Using Domain Knowledge to Improve Machine Translation in Indian Languages

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

Master of Science
in
Computational Linguistics by Research

by

Akshat Gahoi
2018114012
`akshat.gahoi@research.iiit.ac.in`

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled “**Using Domain Knowledge to Improve Machine Translation in Indian Languages**” by **Akshat Gahoi**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. Dipti Misra Sharma

To my Grandfather

Acknowledgments

I started my IIIT journey 5 years ago, which had many ups and downs. Throughout my 5 years, I learned a lot and had a fair share of fun with my friends, who are now family to me.

I would like to thank Prof. Dipti Misra Sharma for her support over all these years. She became my advisor in my 3rd year, but even in the first two years, she helped any CLD student that went to her with any issues. We can go and talk to her even if we have personal issues, and she is there to listen. When it comes to research, she helped in opening our thought process for a research problem. Due to weekly meetings, there were constant discussions, and she always pushed us to think like a researcher and not just take a problem and start coding around it. She always cares for the wellbeing of students around her and that made my 5 years quite easy. After my advisor, I would like to thank Vandan Sir, Arafat Sir, and Pruthwik Sir who helped me constantly throughout my research projects.

My research partners Akshat and Saransh who are my close friends too made my life easier in the lab as I shared many of my projects with these two guys. Our discussions on every topic that Ma'am sent our way made this thesis possible.

I would like to thank my friend group "Daddycated" without whom I can not imagine my life now. Starting with Anirudh for being the best roommate possible in college and one of the best guys to share any problem with. Anshul with whom I have almost spent every minute of my college life. Risubh, Tiwary, Pranav, Karra, Reddy, Duneja, and Mangle, without you guys, it would have been impossible to get out of this college. Avlok, Mohla, Ista, Shourja, Sumba, Ainsley, Winter, and Maharana, I know I can't have fun on that energy level, but thanks for making my life in college full of entertainment. Tanvi for supporting me in not eating mess food. Nikki, and Baz, for making my life easier with gaming. Chan and Chaitanya for helping out whenever I faced any issue related to research.

At the end, I would like to thank my family. My parents and my sister who has supported me all my life. Jijaji who entered my life when half of my college was over but he made sure he was there for me whenever I need him. My grandmother whose daily calls helped me overcome any tension that I had in the day. Phuphaji and Bua whose constant efforts in my education have resulted in what I am right now. Both of my elder first cousins who made me believe that all these things can be achieved. Chacha chachi and rest of family whose constant support made me overcome any challenge that I faced.

Abstract

In this modern world, due to the increased mobility of humans, encountering a foreign language has become a common challenge for many people around the world. This causes a language barrier in their regular lives, which makes communication quite difficult. This makes machine translation a facility that helps people to overcome this language barrier. Research on machine translation has been going on for many decades, and there are many MT models that give good-quality translations, but even the best of these models fail to produce quality output when a domain-specific input is given to them. These models are trained on large general domain data, which makes their domain-specific translations not up to par. This brings up the issue of domain adaptation for different areas. For Indian languages, the issue arises with the lack of domain-specific data and good baseline models. This thesis will try to put forward an approach to improving the scores of domain-specific translations with efficient use of domain data.

Before getting into domain adaptation, this thesis will try to understand how a domain is defined and how domain information is stored in these documents or sentences. For this study, we will discuss two tasks that will help us to understand the importance of domain terminologies. The first one discussed fine-grained domain classification as a task. It tries to get information out of similar domains and what makes those domains different by classifying an unknown document into a similar set of domains. The other task helps to find domain terms in a document in an unsupervised manner. It used an improved TextRank approach, where n-grams are used to get the most important terms in a document. Both of these approaches helped in understanding domain terms and their importance in defining a domain.

After understanding domains, we detail different approaches done for domain adaptation and give a comparative analysis of them. We started with a very basic domain adaptation approach that gave us a good result but proved to be an inefficient task for multiple domains. All the approaches were for the English-Hindi language pairs, but the basic domain adaptation of individual domains was also done for the English-Telugu and English-Bengali language pairs. After multiple experiments, we show in this thesis how we can get better performances in an efficient manner when we use the domain knowledge of different domains in the task of domain adaptation. Different approaches will be discussed for all the domains to create different translation models for our task of domain adaptation.

The work done in this thesis will try to solve the above-mentioned problems and give us a better understanding of the world of domains by tackling the tasks of fine-grained domain classification, domain terminology extraction, and, most importantly, efficient domain adaptation.

Contents

List of Figures

List of Tables

Chapter *1*

Introduction

1.1 Overview

In this thesis, we will try to tackle the problem of domain adaptation for domain-specific translations by studying different approaches to fine-tuning a baseline model over a domain. These approaches will try to improve the translation quality for each domain. The problem of domain adaptation arises when a general machine translation model is given a domain-specific input. Most of the models that are trained on the general domain start giving noisy outputs for these inputs due to a lack of domain knowledge.

To understand the concept of domain adaptation, we will first try to understand how a baseline machine translation model is created for different Indian languages and how we identify what features are used to identify the domain of a phrase or a document. This will help us to understand how we can reduce the number of models that we are going to adapt when we are dealing with multiple domains and also to understand low-resource languages and how they work when it comes to the task of machine translation.

This thesis is mainly divided into three parts. In the first part, we will talk about how we created baseline models for different Indian languages. This will include a brief introduction to transformers[70] and what difference different parameters make in different Indian languages.

The second part of the thesis deals with domain knowledge. It will deal briefly with domain terminologies and how domains are identified. Different basic domain adaptation techniques will also be discussed here.

The third part of the thesis takes this identified domain knowledge from different domains and discusses the approach devised to improve the performance of the baseline model.

1.2 Machine Translation for Indian Languages

Language plays an important role in any society. All the ideas and other related information of a community/region are encoded by the language that is used in that region. It passes on from generation to generation, which changes the language with time. As a group of people moves from one place to another, a mode of communication is always needed, resulting in a change of language for the set of people. If a person does not understand a language that is being spoken or written around her, she cannot complete their day-to-day activities and fulfill her basic needs. Here comes the concept of translation, where a person does not need to learn a language that is foreign to them and just has to understand its meaning and its translation in a language that is known to her.

Machine translation has become one of the most common applications of NLP and machine learning. There is no single solution for this task, as it is dependent on different characteristics of the data used. The first and most important part of this task is the language pair used. Machine translation as a task becomes quite different for different language pairs. This is because of the idiosyncrasies of these languages. The quality and quantity of the data used to determine the output quality change significantly.

Machine translation and the solution to it have changed significantly with time. It started with a rule-based approach dating back to the 1950s. After some time, basic Statistical Machine Translation[1](SMT)[10] method took over, where a sentence is broken down into phrases. Then, for each phrase, the most probable translation is identified from the parallel corpora. Another model is then used to reorder these translated phrases to make a complete sentence in the target language.

Neural Machine Translation (NMT) comes after it. DeepL and Google Neural Machine Translation[75] were a few of the starting key milestones. A Sequence-to-Sequence architecture[68] was introduced to produce basic translations. This is made by RNN/LSTM[15] encoder-decoder model. A seq2seq model takes the input in the encoder layer sequentially and then tries to predict the next word in the output according to the probability likelihood.

This was followed by a major breakthrough with the introduction of Transformers[70] and the concept of attention. A transformer deals with quite a lot of different parts. From positional encoding to finally getting embeddings from the final softmax layer[73]. With time NMT was mixed with reinforcement learning (RL) methods, where the BLEU score of the translation acted as the reward. We will go deep into how transformers are used in machine translation in Chapter 3.

1.3 Domain Adaptation

Before going into domain adaptation, one has to understand what someone means by "domain." It's a field or branch to which a particular text or document belongs to. A piece of text can belong to multiple domains according to the context. Domains can differ from each other where they are not at all similar, i.e., Medicine and governance are two completely dif-

[1]`https://kantanmtblog.com/2019/04/02/a-short-introduction-to-the-statistical-machine-translation-model/`

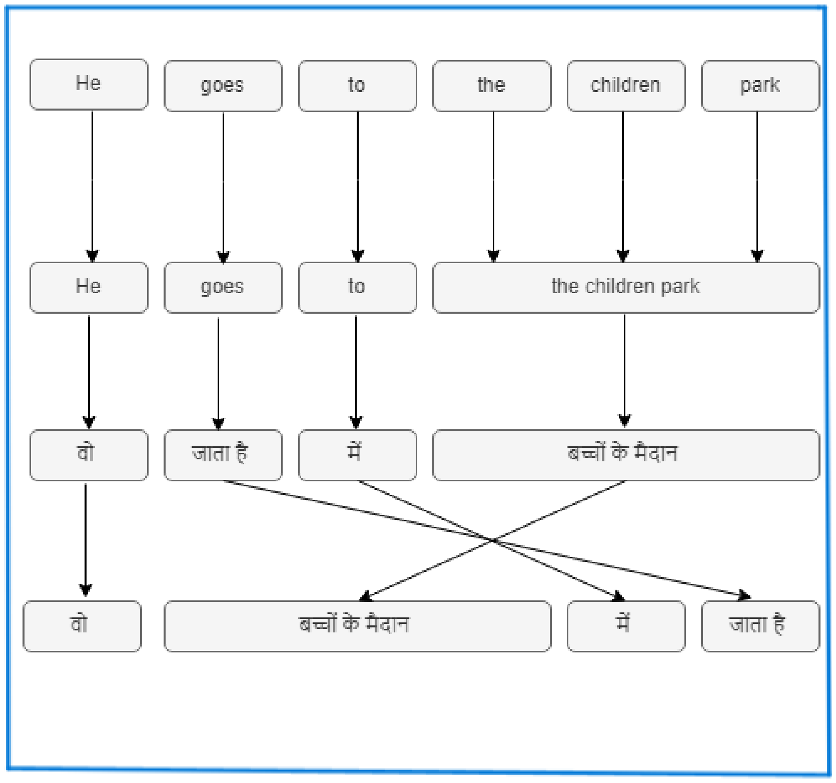


Figure 1.1 Basic SMT Model

ferent domains, and they can also differ on a shallow level, i.e., Artificial Intelligence and Data Analytics are two different domains under a broader category of Computer Science.

Domain Adaptation comes into play when we want our Machine Translation model, which is trained on a general dataset, to perform equally well and give great results on different domains. This challenge is quite common in Machine Translation tasks, as every model faces a challenge when input is given from an unrelated domain and not that of the general domain that it is trained on. As the baseline models are created on a wide variety of domains, it becomes a task to make them perform on a specific topic.

A general idea can be made that domain adaptation is just fine-tuning a model for a specific domain, but the problem can be approached differently to get better results. A better understanding of each domain can help us to get better results. These results, obtained by experimenting with the domain data, make the approach data-centric. Model-centric approaches in domain adaptation play with the configuration of the model and how one can get the best out of it.

One of the biggest challenges, when we talk about domain adaptation, is the quality and quantity of the domain-specific data. This problem is faced in Indian languages, where domain-specific data is not readily available. For example, one has to collect a lot of medical data, which is a parallel corpus for any Indian language, to create a medicine domain-specific model for machine translation. One other challenge is the number of models that are created, i.e., one has to optimize the number of models that are being created, as one can not just create a domain-specific model for every domain out there. So these are the two major challenges in the case of Indian languages.

When a machine has to understand what a domain is, it has to learn about many different things related to it. A domain can be identified by the terms that are used in it and how the language acts in that particular domain, i.e., the style of the language also changes from domain to domain. This can be seen in the way how it is written or the grammar that is being used. So the data and these features of a domain contribute most to the task of domain adaptation.

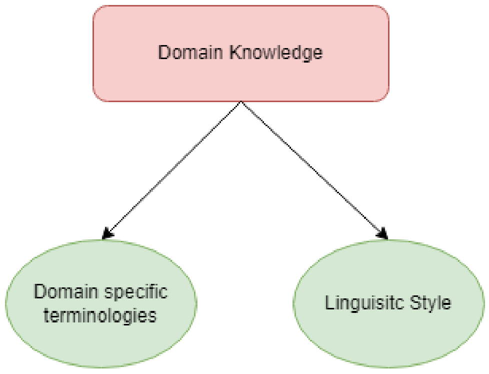


Figure 1.2 Different characteristics of a domain

1.3.1 Domain Identification and Domain Terms

Before getting into domain adaptation, one has to first identify the domain of a particular text. This is a classification task in itself. Domain identification in machine translation helps when we get an unknown document that we have to translate. After identifying the domain, we can give the document or the piece of text to that domain-specific machine translation model and get a better translation. There are many ways to identify the domain of a document, as it is a classification task. Most of the ways take domain terminologies into account, as the most important thing that defines a domain or a field is the terms that are used when a document is written. Experiments are done on BERT embeddings to get the same result.

Domain terms play a very important role in any problem related to domain adaptation. Domain term extraction from a document is a task of its own. There are many supervised and unsupervised solutions for the same. A domain can be very terminology intensive, like medical or corporate law. All these observations can be made linguistically by taking the number of nouns and verbs used per sentence by each domain. All these linguistic features, like grammar and the style of the language, also differ among them. For example, when we see a domain like governance, we see a change in language and how formally each sentence is written for this domain when compared to a general domain. One such example of a sentence related to governance domain will be *"Under the special scheme 2,22,692 house site pattas were issued from 01 . 01 . 2007 to 31 . 03 . 2010 . Under the regular scheme, 5,01,635 house site pattas were issued from 01 . 04 . 2006 to 31 . 03 . 2010 ."* which can be compared to an example sentence from a general domain *"So children who have not attended their school regularly have less chance of getting a good job."* Like this example, all the domains differ from each other in many linguistic features, including the style of the language.

1.3.2 Approaches for Domain Adaptation

There are four different approaches that we will be taking to tackle the problem of domain adaptation:

1. The first one will be the basic approach of fine-tuning for every domain separately, which will provide us with the results of the most basic approach used for this problem and what are the issues it creates.

2. The second approach tries to tackle the problem by making only a single model for all the domains by taking all the domains together.

3. In the third approach, we will still make only a single model, but with domains stacked over each other instead of taking knowledge of all the domains together.

4. The fourth approach will be our final approach, which will help us to understand how clustering can be done on similar domains and how their shared knowledge can be used to improve the translation quality of the output for each domain.

5. Domains can be seen in a hierarchical form where the root is the whole language and different domains branch out of it. Here, the first layer out of the root is the broadest category when we think of a language, and the lower we go in the branch, we get shallower domains that are more interrelated to each other. For example, Biology and Computer Science are two different domains, but there are finer domains like Cardiology and Osteology under Biology and Artificial Intelligence and Software Development under Computer science.

6. This tells us how we can cluster some similar domains and use the common knowledge among those domains for domain adaptation. These similarities between different domains will again come down to knowledge of different domains and their linguistic features.

1.4 Thesis Contribution

This thesis will contribute in the following ways:

1. Through domain classification and domain terminology extraction, this thesis will try to understand the components of a domain and how a domain is defined.

2. A new approach for domain terminology extraction is introduced. TextRank approach of unsupervised term extraction is modified to get better terms for a specified domain.

3. A systematic study of domain knowledge will help us to cluster domains on the basis of similarity and their shared knowledge.

4. Different approaches of domain adaptation for different domains will be discussed, and using the clustering done for similar domains, an efficient approach for domain adaptation using clustering will be introduced, which will improve the output quality for every domain.

1.5 Workflow of the Thesis

In this thesis, the discussion will be divided into five chapters. The description of each chapter is given below:-

- Chapter 1 - It has a brief introduction for all the topics that this thesis is going to cover. A brief discussion of Machine Translation, Domain Adaptation, and Domain Knowledge is present in this chapter.

- Chapter 2 - All the related work related to this thesis will be discussed in this chapter. It will be a short chapter discussing previous work in the related field.

- Chapter 3 - This will cover the basic baseline of Machine Translation for Indian languages, where the models and their configuration will be discussed in detail. How subwords and their length affect the results of different Indian languages will also be discussed for the baseline models.

- Chapter 4 - This chapter will discuss the definition of domain and the basic approaches of domain adaptation. It will try to cover the same using domain identification and domain terms and how these make each domain different from the others. All the different linguistic aspects of each domain will be covered in this chapter. After this, three basic approaches of domain adaptation will be discussed on the domains that we will be dealing with, and their issues will be discussed in this chapter.

- Chapter 5 - The domain knowledge will be further discussed using sentence embeddings, and how we can use it to make domain adaptation better will be discussed in this chapter. This chapter will discuss how similar domains can be used to make the number of models and the results of these machine translation models more efficient.
- Chapter 6 will be the conclusion of the thesis, which will discuss all the results that this thesis achieved and also will try to look into all the future problems that might be solved using the same.

Chapter *2*

Related Work

In this chapter, we will discuss the work which is relevant to machine translation in Indian Languages, domain terms, and domain adaptation which will help us to understand the following three chapters of the thesis in a better way.

2.1 Approaches for MT in Indian Languages

There are both SMT (Statistical Machine Translation) and NMT (Neural Machine Translation) models for the task. We will discuss two approaches in this part and the common transformer approach over which we are building our baseline model in the next chapter.

2.1.1 SMT

SMT, as described by its name, uses statistics of the source and target sentences to predict the translations. It uses the Bayes decision rule along with statistical decision theory to give the best possible translations. It can be explained through the below-given mathematical expressions, where the translations are calculated through probabilities of the occurrence of words and the language model created by the target language.

$$\begin{aligned} h_{best} &= p(h|e) \\ &= argmax_h p(h|e) \\ &= argmax_e p(e|h) \cdot p_{_{LM(h)}} \end{aligned}$$

IBM started the research[9] and proposed a detailed study on SMT again once a large corpora parallel corpus started to become available. This model can also be explained by a log-linear model, where it uses different feature functions for predicting the translations by the use of some random variables.

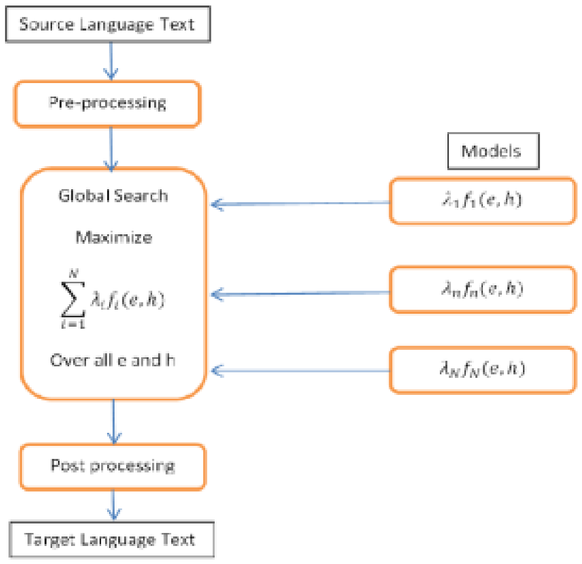


Figure 2.1 Log-Linear Model for SMT[3]

Basic SMT models are divided into two types. The first uses basic word-level translation and then alignment to produce translations. In the second approach, the translation is done on a phrase level, and this helps to get a better quality translation linguistically.

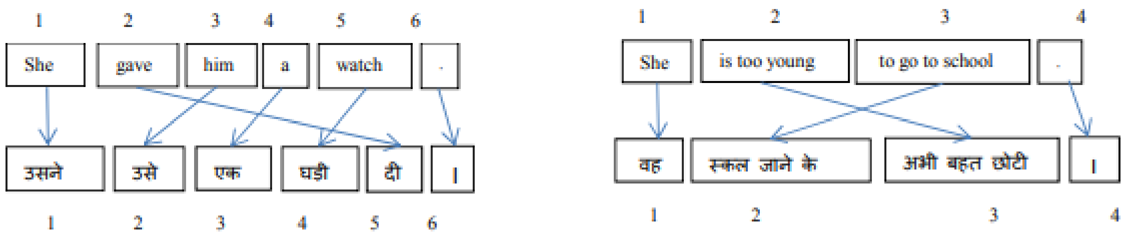


Figure 2.2 Word-Level SMT (left) Phrase-Level SMT (right)[3]

In the paper [41], a model was proposed that produced phrase alignment using phrase-based joint probability. This helped in skipping the step of creating word-level alignment. The paper[71] restricted the window size of the reordering of phrases during alignment. This produced better performance results for Chinese-English translations. There is also a different approach that uses syntax for the SMT model. In the paper[76], the authors proposed a method of transforming the parse tree of one language to another. This method tried to use the linguistic features of syntax, like word order.

A new architecture was discussed in [55]. Here they try to solve the morphological and structural divergence[40]. They used suffix separation and compound splitting to achieve the same. Through suffix separation, the longest matching suffix is taken from the list of all suffixes.

In compound splitting, all the compound words are divided into their constituents. Both the phrase-based models and factored models were used as baseline models. Stem from both the source and target side was used as an alignment factor when factored SMT training was done.

2.1.2 NMT

Basic NMT models [1] use LSTM[28] for tackling the task of translation. One such architecture was provided by [56], where the encoder is divided into two layers. Bi-LSTM is present in the first layer. The second layer has an LSTM. Both the LSTM are basic LSTMs in the decoder. Residual connections help to put the output of one layer and the input of the next layer together to make a new input. In the paper [4], the authors claimed that the bottleneck of quality translation is due to the fixed-length vector that an encoder creates for the decoder to translate and proposed a model 'RNNSearch'. So they proposed a method where the model focused on words that are relevant in the source text for predicting the next word and searched for it on its own. This removed the dependency on the fixed length of the vector and gave better-quality translations. Focusing on select parts of sentences was extended after the introduction of attention by the paper [39]. They provided both a global approach where all words were considered and a local approach where only a select few in a window size is considered, and it gave state-of-the-art results in WMT 14.

[75] introduced GNMT, which provided faster translation with a better performance. They introduced beam search, which helps the output to consider all the words in the source text at the time of translation. Wordpiece and other techniques like length normalization helped this model to achieve such great performance. Through the paper [14], the authors provided a comparative study between RNN and Recursive convolutional neural networks. It discussed how architectures like these perform in different settings and their degrading performance on the length of the input sentence. Subwords were introduced in the paper [63]. This helped in removing the issue of out-of-vocabulary words as it considers different subwords in making up the vocabulary of the data, and subwords can make a new word that is not yet seen by the model. The issue of large target vocabulary is solved by the paper [29], as the authors proposed an approach of taking only a subset of large vocabulary on each update. This helped in solving the issue of large vocabulary and also improving the translation quality in many cases.

2.2 LLMs in Machine Translation

With the introduction of GPT-4[50], one has to talk about different Large Language Models that are present for Machine Translation. XLM-R[17] is one such model which is developed by Facebook. They used a corpus of over 100 languages to train a translational model, which helped

[1] `https://galhever.medium.com/neural-machine-translation-with-transformers-69d4bf918299`

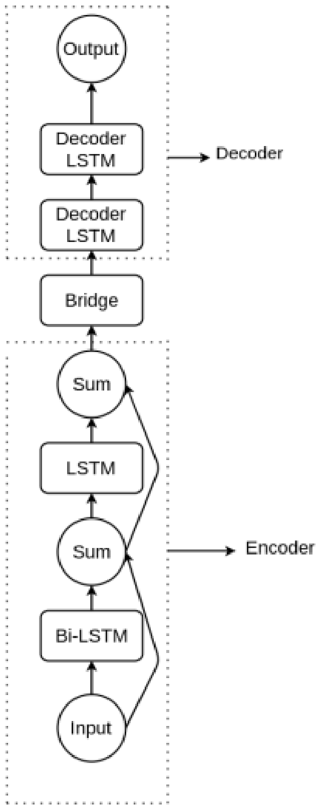


Figure 2.3 NMT Architecture[60]

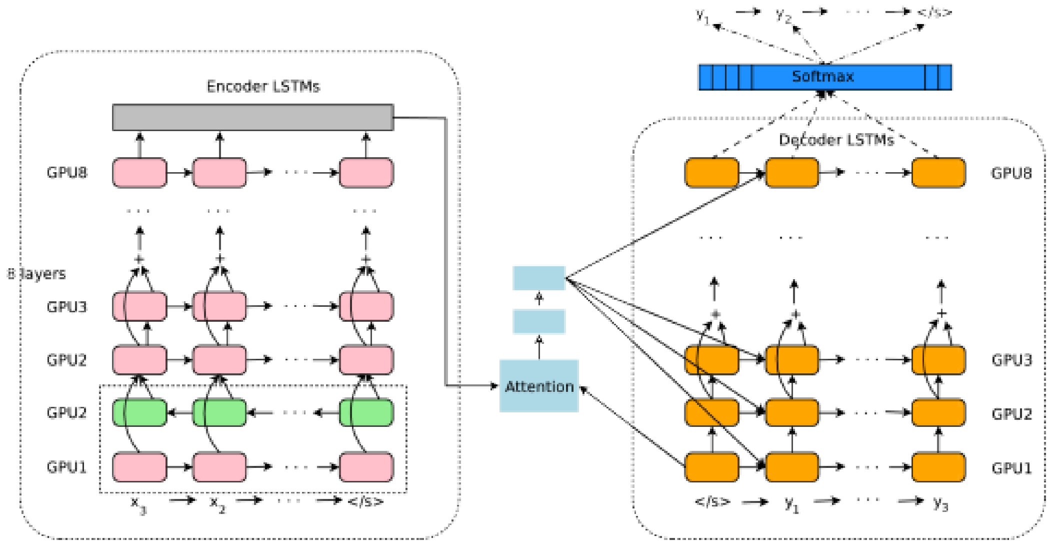


Figure 2.4 GNMT Architecture [75]

to get translations for many language pairs. MASS[66] is another such model developed by Microsoft, which has cross-lingual translation capabilities. mBART[36] was developed by Facebook AI to handle multiple languages with low resources. It was developed on language-specific embeddings, which help it to get good quality translations for all the languages. UniLM[21] was developed by Microsoft and takes care of translations through multimodal inputs. It can be used for training multiple tasks at a time which helps in learning the syntactic and semantic meaning in a better way. GPT-3[11] and GPT-4 are not specifically designed for these tasks but they can be fine-tuned on the data to get translations of a language pair. All of these LLMs gave high-quality results on WMT19[6] and WMT20[5]benchmarks and are based on transformer architecture which will be discussed in the next chapter.

2.3 Domain Classification and Domain Term Extraction

In our fourth chapter, we will talk about domain identification and domain term extraction. To understand the same, this section will discuss some related work in that direction.

Domain classification[2] can be treated as a very basic classification problem, but it is one of the first steps when tackling several other problems. For example, knowing the domain beforehand can help us to choose the best model from a pool of models if we have a number of models to choose from to get the best results. Domain term extraction as a task helps us to define what a domain is as the terms used in a domain become its representation.

This classification problem is tackled through three different types of approaches. These are machine learning models, a rule-based approach, and third which is a hybrid of both. In the rule-based approach, linguistic rules are made on the basis of knowledge of the domains, and that particular language and a rigid structure of rules are made for the same to classify a text into a particular domain. For domain classification, these rules are generally a list of words that are made for every domain that we want to classify a document in, and then the frequency of different terms determines the same. For example, for a domain like Computer Science, terms like Artificial intelligence, Big data, Machine Learning, etc, will make up the list. To form better rules which follow more linguistic patterns, a person with vast domain knowledge is needed.

By the use of supervised machine learning techniques, different associations are learned between all the data points (domain terms) and their respective labels. For example, an ML system can be trained to correctly identify the topic of a news article by presenting the model with thousands or millions of examples from different categories. This learned "knowledge" is either based on intentionally selected features such as bag-of-words representations or tf-idf[78] or on properties of the data discovered by the model itself. It can predict the label for unseen examples as well. Support vector machines, neural networks, and deep learning can be used

[2] `https://www.taus.net/resources/blog/domain-classification-with-natural-language-processing`

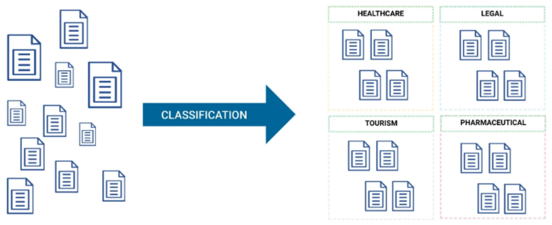


Figure 2.5 Domain Classification Task

for the same. Authors in [33] used attention weights to classify domains as attention scores for each word can give very much information about a domain.

On the other hand, when we see unsupervised systems, the training set does not have any labels which can connect each data point to a domain. Because of this, the model has to learn all the internal features that are seen in the whole dataset. Clustering is one of the main ways of doing domain classification, where sentence embeddings of a text are separated according to different clustering techniques. BERTtopic[25] is one such topic modeling model which helps in identifying topics using sentence embeddings. Latent Dirichlet Allocation[8] is also one such thing where patterns among the words are seen to extract the topics out of many documents. When semi-supervised machine learning ways are discussed, both labeled and unlabeled data points are taken together.

There are many easy-to-use libraries to carry out the above. Scikit-learn[57] gives many options to choose from. Different classifier models and different techniques are well documented and listed to help with the same. When one is using a deep learning technique, it can be carried out using TensorFlow[1] and PyTorch[54].

Automated Term Extraction takes a domain-specific corpus and tries to get domain-specific words out of the same. When handling tasks for knowledge discovery, word extraction methods have the biggest problems with generalization across domains, and there is very less annotated data for the same. Some of the popular datasets for this task are provided below.

Datasets	**Number of documents**	**Number of Words**	**Number of Terms**
GENIA	193	436967	93294
ACL RD-TEC v1.0	10922	36729513	82000
ACL RD-TEC v2.0	300	33216	6818
RSDO5	12	257029	37985

Table 2.1 ATE Datasets

The unsupervised approach of TextRank will be discussed in detail in Chapter 4. There are few machine learning and deep learning approaches. In [77] the classifier was made after extracting features like term frequency with the help of n-grams[12] and stopwords exclusion.

The same approach was put forward by the authors of [18], where they experimented with both linguistic and statistical features. They tried to get information from features like the head of phrases, nouns, n-gram length, term frequency, and domain frequency before putting the data into a classifier.

Authors of [7] used word embeddings in domain-specific medical documents. They tried to stack similar words with similar properties and then use this similarity with a list of technical terms to extract out domain terminologies. [27] used different BERT models to make a simple classifier of term prediction. They took sentences as input and used BERT and other models to predict for each n-gram whether it is a domain term or not, and this became a binary classification problem that outperformed all the other teams in the task.

2.4 Domain Adaptation

Domain adaptation has been tackled in many different ways in recent years. In most ways, the paper tries to modify the basic fine-tuning at some level of architecture. It is generally done in two ways: data-centric and model-centric.

When it comes to data-centric architecture, the best example for multiple domains is [16], where tags of domains were introduced at the time of fine-tuning. This was first introduced in [62] where politeness was given as a tag, and translation was changed according to the tags that were given. The paper [61] discussed three approaches of stacking, concatenating data, and model ensemble, which help to understand the efficiency of various approaches.

When it comes to model-centric domain adaptation, fine-tuning was first introduced by the paper [38]. Through this paper, it was introduced that after training on a general domain, domain adaptation can be done by fine-tuning the same model on a small dataset of an individual domain. Knowledge distillation was used in the paper [19] to keep the score of the general domain high even after fine-tuning the model on a domain.

Mixed domain fine-tuning is another technique that is quite talked about in this area. In this technique, the model is trained on the general domain, and then it is fine-tuned on the data, which is achieved by mixing the domain-specific data and the general domain data. This was shown by [19]. It gives faster training than the multi-domain model with better results than normal fine-tuning on a general domain test set.

One more type of domain adaptation is architecture-centric. Some of the common approaches are fusion techniques in which, apart from a translation model, an LM model is trained for the target language, and the result translation is made through the use of both of these models, which gives a better result for a domain. This was done by [26]. Some other approaches include the one that we discussed earlier, in which domain is specified while training for all sentences, which is considered domain control. In a domain discriminator approach, above the encoder, a domain classifier is built that first verifies the domain and then makes the translations.

Using all the above-related work, we tried to discuss four different approaches for domain adaptation. This will include normal fine-tuning techniques, taking data together, stacking the knowledge, and creating domain similarity clusters to create the best scores.

The above-mentioned approaches for all three tasks will help to understand how these tasks have been solved in the previous times. The aim of the next three chapters will be to provide a new solution for all of these three tasks that are related to domain-specific issues. In the next chapter, we will discuss on building the baseline NMT models for English-Hindi, English-Bengali, and English-Telugu language pairs.

Chapter *3*

Machine Translation for Indian Languages

Machine translation makes it easier for people with different language backgrounds to access information and communicate with one another. This is particularly important in global businesses, where employees, customers, and partners may come from different countries and speak different languages. Machine translation technology can also help in accessing documents in different domains such as Science, Technology, Law, Health, etc. It is well known that the performance of a model trained on generic data drops substantially when a domain-specific document is passed through it. Our primary goal is to create a domain-adapted model for different domains Before jumping to our task of domain adaptation, our goal was to create a good baseline model for an Indian language that gives us good results for sentences that belong to a general domain. There are multiple architectures that deal with training a machine translation model from scratch, and we will study and use transformers for the same. The aim will be to develop a baseline model, and the results will be compared to the Himangy model [45][44][46] so that we can see if our baseline model is giving acceptable scores for an Indian language.

3.1 Introduction

Machine Translation as a task has become one of the biggest challenges for a country like India, where there are so many languages and the sources for the parallel corpus are very few. The efficient translation of most of the content available online for any language will help convey ideas throughout a large and diverse country like India. Machine translation is a critical task in India, where linguistic diversity proves to be a significant challenge. If we can make an efficient machine translation system, it can help break down language barriers and make information more accessible, which is vital in a country as varied as India.

Parallel corpora are the first problem that one faces when dealing with low-resource languages. Our aim was to get as much corpus as possible for machine translation from various domains, which can be used further down the line after our baseline model is created.

Machine translation can be used to train language models and improve the accuracy of translations over time. This is particularly useful in industries where terminology and jargon are constantly evolving, such as the medical, legal, and financial sectors. Machine translation can be programmed to consider cultural nuances and differences, ensuring that translations are not only accurate but also appropriate for the target audience. In this chapter, we build baseline translation models for three Indian languages- Hindi, Bengali, and Telugu.

3.2 Transformers and attention for MT

RNN-based machine translation models gave good baseline models, but they had two major limitations. Firstly, they were not able to tackle the problem of long-range context, and secondly, the dependence of each hidden state on the previous state made it difficult to do the computation in parallel, and this made the training part inefficient. The decoder in this architecture was not able to get all the hidden information out of the processed vector that was made, This issue brings in the concept of attention, where the decoder is able to look at the whole input sequence before generating a word and get the information accordingly. The main issue with all the architectures before the introduction of transformers was that the model worked on the input key-wise individually and not the whole sequence altogether.

In transformers, self-attention[30] is applied between all the words at each step, which helps to extract information and relation between all the words in the sequence input. The basic architecture of the same is given below.

The encoder helps in taking each token in a sequence and comparing it with each and every other word in the same sequence after getting its representation. This gives us an attention score for each word in a sequence. These scores are treated as the weights which will be given to a fully connected network to generate a specific representation for each word. These representations are then used by the decoder to predict every next word in the translation.

So the encoder consists mainly of two things i.e. a self-attention mechanism and a fully connected feed-forward network. All the other hyper-parameters of the same will be discussed later in this chapter.

The decoder tries to predict the next word in every step as it has all the information that is being stored in all hidden states of the encoder. The decoder is mainly divided into three parts. Two things are similar to the encoder i.e., a fully-connected feed-forward network and multi-head self-attention. But in the decoder, this attention is added with masking. When the training is done, it helps to hide the words that have not yet come. So attention score is calculated among the words which are predicted till that step, and all other words after that point are masked so that we can hide this information from the decoder. This helps the decoder to learn the information and not just give out the target sentence again and again.

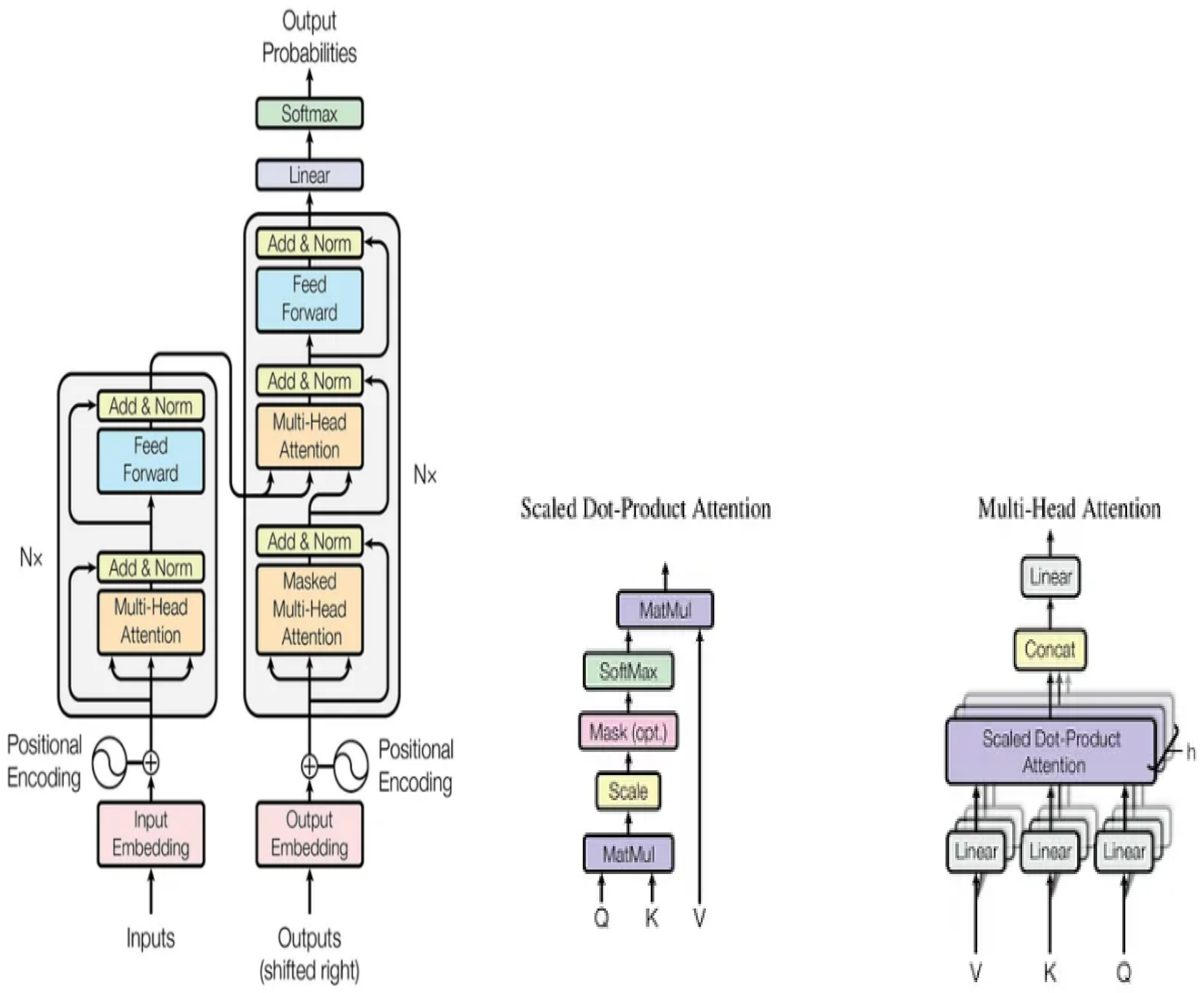


Figure 3.1 Basic Transformer Architecture [70] (left) Scaled Dot-Product Attention which takes Query, Key, and Value (center) Multi-Head Attention helps to complete steps in a parallel (right)

Transformers process the input as a single piece, so the positions of the words in an input can lose their meaning if not preserved. Here the concept of Positional Encoding[13] was introduced. This is made up of sine and cosine functions, and it helps to store the index of each and every word. This helps both the encoder and decoder to maintain the exact location of each word.

To calculate attention [1], one needs to get three things :

- A query vector is a word vector and also the representation in the decoder's hidden states.
- A key vector is for all the words in an input and also the representation in the encoder's hidden states.
- A value vector that gives the attention weights in the hidden states of the encoder.
- The attention score can then be calculated between each word using

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.1}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O \tag{3.2}$$

At the start, the transformer creates embeddings for each word. After that calculation of key, query, and value vectors are done to get the attention scores that create a relation between every word. The decoder then takes the representations generated by the encoder and the last output word of the decoder to generate the next output word of the translation.

3.3 Subword

This section is being discussed separately because subword tokenization played a huge role in the score of baseline models for Indian Languages. Byte Pair Encoding[63] (subword tokenization) is a technique used so that one can tackle the problem of out-of-vocabulary words. BPE was initially used as a data compression algorithm for images. With time the idea was adopted in NLP. Adjacent characters are taken from the text, and the ones with the most frequency after merging are taken as the list of reduced vocabulary. The idea is that this list can act as a better generalization, and most of the words can be made by using this list of subwords[72]. So even if an unknown word is seen by the model, it is not treated as an out-of-vocabulary word.

If there are many subwords, it can increase the complexity of the model. This will show an effect on the training time and also the results. So the number of subwords becomes a very big factor for different languages. This is due to the number of merge operations being different in different languages to form a complete word.

If the number of subwords is kept very less, it can lead to a list that can not form all the words that are going to be encountered during the training. So that language will face a huge case of out-of-vocabulary words. If the number of words is increased beyond a certain number, then the words will start losing their contextual meaning, and languages that have many compound words will start to get affected. As the properties of Indian Languages change from language to language, we will see how the length of subwords changes for different languages.

[1] `https://galhever.medium.com/neural-machine-translation-with-transformers-69d4bf918299`

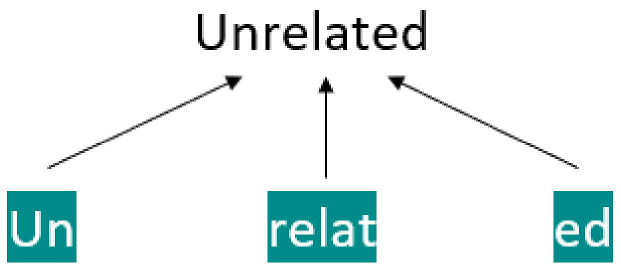


Figure 3.2 An example of subword

In the figure below, one can see if we have *Un, relate, and ed* in the list of our subwords. When the word Unrelated is encountered in a sentence, it can be seen as a word made up of these different subwords. The generalization part of the subwords can also be seen through the same as we can see that many words do have these subwords in them, and they will be recreated using one of these subwords with the others present in the list.

The above-mentioned issue with the subwords will be discussed later in this chapter when we will discuss baseline models for different Indian Languages.

3.4 Fairseq Toolkit

Fairseq toolkit[51] is being discussed in this thesis as all the approaches discussed related to machine translation are done using this toolkit. It is an open-source toolkit made for seq-to-seq models. It is developed by Facebook AI research and can be used for many applications. Some of its key features are:-

- The main benefit of using this toolkit is the command line support that it provides for different approaches from training to evaluation.

- It provides optimized training on multiple GPUs, which is a very big advantage for a task like Machine Translation where a model is trying to learn representations from scratch from a very big dataset.

- It provides an easy approach for subword tokenization and a modular code base where we can make our changes according to our approach.

3.5 Experimental Setup and Training

In this section, we will discuss all the parameters and the training details for our baseline machine translation models. We will first talk about the datasets used for all four languages, i.e., English, Hindi, Bengali, and Telugu. After that, the basic parameters that are set for training will be discussed, along with what different subword lengths were set for different languages. Finally, we will have three models which perform decently on a general domain for machine translation tasks. These models will then be used for domain adaptation and for other approaches that will be discussed in Chapter 5.

3.5.1 Datasets

Three languages were chosen to create a basic machine translation model from the English Language. These were Hindi, Telugu, and Bengali. The Hindi dataset was taken from WAT 2021 task [2]. The other two were a result of online scraping and cleaning. The key points regarding these three datasets were:-

- The dataset used for English-Hindi translations was part of IITP-MT at WAT2021. It had a total of 3,069,725 parallel sentences.
- These translations were compiled from different datasets, from open subtitles to IIT Bombay dataset[35] it had a diverse list of sources.
- For the Bengali dataset, there were 62,07,556 parallel sentences for creating the baseline.
- For the Telugu dataset, there were 40,29,431 parallel sentences for creating the baseline.
- The test data was kept constant for all three languages,i.e., 2500 sentences.

	Number of Sentences	**Number of Tokens**
Train	3069364	67454966
Valid	1500	21528
Test	2500	51536

Table 3.1 Hindi Dataset Distribution

	Number of Sentences	**Number of Tokens**
Train	4025431	54636143
Valid	1500	20177
Test	2500	34413

Table 3.2 Telugu Dataset Distribution

	Number of Sentences	**Number of Tokens**
Train	6203056	77357991
Valid	1500	23988
Test	2500	31322

Table 3.3 Bengali Dataset Distribution

3.5.2 Hyper-parameters

There are many different parameters when it comes to training a transformer a scratch. A few important ones are discussed below :

- The model was trained for 20 epochs for all three languages. Max tokens were kept as 4000, after which we were facing memory issues.
- Dropout[67] was kept at 0.5 to avoid overfitting on the training data. Attention dropout was kept at 0.1 for the same reason.
- Dimensionality of the input embeddings for both encoder and decoder was kept as 512. It was the same for the output. The dimensionality of the feed-forward network was kept as 2048 for both the encoder and decoder.
- Number of the encoder and decoder attention heads was kept at 8.
- Adam optimizer[34] was used for its less memory usage and adaptive learning rate. Initially, the learning rate was defined as 0.0005. The loss function used was LabelSmoothedCrossEntropy[48], which is a normal cross-entropy loss.
- Warmup[23] was used to make the model more familiar with weights when the training starts initially. Patience was also used for
- 4 NVidia GeForce 2080 Ti GPUs were used on the college server to perform these training tasks.

3.5.3 Subword Preprocessing

In preprocessing, subword tokenization was carried out. It helped to break the training data into characters which can be later used to rebuild words so that we can tackle the problem of out-of-vocabulary words. For English, Hindi, and Bengali, the number of subwords was kept at 30,000. But the issue arises as we bring Telugu into account. As Telugu is an agglutinative language, there are many words that are taken together and form one complete token. So we discussed different approaches by varying the subword vocabulary size in the Telugu language, the first one where the number of subwords is same as the other two languages and the second one in which there were only 9000 subwords. This drop in the number of subwords was captured in the final scores, which help us to understand how different properties of a language can change the quality of results in machine translation.

Language	**Number of Subwords**
English	30000
Hindi	30000
Bengali	30000
Telugu (Exp-1)	30000
Telugu (Exp-2)	9000

Table 3.4 Number of Subwords for Different Languages

3.6 Results and Observations

Testing for all three language pairs was done on 2500 sentences. The metric for testing was BLEU score[53]. It's a very common metric in the task of Machine Translation where the number of matching n-grams between predicted output and reference output is used to give the score. The higher the score better is the translation quality. Following are the other key details regarding the results

- Himangy model of Language Technologies Research Center (LTRC) - IIIT Hyderabad was used to compare results for Hindi. Himangy model is trained on more data, so our aim is to get comparative results so that we can do domain adaptation on our model.
- FLoRes Dataset[24] was also used to evaluate the models as for Hindi we didn't want any bias created by WAT dataset over which it was trained. There were total 1012 sentences in this test set.
- On the WAT test set, our model gave similar results to the Himangy Model and gave BLEU score of 29.53 in comparison to 31.29. The same was the case with the Flores dataset, where our model gave a score of 25.63, and Himangy gave 26.56.
- Bengali and Telugu didn't give results as good as Hindi. This is due to the lack of quality in the training data. Bengali gave a BLEU score of 20.1 and Telugu gave BLEU score of 19.2 .
- The main difference that the subwords bring for Telugu was the jump that it gave from the first baseline model. When we used 30000 as our number of subwords the score was 15.3 . It jumped to 19.2 when we decreased the number to 9000. This tells us the property of Telugu is an agglutinative language. Where the words from different categories fuse

together to make one token. Breaking the vocab into many subwords was just increasing the complexity for the model and not helping with the language as the tokens stick together in this language.

- As Hindi doesn't follow the same, it gave great results even on 30000 subword length. Now these models will be used to infer results in other domains.

Model	**Dataset**	**BLEU Score**
Himangy Model	WAT	31.29
English-Hindi Baseline	WAT	29.53
Himangy Model	FLoRes	26.56
English-Hindi Baseline	FLoRes	25.63

Table 3.5 Results for English-Hindi Translation

Model	**BLEU Score**
English-Bengali Baseline	20.1
English-Telugu Baseline (Subwords=30000)	15.3
English-Telugu Baseline (Subwords=9000)	19.2

Table 3.6 Results for English-Bengali and English-Telugu Translation

3.7 Discussion

Before starting the training of the baseline models, it was found that the quantity of data is a big factor in the quality of output in machine translation. To check the pipeline, the English-Hindi model was first trained on only 3 lakh sentences. The model was not able to learn anything and gave a random word as its output. So, the final English-Hindi baseline model was built on around 3 million sentences. The output quality of the English-Hindi language pair was high, but when we jumped to English-Telugu, we found out that it was not only the quantity of data that made the quality of output better as the quantity of English-Telugu sentences was more than English-Hindi. Different numbers of subword tokenization were discussed to improve these results. Telugu being an agglutinative language, was not giving good results on a high number of subwords as the words were getting split into more subwords for the Telugu language due to its agglutinative nature, and increasing the number of subwords was only increasing the complexity of the model. We reduced the number of subwords to take the nature of the language into account. This change in the number of subwords resulted in an improvement in translation quality.

English-Bengali gave the best quality in the same experimental setup as English-Hindi, but the difference in the quality for both languages came due to the high quality of the English-Hindi dataset. This quality drop is due to the presence of symbols at the time of subword tokenization. So this helped us to understand that the translation quality of a machine translation model depends on the quantity and quality of the dataset along with the nature of a language.

3.8 Conclusion

In this chapter of the thesis, we discussed how to make a machine translation model from scratch. We did the same for three language pairs,i.e., English-Hindi, English-Bengali, and English-Telugu. We observed that the English-Telugu model performed differently than the other two models on a different number of subword tokenization. The one with lesser subwords was taken as a better model. English-Hindi baseline gave very good results, so most of the approaches in the next two chapters will be carried out on the same. The other two models will be used to create a fine-tuned model on a single domain to observe the effects of the same on these languages. In the next chapter, we will discuss domain knowledge and basic domain adaptation before going into our approach of domain adaptation in Chapter 4. This will be done using the knowledge of domain terms and domain identification tasks.

Chapter 4

Domain and Domain Adaptation

In this chapter, using the baseline models created in the previous chapter, we will try to get a basic understanding of domain adaptation. We will define what a domain is and how domain terminologies play a role in a specific domain. The issues in using the baseline models will be discussed for a domain-specific translation. To solve this issue, we will discuss two domain-related tasks: Domain Classification and Domain Term Extraction, which will help us to understand the basic definition of a domain through its properties. Basic techniques of domain adaptation will be discussed to help our baseline model to give better performance on domain-specific translations. Through both of these approaches, we will try to understand how different domains can affect language, and then we will dive into the discussion related to domain adaptation in these domains.

4.1 Introduction

As mentioned earlier, when a translation model is trained, it is able to learn basic language rules and representations, but if the model is given a domain-specific document, the performance is bound to drop. The goal of these approaches discussed in this chapter is to translate a domain-specific document with great performance without affecting the performance of other domains. As the domains start to increase, the complexity of this problem starts increasing.

In this section, we will first discuss the domains and the dataset that we are going to use. Domain-specific data extraction is also done. Data preparation for different domains was an important part of the experiment, as there is a lack of parallel data for Indian Languages when it comes to specific domains. To understand the domain, we will try to extract terminologies related to each domain. In this chapter, we will discuss an approach that solves the issue of domain term extraction. The approach is a modified version of the TextRank[42] algorithm, which is based on the PageRank[52] algorithm. We will also modify the TextRank approach, which will help us to take the domain knowledge into account while tackling this problem. After understanding the importance of terms in translation, we will start the discussion related to basic approaches for domain adaptation. Our basic domain adaptation techniques will be able to solve the problems faced by our baseline model, but we will face some other issues with each of these techniques. Some important points related to this setup are:

1. The domains that we focused on were: **Governance, Chemistry, Artificial Intelligence, Corporate Law, Medical, Information Sheets,** and **Consent Forms.**

2. For English-Bengali and English-Telugu, only Information sheets and consent forms were used to validate the fine-tuning process. All the other domains were used for English-Hindi translations. This chapter will try to understand the quality of output after each approach is discussed and will address the issues in the next approach. We explored three approaches for domain adaptation :

   - Fine-tuning every domain separately, creating n different domains for n different models, which is inefficient when we address multiple domains, but it helps us to get domain-specific models.
   - In the second approach, we try to create only one model for multiple domains by taking all the data together.
     - The above approach was carried forward by stacking up the domain knowledge one after the other instead of taking all the domain data together. These approaches had issues like catastrophic forgetting, which affected the output quality of earlier domains.

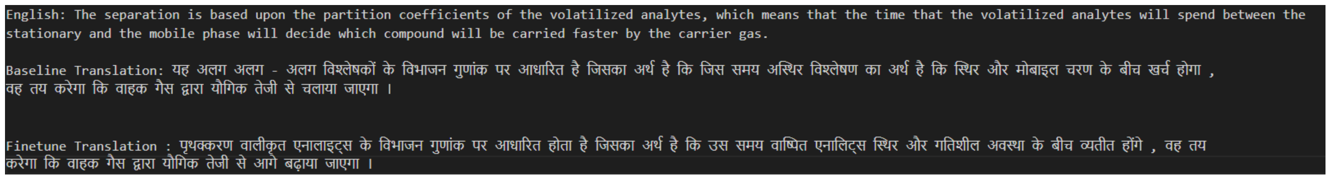
We will look into the questions like What do we mean by "domain" and how domain terminology plays a role in this? After that, we will explain how different approaches can be used to carry out the task of domain adaptation in Indian languages and what are the different ways to make domain-specific models and only a single model for all the domains.

4.2 What is a Domain?

A domain is a distinct subset of sentences that belong to a particular field of study. As every word is given its meaning through embeddings in a model, the sentence embeddings which we have can be grouped into different domains. So when a model is trying to give better performance over a domain, it is trying to learn the translations of these embeddings when that domain is given as its context.

4.2.1 Issue with No Domain Adaptation

When we try to use only a baseline model for a domain-specific task, the main issue that our model faces is the issue of domain-specific terminologies. One such example of the chemistry domain can be seen below:



```
English: The separation is based upon the partition coefficients of the volatilized analytes, which means that the time that the volatilized analytes will spend between the
stationary and the mobile phase will decide which compound will be carried faster by the carrier gas.

Baseline Translation: यह अलग अलग - अलग विश्लेषकों के विभाजन गुणांक पर आधारित है जिसका अर्थ है कि जिस समय अस्थिर विश्लेषण का अर्थ है कि स्थिर और मोबाइल चरण के बीच खर्च होगा ,
वह तय करेगा कि वाहक गैस द्वारा यौगिक तेजी से चलाया जाएगा ।


Finetune Translation : पृथक्करण वालीकृत एनालाइट्स के विभाजन गुणांक पर आधारित होता है जिसका अर्थ है कि उस समय वाष्पित एनालिट्स स्थिर और गतिशील अवस्था के बीच व्यतीत होंगे , वह तय
करेगा कि वाहक गैस द्वारा यौगिक तेजी से आगे बढ़ाया जाएगा ।
```

Figure 4.1 Issue without Domain Adaptation

Our model is able to form a representation for these domain terms after domain adaptation. Through the next section, we will be able to see how a domain is defined. Through the task of domain classification using the domain terms, we will show how a domain is defined due to its terminologies, and through domain term extraction, we will be able to show how under different domains, terminologies have different properties. Using the embeddings of these terms, we will propose a new method of domain adaptation.

4.2.2 Domain Classification

On several distinct natural language processing (NLP) benchmarks, the transformer-based language models have been making encouraging progress. Modern NLP is increasingly using transfer learning techniques in conjunction with large-scale transformer language models, which has produced a number of cutting-edge models. The main limitation of a biLSTM is the lack of parallel training due to its sequential nature. The attention mechanism in transformers takes care of this issue. This helps in creating better representations in a model. Text-Classification is a very general task in NLP. This helps in providing the document-level metadata automatically. We will discuss the task if Fine-Grained Domain Classification. This task deals with domains that are closely related to each other and lie within a particular domain, for example classifying between mechanics, relativity, and rotational mechanics inside physics. We will deal with this task. In this task, a model can get confused between two domains as they are very closely related to each other.

This approach[22] discusses the use of transformers and provides a comparative study between BERT[20] and RoBERTa[37] performance for the task of fine-grained domain classification. The pre-processing of data proved to be a major difference for a task like fine-grained classification compared to the coarse-grained one. Shared vocab among the domains was removed along with stopwords which helped in the removal of a lot of terms. Removal of these terms, which were common in these similar domains, helped the model to differentiate between the closely related domains. On a blind test corpus, the proposed model gave an F1 score of 0.824. The test was done on 1929 samples, and the model outperformed all the other submitted models.

4.2.3 Domain Term Extraction

Ranking algorithms that use graphs are used quite frequently for ranking/ordering task. Ranking webpages or even analyzing citations are examples of these tasks. PageRank was developed by a team at Google and is one such thing that helps rank webpage searches. Global knowledge is used to make local decisions to make up a graph and get information out of it. The importance of a node is determined by recursively going through other nodes and gaining information. This graph-based approach is now applied in other areas as well. In Natural Language Processing, this has been introduced as TextRank. It is used for different tasks like summarisation and word sense disambiguation. We will discuss the approach for domain term extraction and modify the algorithm to get better results. The whole text will be considered while making this graph.

Through this approach[58], we introduced a new step in the TextRank algorithm. This helped us to complete the task of domain term extraction in an unsupervised manner. We consider the fact that domain terms can be multi-word, so we introduced bigrams and trigrams with a weighted approach to the TextRank algorithm. POS tagging filter was also used to get better results.

4.2.3.1 Pre-processing and Data

We applied this method to 800 domain-specific documents. This was provided to us by ICON TermTraction 2020[65]. There were in total four domains: Bio-Chemistry, Communication, Computer Science, and Law. The aim was to get a list of domain terms out of all these documents for each domain.

Standard pre-processing was done as an initial step. Documents were tokenized and non-essential punctuations were removed. POS tagging was an important step in this approach. Only Nouns and Verbs were kept in the document because it was taken as an assumption that domain terminologies are either a Noun or a Verb. This helped us to increase the F-1 score significantly.

4.2.3.2 TextRank

Just like HITS[43], TextRank is a ranking model which is based on graphs. It can be used to process text and find the most relevant keywords in a document. This algorithm is based on PageRank.

PageRank is used to find relevant importance among some webpages by ranking all the webpages linked together by making a graph. All web pages are taken as a directed graph, where each web page act as a node. If webpage A is linked to Web page B, a directed edge is considered between A to B. The following formulae are used to assign weights to each node after constructing the whole graph :

$$S(V_i) = (1-d) + d * \sum_{j \in In(v_i)} \frac{S(V_j)}{|Out(V_j)|} \tag{4.1}$$

In the above equation, S is the weight of each webpage, and d is the damping factor for a node. $In(V_i)$ and $Out(V_i)$ represent all the incoming edges and outgoing edges, respectively, for a node V_i .

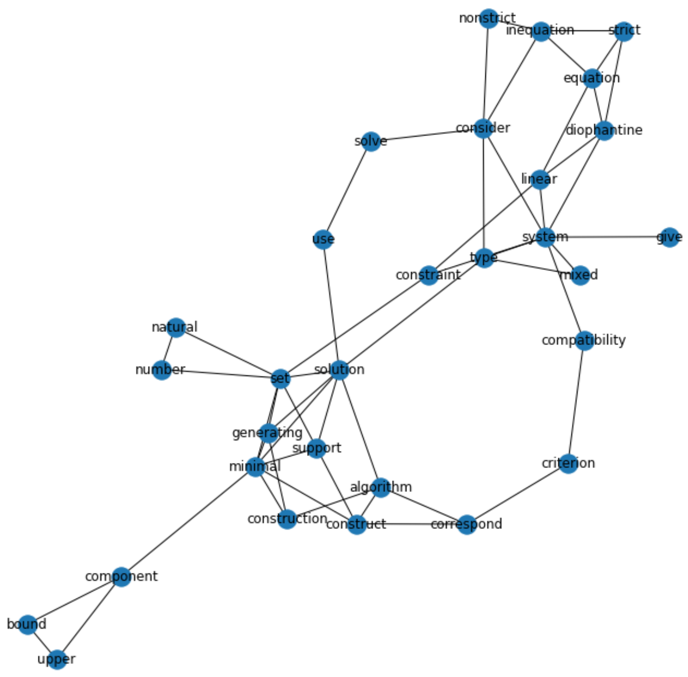


Figure 4.2 An example of TextRank Algorithm

TextRank [1] is similar to the PageRank algorithm. Here the nodes are words of a document instead of webpages. A word graph is created to find important keywords from a document. Consecutive words are taken to make a directed graph. If a word lies in a window size (that we provide) of another word, there is a directed edge between those two words. The more frequent the connection higher will be the weight. The results can be skewed because of many common words among documents. This made the preprocessing step very important.

4.2.3.3 Implementation

The following steps were taken to extract domain-specific keywords from the documents in an unsupervised way :

[1] https://derwen.ai/docs/ptr/explain_algo/

- Documents were split into sentences on the basis of end-of-sentence punctuations and then further tokenized using SpaCy tokenizer. Stop words were removed to make the list of extracted words more specific.
- In the next step, all the sentences were POS tagged for filtering out irrelevant words. The assumption behind the same was that in any domain, the domain terminologies are either Nouns or Verbs. So only these two tags were kept to make the list more exclusive.
- From these tokenized sentences, we made a list of Unigrams, Bigrams, and Trigrams, and each of the n-grams was treated as a different node. We started our experiment with a window size of 4. So for every n-gram, four words are considered around it. Increasing the window size to a big number increased the execution time and did not affect the results. Two unrelated words were also getting an edge due to the larger window size.
- Damping factor was kept at 0.85, which determines how much importance is given to the weight.
- Once the graph is built, weights are calculated for each node. The weight represents how important the word is in a document.
- There are many domains where multi-word terms are the domain terminologies. So to consider this factor, weights were given to Unigram, Bigram, and Trigram nodes i.e. if we think that the domain contains more bigrams terms, we can change the calculation just by changing the importance of Unigram, Bigram, and Trigram nodes. This weighting system was introduced to the TextRank algorithm in this paper. This improved the results drastically for bigrams and trigrams terms.
- If we have a domain expert, these weights can be played around with to get a better understanding of Unigram, Bigram, and Trigram distribution.

4.2.3.4 Results and Evaluation

We were given 10 documents from each domain - Bio-Chemistry, Computer Science, Communication, and Law. Three different runs of experiments were done for the same. The weights for each run are given below.

We returned only 20 results for each document, due to which our recall was low. Increasing the number of results will increase the recall, but we wanted to stick to only the top 20 terms for each domain. The low recall was also noticed for domains like Computer science and Bio-Chemistry, where the length of the documents was greater than the other domains.

Our model attained the best scores in the task for the Law domain. If more experiments are done with the weights and length of the document, then high scores can be achieved for other domains as well. For domains having longer documents, lemmatization[32] improved the score

Domain	**Run**	**Precision**	**Recall**	**F-1**
Law	*Run*1	**0.4**	**0.32**	**0.355**
	*Run*2	0.266	0.32	0.29
	*Run*3	0.133	0.285	0.181
Communication	*Run*1	**0.25**	0.208	0.227
	*Run*2	0.233	**0.291**	**0.259**
	*Run*3	0.1	0.125	0.111
Computer Science	*Run*1	0.251	0.13	0.174
	*Run*2	0.3	0.134	0.185
	*Run*3	**0.466**	**0.152**	**0.229**
Bio Chemistry	*Run*1	**0.501**	0.131	0.208
	*Run*2	0.3	0.173	0.219
	*Run*3	0.466	**0.184**	**0.264**

Table 4.1 Results for different N-gram Weights

Runs	**Bigram Weight**	**Trigram Weight**
Run1	1.8	1.5
Run2	1.8	2.5
Run3 with lemmatization	1.8	2.5

Table 4.2 Different weights for each run

in Run3 of this approach. The negative impact of lemmatization is on those domains where the same root words are used differently in different places.

4.2.3.5 Discussion

From the above two tasks, we found out the following things :

1. By addressing the issue of shared vocabulary in fine-grained domain classification, we observed how a domain is defined by the terms it uses. Keeping those shared terminologies aside helped to define a domain without them and helped in the classification of domains in a better way.

2. When it came to domain terminology identification, we are able to understand how different domains have different properties. The distribution of these terms is different for different domains.

3. For a domain like law, a lower trigram weight helped in getting the best output. This shows a lack of trigram of terminologies in this domain.

4. For domains like Communication, Computer Science, and Bio-Chemistry, higher trigram weights helped in the output. This shows us the importance of multiword terms in these domains.

5. Lemmatization helped in domains like Computer Science and Bio-Chemistry, but it reduced the resulting quality a lot in Communication. This shows the use of the same root word in a different context is more in domains like Communication. On the other hand, lemmatization helped in reducing similar terms in Computer Science and Bio-Chemistry, which helped us to extract more domain-related terminologies.

6. Both of these tasks helped us to understand how a domain is defined and the importance of domain terminologies for each domain.

Now that we have seen that domains are defined by the terms used in the sentences of that domain, we will try to discuss some basic domain adaptation techniques and what are the issues with these techniques. Through this knowledge of term embeddings and domain adaptation, we will propose a new technique of domain adaptation for multiple domains.

4.3 Domain Adaptation for Different Domains

We will start this section with a discussion of data from different domains. As our aim is to carry out domain adaptation on different domains together, we had to extract some data from different sources to complete the experiments. This extracted data was augmented to the already present domain-specific data.

4.4 Data Extraction

Only medical and governance domains were focused on when extracting the data. All the other data were taken from other sources where domain-specific data was present. Governance data is extracted from government magazines that discuss different government policies and IAS coaching center study material related to the same. Medical domain data is extracted from government medical websites and other information sheets related to different diseases. The distribution is given in the table below:

4.4.1 Data Distribution for Each Domain

Finally, all the approaches were discussed in six different domains,i.e., Governance, Chemistry, Artificial Intelligence, Corporate Law, Medical, Information Sheets, and Consent Forms. Training points were kept similar for each domain so we don't face any bias for different domains. The subword tokenization was kept the same as the baseline. For Bengali and Telugu,

Domain	**Number of Sentences**
Governance Magazines (Translations created through Himangy)	11,149
Medical Websites	1000
Medical Information Sheets (Cleaning)	3000
Governance Related PDFs	2000
IAS Study Material	1000

Table 4.3 Data Extracted from Different Domains

fine-tuning was done only on the Information Sheets and Consent Forms. The distribution of all these data is given below:

Set-Split	**Number of Sentences**
Training Set	2525
Dev Set	300
Test Set	500

Table 4.4 Data Distribution for English-Bengali Information Sheet Data

Set-Split	**Number of Sentences**
Training Set	4722
Dev Set	300
Test Set	500

Table 4.5 Data Distribution for English-Telugu Information Sheet Data

Domain	**Training Set**	**Dev Set**	**Test Set**
Artificial Intelligence	4762	400	500
Chemistry	4884	300	495
Corporate Law	4807	300	500
Medical	4599	301	500
Governance	4200	354	600
Information Sheets and Consent Forms	5000	400	490

Table 4.6 Data Distribution for English-Hindi across different domains

4.4.2 Hyper-parameters

Mostly all the parameters were kept the same as the baseline model. The main parameters are mentioned below:

- The model was fine-tuned for 10 epochs, and the max token was kept as 4000.

- Dropout was kept the same as the training phase, i.e., 0.5. The same is the case with attention dropout at 01.

- Input embeddings of the encoder and decoder had 512 as their dimensionality. Feed-forward network had a dimensionality of 2048.

- Attention heads for both encoder and decoder were kept as 8.

- Optimizer, initial learning rate, and the loss function remained the same, i.e., Adam, 0.0005, and LabelSmoothedCrossEntropy, respectively.

- Fine-tune model was loaded with the same vocabulary and subwords tokenization was used to tackle the issue of out-of-vocabulary words of new domains. For the same reason, the same subword list was used at the time of pre-processing the data for the transformer.

- 4 GPUs were used, and the fine-tuning was done on NVidia GeForce 2080 Ti.

4.5 General Fine-Tuning Results on Bengali and Telugu

We will first check the effects of fine-tuning on Bengali and Telugu languages. Some points that were noticed during the fine-tuning were :

- The data that we are fine-tuning are consent forms and information sheets of the Medical area. It is taken at the time of admission or when tests are conducted.

- For Telugu, the Himangy model was also considered in this test set of forms and sheets. It gave a BLEU score of 20.79.

- BLEU score for Telugu increased by almost 10 points from 15.9 to 25.47.

- For Bengali, this increase was also similar, i.e., from 16.8 to 26.64.

- Fine-tuning both models helped us to get a better score than the Himangy model.

Model	**BLEU Score**
Baseline	19.2
Fine-Tuned Model	**25.47**
Himangy Model	20.79

Table 4.7 Results of Fine-Tuning on Information Sheets and Consent Forms Domain for English-Telugu

Model	**BLEU Score**
Baseline	16.8
Fine-Tuned Model	**26.64**

Table 4.8 Results of Fine-Tuning on Information Sheets and Consent Forms Domain for English-Bengali

Fine-tuning for English-Bengali and English-Telugu models gave us good-quality outputs on the domain-specific test set. This shows that for a single domain, fine-tuning proves to be a successful method to get better quality results. We will carry out the same method in the next section for 6 different domains for the English-Hindi pair and build on different approaches on its basis.

4.6 Fine-Tuning on Hindi and Creating Domain-Specific Models

As mentioned earlier, the first approach is to create n different models for n different domains. This provides us with domain-specific models for use but is very inefficient for training and memory as we are making new models for any new domain that we encounter. The results of the same are as follows:

- Each model is compared with the results of the Himangy Model for every domain. All our fine-tuned models outperformed the Himangy model except in the domain of Artificial Intelligence.
- The highest increase is seen in the score of the Medical domain. The score jumped by 24 points.
- On the other hand, Corporate Law had the score increase by 3 points only.
- This approach resulted in us making 6 models for 6 domains. If we increase our domains with time, this method will become very inefficient.
- In the next approach, we will try another method by creating only a single model for all the domains.

4.6.1 Discussion

We observed better-quality outputs for each domain and were able to outperform the Himangy model in most of the cases. But, this is a highly inefficient method as we are increasing our number of models with every domain, which is not a memory-efficient technique. This

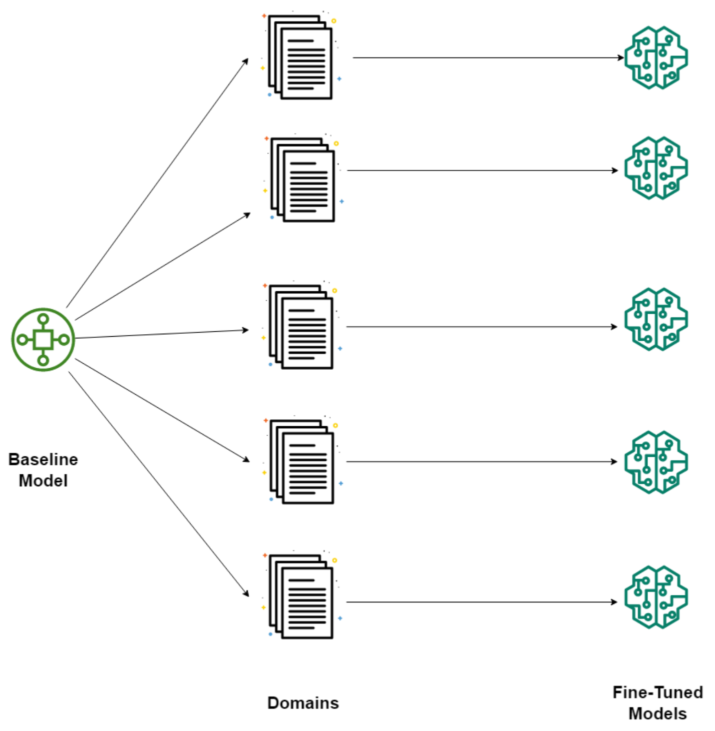


Figure 4.3 n-domain n-model method

Domain	Himangy	Baseline	Fine-Tune Model(Exp-1)
Artificial Intelligence	40.03	28.22	36.57
Chemistry	27.77	21.8	29.84
Corporate Law	21.18	23.52	27.33
Medical	32.52	21.1	45.21
Governance	44.96	38	50.18
Information Sheets and Consent Forms	21.7	23.58	41.7

Table 4.9 Result of Domain-Specific Fine-Tuning for English-Hindi across different domains

technique is also highly dependent on the domain classification task. If a document is classified in the wrong category, its translation quality will be very poor because we have a model for each domain, and there is a chance that the document is translated using a different model. The quality of the output after fine-tuning does not outperform the Himangy model in a huge way, so our aim in the next approach was to tackle the issues that we talked about above and also increase the quality of the output.

4.7 Fine-Tuning on Hindi and Creating Single Models for n Domains(Taken Together)

Through this method, we take all the domain data together and fine-tune a single model over it. The results of this approach were worse than single fine-tuning:

- For most of the models, we were able to able to get a better score than the Himangy model. For the corporate law domain, the score slightly decreased from the baseline model.
- This time, the highest jump was for the Medical domain. The score went from 21.1 to 36.94.
- The results were worse than the last method for every domain, but in this way, we are able to limit our number of models to 1, which makes this type of domain adaptation very efficient.
- In the next approach also, we will not use all the domain data together and will try to stack up the information one after the other and see the results.

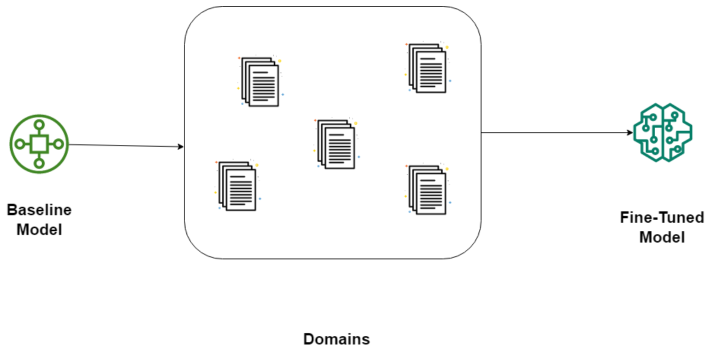


Figure 4.4 n-domain 1-model taken together method

Domain	**Himangy**	**Baseline**	**Fine-Tune Model(Exp-2)**
Artificial Intelligence	40.03	28.22	29.67
Chemistry	27.77	21.8	28
Corporate Law	21.18	23.52	23.26
Medical	32.52	21.22	36.94
Governance	44.96	38	45.26
Information Sheets and Consent Forms	21.7	23.58	33.3

Table 4.10 Result of Exp-2 Fine-Tuning for English-Hindi across different domains

4.7.1 Discussion

Through this method, we were able to tackle the main issues in the first approach. We created only one model, which made the process memory-efficient, and by this approach, dependence on domain classification became minimal as all the documents will be translated through only one model, and misclassification will not affect the translations. There are two issue that arises in this type of setup. First, the translation quality drops if compared with when each domain was given individual models. The other issue is whenever a new domain will come, the data will be augmented with other domain data, and the fine-tuning is done on the baseline model again. The next method tried to see through these issues.

4.8 Fine-Tuning on Hindi and Creating Single Models for n Domains(Stacking One After Another)

In this approach, we create a single model only, but instead of taking all the domains together, we stack the domains one after another on the model and see the results on the domain test data on each iteration:

- Flores dataset is also taken in this approach as the test set.
- In this method, we faced the issue of catastrophic forgetting[31]. After adding a new layer of information on the model of a new domain, the translation quality of the domains that it learned beforehand reduced very much. This is an issue of deep learning where the model starts to forget old information when it starts to learn new information after each iteration of fine-tuning.
- This approach is very bad for the domain that is used at the start. As the translation quality started to degrade, this method was only carried out for six domains.

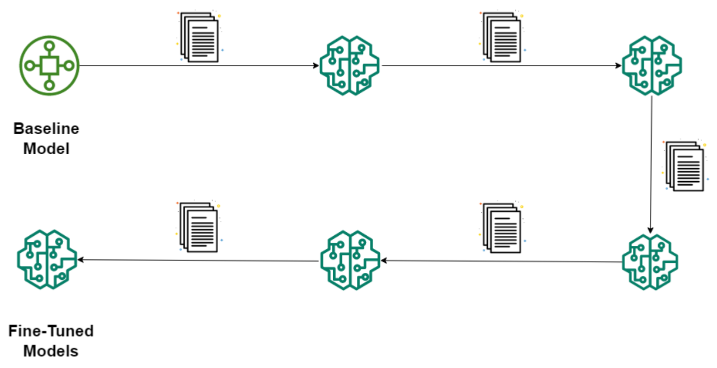


Figure 4.5 n-domain 1-model stacking method

Domain	Base	Gov	Gov-Chem	Gov-Chem-AI	Gov-Chem-AI-Corp-Law	Gov-Chem-AI-CorpLaw-Med
Flores	25.63	23.73	19.71	18.56	19.97	18.4
Gov	38	50.18	35.97	31.07	36.25	34.42
Chem	21.8	20.93	30.6	28.41	27.51	26.58
AI	28.22	28.06	28.31	35.45	29.42	31.92
Corp-Law	23.52	22.31	20.09	17.86	25.72	23.43
Medical	21.1	26.23	24.13	23.37	23.66	41.93

Table 4.11 Result of Exp-3 (Stacking) Fine-Tuning for English-Hindi across different domains

4.8.1 Discussion

This approach showed that we can tackle the issue of the introduction of the new domain. As the domain is stacked upon each other, a new domain can be fine-tuned directly on the latest model. Through this approach, we faced the issue of catastrophic forgetting. This makes the model forget the initial domain knowledge that it has learned, and for those domains, the translation quality drops drastically with every new layer of a domain.

The above three approaches gave us some points to think about. We had to find a way in between the two approaches,i.e., between having n models among n domains and having a single model among n domains. We had to find an approach so that we have fewer models, but our translation quality is increased with the help of similar domains. The next method is the final approach that we are going to propose for the domain adaptation in English-Hindi Language Pair. Before going to the approach, the next chapter will discuss how domains can be similar to each other and how we see which domains are related to which other domains.

4.9 Conclusion

In this chapter, we explored the definition of a domain. Before jumping into domain adaptation, we saw how a baseline model is not able to learn the representation of domain-specific terminologies. To understand it in detail, through the task of fine-grained domain classification, we see how dependent each domain is on these terms. Through the method of domain term extraction, we understand how in different domains, these terms have different properties. Both of these methods can be easily extended to Indian Languages. There are some limitations in the unsupervised approach to finding out important words, but that can be tackled with better data and more experimentation with weights of the n-gram and the length of the documents.

We also discussed some basic techniques and the challenges that they put forward. Through the knowledge of these techniques and domain-related sentence embeddings, we will propose a new method in the next chapter that will try to solve all of these issues.

Chapter *5*

Efficient Domain Adaptation in Indian Languages

After setting the background in the previous chapters, this chapter will focus on how to tackle many domains together when we are doing the task of domain adaptation. The approach will be discussed on the English-Hindi language pair, and it will use the knowledge of sentence embeddings that are given to us by the domain terms and help us to get an efficient way to produce better-quality translations.

5.1 Introduction

Sentence embeddings are representations of a sentence and the meaning that it carries. These embeddings are calculated using the word embeddings of the words used in those sentences. As it was seen in the last chapter that a domain is defined by the terms it uses, our approach takes these sentence embeddings and tries to cluster them in a way so that our machine translation model can give better translations. The idea behind this approach is that similar domains will have similar domain-specific vocabulary and will help each other in translations when they are taken together in a cluster.

We will start by getting these representations and then clustering them on the basis of distances from the centroid of a domain. After this clustering, we will apply the fine-tuning approach to get results on these clustered models.

The major question that we will be looking at is how we can use domain knowledge and get a method that will solve the issues discussed in the last chapter and not create only one model for all domains or individual models for each domain. We will also see how sentence embeddings can be used in this task to get good-quality translations for each domain.

5.2 Using Domain Similarity for Domain Adaptation

In this approach, we will first discuss how to define similar domains, and will the approach of grouping similar domains help us in increasing the score of each domain. Sentence embeddings will be introduced before we start the discussion on our approach to domain adaptation.

5.2.1 Sentence Embeddings For Different Domains

Sentence Transformers from the Hugging Face library[74] were used to get sentence embeddings. "all-mpnet-base-v2"[59] gives an embedding of size 768, which encodes all the information regarding the sentence. These embeddings can be used for semantic search[47] and clustering. t-distributed stochastic neighbor embedding(t-SNE)[69] is used to get these high-dimensional embeddings to a 2-Dimensional space where we can analyze the same. t-SNE tries to maintain similarity between two points in two different dimension spaces.

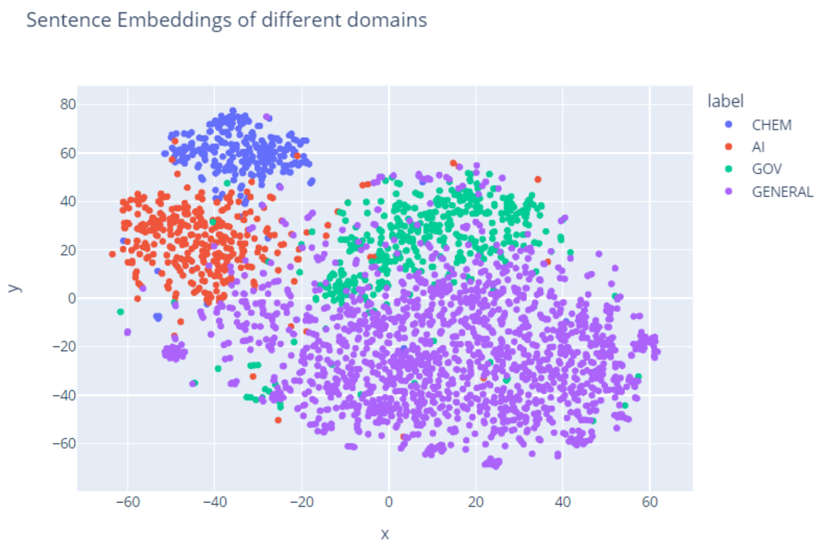


Figure 5.1 Sentence Embeddings on a Small Dataset

Around 30,000 data points were used to plot these sentence embeddings. After plotting these points, two experiments were conducted on these embeddings. The first one gave us a way to cluster these domains to get better results. The second approach verified the cluster we made without taking in labels as input.

5.2.1.1 Centroid Distance

Centroid was calculated for each cluster through all the points. After getting a centroid for each cluster, those centroids were considered as representations of each domain. Distances

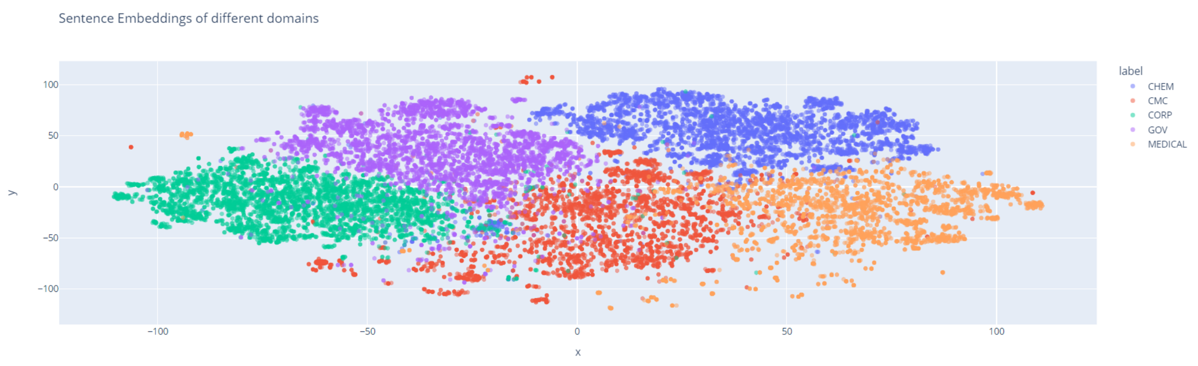


Figure 5.2 Sentence Embeddings of the Whole Dataset

between these centroids were calculated, and domains that are closest to each other were taken together for domain adaptation. The closest distances are shared in the table below:

Domains	**Distance**
Information Sheets and Consent Forms	58.2
Chem	85.52
Corp-Law	126.96
Governance	113.14

Table 5.1 Distance of Centroid of other Domains from Centroid of Medical Domain

Domains	**Distance**
Information Sheets and Consent Forms	79.63
Medical	113.14
Corp-Law	60.19
Chem	71.89

Table 5.2 Distance of Centroid of other Domains from Centroid of Governance Domain

Domains	**Distance**
Information Sheets and Consent Forms	95.67
Medical	85.52
Corp-Law	123.3
Governance	71.89

Table 5.3 Distance of Centroid of other Domains from Centroid of Chemistry Domain

From the above tables, one can see that Corporate Law and Governance are domains that are very close to each other. The cluster of the Medical domain is also surrounded by the cluster of Chemistry and Information sheets domains. So now the approach will be to carry out domain adaptation for these two clusters to see will these common clusters help each to increase their translation quality and will domains that are separated as Chemistry and Information Sheets, can be taken together without dropping their translation quality. Before doing domain adaptation, we are going to verify our cluster through an unsupervised technique of clustering.

5.2.1.2 Unlabelled Clustering of the Domain Data

This is a small step just to see how good centroid distances hold when it comes to clustering. We took all the domain labels out of the data and tried to do simple Kmeans[49] clustering where the embeddings are divided into k clusters according to the density around the centroid clusters that are created. As we have 5 domains in hand, we did clustering from k=5 to k=1, as we want to get how many models we have to make from 1 to n if we have n domains. Silhouette score[64] is calculated for each k from 2 to 5. This score gives us the quality of clustering as it calculates both the intercluster and intracluster distance of data points. Higher the score better the clustering quality. The scores are given below:

k-value	**Silhouette Score**
2	0.366
3	**0.396**
4	0.376
5	0.381

Table 5.4 Silhouette Score for k=2 to k=5

For k=3, we get the best clustering, and when we give back the labels to the clustered data, we see that the medical domain is clustered with domain sentences of Information Sheets and Chemistry. The same is the case for the clustering of corporate law and governance. The third cluster was of general domain points mixed with some data points from all the other domains. As this was unlabelled clustering, all the clusters had some domain points of all the domains but the main distribution was as we discussed above.

5.2.2 Domain Adaptation and Results

Two groups were made before doing fine-tuning. First was Governance and Corporate Law. Second was Medical and Chemistry, along with Information Sheets and Consent Forms. The results of the same are shown below:

When we look at the first clustered model of Governance and Corporate Law, we see that common knowledge of these domains has given us the best score for these two domains. This shows us how different domains can help each other in getting better translations.

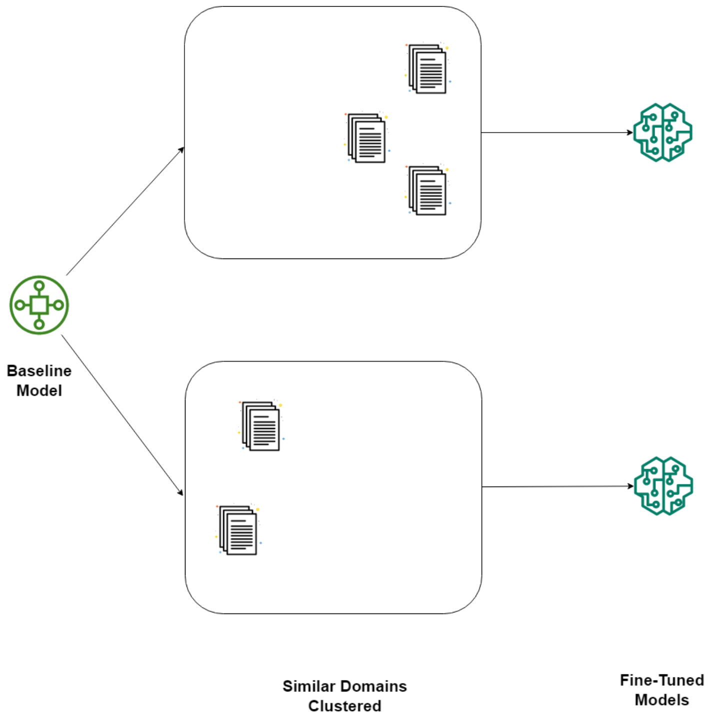


Figure 5.3 Clustering Similar Domain method

	Governance Test Set	**Corporate Law Test Set**
Baseline	38.8	24.35
Fine-Tuned Individually	49.56	27.33
Fine-Tuned Together	45.27	23.26
Fine-Tuned GoCorp	**50.73**	**27.4**

Table 5.5 Results of Fine-Tuning on Governance and Corporate Law Together

	InfoSheet-Consent Forms Test Set	**Medical Test Set**	**Chemistry Test Set**
Baseline	23.58	21.1	21.8
Individually	41.7	45.21	29.84
Together	33.3	36.94	28
Med-Info Sheets+Consent Form	**43.73**	47.45	23.25
Med-Info Sheets+Consent Form - Chem	42.31	**48.74**	**30.11**

Table 5.6 Results of Fine-Tuning on Medical, Information Sheets and Consent Forms and, Chemistry Together

In the second cluster model, we see that the Medical domain gets the best result due to shared knowledge with Information Sheets and Chemistry. As we saw that Information Sheet is far from Chemistry, its scores slightly decreased due to the introduction of Chemistry, but the decrease is so less that we can argue that's it better to decrease our models by one by taking both Information Sheets and Chemistry with Medical than clustering only a single domain with Medical. The increase in the translation quality other two domains outweighs the slight decrease in the quality of Information Sheets and Consent Forms.

5.3 Discussion

Through the last approach, we were able to find an efficient method of domain adaptation for multiple domains. The number of models was reduced from five to two, and we got the best quality translation through this approach. Domains that were similar to each other helped to improve the knowledge gained by the model, which resulted in better translations.

If a new domain is introduced, we have to first check in which cluster the domain will lie, and then it can be grouped together in that cluster. The current approach of calculating the distance between the centroids be applied to any set of domains to get the best output. As we saw while comparing the Med-Info Sheets+Consent Form and Med-Info Sheets+Consent Form - Chem model the quality can be slightly better for a model with a smaller set, but decreasing the number of models is a very important step. This approach also does not depend on domain classification very much as a classifier mostly confuses similar domains, and similar domains are given the same models in this approach so there is less chance of bad quality translation due to misclassification of a document.

5.4 Conclusion

In this chapter, we used domain knowledge to get the best results out of domain adaptation. We tried different approaches to get an efficient way of training models where we are not wasting memory with great scores. We started with n different models and ended up with 2 models at the end.

The domain adaptation models were based on how close the domains were. This was calculated through centroids of these domain representations. After getting these two models, we also saw how we increased the score by using common knowledge among these domains. This technique can easily be applied to new domains when they are introduced just by calculating the distance of new domains with models that are built previously.

Chapter *6*

Conclusion and Future Work

6.1 Conclusion

From the start of this thesis, we tried to tackle the issue of machine translation and domain adaptation in Indian languages. We mainly focused on the English-Hindi language pair, but through the baseline and normal fine-tuning on English-Telugu and English-Bengali, we saw that these approaches can be extended to other Indian languages too. We also tackled the issue of fine-grained domain classification and domain terminology extraction, which helped us to understand more about how a domain is defined and how domain terminologies weigh the most in the same.

Chapter 2 was a brief discussion of work that has been done in the field of domain adaptation and other domain-related tasks. This chapter gave us a basic understanding of how these tasks have been tackled in recent times.

In Chapter 3, we started building baseline translation models for three language pairs. The end goal of this chapter was to get a model that gave comparable results with the Himangy model, which has been trained on. We achieved the same for all three models with some initial problems of subword lengths for the Telugu language. It was increased after some experiments with subword length. The models achieved in this chapter were used for domain adaptation experiments in future chapters.

Chapter 4 talked about domain adaptation and how it is important for domain-specific translations. To understand this importance, we first defined a domain. This was defined through the two experiments of fine-grained domain classification and domain term extraction. We saw how domain terms are the building blocks of a domain. Domain adaptation experiments that we did were based on similar domains, and the domain classification on a fine-grained scale helped us to differentiate between these domains, which are very similar to each other. Apart from that, an unsupervised experiment related to domain term extraction was done, which helped us to understand how different domain terms are in different types of domains. This chapter helped us to understand what it means when we define different domains. After that, we started a discussion on our basic approaches for domain adaptation. We discussed how the data is distributed throughout the domains. At the start, we tested our fine-tuning by doing it on English-Telugu and English-Bengali pairs for the domain of Information Sheets and Consent Forms. After that, our first experiment involved a specific fine-tuned model for each domain. This presented acceptable results but proved to be quite an inefficient method. Then we did two experiments on single model domain adaptation, the first being taking all the data together and the second being stacking the domain one after the other. Taking all the data together did not give as good results as individual fine-tuning and stacking the domain one after another resulting in the issue of catastrophic forgetting. The domains that are getting stacked earlier on were getting forgotten by the model with each iteration. So the aim was to get a way to get fewer models than the number of domains and have a better performance.

Chapter 5 discusses the new approach that is proposed for domain adaptation across multiple domains. At the start of the chapter, this was tackled by getting sentence embeddings representation of each domain and then clustering domains related to their similarity. Through this, we were able to create a cluster where domain terms from different domains helped in translations. This helped us not only in reducing the number of models but also giving us the best results, as similar domains were helping each other in learning better translations. At the end of the chapter, we got an approach that not only help us to do domain adaptation in a structured manner but also get the best scores out of it.

This thesis provided three techniques to get results in three different domain-related tasks, which can be extended to get better results in the future in some different domains.

6.2 Future Work

This thesis discussed fine-grained domain classification, unsupervised domain term extraction, and domain adaptation as three tasks. All these three tasks can be extended in the following ways.

6.2.1 Fine-Grained Domain Classification

The approach that was discussed helped in domain classification for domains that are closely related to each other. The word embeddings that were used to differentiate between similar domains were the key to getting better results. If word embeddings of good quality for all Indian Languages can be made, this approach can be extended to all Indian Languages easily. Better embeddings will help in distinguishing the uncommon knowledge better between similar domains and getting higher scores. The technique of removing common knowledge between sentences can help in some other NLP tasks also.

6.2.2 Unsupervised Domain Terminology Extraction

This method also can be easily extended to Indian Languages as it is an unsupervised approach and does not depend on sentence embedding quality and other data-related issues. Apart from Indian languages also, this approach can be easily used for other languages too. The issue here arises with the introduction of n-grams, and that has to be experimented with for different languages due to the different structures of each language. Apart from that, there can be a study on achieving the best n-gram weights automatically by studying all the documents of a domain.

6.2.3 Domain Adaptation in Indian Languages

The method discussed in the last chapter can be used for different language pairs. Finding data from different domains and the availability of baseline models will be a challenge for this task for different language pairs. Once both of them are available, this clustering method can be applied without any issue. Apart from that, the technique of getting domain clusters can be improved to see if there is any other way to group these domains to get even better results. Better distribution of these clusters will make our results better for each translation.

Related Publications

1. Saransh Rajput, **Akshat Gahoi** , Dipti Mishra Sharma **N-Grams TextRank A Novel Domain Keyword Extraction Technique.** Proceedings of the 17th International Conference on Natural Language Processing (ICON): TermTraction 2020.

Other Publications

1. **Akshat Gahoi**, Akshat Chhajer, Dipti Mishra Sharma **Fine-grained domain classification using Transformers.** Proceedings of the 17th International Conference on Natural Language Processing (ICON): TechDOfication 2020 Shared Task.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] R. Appicharla, K. K. Gupta, A. Ekbal, and P. Bhattacharyya. IITP-MT at WAT2021: Indic-English multilingual neural machine translation using Romanized vocabulary. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*, pages 238–243, Online, Aug. 2021. Association for Computational Linguistics.

[3] A. Babhulgaonkar and S. Bharad. Statistical machine translation. pages 62–67, 10 2017.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[5] L. Barrault, M. Biesialska, O. Bojar, M. R. Costa-jussà, C. Federmann, Y. Graham, R. Grundkiewicz, B. Haddow, M. Huck, E. Joanis, T. Kocmi, P. Koehn, C.-k. Lo, N. Ljubešić, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, S. Pal, M. Post, and M. Zampieri. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online, Nov. 2020. Association for Computational Linguistics.

[6] L. Barrault, O. Bojar, M. R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. Müller, S. Pal, M. Post, and M. Zampieri. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[7] M. Bay, D. Bruneß, M. Herold, C. Schulze, M. Guckert, and M. Minor. Term extraction from medical documents using word embeddings. In *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, pages 328–333, 2020.

[8] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. volume 3, pages 601–608, 01 2001.

[9] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

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

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[12] W. Cavnar and J. Trenkle. N-gram-based text categorization. *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, 05 2001.

[13] P.-C. Chen, H. Tsai, S. Bhojanapalli, H. W. Chung, Y.-W. Chang, and C.-S. Ferng. A simple and effective positional encoding for transformers, 2021.

[14] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[15] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[16] C. Chu, R. Dabre, and S. Kurohashi. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[17] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020.

[18] M. Conrado, T. Pardo, and S. Rezende. A machine learning approach to automatic term extraction using a rich feature set. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 16–23, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[19] P. Dakwale and C. Monz. Fine-tuning for neural machine translation with limited degradation across in- and out-of-domain data. In *Proceedings of Machine Translation Summit XVI: Research Track*, pages 156–169, Nagoya Japan, Sept. 18 – Sept. 22 2017.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[21] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon. Unified language model pre-training for natural language understanding and generation, 2019.

[22] A. Gahoi, A. Chhajer, and D. Mishra Sharma. Fine-grained domain classification using transformers. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON): TechDOfication 2020 Shared Task*, pages 31–34, Patna, India, Dec. 2020. NLP Association of India (NLPAI).

[23] A. Gotmare, N. S. Keskar, C. Xiong, and R. Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation, 2018.

[24] N. Goyal, C. Gao, V. Chaudhary, P.-J. Chen, G. Wenzek, D. Ju, S. Krishnan, M. Ranzato, F. Guzman, and A. Fan. The flores-101 evaluation benchmark for low-resource and multilingual machine translation, 2021.

[25] M. Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure, 2022.

[26] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio. On using monolingual corpora in neural machine translation, 2015.

[27] A. Hazem, M. Bouhandi, F. Boudin, and B. Daille. TermEval 2020: TALN-LS2N system for automatic term extraction. In *Proceedings of the 6th International Workshop on Computational Terminology*, pages 95–100, Marseille, France, May 2020. European Language Resources Association.

[28] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[29] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation, 2015.

[30] B. Jung, Y. Mukuta, and T. Harada. Grouped self-attention mechanism for a memory-efficient transformer, 2022.

[31] P. Kaushik, A. Gain, A. Kortylewski, and A. Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping, 2021.

[32] D. Khyani and S. B S. An interpretation of lemmatization and stemming in natural language processing. *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology*, 22:350–357, 01 2021.

[33] J.-K. Kim and Y.-B. Kim. Supervised domain enablement attention for personalized domain classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 894–899, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[35] A. Kunchukuttan, P. Mehta, and P. Bhattacharyya. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[36] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer. Multilingual denoising pre-training for neural machine translation, 2020.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[38] M.-T. Luong and C. Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 76–79, Da Nang, Vietnam, Dec. 3-4 2015.

[39] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[40] K. Manger. Morphological divergence in hindinepali language pair. 03 2023.

[41] D. Marcu and W. Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, page 133–139, USA, 2002. Association for Computational Linguistics.

[42] R. Mihalcea and P. Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[43] A. Mirzal and M. Furukawa. A method for accelerating the hits algorithm paper: A method for accelerating the hits algorithm. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 14, 09 2009.

[44] V. Mujadia and D. Sharma. Low resource similar language neural machine translation for Tamil-Telugu. In *Proceedings of the Sixth Conference on Machine Translation*, pages 288–291, Online, Nov. 2021. Association for Computational Linguistics.

[45] V. Mujadia and D. Sharma. The LTRC Hindi-Telugu parallel corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3417–3424, Marseille, France, June 2022. European Language Resources Association.

[46] V. Mujadia and D. M. Sharma. English-Marathi neural machine translation for LoResMT 2021. In *Proceedings of the 4th Workshop on Technologies for MT of Low Resource Languages (LoResMT2021)*, pages 151–157, Virtual, Aug. 2021. Association for Machine Translation in the Americas.

[47] E. Mäkelä. Survey of semantic search research. 07 2008.

[48] R. Müller, S. Kornblith, and G. Hinton. When does label smoothing help?, 2020.

[49] S. Na, L. Xumin, and G. Yong. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 63–67, 2010.

[50] OpenAI. Gpt-4 technical report, 2023.

[51] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling, 2019.

[52] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking : Bringing order to the web. In *The Web Conference*, 1999.

[53] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[54] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[55] R. N. Patel, P. B. Pimpale, and S. M. Statistical machine translation for indian languages: Mission hindi, 2016.

[56] A. Pathak and P. Pakray. Neural machine translation for indian languages. *Journal of Intelligent Systems*, 28(3):465–477, 2019.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018.

[58] S. Rajput, A. Gahoi, M. Reddy, and D. Mishra Sharma. N-grams TextRank a novel domain keyword extraction technique. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON): TermTraction 2020 Shared Task*, pages 9–12, Patna, India, Dec. 2020. NLP Association of India (NLPAI).

[59] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[60] K. Revanuru, K. Turlapaty, and S. Rao. Neural machine translation of indian languages. 11 2017.

[61] H. Sajjad, N. Durrani, F. Dalvi, Y. Belinkov, and S. Vogel. Neural machine translation training in a multi-domain scenario, 2018.

[62] R. Sennrich, B. Haddow, and A. Birch. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California, June 2016. Association for Computational Linguistics.

[63] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units, 2016.

[64] K. R. Shahapure and C. Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 747–748, 2020.

[65] D. M. Sharma, A. Ekbal, K. Arora, S. K. Naskar, D. Ganguly, S. L, R. Mamidi, S. Arora, P. Mishra, and V. Mujadia, editors. *Proceedings of the 17th International Conference on Natural Language*

Processing (ICON): TermTraction 2020 Shared Task, Patna, India, Dec. 2020. NLP Association of India (NLPAI).

[66] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mass: Masked sequence to sequence pre-training for language generation, 2019.

[67] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[68] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014.

[69] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[71] S. Vogel. Smt decoder dissected: word reordering. In *International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003*, pages 561–566, 2003.

[72] C. Wang, K. Cho, and J. Gu. Neural machine translation with byte-level subwords, 2019.

[73] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang. A high-speed and low-complexity architecture for softmax function in deep learning. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 223–226, 2018.

[74] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[75] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[76] K. Yamada and K. Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, page 523–530, USA, 2001. Association for Computational Linguistics.

[77] Y. Yuan, J. Gao, and Y. Zhang. Supervised learning for robust term extraction. In *2017 International Conference on Asian Language Processing (IALP)*, pages 302–305, 2017.

[78] W. Zhang, T. Yoshida, and X. Tang. A comparative study of tf*idf, lsi and multi-words for text classification. *Expert Syst. Appl.*, 38:2758–2765, 03 2011.