Two-phased Architecture for Dynamic Language Model

Debajyoty Banik, Asif Ekbal, and Pushpak Bhattacharyya
{debajyoty.pcs13, asif, pb}@iitp.ac.in

Indian Institute of Technology Patna, India

Abstract. We discuss the importance of domain specific language model in statistical machine translation system. Both the structures and phrase selection are not the same for different domains. So, the language model trained with the general domain data or other domain data can not provide better accuracy. Moreover, there may have some specific focus in different texts of the same domain. Hence, the language model trained with data from the default domain may not yield significant output. In this paper, we learn our system dynamically based on the better matches with the input text. Instead of directly selecting pre-trained language model we prepare the prioritized language model according to the situation. The proposed model is evaluated for Hindi-English translation. It shows a significant improvement on the translated output in terms of the BLEU score. Our evaluation shows that automated domain adoption to predict better language model improves the translation quality.

Keywords: Natural Language Processing; Statistical method; Language Model; Web scraping.

1 Introduction

Domain adaption in the statistical machine translation (SMT) system has resulted in improved translations once the domain is properly classified [1]. In order to improve the translation quality, we adapt context wise language models for better translation. There is a significant improvement in translation instead of just using a general monolingual corpus.

We want to compute the probability of a string $W = w_1, w_2, ..., w_n$. Intuitively, $p(W)$ is the probability of a sequence of English (target language) tokens which turns out to be W. To find out $p(W)$, the typical statistical approach estimates the counting how often W occurs in the training data. However, most long sequences of words do not occur in the text at all. So, we have to break down the computation of $p(W)$ into smaller steps, for which we can collect sufficient statistics and estimate probability distributions. In n-gram language modeling, it is decomPoSed the probability using the chain rule:

$$p(w_1, w_2, ..., w_n) = p(w_1)p(w_2|w_1)...p(w_n|w_1, w_2, ..., w_{n-1}) \tag{1}$$

The language model probability $p(w_1, w_2, ..., w_n)$ is a product of word probabilities where given a history of preceding words as discussed in Equation 3. To be able to estimate these word probability distributions, we limit the history to m words:

$$p(w_n|w_1, w_2, ..., w_{n-1}) \simeq p(w_n|w_{n-m}, ..., w_{n-2}, w_{n-1}) \tag{2}$$

In Statistical machine translation, language models are used to increase the fluency of translated texts. It is a common concept that the language model using large data-set may be more trustable since we would get more histories (the basis of language modeling). But it may leads to over fitting if that is not domain specific. To make it domain specific, data sparsity problem can be noticeable. In this paper, we are trying to consider both the problems. Depends on a belief that a language model that embraces a larger context provides better prediction ability, researchers in [2] presented two different language models: a backward language model that augments the conventional forward language model, and a mutual information trigger model which captures long-distance dependencies that go beyond the scope of standard n-gram language models. The traditional language model cannot provide best fluency for different type of the texts from various domains. We reconstruct the language models in different way by scrapping the web for different domains/context. Finally, suitable language model is used for the translation based on the input text.

1.1 Motivation

Developing better language models often results in models that perform better on their intended natural language processing task. This is the motivation for developing better and more accurate language models. Our motivation is drawn from the fact that usually a document is based on a topic and has a particular domain that it is discussing. The topics may be related to technology, politics, sports, health etc. Thus if the language models are built specifically for a particular domain and used for translation, the results are expected to be better. This is useful in speech recognition [3] where the input speech is mostly based on a particular topic. For a particular domain, some words or phrases are more commonly used in texts, which help us to analyze what sort of topic is being spoken about. When we mostly use a particular set of words or phrases, we become sure of the topic expressed. These phrases (cue words) can also be used to check the domain of the input text and extract data from the web to create the domain-wise language model. This can be used to adapt the language model and use it for better translation.

The paper is organized as follows. The overall methodology of creation of the domain-wise language model is described in section 2. Section 3 deals with the experimental setup used in this process and analyzes the results obtained and finally Section 4 gives the conclusions for domain-wise language modeling for Hi-En translations.

1.2 Related work

From the literature survey, we observe that there has not been much detailed discussion as to how statistical machine translations (SMT) system can be improved by improving Language Model, more specifically on domain adaption by using n-gram modelling.

In [4] the paper reports on how to build a large language model that allows scaling to a enormous amount of training data, and how much does translation performance improve as the size of the language model increases.

In [5], they have tried a variety of sizes and domains of training data; but this has been tried on the parallel corpora and not on the language models to investigate the effect on translation. They have investigated the behaviour of an SMT system exPoSed to training data of different sizes and types. Their experimental results showed that even parallel corpora of modest sizes could be used for training purPoSes without lowering too much the evaluation scores considering two language pairs in both translation directions for the experiments: English-Romanian and German-Romanian.

Recently there has been work in using Neural Language Models (NLMs). In [6] they have demonstrated that deep NLMs having three or four layers which perform well than the fewer layers in terms of both the perplexity and the translation quality. They have combined various techniques to successfully train deep NLMs that jointly condition on both the source and target contexts. They also use this concept for domain adaption on the sms-chat domain. But the adaption is done to tune the decoder weight by using a tune set of 260K words in the newswire, web, and the sms-chat domains to tune the decoder weights and a separate small, 8K words set to tune re-ranking weights. To train adapted NLMs, they have used trained models on general in-domain data and further fine-tune with out-domain data for about four hours.

A significant work to compare language model has been carried out in [7]. They have investigated the differences between language models compiled from the original target-language texts and those compiled from texts manually translated to the target language. Corroborating established observations of Translation Studies, they demonstrated that the latter are significantly better predictors of translated sentences than the former, and hence fit the reference set better. Furthermore, translated texts yield better language models for statistical machine translation than the original text.

Like we have stated earlier, the significant work in developing topic wise language models has been done only to benefit automated speech recognition. In [3] they have used a similar process, generate queries and obtain outputs from the World Wide Web, to develop language models. They have described an iterative web crawling approach which uses a competitive set of adaptive models comprised of a generic topic independent background language model, a noise model representing spurious text encountered in web-based data (Web data), and a topic specific model to generate query strings using a relative entropy based approach for WWW search engines and to weight the downloaded Web data appropriately for building topic specific language models. They have demonstrated how this system can be used to build language models for a specific domain rapidly given just an initial set of example utterances and how it can address the various issues attached with Web data. They were able to achieve a 20% reduction in perplexity for the target medical domain. The gains in perplexity translated to a 4% improvement in ASR word error rate (absolute) corresponding to a relative gain of 14%

In [8] adaptive language modelling has been done for the domain of blogging and micro-blogging. They have used information retrieval (IR) language modeling based on the large volumes of constantly changing data. This data fulfill the needs to frequently integrating and removing data from the model. They have identified a set of matches from a corpus given a query sentence, then the likelihood estimation are calculated for that vary query. The IR-LM can be beneficial when the language model needs to be updated with adding and removing the data which works for the social data where new content is constantly generated.

2 Domain Adoption in the Language Model

To prepare a language model we usually use n-gram modeling. n-gram language models are based on statistics of how likely words are to follow each other. If we analyze a large amount of text, we will observe that the word 'home' follows the word 'going' more often than the word 'house' does [9]. This information is learned by the language model to assure fluency.
Language modeling (LM) is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede it. There are two challenges to prepare best language model which fit for specific text: 1. Domain specific language model cannot provide better accuracy for other specific domain. We may not familiar with the domain of the upcoming input text. 2. The language model trained with General domain data may lead overfitting problem. Considering both issues, we prepare a prioritized language model based on the specific topic of input text addition with general language model.

The Primary concept of this prioritized language model is Let $w_1^L = w_1, w_2, ..., w_L$ denotes a string of L tokens over a fixed vocabulary. An n-gram language model assigns a probability to W_1^L.

$$(w_1^L) = \prod_{L}^{i=1} P(w_i|w_1^{i-1}) \approx \bar{P}(w_i|w_{i-n+1}^{i-1}) \tag{3}$$

The Markov assumption is the base of the language modelling. Let, f (w_i^j) represents frequency of occurrence of the sub-string w_i^j in a long target language string w_1^L, known as the training data. Using their relative frequencies the maximum-likelihood (ML) probability computes for the n-grams.

$$r(w_i|w_i^{i-1} - n + 1) = \frac{f(w_{i-n+1}^{i})}{f(w_{i-n+1}^{i-1})} \tag{4}$$

The Markov assumption states that only a limited number of previous words affect the probability of the next word. It is technically wrong, and it is not too hard to come up with counter examples which demonstrate that a long history is required. However, limited data restricts the collection of reliable statistics to short histories.

Typically, we choose a number of words in the history based on the training data we have. More training data allows for longer histories. Most commonly, trigram language models are used. They consider a two-word history to predict the third word. We have also used tri-gram language model in all the cases. The estimation of trigram word prediction probabilities $p(w_3|w_1, w_2)$ is straightforward. We count how often in our training corpus the sequence w_1, w_2 is followed by the word w_3 , as opPoSed to other words. According to maximum likelihood estimation, we compute:

$$p(w_3|w_1, w_2) = count(w_1, w_2, w_3)|sum(w)count(w_1, w_2, w_3) \tag{5}$$

Finally, individual models (domain specific language model and general language model) combine using joint probability model based on their priority. It is quite obvious that the domain specific language model gets more priority than the general language model. The primary reason to keep the general language model to cover all situations which miss at domain specific model.

The first step is devoted to construct the general language model. The second step is devoted to retrieve cue words from the input text (test data). Based on those cue words we identify the online documents using the matching ratio. We don not use Term frequency-inverse document frequency (TF-IDF) [10] calculation to mine the documents because the input text may be a new copy and it will take much to cover all variety documents searching. These documents are used to prepare the domain specific language model and merge with the general language model to get final language model which is best fit for the machine translation and other application. The detailed flowchart is shown in Figure 1. As discussed earlier, cue words are extracted based on the TF-IDF computing. The term frequency $tf(t, d)$, is the simplest choice to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d. If we denote the raw count by $f_{t,d}$, then the simplest tf scheme is $tf(t, d) = ft, d$. The term frequency adjusted for document length : $\frac{f_{t,d}}{(numberofwordsind)}$. The augmented frequency, to prevent a bias towards longer documents. $tf(t, d)$ is shown in Equation 6:

$$tf(t, d) = 0.5 + 0.5(\frac{f_{t,d}}{max(f_{t'd}) : t' \in d}) \tag{6}$$

The inverse document frequency is a measure of how much information the word provides, i.e., if it is common or rare across all documents[1]. It is shown in Equation 7.

$$idf(t, D) = log\frac{N}{|D \in d : t \in d|} \tag{7}$$

[1] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Where, N refers total number of documents in the corpus $N = |D| N = |D| |\{d \in D : t \in d\}| |\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t,d) \neq 0$ $\mathrm{tf}(t,d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}| 1 + |\{d \in D : t \in d\}|$

We prepare the language models for translations from Hindi to English. A language model is created in the target language. Hence, here we will be making language models in English.

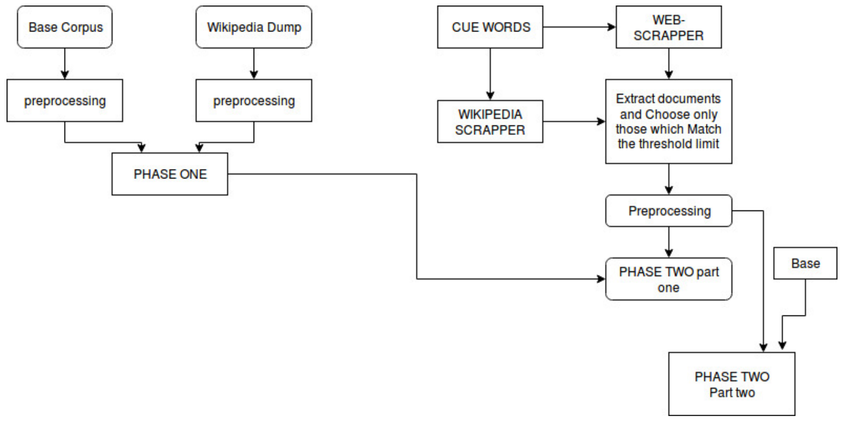

Fig. 1. Flowchart for the adaptive language model

2.1 Preprocessing

Documents should be preprocessed so that relevant and clear information can be obtained. Preprocessing helps to improve the quality of texts and makes all the documents to be at par with each other. The steps of preprocessing followed for all documents are:

Tokenization: Tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. We chop on whitespace and throw away punctuation characters.

True casing: True casing means converting all the characters to lowercase.

2.2 Base corpus

For any language model creation, we need a huge amount of texts. By analyzing patterns in that text, n-gram probabilities are calculated.

2.3 Phase-one

Here, we calculate the n-gram probability for base corpus plus Wikipedia dump[2]. Wikipedia has a variety of domains and topics so it can be a valuable addition to the language model. It can cover various types of sentences with more confident and able to solve the data sparsity problem. The language model, which is prepared using the general data-sets, is waiting to take participation at the phase-two after preparing the domain specific language model.

2.4 Phase-two

In this section, we define the process used to make the adaptive language model; i.e. one which is made strictly for the topic which is being spoken of. No need to have any prior knowledge about the topic or domain. This real life problem is solved in real time using the proposed approach. The primary processes are 1. cue-word extraction, 2. web-scrapping and wikipedia scrapping based on the cue-word, then 3. combining the general language model with the prepared language model based on the extracted documents (with assigned priority) to create the language model.

Cue words: Cue words are the significant or important words present in the text. Cue words express the ideas of topics that provide some information about the content. Instead of the individual word we consider phrases as cue-word for our work which makes the search more realistic and contextual. We have parsed each sentence into trees based on phrases and to obtain the required cue words we have extracted the NOUN PHRASE (NP) and VERB PHRASE (VP). We have not extracted ADJECTIVE PHRASES (ADJP) separately because they are present inside the VP already. Considering phrases are better than words because if we use just some specific PoS tags we might miss out some contextual information. If we use PoS tags for cue word extraction and fix the tags for noun forms to just NNS (Noun Plural) and NN (Noun Singular), they may not be adequate enough for improvisation. NP for all forms of nouns are taken into consideration in this case. Other tags such as adverbs, prepositions do not contribute to spot the cue words and add only to the noise in retrieving relevant documents. All of the abbreviations of the PoS tag follow standard form (i.e. penn tree bank[3]). Some examples of phrases is shown in Table 1.

We realized that all the retrieved phrases would not be helpful in retrieving more information from the Web and Wikipedia. Based on the observation and imperial analysis, we have set the threshold for size of the cue-word phrases as six words because too big query will not produce any suitable result from the web at all and will be just a loss of time. We have used the Stanford Parser for parsing [11].

[2] $https : //www.wikidata.org/wiki/Wikidata : Database_download$

[3] $https : //www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_PoS.html$

TAGS	TEXT
NP	vegan children
VP	are not suitable in for infants
ADJP	suitable for infants
NNS	infants
JJ	suitable

Table 1. Examples of tags

Web Scraper and Wikipedia Scraper: The cue words obtained in section 2.4 are sent as a query to the Web. The World Wide Web consists of a lot more textual information. The amount of information related to a topic increases steeply in the web. So if cue words are sent as a query, it would retrieve documents relevant to topic and domain. We have used Webhose-python[4] as our web Scraper. It gives us the news domain pages related to the query.

We have further used a python program to return just those wikipedia pages for which cue words/phrases are sent in as queries. This also helps us to find the related pages as required.

Setting up threshold: Not all the pages which are received from the web or the wikipedia may be relevant to one domain. For example, while retrieving pages from Wikipedia, we observed that for the same cue word “Air”, we received a no a no pages ranging from “Air guitar” to “air”. As we are dealing with health corpus, we do not need ”air guitar” to be included. So, we set up a threshold which would help in the inclusion of files. It (R) is computed in Equation 8.

$$R = \frac{|K|}{|T|} \tag{8}$$

Where K refers to the keywords, present in the document, which match the list of keywords except the stop words and T denotes a total number of words in the document except stop words. This equation is also used to fine-tune the threshold value δ. Then we compute R for different files which are obtained from the web or the input text to compare with δ.

Table 2 shows the file type and obtained R for the different texts. These texts can be treated as a specialized version of the different domains. So, computed R for different files crawled from the web are shown here altogether.

In Table 3, the ”air” expelled in one section is a misleading title. It is about a video game and not related to health as the title assumes us to think; thus the threshold 0.2 is good. Hence, we set this ratio to 0.22 for inclusion in case of webhose files and 0.2 in the case of wikipedia files.

[4] https://github.com/Webhose/webhoseio-python

Filename	**Ratio (R)**
social2.txt	0.189247311828
new3.txt	0.1483198146
new_danielpipes.en.txt	0.149747834533
new_emille.en.txt	0.237342128443
tourism1.txt	0.175356615839
social1.txt	0.194812069878
tech1.txt	0.185867895545
health1.txt	0.242798353909
tech2.txt	0.183632734531
health2.txt	0.235188509874
new1.txt	0.155754651964
health3.txt	0.227241615332

Table 2. Matching ratio (R) of different crawled files

Filename	**Ratio (R)**
are filled with airen_wiki_article.txt	0.2000664673
cognitive impairment_en_wiki_article.txt	0.2082551595
clean air_en_wiki_article.txt	0.2476382416
the air expelled in one second_en_wiki_article.txt	0.1226463104
Orcas Island Airport_en_wiki_article.txt	0.1226415094
spreads in the lungs and airways_en_wiki_article.txt	0.226668751
Mayor Buenaventura Vivas Airport_en_wiki_article.txt	0.0379746835

Table 3. Table for checking ratios of wikipedia pages32

2.5 Final Language model

All these files are then combined to form language models. We have done this in two ways.

1. Base+Wikipedia dump+webhose files+specific wikipidia pages
2. Base+webhose files+specific wikipidia pages

We call them phase-two part 1 and phase-two part 2, respectively. We will then test which of the language models,the base,phase-one or phase-two give us the best results.

The main application of this experiment lies in identifying the domain of incoming test file and using that language model which matches the domain. For this, we use concept the match ratio (R). We keep another set of cue words, in the source language and match the incoming test file to each of these lists. The maximum match ratio will provide us the idea about the domain the input test file. The preferred domain specific language model get more priority to built the final language model.

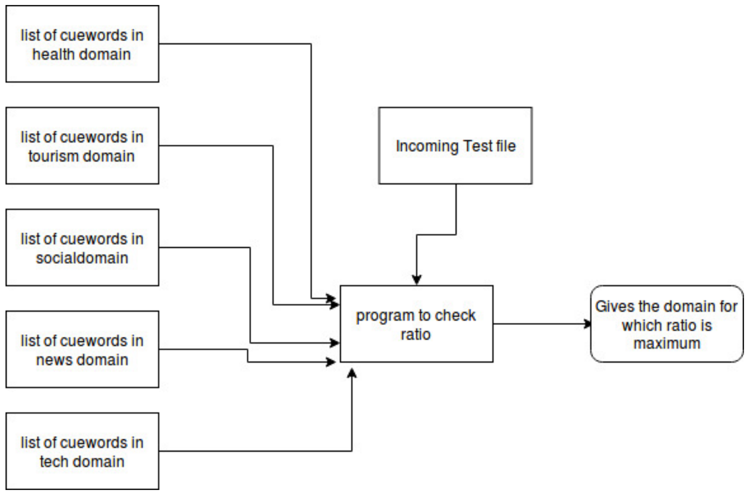
list of cuewords in health domain
list of cuewords in tourism domain
list of cuewords in socialdomain
list of cuewords in news domain
list of cuewords in tech domain
Incoming Test file
program to check ratio
Gives the domain for which ratio is maximum

Fig. 2. Flowchart demonstrating checking of domain of test file

3 Experimental Evaluation

We detail the experiments in this Section to check which out of the four models would perform best for a domain. We train 3-gram LMs of each of the models for the following domains: health, news, social, tech and tourism. We have performed the experiment on Hindi to English Translations.

The base corpora has been obtained from monolingual sets of News Crawl articles and Europarlv7[9], [12]. The combination of these gives us 1492827 sentences. The Wikipedia Dump has been downloaded from the web[5]. We have taken the pages and articles of 2017-05-20. The processing of wikipedia files from xml format to plain text has been done with the help of software available at [6] which uses BeautifulSoup and PHP for the conversion.

Preprocessing of the texts also needed to be done. For English, the Moses tokenizer was used and For Hindi, the Indic NLP tokenizer[7] was used. Some other independent python programs were also written to get rid of unwanted patterns and to clean the corpus.

For cue word extraction, we have used Stanford Parser [13]. It converts the text into tree format from which we can extract the required phrases using python programs. The following corpora are used for the cue-word extraction:
The cue words for health are extracted from HindEnCorp corpus [14] which contains the parallel corpus consisting of 200,000 words of English text (8.9k sentences) and IIT-Bombay annotated corpus [15] which consists of 15,589 sen-

[5] https://dumps.wikimedia.org/en
[6] http://www.evanjones.ca/software
[7] $https://anoopkunchukuttan.github.io/indic_nlp_library/$

tences. The cue words for news, social, tech are extracted from HindEnCorp corpus.

The news domain cue words are extracted from the Tides and DanielPipes Commentary section which consist of 50k and 6.6k sentences, respectively. The text used for social domain consists of 39.8k sentences and for tech domain contains 66.7k sentences. The cue words for tourism are extracted from IIT-Bombay annotated corpus [15] which contains 16222 sentences and The TDIL Program and the Indian Language Corpora Initiative [16] which contains 25000 sentences.

After the filtering of cue words (threshold filtering of 6 words maximum), for the health domain we are left with around 42,000 phrases which are then sent as queries to Webhose and a web crawler which returns the pages as found in the news in text format and to the wikipedia Scraper.

Each of the pages are then checked for keyword match ratio and the ones not fulfilling in the criteria are discarded.

Then the pages are combined in two formats as mentioned in section 2.5.

Then Language Models are made for each of the four formats described in section 2 using the SRILM toolkit.

For the data on health domain, the prepared parallel corpus was from the corpus of the TDIL Program and the Indian Language Corpora Initiative. It was trained using Moses. It contains 25,000 sentences in the parallel corpora which are divided accordingly:

We then translate and evaluate the quality using the BLEU score and RIBES score to see which of the four models worked better for the test file which was of health domain. We also compared the scores for test file to a widely available existing Language Model, Gigaword LM [8]https://www.keithv.com/software/giga). It is trained using the newswire text provided in the English Gigaword corpus (1200M words of NYT, APW, AFE, XIE).

RIBES is an automatic evaluation metric for machine translation, developed in NTT Communication Science Labs. Its implementation is commonly distributed in Python.

Although, BLEU (biligual evaluation understudy) is the score, mostly used for MT evaluation in the last couple of years. Its measures the accuracy based on the n-grams match.

LANGUAGE MODEL TYPE	**BLEU**	**RIBES**
BASE [9]	17.82	0.703778
PHASE 1	17.83	0.706083
BASE+WIKI_DUMP+WEB+WIKI	18.37	0.703172
BASE+WEB+WIKI	18.47	0.706040
GIGA LM	13.83	0.673181

Table 4. Table showing BLEU and RIBES score for health-based test data

[8] (

	Language	Train	Dev	Test
#Types	hin	343,601	2,625	8,489
	eng	250,782	2,569	8,957
#Sentences	hin/eng	1,492,827	520	2,507
#Tokens	hin	22,171,543	10,174	63,853
	eng	20,667,259	10,656	57,803

Table 5. Snapshot of the IIT Bombay English-Hindi corpus.

Set	#Sentences	#Tokens	
		En	Hi
Training	64724 × 2	458831	532985
Test	1002	7229	8519
Development	1001 × 2	7241	8271

Table 6. Statistics of HindEnCorp data-set

In Table 4, we see that for a test file which is in health domain, the maximum BLEU score is obtained for the language model which is prepared using the base+selective webhose pages+selective wikipedia pages. It is higher than the one with base+selective webhose pages+selective wikipedia pages+ Entire Wikipedia Dump, even though the latter has more data. We find 3.65% increase in BLEU point if we use the language model which is prepared by the part-two of the phase-two architecture rather than the traditional language model. After considering the widely available Gigaword language model[9] we find 33.5% improvement in BLEU with the proposed approach.

The second part of the experiment comprises of choosing of language model; i.e. the understanding of which domain the test file belongs to. The cue words come in handy again, though this time we use a translated version of the previously obtained cue words. We have directly translated the list file to source and kept them ready for use. Using the similar ratio formulation, we print the name of the list which shows the domain for which the ratio is maximum. We then decode our test file using that particular LM only.

Also, we have used the IIT Bombay English-Hindi corpus [17] and the HindEnCorp [18] for more experiment which is based on miscellaneous domain. The statistics of the IIT Bombay English-Hindi and HindEnCorp data-sets are shown in Table 5 and Table 6, respectively. We call these data-sets as data-set-1 and data-set-2, respectively. Moreover, we have collected some random sentences and prepare data-set-3, data-set-4, and data-set-5. The comparison analysis of the translated outputs between the proposed method and the phrase-based SMT [9] for these data-sets are shown Figure 3. Similar comparison with hierarchical phrase-based SMT [19] is shown in Figure 4. In every case, we incorporate

[9] https://catalog.ldc.upenn.edu/LDC2003T05

the proposed methodology with that specific approach to prepare the language model. It shows a significant improvement in every cases using our method.

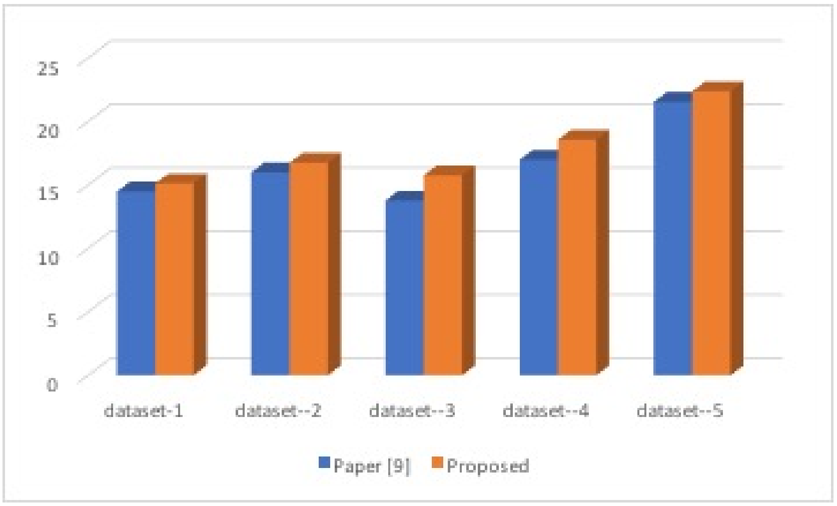

Fig. 3. Comparison analysis 1 with [9]

4 Conclusion and Future Work

In this paper, we have experimented the usage of domain-wise language models for Hindi to English translations. We identify the keywords and use them to incorporate the relative keywords. We have proposed a two-phased architecture to adapt a language model in such a way that the language model has a close proximity to different domains. By using this two-phased architecture, we showed a wide improvement in the health-domain related data. It is also proved that just because an LM contains a larger number of histories, does not mean that it will be performing a better translation as seen with the inclusion of wikipedia dump. The extraction of phrases as keywords not only give a lot of related data to add to the base, but also helps us finding the domain of test file if used in the source language. There was an although out increase in BLEU for the new adapted model of part-two of phase-two architecture. On the other hand, if enough parallel corpus can be obtained for each domain, we can also keep a language model set which is ready for each domain and use it for translation with the preferred domain of the input text. We can also test this concept on more languages and confirm and validate for more language pairs that the two-phase approach of language model works better for translation. In recent future, we will combine all statistical models to achieve better accuracy in the real time.

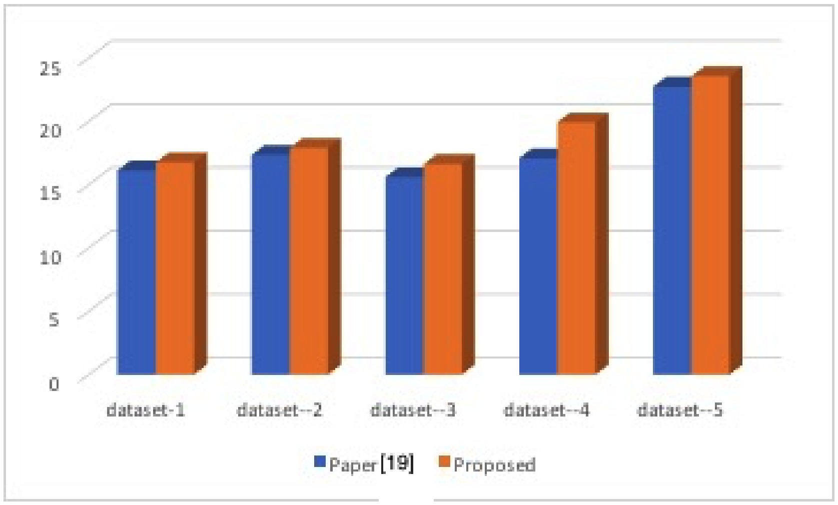

Fig. 4. Comparison analysis 2 with [19]

This approach may release from the bounding of the domain specific translation problem.

References

1. Banerjee, P., Du, J., Li, B., Kumar Naskar, S., Way, A., van Genabith, J.: Combining multi-domain statistical machine translation models using automatic classifiers. AMTA -9th Conference of the Association for Machine Translation in the Americas, USA (2010)
2. Deyi Xiong, Min Zhang, H.L.: Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (2011) 1288–1297
3. Abhinav Sethy, Panayiotis G. Georgiou, S.N.: Building topic specific language models from webdata using competitive models. (2005)
4. Thorsten Brants, Ashok C. Popat, P.X.F.J., Dean, J.: Large language models in machine translation. (2007)
5. Monica Gavrila, C.V.: Training data in statistical machine translation – the more, the better? –. Proceedings of Recent Advances in Natural Language Processing Hissar, Bulgaria, 12-14 September 2011 (2011)
6. Luong, T., Kayser, M., Manning, C.D.: Deep neural language models for machine translation. In: Proceedings of the 19th Conference on Computational Natural Language Learning, CoNLL 2015, Beijing, China, July 30-31, 2015. (2015) 305–309
7. Gennadi Lembersky, Noam Ordan, S.W.: Language models for machine translation:original vs. translated texts. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (2011)

8. Huerta, J.M.: An information-retrieval approach to language modeling : Applications to social data. Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media (2010) 7–8
9. Koehn, P., Och, F.J., Marcu, D.: Statistical phrase-based translation. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, Association for Computational Linguistics (2003) 48–54
10. Hiemstra, D.: A probabilistic justification for using tf$\times$ idf term weighting in information retrieval. International Journal on Digital Libraries **3**(2) (2000) 131–139
11. De Marneffe, M.C., MacCartney, B., Manning, C.D., et al.: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC. Volume 6., Genoa Italy (2006) 449–454
12. Koehn, P.: Europarl: A parallel corpus for statistical machine translation. In: Proceedings of MT Summit X, Phuket, Thailand (2005) 79–86
13. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. (2003)
14. Bojar, O., Diatka, V., Rychlỳ, P., Stranák, P., Suchomel, V., Tamchyna, A., Zeman, D.: Hindencorp-hindi-english and hindi-only corpus for machine translation. In: LREC. (2014) 3550–3555
15. Mitesh M. Khapra, Anup Kulkarni, S.S.P.B.: All words domain adapted wsd:finding a middle ground between supervision and unsupervision. Conference of Association of Computational Linguistics (ACL 2010) (2010)
16. Jha, G.N.: The tdil program and the indian langauge corpora intitiative (ilci). In: LREC. (2010)
17. Kunchukuttan, A., Mehta, P., Bhattacharyya, P.: The iit bombay english-hindi parallel corpus. arXiv preprint arXiv:1710.02855 (2017)
18. Bojar, O., Diatka, V., Rychlỳ, P., Stranák, P., Suchomel, V., Tamchyna, A., Zeman, D.: Hindencorp-hindi-english and hindi-only corpus for machine translation. In: LREC. (2014) 3550–3555
19. Chiang, D.: Hierarchical phrase-based translation. computational linguistics **33**(2) (2007) 201–228