, etc.

Bound morphemes can be further classified as derivational or inflectional.

- Derivational morphemes, when combined with a root, change either the semantic meaning or part of speech of the affected word. For example, in the word *happiness*, the addition of the bound morpheme *-ness* to the root *happy* changes the word from an adjective (*happy*) to a noun (*happiness*). In the word *unkind, un-* functions as a derivational morpheme, for it inverts the meaning of the word formed by the root *kind*.

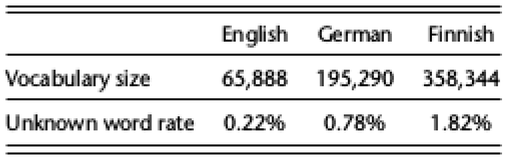
- Inflectional morphemes modify a verb's tense or a noun's number without affecting the word's meaning or class. Examples of applying inflectional morphemes to words are adding *-s* to the root *dog* to form *dogs* and adding *-ed* to *wait* to form *waited*.

Allomorphs, content morphemes and function morphemes are other types of morphemes.

3.3 Sparse Data Problem

Languages differ in morphological complexity. English morphology is limited mainly to verb form changes due to tense, and noun form changes due to count. Other languages such as German or Finnish have a much richer morphology. Rich morphology implies that for each lemma (for instance, house), many word forms exist (house, houses). This leads to **sparse data problems** for statistical machine translation [5]. One of the short-comings of traditional surface word approach in SMT is poor handling of morphology. Each word form is treated as token in itself. This means that the model treats, for example, the word ‘house’ independent of the word ‘houses’. Any instance of ‘house’ in training data doesn't add knowledge for the translation of houses. So, while the translation of ‘house’ may be known to the system, ‘houses’ will be left untranslated. This problem is much more severe in case of morphologically rich languages like German or the Indian languages like Gujarati, Dravidian languages, etc.

Figure 6 illustrates the effect of rich morphology on vocabulary sizes and out-of-vocabulary (OOV) rates.



	English	German	Finnish
Vocabulary size	65,888	195,290	358,344
Unknown word rate	0.22%	0.78%	1.82%

Figure 6: Vocabulary sizes and OOV rates
(Referenced from Statistical Machine Translation by Philip Koehn)

The sparse data problem can be resolved in several ways. One way is to model the translation between morphologically rich languages at the level of lemmas and thus, pool the evidence for different word forms that derive from a common lemma. In this case, the lemma and morphological information can be translated separately and combined at target side to obtain translated surface word forms. For example, consider the following English past tense words:

talked → talk (stem) + ed (morpheme)
produced → produce (stem) + d (morpheme)

Here, both the morphemes 'ed' and 'd' play the same role of indicating the past tense. Hence, one may think of representing them not by such change patterns (attaching 'ed' to the stem acts as a pattern) but by their syntactic roles (+PAST).

But morphology is often ambiguous. A morpheme may have several different meanings. For example, the German adjective change pattern '+e' indicates either (a) singular, indefinite, nominative, (b) singular, definite, female, nominative or dative, (c) accusative, plural, or (d) indefinite, nominative, plural. Similarly, in Gujarati, the postposition એ *(e)* is either used as an ergative marker in verbs while representing third person future tense, or a general locative, specifying senses such as "at", "during", etc and in case termination for masculine and neuter genders among nouns, or a simple postpositional addition for unmarked nouns or marked feminine. For these reasons, representing morphemes by their syntactic roles is not appropriate and so they are generally represented as change patterns only.

3.4 Simplifying Rich Morphology

Often, the input language has a much richer morphology, as is the case with German, Turkish, and Arabic when translated to English or in case of Indian languages, Gujarati and Malayalam when translated to Hindi. The additional word forms on the input side create sparse data problems that make the translation of rare or unseen word forms difficult. As stated above in section 3.3, one way to handle this is to split each word into its stem and morpheme. As an example, consider the following Gujarati sentences and its English translation:

તમારા$_{(1)}$ ઘરમાં$_{(2)}$
(tamaara gharma)
In your$_{(1)}$ home$_{(2)}$

Word alignments are indicated inside brackets. Many of the English function words have no equivalent in Gujarati such as 'in'. Their role is realized as morphological suffixes – the word માં which is referred to as a case marker. A morphological analyzer may give the following analysis:

તમારા$_{(1)}$ ઘર$_{(2)}$ + માં$_{(3)}$

At this level of analysis, we are able to establish alignments to English function words and morphemes:

તમારા$_{(1)}$	ઘર$_{(2)}$	માં$_{(3)}$
your	home	in

which on reordering using language model gives proper translated output – in your home.

This example suggests that languages such as Gujarati have morphemes (generally the case markers) that we may want to treat like independent words when translating into English or some other language. Similarly, the translation of other morphologically rich languages can be handled.

Chapter 4

Unsupervised Word Segmentation

4.1 Why Unsupervised approach to Segmentation

Unsupervised morphological word segmentation is attractive, because in every language there are virtually unlimited supplies of text, but very few labeled resources.

Since a word form can be expressed as a combination of different morphemes, automated morphological analysis would be beneficial for many natural language applications dealing with large vocabularies, such as speech recognition and machine translation. A morphological analyzer is a program to perform automated morphological analysis. But there exist morphological analyzers designed by experts for some Indian languages only. Moreover, expert knowledge and labour are expensive and analyzers need to be built separately for each language, and also to be updated on a continuous basis in order to cope with language change (mainly the emergence of new words and their inflections).

For this reason, as an alternative to the hand-made systems, there exist algorithms that work in an unsupervised manner and autonomously discover morpheme segmentations for the words in un-annotated text corpora and Morfessor is a general model, in this category, for the unsupervised induction of a simple morphology from raw text data [7]. Moreover, Morfessor is language independent. It has been designed to cope specially with agglutinative languages and where the number of morphemes per word can vary much and is not known in advance. This distinguishes Morfessor from other resembling models, e.g., Goldsmith (2001), which assume that words consist of one stem possibly followed by a suffix and possibly preceded by a prefix [7].

4.2 Introduction to Morfessor

Morfessor is an unsupervised data-driven method for the segmentation of words into morpheme-like units [6]. Morfessor is a tool for finding the morphological segmentations of words. Given the training corpus which is a list of words, a vocabulary of a language with one word per line, Morfessor trains the system and outputs a model consisting of all possible morphemes obtained from the training corpus based on the training parameters and the training algorithm used. The model consists of morphs. A set of properties is stored for each morph - the frequency, the string and the probability. This model can then be used to segment input sentences.

4.3 Working of Morfessor

Just like for machine learning methods, there are two phases when using Morfessor: *training* and *decoding*. Input for the training phase is a set of compounds and output is the model parameters. The performance of this trained model is evaluated by decoding the test data to obtain the morphs.

The training algorithm of Morfessor Baseline can be characterized as greedy and local search. It starts with an initial lexicon, which usually consists of all the compound forms seen in the training data. Then it selects one compound at a time and tries simultaneously to find the optimal segmentation for the compound and the optimal lexicon given the new segmentation and the segmentations of all the other known compounds. The training is normally done as a batch job. For decoding - finding the optimal segmentations for new compound forms without changing the model parameters - Morfessor Baseline applies a variation of the Viterbi algorithm. The Viterbi algorithm can be used for training too, but generally the final cost of segmentation using Viterbi is more as compared to the default recursive algorithm and so it is not as useful as recursive.

As an example of the output generated by Morfessor, consider the following compounds:

અંક (ank)

અંકગણિત (ank ganit)

અંકગણિતથી (ank ganit thi)

અંકગણિતનું (ank ganit nu)

When these compounds are provided as input to Morfessor, the trained model obtained is as follows:

1 અંક

1 અંક + ગણિત

1 અંક + ગણિત + થી

1 અંક + ગણિત + નું

Such a trained model is used while decoding the test data to generate the morphological segmentation of each word in the test data. The morphs generated in the test data have a negative log probability associated with them.

4.4 Features of Morfessor

Morfessor has various training features:

- **Batch Training**: In batch training, each epoch consists of iteration over the full training corpus. Epochs are repeated until the model cost is converged. All training data needed in the training needs to be loaded before the training starts.

- **Online Training**: In online training, the model is updated while the data is being added. This allows for rapid testing and prototyping. All data is only processed once, hence it is advisable to run batch training afterwards. The size of an epoch is a fixed and predefined number of compounds processed. The only use of an epoch for online training is to select the best annotations in semi-supervised training.

- **Recursive Training**: In recursive training, each compound is processed in the following manner: The current split for the compound is removed from the model and its remaining constructions are evaluated. After this, all possible splits are tried, by choosing one split and running the algorithm recursively on the created constructions. In the end, the best split is selected and the training continues with the next compound.

- **Local Viterbi Training**: In Local Viterbi training, the compounds are processed sequentially.

Each compound is removed from the corpus and afterwards segmented using Viterbi segmentation. The result is put back into the model. In order to allow new constructions to be created, the smoothing parameter must be given some non-zero value.

- **Random Skips**: In Random skips, frequently seen compounds are skipped in training with a random probability. This speeds up the training considerably with only a minor loss in model performance.

- **Corpusweight Tuning**: An important parameter of the Morfessor Baseline model is the corpusweight (α), which balances the cost of the lexicon and the corpus. There are different options available for tuning this weight:
  - Fixed weight (--corpusweight): The weight is set fixed in the beginning of the training and does not change
  - Development set (--develset): A development set is used to balance the corpusweight so that the precision and recall of segmenting the developmentset will be equal
  - Morph length (--morph-length): The corpusweight is tuned so that the average length of morphs in the lexicon will be as desired
  - Num morph types (--num-morph-types): The corpusweight is tuned so that there will be approximate the number of desired morph types in the lexicon.

The default training mode is 'init+batch' which creates a new model and uses 'batch' training. In order to use different training mode, the desired mode can be specified as command line option.

If one doesn't need a word to be split at some specific character, then –nosplit-re option can be used which takes a regular expression as its argument and if the split location for a word matches the expression then the split is not performed. For example, if suppose the English training corpus has the words – cash and ash – then by default the output model will be: ash and c+ash. Now to prevent such single letter occurrences (c+ash instead of cash) except for some characters like house+s which are suffixes in general, --nosplit-re 'c' can be mentioned in the command line before training.

Morfessor includes --skips option for randomly skipping compounds and constructions that have been recently analyzed to speed up batch training.

The command line usage for Morfessor tool can be referred from the Morfessor Documentation: http://www.morfessor.readthedocs.org

4.5 Algorithms used in Morfessor

There are two algorithms that can be used for training:

- Recursive algorithm: It is the default algorithm that uses recursive training.
- Viterbi algorithm ('-a viterbi'): It uses viterbi training where viterbi segmentation is performed using the viterbi algorithm (a dynamic programming algorithm for finding the most likely sequence of hidden states for a sequence of observed states) which finds the most likely sequence of segmentations from the set of observed words present in the corpus.

While performing decoding, numerous methods of BaselineModel class are used such as:

- segment(): for segmenting a compound by looking up in modal analyses.
- viterbi_segment(): for segmenting new words, this method is used. It performs viterbi segmentation using viterbi algorithm for searching the most optimal segmentation.
- viterbi_nbest(): It searches for segmentations just like viterbi_segment() but returns top n segmentations of the compound.

Chapter 5

Word Lattices

5.1 Introduction to Word Lattices

A word lattice is a directed acyclic graph with a single start point and edges labeled with a word and weight that can represent exponential number of sentences. Unlike confusion networks which additionally impose the requirement that every path must pass through every node, word lattices can represent any finite set of strings (although this generally makes word lattices slightly less space-efficient than confusion networks). However, a word lattice can represent an exponential number of sentences in polynomial space. Here is an example lattice showing possible ways of decompounding some compound words in German (referenced from http://www.statmt.org/moses/?n=Moses.WordLattices):

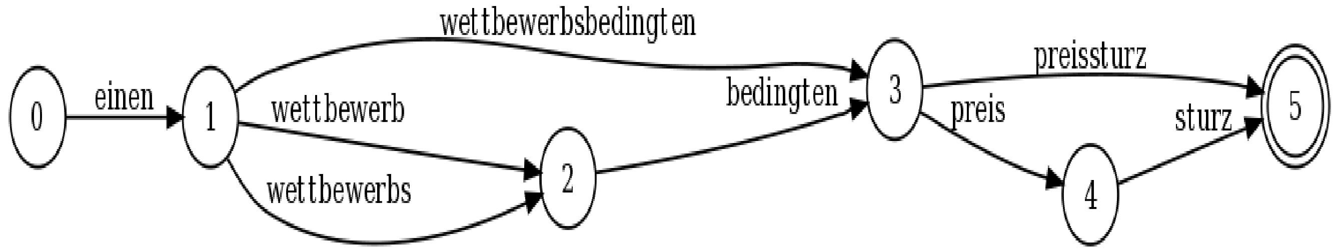


Figure 7: Word Lattice

Moses translates input encoded as a word lattice and the translation it chooses maximizes the translation probability along *any* path in the input.

A word lattice is useful because it permits any finite set of strings to be represented and allows for substrings common to multiple members of the set to be represented with a single piece of structure. Additionally, all paths from one node to another form an equivalence class representing alternative expressions of the same underlying communicative intent. There are several other reasons why a word lattice is useful as stated below:

Today, virtually all statistical translation systems seek the best hypothesis e for a given input f in the source language, according to e= arg max Pr(e|f). An exception is the translation of speech recognition output where there may be errors in understanding the speech and the context of the sentence. Even noise can be there and so the acoustic signal generally under-determines the choice of source word sequence f. So an approach here is to provide multiple inputs to the decoder. Rather than translating a single-best transcription f, it is advantageous to allow the MT decoder to consider all possibilities for f, that sound similar, by encoding the alternatives compactly as a confusion network or a lattice [8]. To this Christopher Dyer argued that even for text, there are often multiple ways to derive a sequence of words from the input string [6]. For example, there can be different morphemes of a word. There can be ambiguity in the input for example: out of multiple segmentations of a word, the optimal segmentation might not be known. So a way to represent all possible contexts/meanings of such ambiguous input is required. One such way to provide multiple inputs to the decoder is a lattice structure where given a signal o with different inputs f, the translation e can be e = argmax(e) max(f) pr(e,f|o). Thus it is basically used for representing ambiguous input.

However, when it comes to ambiguous input, a simple way is to just list all possible meanings successively and then translate each but this is expensive and also there can be many common words in this top n listing. So for representing such a top n listing with common words as nodes, word lattice is used which is a subset of finite state automata.

Word lattice translation offers an improvement over the conventional approach: a lattice may represent multiple segmentations of a sentence. Input represented in this way will be more robust to errors.

Apart from this, word lattices are widely used in machine translation for several other purposes. The search graph of a phrase based decoder can be thought of as a word lattice (Ueffing et al., 2002).

5.2 Relevance of Lattice Structure in Morphology

Word lattice is a subset of finite state automata. Graph theory is also used in natural language and linguistics since natural language represents discrete structures well. Many methods are used in phonology and morphology to perform analysis of a language as a graph. In morphology, a morphological analyzer converts an input to output i.e. word forms to strings. Since the function of a finite state transducer is to translate input to output, morphological analyzer is implemented as transducer. A finite state transducer is bidirectional. So conversion can be performed in any direction. The general practice is to use it by providing the lemma and tags as input and obtaining the word forms. A transducer is designed as two parts: first the lexicon part which generates morphemes ('s' for plural) from the tags and second the morphophonological part which validates the morphemes (watch+s converted to watch+es) and then both the parts are combined. Since it is bidirectional, even if it is constructed in forward direction with input as tags, intermediate output as morphemes and final output as word forms, it can be used with word forms as input and then traversing the paths in reverse direction based on which path is matched. This is the main reason why transducer is used in morphological analysis. This way for a given word form, all possible tags associated with that word form can be determined i.e. if word form is ambiguous for e.g. watches (Noun as well as Verb), then the output will be both tags. Thus, ambiguity in a word form can be taken care of.

A finite state transducer is basically a finite state machine which is one of the applications of a graph since the different machine states can be represented as nodes and the transitions can be represented using edges. In fact, it is a directed graph. This is how graph theory is used in morphology. Since word lattice is a subset of finite state automata, it finds its usage in representing the ambiguous input.

5.3 Representation of Word Lattice

Lattices are encoded by ordering the nodes in a topological ordering and using this ordering to assign consecutive numerical IDs to the nodes. Then, proceeding in order through the nodes, each node lists its outgoing edges and any weights associated with them. For example, consider the following Gujarati sentence:

પરીલોકનું (pariloknu)

પરી + લોક + નું

The word lattice representing top 2 segmentations of this input sentence is:

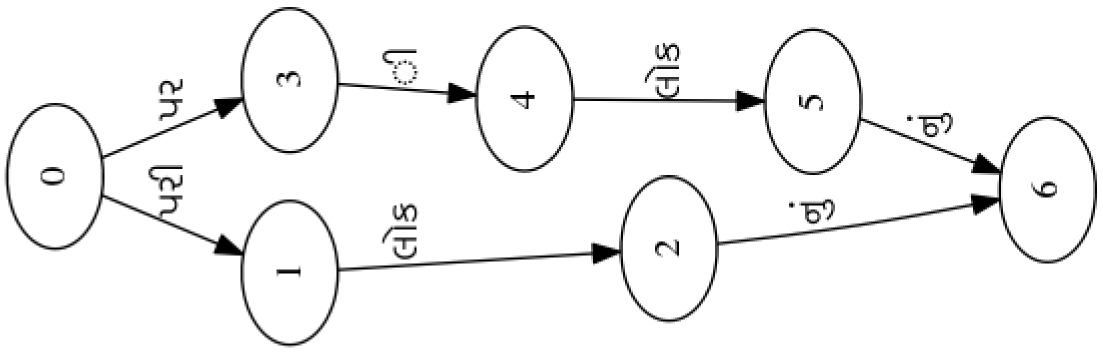


Figure 8: Lattice Example

This lattice can be written in Moses format (also called Python Lattice Format PLF) as follows:

```
(
 (
  ('પરી',2.30664347567,1),
  ('પર',2.6537259109,3),
 ),
 (
  ('લોક',1,1),
 ),
 (
  ('નું',1,4),
 ),
 (
  ('ી',1,1),
 ),
 (
  ('લોક',1,1),
 ),
 (
  ('નું',1,1),
 ),
)
```

The second number is the probability associated with an edge. The third number is distance between the start and end nodes of the edge, defined as the numerical ID of the end node minus the numerical ID of the start node. The edge distance indicates how many nodes to jump from the current node for accessing the next node. Say if the current node is numbered ‘1’ and the edge distance of the edge followed from that node is ‘3’, then next node accessed will be the node numbered ‘4’. Typically, lattices are written, with no spaces, on a single line:

```
((('પરી',2.30664347567,1),('પર',2.6537259109,3),),(('લોક',1,1),),(('નું',1,4),),(('ી',1,1),),(('લોક',1,1),),(('
નું',1,1),),)
```

5.4 Configuring Moses to translate Word Lattice

To operate on word lattice, first of all, we need to binarize the phrase and reordering tables since lattice decoding can't be performed on normal non-binary phrase and reordering tables. To indicate that Moses will be reading lattices in PLF format, we need to specify **-inputtype 2** on the command line or in the moses.ini configuration file. Additionally, it is necessary to specify the feature weight that will be used to incorporate arc probability (may not necessarily be a probability!) into the translation model. To do this, we need to use:

(for old moses.ini format)
-weight-i X on the command line where X is any real number.

(for new moses.ini format):

under [feature] section:
InputFeature name=InputFeature0 num-features=1 num-input-features=1 real-word-count=0

and under [weight] section:
InputFeature0= X where X is any real number

Once Moses is configured, we can decode lattice input in a manner similar to normal decoding by providing the lattice file as the input.

In word lattices, the phrase length limit imposed by the **-max-phrase-length** parameter (default: 20) limits the difference between the indices of the start and the end node of a phrase. If the difference exceeds the default, then we can specify new max length using this command line option while decoding as **-max-phrase-length X** where X is the new difference between the start and end node indices.

We can also verify if our PLF input file is valid using checkplf (which is provided in Moses) which reads a PLF (lattice format) input file and verifies the format as follows:

```
./checkplf < input_plf_file
```

Chapter 6

Experiments

6.1 Experimental Setup

6.1.1 Codes Developed

- *viterbi_nbest_sentence.py*: Generates top n segmentations of a word, appends that in morph_tokens list and also outputs the same to the output file in such a way that top n sentences of a given sentence are arranged successively.
  Usage: python viterbi_nbest_sentence.py <infile> <outfile> <language> <indic_resources_path> <nbest_value>

- *nsentence.py*: To make each input sentence appear n times (generally used for target data)
  Usage: python nsentence.py <infile> <outfile> <n-value>

- *mulfile.py*: To combine the sentences from two different resource files
  Usage: python mulfile.py <infile1> <infile2> <outfile>

- *morpher_sentence.py*: Creates a morpher_tokens list comprising of case marker separated tokens for the given Gujarati input sentence
  Usage: python morpher_sentence.py <infile1-morphedfile> <infile2-originalfile>

- *lattice.py*: Creates a top n lattice representation of given input sentence
  Usage: python lattice.py <infile> <outfile> <language> <indic_resources_path> <n_value>

- *multiple-resource-lattice.py*: Creates a lattice representation using sentences from two different sources
  Usage: python multiple-resource-lattice.py <infile1> <infile2> <outfile> <language> <nvalue>

- *graph.py*: Uses pydot python library to create a graphical representation of lattice structures
  Usage: python graph.py <infile> <language> <nvalue>

- *binarize_model.sh*: Script developed to perform binarizing of phrase table and reordering table using just one command.
  Usage: sh binarize_model.sh input_phrasetable input_reorderingtable output_phrasetable output_reorderingtable

- Modified moses_jobscripts/moses_run.sh to perform lattice creation and lattice decoding

6.1.2 Experiments Performed

To test the usefulness of word lattice in representing ambiguous input to Moses, we used two morphologically rich languages – Gujarati (Gu) and Malayalam (Ml) – as source and Hindi (Hi) as target.

For the systems using word lattices, the training data should contain the versions of the corpus appropriate for the segmentation schemes used in the input.

For both, Gu-Hi and Ml-Hi language pairs, we trained three systems. In each of the experiments, the target was unsegmented while the source was used as unsegmented source, top 1 segmented source and top 2 segmented source at the time of training the model:

Table 1: Experiments Performed

Experiment		Corpus size
Exp #/Model #	(Segmented/Unsegmented source, and unsegmented Hi target)	# of sentences
1	Unsegmented Gu/Ml as source (surface)	46277
2	Top-1 segmented Gu/Ml as source (top 1)	46277
3	Top-2 segmented Gu/Ml as source (top 2)	92554

(brackets indicate the short representations for source)

In addition, we performed three more tests for Gu-Hi language pair: (a) using case marker separator – another tool to perform morphological segmentation of Gu language instead of Morfessor which trains a model using **semi-supervised approach** and **active learning**, (b) combining case marker separator with Morfessor top 1 output, and (c) combining phrase based data with top 1 data on the source side.

Table 1 Experiments Performed (cont.)

Experiment		Corpus size
Exp #/Model #	(Unsegmented Hi target in all)	# of sentences
4	Source GU -> surface + case marker split	46277
5	Source GU -> top 1 + case marker	92554
6	Source GU -> surface + top 1	92554

6.2 Experimental Results

The BLUE (Bi-Lingual Evaluation Understudy – an evaluation metric for determining the quality of translations obtained) scores obtained in various tests are as follows:

Gu-Hi Results: (Here Model # indicates the system model on which the test was performed, surface form indicates unsegmented test input, top 1 indicates best segmented test input and lattice content indicates the different forms which lattice comprises)

Table 2: Gu-Hi Results

Model #	% BLUE Scores for surface form	% BLUE Scores for top 1 source	Lattice Input	
			Lattice Content	% BLUE Scores
1	45.09	-	-	-
2	-	49.27	Top 2 segmentations	47.45
2	-	-	Surface + top 1	49.15
3	-	49.70	Top 2 segmentations	48.33
4	-	49.43	Top 1 + case marker	47.78
5	-	48.68	Top 1 + case marker	49.03
6	-	49.36	Surface + top 1	48.87

Ml-Hi Results:

Table 3: Ml-Hi Results

Model #	% BLUE Scores for surface form	% BLUE Scores for top 1 source	Lattice Input	
			Lattice Content	% BLUE Scores
1	13.34	-	-	-
2	-	18.51	Top 2 segmentations	17.74
3	-	18.59	Top 2 segmentations	18.17

6.3 Evaluation

This section focuses on the evaluation of experiments performed in the previous section. The question that needs to be answered is:

- Is lattice representation of ambiguous inputs useful for Indian languages?

6.3.1 Analysis

The results above indicate:

- The BLUE scores for phrase based experiments are **lowest of all** for both Gu-Hi and Ml-Hi.

This is due to **sparse data problems** arising in morphologically rich Gu and Ml languages.

- The BLUE scores for decoding top 1 segmented test data, obtained using Morfessor, on model #2 improve for both Gu-Hi (from **45.09** for unsegmented to **49.27**) and Ml-Hi (from **13.34** to **18.51**). These improvements justify the point that rather than using a system and performing decoding at word form level, results improve if decoding is performed at lemma level.

- The BLUE scores for decoding top 1 segmented test data on model #3 are **49.70** for Gu-Hi and **18.59** for Ml-Hi. One reason for this slight improvement, as compared to systems trained using top 1 segmented data only, is the larger corpus size. The **corpus size** in these systems is **double** the size in systems trained using top 1 segmented data.

- On training the system using the case marker splitter for Gu, which generally splits a word at start position of case marker (for example: બંદૂકની - bandukni - is split as બંદૂક+ની), and then decoding the test data also segmented using the case marker splitter, the BLUE score obtained is **49.43** which is slightly better than **49.27** score obtained using top 1 segmentation of Morfessor (the corpus size for both experiments is same). This result indicates that case marker splitter for Gu language yields results similar to Morfessor and so can be used as its alternative.

  As example, consider the following sentences:

- ચારેબાજુથી ઘેરાયેલ વન પ્રદેશ , ઘાટીમાં વાદળોની આંખમિચોલી , ઠંડી હવા અને ઝાકળની ચાદરમાં લપેટાયેલા બરફવાળાં શિખરો , કોઈ પરીલોકનું દ્રશ્ય ઉપસ્થિત કરે છે . (source sentence)

  चारों ओर से घिरा वन प्रदेश, घाटी में बादलों की आँख मिआओ ચોલી , ठंडी हवा और पाले की चादर में लिपटी बर्फ वाले चोटियों , किसी परीलोक का दृश्य उपस्थित करता है। (obtained using Morfessor resource)

  चारों ओर से घिरा वन प्रदेश, घाटी में बादलों की આંખમિચોલી, ठंडी हवा और पाले की चादर में लिपटी બરફવાળાં चोटियों , किसी પરીલોકનું दृश्य उपस्थित करता है। (obtained using case marker splitter)

Here Morfessor splits આંખમિચોલી (aankhmicholi) as આંખ મિ ચોલી as a result of which incorrect translation is obtained - आँख मिआओ ચોલી. મિ is transalted as मिआओ which is incorrect.

On the other hand, the word બરફવાળાં (barfwara) remains untranslated in case of case marker splitter as it appears as a complete word in case marker separated data. While, Morfessor splits બરફવાળાં as બરફ+વાળાં and so better translation is obtained as બરફ+વાળાં translates to बर्फ वाले.

This one sentence indicates both cases where Morfessor performs better and where it performs poorly. In some cases, Morfessor produces unnecessary, incorrect segmentations which result in irrelevant translations reducing the score. Similarly, in some cases, case marker splitter performs poorly leaving the word untranslated where segmentation of word yields proper segments. As a result of this, the scores of both are almost similar.

- આંખોમાં મેંશ ન આંજો . (source)

  आँखों में मैंने नष्ट न चकाचौंध करने । (obtained using Morfessor resource)

  आँखों में મેંશ न આંજો । (obtained using case marker splitter)

  Here, Morfessor splits મેંશ as મેં+શ and આંજો as આંજ ો which results in irrelevant translation.

- આગલા દિવસે તાઓરમિનાથી વિદાઈ લઈને અમે ` મંદિરોની ઘાટી ` એગ્રેજૈન્ટો પહોંચ્યા . (source)

  अगले दिन ताओरमिना से विद हो लेकर हम ' मंदिरों की घाटी ' ने ग्रैंड जैन ्टो पहुँचे । (obtained using Morfessor resource)

  अगले दिन ताओरमिना से વિદાઈ लेकर हम ' मंदिरों की घाटी ' એગ્રેજૈન્ટો पहुँचे । (obtained using case marker splitter)

  Here, વિદાઈ (vidaai) is split as વિદ ાઈ and એગ્રેજૈન્ટો (aegrezento) as એ ગ્રે જૈન ્ટો. Due to such splitting, વિદ ાઈ is translated as विद हो and એ ગ્રે જૈન ્ટો as ने ग्रैंड जैन ्टो with the mapping એ → ने , ગ્રે → ग्रैंड, જૈન (jain) → जैन and ્ટો → ્ટો left untranslated.

- However, the combination of top 1 segmentation using Morfessor and case marker splitter (model #5) yields a BLUE score of **48.68** which is slightly lower as compared to top 1 segmentation system. Basically, both the translations differ in few words which are represented by their synonyms (synonymy) and this fact is reflected by the METEOR scores of both:

  For combined system, METEOR of top 1 segmented test data - 0.718 while for top 1 system, METEOR – 0.717

  As an example, consider the following sentence:

  यदि श्वास प्रणाली में सूजन आ जाए तो भी रक्त मुँह के रास्ते बाहर आने लगता है । (combined)
  यदि श्वास प्रणालिका में सूजन आ जाये तो भी रक्त मुँह के रास्ते बाहर आने लगता है । (top 1)

  ब्रेकफास्ट न करने से आपका पाचन दोपहर के भोजन तक शुरू नहीं होता । (combined)
  ब्रेकफास्ट न करने से आपका उपापचय दोपहर के भोजन तक शुरू नहीं होता | (top 1)

- The combination of phrase based and top 1 segmented data (model #6) yields a BLUE score of 49.36 for decoding top 1 segmented test data while the score obtained using top 1 segmented data individually is 49.27. Again, one reason for this slight increase is double the corpus size.

As far as lattice decoding is considered:

- The BLUE scores for all lattice decodings are lower as compared to corresponding system's top 1 segmentation decoding.
- For model #2, BLUE score for top 1 segmented test data is 49.27 while the score for a lattice of top 2 segmentations is 47.45. The reason for this reduction is that if a string in first of the two paths in a lattice is not found in the phrase table i.e. the string is an OOV (out of vocabulary) word then the decoder automatically chooses the second path which in most cases leads to poor translations.

  As an example, in the sentence:

  રાજસ્થાનની પહેલી મહિલા પાઈલોટ નમ્રતા ભટ્ટ છે .

  rajasthanni pehli mahila paailot narata bhatt che.

  the top 2 segmentations of the word પાઈલોટ are પાઈલોટ and પાઈ+લોટ.

  The translation of this sentence considering only the top 1 segmentation પાઈલોટ is:

  राजस्थान की पहली महिला પાઈલોટ नम्रता भट्ट है।

  Here as પાઈલોટ is an OOV word, it remains untranslated.

  Now if lattice of top 2 segmentations is decoded, the resulting translation is:

  राजस्थान की पहली महिला कट आटा नम्रता भट्ट है।

  Here, as પાઈલોટ in first path is OOV, પાઈ+લોટ in second path is considered and it can even be subjectively seen since લોટ (lot) in Gujarati means आटा.

- For model #3, the BLUE score obtained by decoding lattice of top 2 segmentations is 48.33 which is again less than the score (49.70) obtained by decoding only the top 1 segmented data in the same system. The reason being the same that for OOV words, incorrect and meaningless translations are obtained. However, this score is better than the score obtained by decoding the same lattice representation on model #2 with the reason being larger corpus size.

  For example:

  A word lattice translates to - ध्यान से देखने पर सामने आया कि वह लंबे पेड़ नहीं while decoding on model #3. But while decoding on model #2, the same lattice translates to: ध्यान से देखने से कम पाया गया वह

लंबे पेड़ नहीं. Large corpus helps here in better translation.

Another reason is due to large corpus, some of the previous OOV words may be present resulting in improved translations. For example: in lattice decoding, a translation obtained using model #2 is - हम आठ मील (ધરમપુર - dharampur) तक गाड़ में आए । Here, ધરમપુર has two segmentations ધરમપુર and ધરમ+પુર (dharam+pur) but both are OOV and so the word remains unchanged. On decoding using model #3, the obtained translation is - हम आठ मील (घर पुर) तक गाड़ी में आए । Here, either of ધરમ or પુર is present in phrase table and so the word gets translated.

- BLUE score of 49.03 is obtained by decoding the lattice comprising of top 1 segmented data and case marker separated data on the system trained using top 1 segmented and case marker separated data i.e. model #5. But this score is still less than the score obtained for top 1 segmented test data decoded on model #2.

- BLUE score of 49.15 is obtained by decoding the lattice comprising of phrase based surface forms and top 1 segmented data on model #2. This score is very near to the score (49.27) of decoding only top 1 segmented on the same system with most of the translations being same as that obtained from decoding top 1 segmetned data. The reason for this being: phrase based surface forms are in the first edge and if they are OOVs then mostly second path will be followed which therefore makes the decoding equivalent to directly decoding top 1 segmented data.

- So, for both Gu-Hi and Ml-Hi systems, the usage of word lattices to represent ambiguous input in several alternative ways is not improving the translation obtained using only the best segmentation of source sentences.

- These results do not satisfy the conclusions made by Christopher Dyer, Smaranda Muresan and Philip Resnik as cited in Generalizing Word Lattice Translation where they state substantial gains in translation performance by decoding compact representations of alternative source language analyses, rather than single-best representations. The results stated by them show that the BLUE scores for surface+morph are more than the scores for only surface form and only morph form.

6.3.2 Few Examples of Translation

Few of the examples where lattice structure performs poorly compared to top 1 segmentation decoding:

- Reference: ठहरने की व्यवस्था सरकारी अवकाशीय कैंप व कारला होटल में है ।

  ठहरने की व्यवस्था सरकारी अवकाश ીય કैंप और कारला होटल में है। (top 1)

  ठहरने की व्यवस्था सरकारी अवकाश जा जाता कैंप और कारला होटल में है। (lattice)

- Reference: क्लोम ग्रंथि हमारे अंदर इन्सुलिन की मात्रा को नियंत्रित करता है ।

ક્લો ग्रंथीं हमारी इंसुलिन की मात्रा को नियंत्रित करता है। (top 1)

क्लब का ग्रंथीं हमारी इंसुलिन की मात्रा को नियंत्रित करता है। (lattice)

- Reference: सीढ़ी चढ़ना , बागवानी , घर के छोटे - मोटे काम या डांस करने जैसी हल्की - फुल्की शारीरिक गतिविधियाँ जरूर करते रहें ।

  सीढ़ियाँ चढ़ना , गार्डनिंग , घर के छोटे - बड़े काम या नृत्य जैसी हल्की - फुल्की शारीरिक गतिविधियाँ જરૂર करते रहें। (top 1)

  सीढ़ियाँ चढ़ना , गार्डनिंग , घर के छोटे - बड़े काम या नृत्य जैसी हल्की शारीरिक गतिविधियाँ नजर आतुर करते रहें। (lattice)

- Reference: शहरों में रहने वाली आम मध्यवर्गीय महिलाओं का भी यह स्वभाव बन चुका है और यह स्वभाव देन है , आज के इस आधुनिक युग की नई जीवन शैली की।

  शहरों में रहने वाली सामान्य मध्यवर्ग ીય महिलाओं का भी इस स्वभाव बन गया है और यह स्वभाव देन है , आज के इस आधुनिक युग की नई जीवनशैली की। (top 1)

  शहरों में रहने वाली सामान्य मध्यम वर्गीय जाता महिलाओं का भी इस स्वभाव बन गया है और यह स्वभाव देन है , इसी के इस आधुनिक युग की नई जीवनशैली की। (lattice)

6.3.3 Graphical Representation of few Word Lattices

A word lattice can be represented graphically using python's pydot library. For a given input sentence, the lattice tokens are first generated and then they are used to create a graph. The graph generated for the following sentence using graph.py is as follows:

Input sentence: નિરંતર સ્વસ્થ્ય રહેવાનો મૂળમંત્ર

nirantar swasth rahevano murmantra

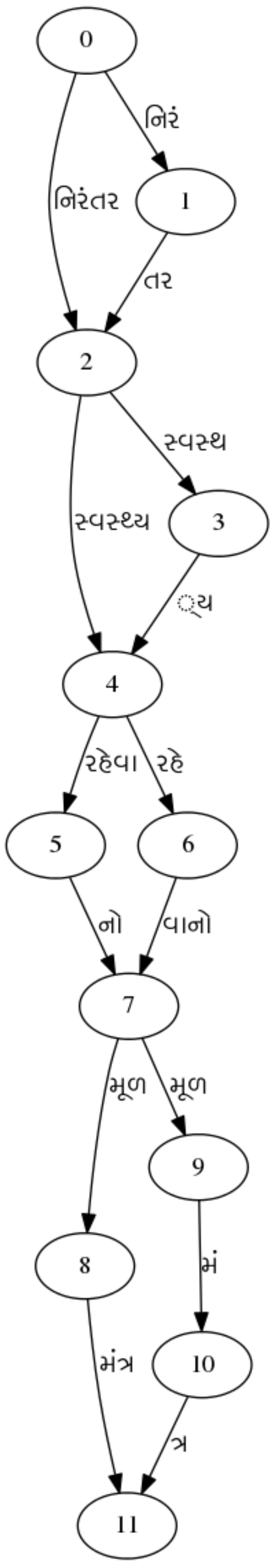


Figure 9: Graphical Representation of Word Lattice

Chapter 7

Summary, Conclusion and Future Work

7.1 Summary

In this report, the following topics have been covered:

- Introduction to Natural Language Processing
- Machine Translation with in-depth study of Statistical Machine Translation
- Morphology and unsupervised word segmentation using Morfessor
- Word Lattice usage for ambiguous input
- Experiments to observe the impact of word lattice input for morphologically rich languages

7.2 Conclusion

After conducting experiments on word lattices, from the results and analysis shown above, one general observation that can be made is that out of vocabulary (OOV) words have a great impact on the overall translation and the dip in translation quality. As we saw, the lattice decoding for an input lattice comprising of surface forms and their best segmentations i.e. surface+morph on the system trained using the best segmentation yielded the score very near to that obtained by decoding the best segmentation alone. However, the same lattice when decoded on the system trained using combination of both – surface forms and best segmentation – yielded lower score.

Another point is that the lattice structure used here represents one complete segmentation of a word on a single path i.e. each segmentation of a word has its own path towards the node representing next word. This is in contrast to the proper lattice structure where common morphs in all possible segmentations are represented only once and paths representing different segmentations merge at and then split again after the common edges. For example in our lattice structure, the words 'લોક' (lok) and 'નું' appears twice - once for each path.

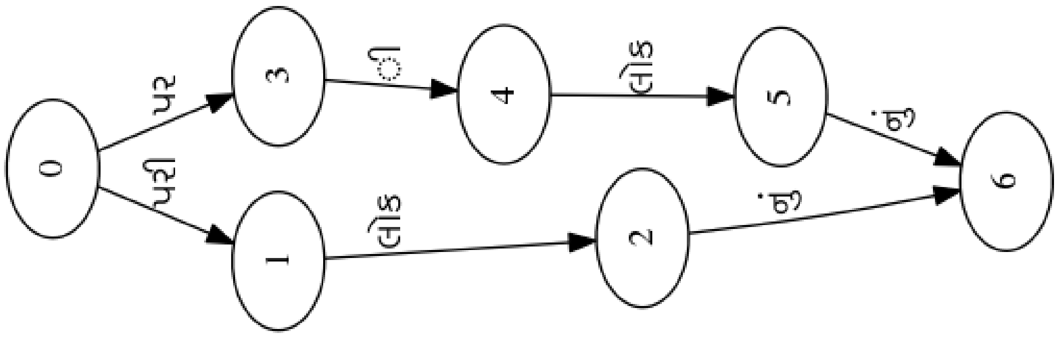


Figure 8: Lattice Example

However, a proper lattice structure for the same word પરીલોકનું (pariloknu) would be:

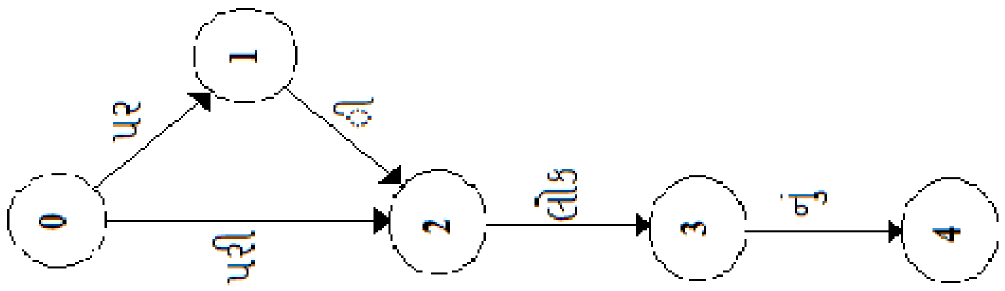


Figure 10: Another Lattice Structure

From this and also from the prior research carried out on the use of word lattices by Christopher Dyer on Arabic-English translation where the lattice of surface+morph with the proper lattice structure yields a score better than both - only surface and only morph, decoding the surface + morph lattice after a change in lattice structure might be helpful.

7.3 Future Work

The future of this study lies in observing the impact of lattice structure on the translation results by implementing a different lattice structure as stated above. One general observation from the experiments performed was that the OOV (out of vocabulary) words cause reduction in translation quality. Hence further task is to find out a way to minimize this reduction due to OOV words.

While performing experiments involving two different sources, be it the segmentation obtained from multiple tools (Morfessor and case marker splitter) or the combination of two forms of input (surface and morph), the approach followed here was to combine the different forms in a single file and then train the model which meant larger corpus. Another approach that can be followed is to train the model for each form (surface as well as morph) separately and then combine their individual phrase tables to perform translation using the concept of multiple phrase tables while decoding. This is another task to work upon in future.

Chapter 8

References

[1] Pushpak Bhattacharyya, *"Natural Language Processing: A Perspective from Computation in Presence of Ambiguity, Resource Constraint and Multilinguality"*, CSI Journal of Computing | Vol. 1 • No. 2, 2012

[2] Carl Vikner, Copenhagen Business School and Sten Vikner, University of Aarhus, *"Hierarchical Morphological Structure and Ambiguity"* from Merete Birkelund, Maj-Britt Mosegaard Hansen and Coco Norén (eds.), ***L'énonciation dans tous ses états – Mélanges offerts à Henning Nølke***, Bern: Peter Lang Verlag, pp. 541-560.

[3] Yihai SHEN, Chi-kiu LO, Marine CARPUAT, and Dekai WU, *"HKUST Statistical Machine Translation Experiments for IWSLT 2007*"

[4] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst; "*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 2007. Association for Computational Linguistics

[5] Philipp Koehn, *"Statistical Machine Translation*", Cambridge University Press, 2009

[6] Vesa Siivola, Mathias Creutz and Mikko Kurimo, *"Morfessor and VariKN Machine Learning tools for Speech and Language Technology",* in Proceedings of the 8th International Conference on Speech Communication and Technology (INTERSPEECH'07), 2007

[7] Mathias Creutz and Krista Lagus, 2005b, *"Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0",* Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology

[8] Christopher Dyer, Smaranda Muresan, and Philip Resnik, "*Generalizing Word Lattice Translation",* in Proceedings of ACL-08: HLT, pages 1012–1020, Columbus, Ohio, USA, June 2008

[9] Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh, Pushpak Bhattacharyya, *"Case markers and Morphology: Addressing the crux of the fluency problem in English-Hindi SMT"*, in Proceeding of ACL-09

[10] Joshua D. Schroeder, *"Word Lattices for Multi-Source Translation",* Master of Science by Research Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2009

[11] Hieu Hoang and Philip Koehn, "*Design of the Moses Decoder for Statistical Machine Translation",* in Software Engineering, Testing, and Quality Assurance for Natural Language Processing, pages 58–65, Columbus, Ohio, USA, June 2008

[12] Sami Virpioja, Peter Smit, Stig-Arne Gronroos, Mikko Kurimo, *"Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline"*, Technical Report, Department of Signal Processing and Acoustics, Aalto University

Web References

[13] Natural Language Processing - http://en.wikipedia.org/wiki/Natural_language_processing

[14] Machine Translation - http://en.wikipedia.org/wiki/Statistical_machine_translation

[15] Statistical Machine Translation - http://en.wikipedia.org/wiki/Machine_translation

[16] AI-Complete - http://en.wikipedia.org/wiki/AI-complete

[17] Moses Overview – http://www.statmt.org/

[18] Morfessor Documentation - http://morfessor.readthedocs.org/en/latest/

[19] Word Lattices - http://www.statmt.org/moses/?n=Moses.WordLattices

Other References

[20] NPTEL and Coursera videos of Natural Language Processing