Exploring Cross-lingual Summarization and Machine Translation Quality Estimation

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

Master of Science

in

Computer Science and Engineering

by Research

by

Nisarg Ketan Jhaveri
201302195
nisarg.jhaveri@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2018
CERTIFICATE

It is certified that the work contained in this thesis, titled “Exploring Cross-lingual Summarization and Machine Translation Quality Estimation” by Nisarg Ketan Jhaveri, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Vasudeva Varma
Acknowledgments

I take this opportunity to thank everyone who helped me in completing this thesis.
I would like to extend special thanks to my guide Prof. Vasudeva Varma for his constant support and guidance since I joined the lab. I would also like to thank Dr. Manish Gupta who constantly motivated and guided me through different parts of my research for this thesis.
I would like to thank my parents for their support, encouragement and everything else! They have always motivated me to work on and complete the thesis on time. Thanks to my brother Pranjal for random talks and ideas.
I would like to thank my friends Harshit, Madhuri, Smriti, Srishti and Vivek for helping me and keeping me motivated throughout my stay at IIIT Hyderabad. They have made my stay at IIIT pleasant and memorable.
I would like to thank all my lab mates at Information Retrieval and Extraction Lab for their valuable ideas and advices at various points. Special thanks to Nikhil Priyatam, Nitin Ramrakhiyani and Priya Radhakrishnan for their incredible support.
I would like to thank Ms. Poonam Valera, my parents Ketan and Mita Jhaveri, aunt Grishma Shah and Neel Trivedi for helping me with preparation of different datasets used in this thesis.
Lastly, I would like to thank the endless number of known or unknown people who have directly or indirectly helped me during my stay at IIIT, making it a memorable experience.
Abstract

The need for cross-lingual information access is more than ever with the easy access to the Internet. This is especially true in vastly multilingual societies like India. Cross-lingual summarization (CLS) aims to create summaries in a target language from a document or document set given in a different, source language. Cross-lingual summarization can have significant impact on enabling cross-lingual information access for millions of people across the globe who do not speak or understand languages that have a large representation on the web by making the most important information available in the target language in the form of summaries. It can also make documents originally published in local languages quickly accessible to a large audience which does not understand those local languages.

Working towards a better cross-lingual summarization system, we first create a flexible, web-based tool, referred to as the workbench, for human editing of cross-lingual summaries to rapidly generate publishable summaries in a number of Indian languages for news articles originally published in English. The workbench simultaneously collects detailed logs about the editing process at article, summary and sentence level. Similar to translation post-editing logs, such logs can be used to evaluate the automated cross-lingual summaries in terms of effort needed to make them publishable. We use the workbench to generate two manually edited datasets for different tasks.

We observed that quality of automatic translation is a major bottleneck when working on CLS. Translation Quality Estimation (QE) aims to estimate the quality of an automated machine translation (MT) output without any human intervention or reference translation. With the increasing use of MT systems in various cross-lingual applications, the need and applicability of QE systems is increasing. We study existing approaches and propose multiple neural network approaches for sentence-level QE with a focus on MT outputs in Indian languages. For this, we also introduce five new datasets for four language pairs: two for English–Gujarati, and one each for English–Hindi, English–Telugu and English–Bengali, which includes one manually post-edited dataset for English–Gujarati created using the workbench. We compare results obtained using our proposed models with multiple existing state-of-the-art systems including the winning system in the WMT17 shared task on QE and show that our proposed neural model which combines the discriminative power of carefully chosen features with Siamese Convolutional Neural Networks (CNNs) works significantly better for all Indian language datasets.

Later, we integrate our efforts on QE with cross-lingual summarization to study its effect on CLS. We extend a popular mono-lingual summarization method to work with CLS, along with a new objective function to take QE scores into account while ranking sentences for summarization. We experiment
with a number of existing methods for CLS with different parameters and settings and show comparative analysis.

At the end, we publish an end-to-end CLS software called clstk to make CLS accessible to a larger audience. Besides implementing a number of methods proposed by different CLS researchers over the years, the tool-kit also includes bootstrap code for easy implementation and experimentation with new CLS methods. We hope that this extremely modular tool-kit will help CLS researchers contribute more effectively to the area as well as developers to easily use existing methods in end-user applications.
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Major Challenges and Scope of the Work</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Methodology</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Results and Impact</td>
<td>3</td>
</tr>
<tr>
<td>1.5 Key Contributions</td>
<td>4</td>
</tr>
<tr>
<td>1.6 Organization</td>
<td>4</td>
</tr>
<tr>
<td>2 Related Work</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Cross-lingual Summarization</td>
<td>6</td>
</tr>
<tr>
<td>2.2 The Workbench</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Machine Translation Quality Estimation</td>
<td>8</td>
</tr>
<tr>
<td>2.3.1 Manual Feature Engineering based Approaches</td>
<td>8</td>
</tr>
<tr>
<td>2.3.2 Neural Network based Approaches</td>
<td>8</td>
</tr>
<tr>
<td>2.4 clstk</td>
<td>9</td>
</tr>
<tr>
<td>3 The Workbench</td>
<td>10</td>
</tr>
<tr>
<td>3.1 Main Features</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Interface</td>
<td>11</td>
</tr>
<tr>
<td>3.3 Architecture</td>
<td>13</td>
</tr>
<tr>
<td>3.4 Log Collection</td>
<td>14</td>
</tr>
<tr>
<td>3.5 Examples</td>
<td>15</td>
</tr>
<tr>
<td>3.6 Pilot Study</td>
<td>17</td>
</tr>
<tr>
<td>3.7 Data Collection</td>
<td>18</td>
</tr>
<tr>
<td>3.7.1 news.gu: A Dataset for Translation Quality Estimation</td>
<td>18</td>
</tr>
<tr>
<td>3.7.2 DUC 2004 Gujarati: A Dataset for Cross-lingual Summarization Evaluation</td>
<td>19</td>
</tr>
<tr>
<td>3.8 Discussion and Possible Extensions</td>
<td>19</td>
</tr>
<tr>
<td>4 Machine Translation Quality Estimation</td>
<td>21</td>
</tr>
<tr>
<td>4.1 Datasets</td>
<td>22</td>
</tr>
<tr>
<td>4.1.1 WMT17: English-German Dataset</td>
<td>22</td>
</tr>
<tr>
<td>4.1.2 news.gu: English-Gujarati Dataset</td>
<td>23</td>
</tr>
<tr>
<td>4.1.3 ILCI Parallel Corpora</td>
<td>23</td>
</tr>
</tbody>
</table>
## CONTENTS

4.2 Models for Translation Quality Estimation ........................................... 24  
4.2.1 Baseline Models .................................................................................. 24  
4.2.1.1 SVR Baseline .................................................................................. 24  
4.2.1.2 POSTECH Approaches .................................................................... 25  
4.2.1.3 SHEF/CNN Approach ..................................................................... 26  
4.2.2 Proposed Models ................................................................................ 27  
4.2.2.1 Recurrent Neural Network (RNN) Approaches ................................. 27  
4.2.2.2 Convolutional Neural Network (CNN) Approaches .......................... 28  
4.3 Experimental Settings .......................................................................... 30  
4.4 Evaluation and Results ......................................................................... 31  
4.5 Conclusions ........................................................................................... 33  

5 Cross-lingual Summarization .................................................................... 34  
5.1 Datasets ................................................................................................ 34  
5.1.1 DUC 2004 Gujarati ............................................................................. 35  
5.1.2 TAC 2011 MultiLing Pilot Dataset ..................................................... 35  
5.2 Methods ................................................................................................ 35  
5.2.1 SimFusion ............................................................................................ 36  
5.2.2 CoRank ............................................................................................... 36  
5.2.3 LinBilmes ......................................................................................... 36  
5.3 Evaluation Metrics ................................................................................ 37  
5.3.1 ROUGE Score ..................................................................................... 37  
5.3.2 Perplexity ............................................................................................ 37  
5.4 Experiments and Results ....................................................................... 38  
5.4.1 Quality Estimation ............................................................................. 39  
5.4.2 Sentence Simplification ....................................................................... 41  
5.4.3 Early-translation .................................................................................. 42  
5.5 Conclusions ........................................................................................... 42  

6 clstk ........................................................................................................... 43  
6.1 Components ........................................................................................... 44  
6.1.1 The Core ............................................................................................. 44  
6.1.2 Evaluation ........................................................................................... 44  
6.1.3 Translation .......................................................................................... 45  
6.1.4 Sentence Simplification ....................................................................... 45  
6.1.5 Translation Quality Estimation ............................................................. 45  
6.1.6 Summarizers ....................................................................................... 46  
6.2 Companion Tools and Roadmap ............................................................... 46  
6.3 Conclusions and Discussion ................................................................... 46  

7 Conclusions and Future Work ................................................................... 47  

Related Publications .................................................................................... 49  

Bibliography ................................................................................................. 50
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Overall flow of information</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Screenshot of the workbench, with a sentence highlighted.</td>
<td>12</td>
</tr>
<tr>
<td>3.3 Screenshot of a summary sentence being dragged for deletion.</td>
<td>13</td>
</tr>
<tr>
<td>3.4 Overall architecture diagram</td>
<td>13</td>
</tr>
<tr>
<td>3.5 Edits made to the mono-lingual summary</td>
<td>16</td>
</tr>
<tr>
<td>3.6 Edits made to the Hindi translation of the summary</td>
<td>16</td>
</tr>
<tr>
<td>4.1 Architecture of the $SHEF/CNN-C+F$ model</td>
<td>26</td>
</tr>
<tr>
<td>4.2 Architecture of the $RNN$ model</td>
<td>27</td>
</tr>
<tr>
<td>4.3 Architecture of the $RNN.summary-attention$ model</td>
<td>28</td>
</tr>
<tr>
<td>4.4 Architecture of the $CNN.Siamese$ model</td>
<td>29</td>
</tr>
<tr>
<td>4.5 Architecture of the $CNN.Combined$ model</td>
<td>30</td>
</tr>
<tr>
<td>5.1 Effect of QE on summarization for DUC 2004 Gujarati dataset</td>
<td>40</td>
</tr>
<tr>
<td>5.2 Effect of QE on summarization for MultiLing Pilot 2011 dataset</td>
<td>41</td>
</tr>
<tr>
<td>6.1 High-level architecture of $clstk$</td>
<td>44</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Summary of various kinds of logs collected by the workbench</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Statistics for summary level editing of example articles</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Statistics of example articles edited</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>Translation statistics for example articles</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>Comparison between human translation and Post-Editing machine translation</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Target languages and the number of sentence pairs in each dataset</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>Results for the Scoring Task, Pearson’s Correlation (* and † indicate statistically significantly better or worse ($p &lt; 0.05$) compared to POSTECH.two-step respectively)</td>
<td>31</td>
</tr>
<tr>
<td>4.3</td>
<td>Results for the Ranking Task, Spearman’s Correlation (* and † indicate statistically significantly better or worse ($p &lt; 0.05$) compared to POSTECH.two-step respectively)</td>
<td>31</td>
</tr>
<tr>
<td>4.4</td>
<td>Example of output by baseline (POSTECH.two-step), compared with our proposed model (CNN.Combined+fastText), across all datasets</td>
<td>33</td>
</tr>
<tr>
<td>5.1</td>
<td>Evaluation of different CLS methods on DUC 2004 Gujarati dataset</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Evaluation of different CLS methods on MultiLing Pilot 2011 dataset for Hindi</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>Evaluation of different CLS methods on MultiLing Pilot 2011 dataset for different languages</td>
<td>39</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this thesis we describe our efforts towards a better cross-lingual summarization system, with a focus on English to Indian languages summarization. For this, we study automatic Machine Translation Quality Estimation for a number of Indian languages and its applicability to improve CLS systems. We also publish a couple of tools related to CLS: one for rapid generation of human-edited cross-lingual summaries and another for experimentation with different CLS methods.

1.1 Motivation

Although English is the most popular language on the web, many highly-populated countries like Egypt, China and India have other (non-English) languages like Arabic, Chinese, and Hindi respectively as the most spoken languages. In Indian news media, most of the content gets published in English first and then in regional languages, especially for news categories such as national or international news, technology or lifestyle news. The delay can be just a few hours, days or sometimes the news does not appear in the regional languages at all. On the other hand, some content gets generated and consumed in regional languages alone. With the Internet becoming easily accessible and the rise of digital journalism, it is now crucial to make the large amount of information published in English or other popular languages on the Internet available to the readers of other languages having fewer native publications.

Advances in Machine Translation (MT) and other fields of Computational Linguistics in recent years make it possible to automate cross-lingual information access. However, the current state of machine translation is not able to generate publishable articles in most Indian languages from English. Although post-editing MT output has been shown to increase translators’ productivity over translating from scratch [1], it still requires a significant amount of human effort to produce articles consumable by end-user.

Cross-lingual summarization (CLS), which makes the highlights or summaries of articles originally published in one language available in another language, helps in making a large amount of critical information accessible as fast as possible with minimal human effort. In other words, automatic cross-lingual summarization systems can help in summarizing the information contained in a “rich language”
document to a “poor language”. In this work, we aim to make the process completely or partially automatic so that the gist of the articles can be published in regional languages with minimal delay.

Machine Translation is an important component of such cross-lingual summarization systems. However, the quality of the output obtained from these MT systems is neither perfect nor consistent across multiple test cases. This motivates us to work on Translation Quality Estimation (QE), which aims to estimate the quality of an MT output without any reference translation.

QE is now critically important with the increasing deployment of MT systems in practical environments. QE can also be extremely useful in various applications and systems such as cross-lingual summarization, cross-lingual information retrieval, etc., which rely on high quality translations [45, 9]. With the help of QE, a cross-lingual summarization system can automatically pick the sentences with good translation to be included in the summary. If the estimated quality is still unsatisfactory the system can alert the user about the poor quality or fall back to some alternate way to find a better translation. We experiment with usage of QE to improve CLS in Chapter 5 and show that QE can indeed be useful in generating better cross-lingual summaries.

1.2 Major Challenges and Scope of the Work

Working towards the larger goal of making information available across language boundaries, we scope this work to cross-lingual summarization. Cross-lingual summarization is a well-established, well-defined problem in the research community. An automatic cross-lingual summarization system aims to generate summaries in a target language, from a document or document set given in a different, source language.

For the purpose of this work, we limit the source language to English and experiment with two datasets having Gujarati and Hindi as target languages. We study different existing methods for our language pairs as well as experiment with several proposed methods for CLS. We also study the effectiveness of translation quality estimation on CLS.

We study the state of translation quality estimation for English to Indian language translations. Specifically, we experiment with datasets for English–Gujarati, English–Hindi, English–Telugu and English–Bengali language pairs.

Many of the existing methods for QE are resource-hungry. Similarly, many CLS techniques also require rich linguistic resources, especially methods focusing on abstractive summarization. These methods cannot work without certain linguistic resources such as dependency parsers or large parallel corpora, whereas most of the languages we focus on are resource-scarce from a computational linguistics perspective. Therefore, we scope our work to a low-resource setting, in which we minimize dependency on large manually-annotated resources. In this work we explore methods that work well despite small datasets or lack of external knowledge in terms of rich linguistic resources.

Another major challenge while working with QE is that most existing QE models are specific to particular language pairs, which may or may not work with other languages with different characteristics.
Additionally, the neural network models used for QE are sensitive to the differences in domains while training and testing. That is, if the model is trained on a dataset from one domain, the model doesn’t work well on test data from a different domain.

Annotated datasets, either for QE or cross-lingual summarization, are difficult to find and costly to prepare for our choice of languages. In this work, we explore ways to easily generate datasets required for cross-lingual summarization, as well as translation quality estimation. We also propose two manually annotated datasets, one each for cross-lingual summarization and translation quality estimation, along with several other datasets compiled from different sources.

1.3 Methodology

We first developed a workbench to rapidly generate human-edited cross-lingual summaries. The workbench can be used for translating articles and summaries for end-user consumption as well as various kinds of log collection. We use the workbench to collect different datasets used for translation quality estimation and cross-lingual summarization evaluation.

With the use of several datasets, including one dataset prepared using the workbench, we study translation quality estimation for Indian languages in a low-resource setting. We propose and compare various models and techniques for the same and show significant improvement over existing state-of-the-art methods. We study QE for four language-pairs, English–Gujarati, English–Hindi, English–Telugu and English–Bengali using five different datasets, including one manually post-edited dataset for English–Gujarati.

Later, we integrate our work on QE with CLS. We experiment with and compare different existing methods, originally proposed for different language pairs, on our datasets. We also experiment with several new techniques for CLS which uses QE and Sentence Simplification at different stages of the summarization process.

At the end, to catalyse future growth of the field, we publish a tool-kit for easy experimentation and implementation of different CLS techniques, called clstk. clstk currently contains implementation of several existing CLS methods along with bootstrap code and other modules required to implement new CLS methods.

1.4 Results and Impact

We compare results obtained using our proposed models for QE with multiple existing state-of-the-art systems including the winning system in the WMT17 shared task on QE. We show that our proposed neural model for QE which combines the discriminative power of carefully chosen features with Siamese Convolutional Neural Networks (CNNs) works significantly better for all Indian language datasets and works comparably for WMT17 English–German dataset.
This result, in addition to pushing the state-of-the-art of QE for Indian languages, signifies the need for different techniques for languages with different characteristics as well as the need for language-independent systems.

From our work on CLS, we can conclude that the use of QE has the potential to greatly improve CLS. Similarly, our preliminary experiments with sentence simplification also show promising results. We believe this result will encourage future researchers to work towards finding more efficient ways to use QE and sentence simplification to improve CLS further.

We believe the two systems we publish can have great impact in enabling research and end-user applications in the area. The workbench can be used as a go-to solution to generate cross-lingual summarization or translation datasets. Similarly, clstk could help end-users greatly in experimenting with different existing CLS techniques as well as researchers in implementing and publishing their models easily.

1.5 Key Contributions

- A workbench for rapid generation of cross-lingual summaries and translations.
- Study and proposal of different methods for Translation Quality Estimation for Indian languages in low-resource setting, along with several new datasets for the task.
- A new dataset for cross-lingual summarization evaluation for English to Gujarati summarization, which was manually created by translating summaries from DUC 2004 dataset.
- Study on effectiveness of QE and Sentence Simplification for cross-lingual summarization
- A tool-kit for easy experimentation with cross-lingual summarization, which contains different existing CLS methods along with bootstrap code to help implement and experiment with new methods easily.

1.6 Organization

In Chapter 2, we describe related work in the areas of cross-lingual summarization and translation quality estimation.

In Chapter 3, we describe the workbench in detail. We describe the design and architecture of the workbench along with the details on log collection and other aspects of the workbench. We also describe the datasets created with the help of the workbench in this chapter.

We describe our work on translation quality estimation for Indian languages in Chapter 4. We describe different existing baseline models used for comparison and later propose several new models for
the task. We also show comprehensive comparison of performance of different methods on different datasets.

In Chapter 5, we describe our experiments with different CLS methods. We also explain different experiments which used QE and sentence simplification at different stages in CLS process and compare their performance over different datasets.

Chapter 6 contains details on the cross-lingual summarization tool-kit, clstk. We explain the design and different components included in the tool-kit.

At the end, we conclude in Chapter 7 along with possible future work.
Chapter 2

Related Work

Cross-lingual summarization has seen some interest in last decade. Several innovative methods to improve the state of cross-lingual summarization have been proposed, which explore different aspects of the problem. We list and explain different methods proposed for CLS by different researchers over the years in Section 2.1.

To the best of our knowledge, there is no available end-to-end system that allows post-editing of automated cross-lingual summaries to generate publishable summaries and at the same time can collect useful data about the process. Section 2.2 describes different tools related to the workbench we developed along with fundamental patterns seen in works in the area which motivate the design of the workbench.

Translation quality estimation has been explored extensively in recent years. WMT12-17 (the 7th to 10th workshops on statistical machine translation and the 1st and 2nd conferences on machine translation) held a shared task on QE [10, 4, 5, 6, 7, 8]. The shared task has explored QE on several datasets and settings for English–Spanish and English–German language pairs over the years. Despite the high interest in the area, little work has been done to study QE for Indian languages. Related previous work on translation quality estimation can be organized into two broad kinds of approaches: manual feature engineering based approaches, and neural network based approaches. In Section 2.3, we explain the different approaches traditionally used for translation quality estimation, along with some recent state-of-the-art methods proposed in the area in more detail.

Despite the significant number of papers published on CLS in the past decade, there are no available usable tools or packages to help users try different methods or generate cross-lingual summaries without having to implement everything from scratch. There are multiple tools containing implementation of particular methods or a collection of different methods for mono-lingual text summarization. However, none of these contains methods designed for CLS nor they are suitable for working with CLS. More details about related tools are provided in Section 2.4.

2.1 Cross-lingual Summarization

In past decade, there has been a lot of work on CLS.
Most recently, Wan et al. [46] proposed a new framework for the task which extracts and ranks multiple summaries in the target language. They first extract multiple candidate summaries using several methods with different parameters. Then a new ensemble ranking method to rank candidate summaries using bilingual features is applied to get the best summary in the target language. This method is based on supervised learning and thus requires a large amount of annotated data for training, which is not easy to obtain for different language pairs.

Zhang et al. [49] proposed abstractive CLS via bilingual predicate-argument structure fusion. They first extract aligned predicate-argument structures (PAS) from the source and automatically translated documents. Later, new summary sentences are generated by fusing bilingual PAS elements to maximize both, salience and translation quality of the PAS elements using an integer linear programming algorithm.

Yao et al. [48] proposed a phrase-based compressive summarization model inspired by phrase-based translation models. The model relies on phrase-alignment information from a phrase-based MT system. With the transition to neural MT systems, the alignment information is no longer available in MT system. Additionally, getting the information from external word-alignment is also not desirable as NMT systems doesn’t guarantee phrase-aligned output.

Wan [44] proposed sentence ranking frameworks which simultaneously use both source-side and target-side information. The two methods proposed, SimFusion and CoRank are described in Section 5.2 in detail. We include these methods in our experiments.

Wan et al. [45] and Boudin et al. [9] proposed different approaches for summarization which also consider the quality of translation while extracting sentences. We experiment with the use of QE in our experiments using an adapted version of the popular sub-modular function maximization based summarization algorithm [28]. Our work on QE is described in detail in Chapter 4. Related work on QE is described in Section 2.3.

Sentence simplification has also been a popular topic in research community for quite some time now. For simplicity, we use the state-of-the-art neural text simplification models published by Nisioi et al. [34] for our experiments.

More details about our experiments and methods are present in Chapter 5.

2.2 The Workbench

The workbench is the first end-to-end system that allows post-editing of automated cross-lingual summaries to generate publishable summaries and at the same time can collect useful data about the process.

Computer Aided Translation (CAT) tools or translation post-editing tools like SDL Trados Studio¹, MateCat², OmegaT³, PET [1] and CATaLog online [37] are available. They compare with our system

¹http://www.sdltrados.com/
²https://www.matecat.com/
³http://omegat.org/
in the following ways: 

a) While they support translation post-editing, the workbench has support for post-editing mono-lingual or cross-lingual summaries as well. 
b) A few of them allow the recording of various kinds of logs about the translation post-editing process while we allow recording comprehensive logs about the human editing of summary and translations.

Some work exists on cross-lingual summarization as described in Section 2.1. Most extractive cross-lingual summarization systems have a sequential pipeline architecture. Additionally, most of them output a proposed mono-lingual summary and its translation at the end [29, 35, 40, 45, 44, 48]. This motivates the design of the workbench where the annotator can edit the mono-lingual summaries and their translations easily to get publishable cross-lingual summaries, and which can also collect various logs.

Chapter 3 contains more details about the design and usage of the workbench.

2.3 Machine Translation Quality Estimation

Related previous work on translation quality estimation can be organized into two broad kinds of approaches: manual feature engineering based approaches, and neural networks based approaches. WMT12-17 shared task on QE [10, 4, 5, 6, 7, 8] has recorded the overview and progress of the field over the years.

2.3.1 Manual Feature Engineering based Approaches

Many previous studies on QE were predominantly based on feature engineering. Manual feature engineering can be costly, especially because it needs to be done for each language pair separately.

For Indian languages, few studies have been done, predominantly for English–Hindi language pair. Most of the approaches, most recently by Joshi et al. [19], are based on manual feature engineering, and traditional classification methods. We show in our experiments, that the neural network based models perform significantly better for all language pairs and datasets.

2.3.2 Neural Network based Approaches

In recent years, many deep learning methods have also been proposed for QE. Patel and Sasikumar [38] proposed the use of Recurrent Neural Network Language Modeling (RNN-LM) to predict word-level quality labels using the bilingual context window proposed by Kreutzer et al. [25]. Several other neural models also use the bilingual context window approach to compose the input layer, which takes the target word and the aligned source word and their contexts as input [32, 33, 31]. These models, however, require word alignment information from the MT system or need to align the words using some external parallel corpora. Since our datasets are prepared using neural MT systems, we do not have alignment information from MT system. Additionally, we do not have enough resources to create external word-aligners for each language-pair. As a result, we do not include systems that need word alignment information in our experiments.
Kim and Lee [21, 20] and Kim et al. [22, 23] have studied and proposed different end-to-end neural network based models, primarily based on predictor-estimator architecture. We compare with the architecture described by Kim et al. [22] in our experiments. The architecture is explained in Section 4.2.1.2.

Paetzold and Specia [36] propose a character-level Convolutional Neural Network (CNN) architecture combined with engineered features. The system is comparable to our proposed work in two ways: 1) They do not use any external data or resources. 2) They also use a CNN-based architecture for QE. However, the final architectures are significantly different. Their best system, SHEF/CNN-C+F, is explained in Section 4.2.1.3.

Chapter 4 contains more details about our work on QE, including detailed description of different models used.

2.4 clstk

clstk is a tool-kit for cross-lingual summarization. This section lists different related tools, generally designed for mono-lingual summarization.

There are multiple tools containing implementation of particular methods or a collection of different methods [15] for mono-lingual text summarization4. Such tools include MEAD5, pytextrank6, sumy7, texteaser8, summarization using TensorFlow9, and TextRank Summarization using gensim10. None of these tools contains methods designed for cross-lingual summarization. Additionally, most existing packages do not even contain critical components like the translation module required for implementing CLS methods.

With an aim of filling the gap between research work in CLS and its usability in real-world applications, we publish clstk for CLS, described in Chapter 6. clstk is the first such tool-kit designed to natively support cross-lingual summarization, along with a collection of different methods for the task.

---

5http://www.summarization.com/mead/
6https://github.com/ceteri/pytextrank
7https://github.com/miso-belica/sumy
8https://github.com/MojoJolo/textteaser
9https://github.com/tensorflow/models/tree/master/research/textsum
10https://radimrehurek.com/gensim/summarization/summariser.html
Chapter 3

The Workbench

We have developed a workbench for human editing of cross-lingual summaries to rapidly generate publishable summaries in target languages, with the aim of making news content in one language accessible in another language as fast as possible. The workbench is a pluggable web-based tool, which implements a pipeline for cross-lingual summarization of news articles. The news articles in the source language are first preprocessed by the workbench. After automatic processing, the articles are sent to humans for post-editing the summary and the automatically translated text to produce publishable cross-lingual summaries.

In this chapter, we describe the workbench, a web-based tool for post-editing cross-lingual summaries, and briefly describe a pluggable pipeline for cross-lingual summarization. The source code of the workbench is available at https://github.com/nisargjhaveri/news-access.

We have a number of machine translation and summarization systems available, working with different languages, and the list is ever-growing. We may also have different translation and summarization systems available to work with different languages. Keeping this in mind, we have made the workbench pluggable by design. The details about the architecture is presented in Section 3.3.
With the carefully designed user experience, we have tried to ensure high productivity while using the workbench. The user interface is available online so that the annotators can work from virtually anywhere without requiring local installations of the tool in a number of machines. Additionally, an online system makes it easier to make the cross-lingual summaries available in real time.

The workbench also records edit logs and other parameters while editing the automatic summary and translation. These logs can give meaningful insights for the task or can also be used as continuous feedback to the system.

The rest of the chapter is organized as follows: Section 3.1 list main features of the workbench. Section 3.2 and Section 3.3 describes the interface and architecture of the workbench in detail. Section 3.4 lists the details about various kinds of logs collected by the workbench. In Section 3.5 and Section 3.6 we show some example usage and results of a pilot study. We describe two datasets collected using the workbench in Section 3.7. At the end, Section 3.8 we discuss other possible use-cases for the workbench and possible extensions of the workbench.

3.1 Main Features

The workbench is a flexible, language-independent tool for editing automatically generated cross-lingual summaries. The main features of the workbench are:

- The workbench provides a unique user-friendly environment for annotators to edit summaries in the source language, the cross-lingual summaries, and optionally, translation of the original article, in a seamless way.
- The pluggable and generic architecture provides the possibility of using the workbench for almost any language pair, and with any set of external tools to plug into the pipeline.
- The workbench collects a wide range of logs from the editing jobs, which can be used as feedback by any module in the pipeline to improve the automatic process over time, and can also provide useful insights for the task in question.

3.2 Interface

Figure 3.2 shows the default interface of the workbench with source and target languages being English and Hindi respectively.

The default interface for the workbench splits the working area in three vertical columns. The first column contains the source article. The second column contains the translation of the article in the target language. Paragraphs are aligned in both these columns and the scrolling is synced. The third column contains the mono-lingual extractive summary in source language and the translation of the summary in the target language.
Naval officer dies of gunshot wounds

Kochi, Oct 1 (PTI). A Naval officer died of gunshot wounds at the Naval base here this morning, a defence spokesman said.

Further details about the incident were awaited.

The spokesman said a sailor on duty at the naval base sustained fatal bullet injury due to the firing of his duty weapon.

The injured sailor was rushed to the naval hospital INHS Sanjivani where all efforts to save him remained unsuccessful, he said.

The naval base here houses the headquarters of the Southern Naval Command, which is one of the three main formations of the Indian Navy.

A Naval officer died of gunshot wounds at the Naval base here this morning, a defence spokesman said. Further details about the incident were awaited. The naval base here houses the headquarters of the Southern Naval Command, which is one of the three main formations of the Indian Navy.

The annotator can edit the sentences in the mono-lingual summary and the translations of the sentences in the summary with the source and automatically translated article as the context. Optionally, the sentences in the translation of the complete article can also be edited.

To edit the mono-lingual summary, a simple drag-drop interface is provided to make the usage intuitive. The annotator can add or remove sentences from the mono-lingual summary with an easy drag-drop interface. When the annotators starts dragging a sentence from the mono-lingual summary, a bin is shown near the end of the summary box, see Figure 3.3. The annotator can either drop the sentence at some position inside the box to reorder or they can drop it on the bin to remove the sentence from the summary. To add a sentence from original article into the summary, the annotator has to pick a sentence from the source article and drop into the summary box at the desired position. The position of the sentence is previewed live while the sentence is being dragged. Additionally, the content of the sentences in the summary can be changed in-line by clicking on it. Any changes in mono-lingual summary reflects in the cross-lingual summary immediately. The number of characters in the summary is also shown below the summary.

Contrary to the translation post-editing tools, since the structure of the article is important for summarization, we cannot show the original article broken into segments, or even sentences. The view shows paragraphs of the source and target article aligned with synced scrolling for easy navigation. To distinguish between multiple sentences in the paragraph, the sentence which is hovered on or is active for editing is highlighted along with all the linked sentences, such as the source sentence, the correspond-
ing sentence in mono-lingual summary if included, and the corresponding sentence in cross-lingual summary.

3.3 Architecture

Figure 3.3: Screenshot of a summary sentence being dragged for deletion.

Figure 3.4: Overall architecture diagram
Figure 3.4 shows the overall architecture of the system. The dotted arrows in the diagram indicate a possible flow, which is quite flexible.

The modular architecture of the system results in high flexibility in terms of use of external resources. The article provider module retrieves news article from either a corpus or other external sources.

The article is then passed through the automated pipeline, which depending on specifics, populates the article object with translation of the article, mono-lingual summary and the cross-lingual summary obtained based on the mono-lingual summary.

Each module in the pipeline can potentially access external resources, make API calls or access previous edit logs in order to generate the output. The modules themselves can be composite of other components.

In our system, summarization and translation are the two major modules along with pre-processing stage. Multiple summarizers and translators are integrated in summarization and translation module respectively. For example, in our case, the translation module may use Google Translate\(^1\) or Microsoft Translator\(^2\) depending on the language pair or a configuration option.

### 3.4 Log Collection

<table>
<thead>
<tr>
<th>Log level</th>
<th>Log type</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence-level</td>
<td>Focus/Blur</td>
<td>When the sentence is activated for editing or de-activated</td>
</tr>
<tr>
<td></td>
<td>Keystrokes</td>
<td>Key presses, along with IME compositions</td>
</tr>
<tr>
<td></td>
<td>Text selection</td>
<td>Text selection by any means, e.g. with mouse or shift+arrow keys</td>
</tr>
<tr>
<td></td>
<td>Text input</td>
<td>When the text is changed, including copy/cut/paste events</td>
</tr>
<tr>
<td>Summary-level</td>
<td>Add/Remove sentence</td>
<td>When a sentence is added or removed from the summary</td>
</tr>
<tr>
<td></td>
<td>Sentence reordering</td>
<td>When the order of the sentences in the summary is changed</td>
</tr>
<tr>
<td>Article-level</td>
<td>Page events</td>
<td>Events about page load, article load, and completion of editing</td>
</tr>
</tbody>
</table>

Table 3.1: Summary of various kinds of logs collected by the workbench

One of the primary goal of the workbench is to generate a large amount of cross-lingual news summaries in multiple languages, which can be used to build new systems in the area. Along with that, various kinds of user-activity logs are collected by the workbench, which can also be used to evaluate or improve new systems.

Table 3.1 summarises the different kinds of logs collected by the workbench. All events are logged with precise time in milliseconds.

The following sentence-level editing events are logged for all editable sentences in the source language or the target language.

- Focus/Blur: Focus and blur events are logged for each editable item when the item is activated for editing and when leaving the focus from the item respectively. A single item can be focused

\(^1\)https://translate.google.com/
\(^2\)https://www.microsoft.com/translator
multiple times. The time spent on the item can be estimated by adding difference between all focus-blur pairs for the item.

- **Keystrokes**: All keystroke information available are logged, including IME (Input Method Editor) composition information.

- **Text selection**: Text selection by any means (mouse, shift+arrow) is logged.

- **Text input**: For all items, all the changes in text are logged, along with all copy/cut/paste events. This is specifically important as keystroke logging doesn’t provide accurate and wholesome information in case of complex scripts and use of IMEs.

For translation post-editing, Human-targeted Translation Edit Rate (HTER) [42], along with information about insertion, deletion, and substitution of single words as well as shifts of word sequences can be calculated and stored when the translation is finalized.

For editing of mono-lingual summary, along with the sentence level editing logs, following summary-level events are also logged by the workbench.

- **Add/Remove sentence**: An event is logged when a sentence is added to the summary or removed from the summary, along with its position in the summary before adding or after removing.

- **Sentence reordering**: When the ordering of the sentences in the summary is changed, an event is logged with the information about previous and new ordering of sentences in the summary.

Additionally, some article-level events described below are also logged.

- **Page events**: The events about page load, article load, and completion of editing of an article are logged.

The total time taken for editing can be calculated as the time difference between completion of editing and article load time, and reducing the difference by the amount of time the annotator was marked away.

In addition to these logs, annotator’s browser and platform information is also collected. This information is important to give us a better idea of client’s environment, and helps interpreting the logs with a better context.

With all these logs, we can virtually replay the complete editing process for any article.

### 3.5 Examples

Figure 3.2 shows an example article without editing. The title of the article is wrongly translated to “नौसेना अधिकारी गोलीबारी घाव से मौत (nausenA adhikArI golIbArI ghAvoM se mauta)”, which is neither syntactically nor semantically correct. Using the workbench, an annotator can fix the translation to “नौसेना अधिकारी गोलीबारी घाव से मौत (nausenA adhikArI golIbArI ghAvoM se mauta)”.

15
A Naval officer died of gunshot wounds at the Naval base here this morning, a defence spokesman said. Further details about the incident were awaited. The naval base here houses the headquarters of the Southern Naval Command, which is one of the three main formations of the Indian Navy. The spokesman said a sailor on duty at the naval base sustained fatal bullet injury due to the firing of his duty weapon.

**Removed sentences, Added sentences**

![Figure 3.5: Edits made to the mono-lingual summary](image)

Additionally, we can see that the summary shown in Figure 3.2 is not very informative, as it is not completely coherent with the title. Figure 3.5 shows the edits made to the mono-lingual summary using the workbench, which is also reflected in cross-lingual summary. Once the sentences in mono-lingual...
summary are fixed, with a few corrections in the translation of the summary, similar to what we have shown for the title, we can generate a publishable cross-lingual summary. Figure 3.6 shows an example of edits made to the translation of the summary in Hindi.

For demonstration, we used the workbench to generate cross-lingual summaries of three randomly selected articles of similar sizes. The articles were originally in English and we set the target language to Hindi. Table 3.2 shows the total time taken to generate human edited cross-lingual summaries and the summary-level edits made to the articles. Table 3.3 shows the number of sentences, words and characters in source articles, as well as mono-lingual and edited cross-lingual summaries. Table 3.4 shows the estimated time taken for editing erroneous translations and HTER along with number of insertions, deletions, substitutions and shifts performed.

### 3.6 Pilot Study

We conducted a pilot study to understand the usability and effectiveness of the workbench. One language expert was hired to generate translations of the articles and cross-lingual summaries for English-Gujarati language pair using the workbench.

For the pilot study, the workbench was configured to allow editing of the translation of the entire article along with the mono-lingual and the cross-lingual summary. The mono-lingual extractive summaries were provided by Veooz and Google Translate was used for automatic translations.

The language expert was told to follow the following sequence to correct the article.

- First, correct the translation of all the sentences in the source article.
- Once the translation is corrected, fix the mono-lingual summary.
- The cross-lingual summary automatically picks up the sentences and their translation included in the mono-lingual summary from the article. Fix the translation errors in cross-lingual summary if required.

In this setting, we observed that the mono-lingual summary was never getting changed. On investigating, the feedback from the language expert was that the summaries are “good enough”. Although, the summaries were not always good, we observed that in this setting, due to multiple tasks (translation of the article, mono-lingual and cross-lingual summary), the “good enough” summaries were not getting enough attention.

Another feedback from the language expert was that “it would be easier and faster to translate by hand instead of post-editing the machine translation”. To verify this claim, we did a small experiment to compare time and effort taken in both the cases. The results and statistics of the experiment are shown in Table 3.5.

---

3 [https://www.veooz.com/](https://www.veooz.com/)
Table 3.5: Comparison between human translation and Post-Editing machine translation

<table>
<thead>
<tr>
<th></th>
<th>Translation by hand</th>
<th>Post-editing machine translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. time per article</td>
<td>14.9 min</td>
<td>9.85 min</td>
</tr>
<tr>
<td>Avg. number of sentences per article</td>
<td>13.70</td>
<td>13.17</td>
</tr>
<tr>
<td>Avg. number of sentences edited per article</td>
<td>100%</td>
<td>57.8%</td>
</tr>
<tr>
<td>Total number of articles included in study</td>
<td>112</td>
<td>395</td>
</tr>
</tbody>
</table>

We can clearly see that the post-editing approach is faster. We see that post-editing machine translations takes about 33% lesser time compared to translating by hand. Though, post-editing machine translation takes lesser time, we notice that it might be more difficult to correct an erroneous translation compared to translation by hand and the difference noticed in time taken might be simply due to the fact that all sentences do not need to be corrected.

3.7 Data Collection

With the use of the workbench, we prepared two different types of datasets. This also shows the usefulness and flexibility of the workbench in practical use.

One dataset is for translation quality estimation, which fundamentally contains source sentences, automatic translation and post-edited translation to get quality scores of for the translations.

Another dataset is for cross-lingual summarization evaluation, in which the summaries are available in the target language for the documents in different, source language. We created such a dataset by manually translating summaries from a benchmark mono-lingual summarization dataset, DUC 2004.

3.7.1 news.gu: A Dataset for Translation Quality Estimation

We compiled manual post-editing data generated during the pilot study described in Section 3.6 to prepare a dataset for translation quality estimation. News articles from various sources were translated to Gujarati from English using the Neural Machine Translation (NMT) API provided by Google Translate and post-edited by one professional translator, who is also a native Gujarati speaker, over a duration of two months.

The dataset contains source sentences, their automatic translation and manually post-edited sentences, along with other measures calculated by the workbench, for example, time taken to edit the sentence, keystrokes, HTER score, etc. The dataset contains a total of 5612 sentences.

The specifics of usage of the dataset is described in Chapter 4.
3.7.2 DUC 2004 Gujarati: A Dataset for Cross-lingual Summarization Evaluation

We also used the workbench to create a new dataset for CLS evaluation for English to Gujarati summarization. The dataset was created by manually translating all summaries of all 50 document sets from DUC 2004 multi-document summarization dataset to Gujarati using the workbench.

For the preparation of the dataset, the workbench was configured to allow annotators to edit translations only. Summaries in English from the DUC 2004 dataset were given for translation. Automatic translations from Google Translate were provided as a reference. The translators were told to make the translations of summaries as natural as possible in Gujarati and not to minimize edits. Five native Gujarati speakers, including the author, translated the summaries to Gujarati depending on their availability.

The dataset now contains source documents in English from the original DUC 2004 corpus, and summaries in Gujarati. This dataset can be used as a benchmark dataset for CLS evaluation, for the given language pair.

We use this dataset later in Chapter 5 to compare different CLS methods.

3.8 Discussion and Possible Extensions

The workbench can be used to collect human edited cross-lingual summaries along with monolingual summaries and translations of the original article for English to a number of Indian languages. Such a dataset can be used for training statistical mono-lingual or cross-lingual summarizers as well as can help research efforts on machine translation. We also aim to collect comprehensive logs and use them as continuous feedback to some of the modules in our pipeline.

As the workbench and the architecture is not limited to a specific set of languages, the same can be used with a number of other language pairs too.

The flexibility of the workbench and the pipeline makes it possible to use the system for a number of other related tasks. The workbench can be used for extractive or abstractive mono-lingual summary generation or post-editing. It can also be used just as another translation post-editing tool, or can be used to prepare paraphrasing datasets.

Apart from the human edited data collected by the workbench, the logs collected about the process can also be important. Keystroke logs, along with translation time and HTER is a common measure of Translation Quality Estimation [1]. Automated Post-Editing (APE) systems also use similar information to automatically post-edit and try and remove systematic errors made by a particular MT system.

The workbench can also be used to compare different settings of the pipeline such as different MT systems or different approaches to summarization, etc., by comparing time taken to edit or other relevant measures.

In future, following possible modules could be integrated with the workbench to improve the usability and the effectiveness of the workbench.
• Cross-lingual dictionaries to refer words and get possible translations. The dictionary can be triggered by double-clicking or selecting a word or phrase and can be shown as a pop-up to ease the work-flow.

• A Translation Quality Estimation module that adapts and learns from previous edits and can highlight sentences or part of sentences that need attention.

• An Automatic Post-Editing module, to automatically remove common errors made by the MT system, based on previous usage of the workbench.
Chapter 4

Machine Translation Quality Estimation

In recent years, Machine Translation (MT) systems have seen significant improvements. However, the quality of the output obtained from these MT systems is neither perfect nor consistent across multiple test cases. The task of Translation Quality Estimation (QE) aims to estimate the quality of an MT output without any reference translation.

Word, phrase, sentence or document level QE has been studied extensively by various researchers. WMT12-17 (the 7th to 10th workshops on statistical machine translation and the 1st and 2nd conferences on machine translation) held a shared task on QE [10, 4, 5, 6, 7, 8]. The shared task has explored QE on several datasets and settings for English–Spanish and English–German language pairs over the years.

Little work has been done to study QE for Indian languages. In this work, we focus on four Indian languages: Telugu, Hindi, Gujarati and Bengali. While English is a West Germanic language\(^1\) that originated from Anglo-Frisian dialects, Gujarati, Hindi and Bengali are Indo-Aryan languages\(^2\), and Telugu is a Dravidian language\(^3\).

Indian languages are relatively free word order languages and morphologically richer when compared to English. Additionally, some Indian languages, for example Telugu, are highly agglutinative. In comparison with English, Hindi has approximately twice as many vowels and consonants. Although Hindi has tenses similar to those used in English, there is a lack of correspondence in their use to express various meanings. Gender and status relations between speakers cause morphological changes in Hindi words, unlike in English. Compared to English, Bengali uses onomatopoeia extensively, and so one has to convey that through particular adjectives and adverbs. Besides these differences, there are some phrases, idioms and compound words in English which do not have equivalents in Indian languages due to significant cultural differences.

Because of the differences in the characteristics of the languages involved, existing methods for QE may or may not be effective for all language pairs. We experiment with multiple datasets in different language pairs, each involving English and an Indian language, to study the effectiveness of various models on these datasets.

---

\(^1\)https://en.wikipedia.org/wiki/West_Germanic_languages

\(^2\)https://en.wikipedia.org/wiki/Indo-Aryan_languages

\(^3\)https://en.wikipedia.org/wiki/Dravidian_languages
In addition to the different characteristics of Indian languages, many of these languages are resource-scarce, from a Computational Linguistics perspective. Linguistic resources like dependency parsers or semantic role labelers are not available for most languages we experiment with. Additionally, large amount of manually annotated data, such as parallel corpora are also difficult and costly to obtain. Hence, in this work, we try to minimize dependency on external large datasets, especially ones which require manual annotation. We hope that the QE accuracy can be further improved using such extra information and plan to explore it as future work.

We study QE for Indian languages with the use of five datasets, for four different language pairs. One dataset, news.gu, described in Section 3.7.1 and Section 4.1.2 has been prepared by manually post-editing MT outputs. The other four datasets described in Section 4.1.3 make use of existing parallel corpora to create datasets for QE. All datasets are prepared using Neural Machine Translation (NMT) API provided by Google Translate\textsuperscript{4}. To the best of our knowledge, we are the first to study QE when using the NMT system.

In this chapter, we evaluate the effectiveness of various state-of-the-art systems (proposed for other language pairs) including the winning system of the WMT17 shared task on various Indian language datasets. We also propose and evaluate multiple neural network models for QE. Finally we show that one of our proposed models CNN.Combined, described in 4.2.2.2, gives best results on most Indian language datasets.

The rest of the chapter is organized as follows: Section 4.1 describes the datasets used for the experiments. Section 4.2 describes different methods and proposed models used for our experiments. Section 4.3 contains a few notes about the experimental settings. Section 4.4 provides analysis and related discussions. Finally, we conclude this chapter with a brief summary in Section 4.5.

### 4.1 Datasets

We used six different datasets for five different language pairs for our experiments. Source language is English for all the datasets. All datasets are split into the typical train, development and test sets. We use the train set for training our models, development set for tuning parameters and the test sets to report various metrics. Table 4.1 shows the target languages and sizes of all the datasets. We describe these datasets in detail in this section.

#### 4.1.1 WMT17: English-German Dataset

We use the English–German dataset released as part of the WMT17 QE Shared Task [8]. The dataset contains text from the Information Technology domain, translated from English to German using a statistical MT system and post-edited by professional translators. The dataset contains source sentences, MT sentences and post-edited sentences, along with Human-targeted Translation Edit Rate (HTER) scores [42] for each sentence pair.

\textsuperscript{4}https://translate.google.com/
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Target Language</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmt17</td>
<td>German (de)</td>
<td>23,000</td>
<td>1,000</td>
<td>2,000</td>
</tr>
<tr>
<td>news.gu</td>
<td>Gujarati (gu)</td>
<td>4,489</td>
<td>561</td>
<td>562</td>
</tr>
<tr>
<td>ilci.gu</td>
<td>Gujarati (gu)</td>
<td>40,000</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>ilci.hi</td>
<td>Hindi (hi)</td>
<td>40,000</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>ilci.te</td>
<td>Telugu (te)</td>
<td>40,000</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>ilci.bn</td>
<td>Bengali (bn)</td>
<td>40,000</td>
<td>5,000</td>
<td>5,000</td>
</tr>
</tbody>
</table>

Table 4.1: Target languages and the number of sentence pairs in each dataset

Translation Edit Rate (TER) is computed as the minimum number of insertion, deletion, substitution and shift operations needed to be done on MT sentence to match a reference sentence, normalized by the length of the reference sentence. The way the HTER differs from TER is that for HTER, there is no pre-decided reference sentence. There is a human in the loop. The human expert generates the targeted reference by editing the system hypothesis, until it is fluent and has the same meaning as the original source sentence. We use the HTER scores reported as quality scores for this dataset. The dataset contains 23,000, 1,000 and 2,000 sentences in the training, development and test sets respectively.

### 4.1.2 news.gu: English-Gujarati Dataset

This is a new QE dataset for the English–Gujarati language pair, prepared using the workbench described in Chapter 3. The method of preparation of the dataset is described in Section 3.7.1. The quality scores were computed as HTER between automatically translated sentence and the post-edited sentences using the *tercom 0.7.2* tool.

The dataset contains a total of 5612 sentences, which was split randomly into training, development and test sets of sizes 4489, 561 and 562 sentences respectively.

### 4.1.3 ILCI Parallel Corpora

A parallel corpora for many Indian language pairs, including English has been released by the Indian Languages Corpora Initiative (ILCI) [13]. We use the parallel corpora of the health and the tourism domain, having 25,000 sentences for each of the domains for each language pair. We prepare the QE datasets using this for translation from English to four Indian languages, namely, English–Gujarati, English–Hindi, English–Telugu and English–Bengali.

To use the parallel corpora for the QE task, we obtain translations using Google Translate from English to all the target languages. We computed the quality scores as the TER between the MT output and the reference sentences using *tercom 0.7.2*.

---

6. [http://tdil-dc.in](http://tdil-dc.in)
7. [https://translate.google.com/](https://translate.google.com/)
The datasets contain a total of 50,000 sentences each, which was divided randomly into training, development and test sets of sizes 40,000, 5000 and 5000 sentences respectively.

4.2 Models for Translation Quality Estimation

This section describes various models used for experiments and evaluation. We first discuss the baseline models in Section 4.2.1 and then the proposed models in Section 4.2.2.

4.2.1 Baseline Models

In this sub-section, we discuss previously proposed models for translation quality estimation and their variations. Section 4.2.1.1 describes baseline model based on Support Vector Regression (SVR). Section 4.2.1.2 describes POSTECH.two-step and POSTECH.multi-task models. Section 4.2.1.3 describes the SHEF/CNN-C+F model.

4.2.1.1 SVR Baseline

The official baseline for WMT17 QE shared task is a Support Vector Regression (SVR) [16] model trained with 17 features for the task. Some of these features use external data such as language models or word alignments trained on large parallel corpora. These features were adapted to use whatever scarce resources are available for our set of target languages as follows. Two features requiring word alignment tables were removed. No external data was used to compute the language models or n-gram counts. Additionally, a few features were added such as, average target token length and depth of parse tree of source sentence. The source parse tree were computed using Stanford CoreNLP toolkit [30], this was possible as all the datasets have English as the source language. We call this model SVR.baseline.

The features used are:

- Number of tokens in the source sentence
- Number of tokens in the target sentence
- Average source token length
- Average target token length
- LM probability of source sentence
- LM probability of target sentence
- Number of occurrences of the target word within the target hypothesis (averaged for all words in the hypothesis - type/token ratio)
- Number of words in target per source word (number of words in target divided by number of words in source sentence)
- Number of words in source per target word
- Percentage of unigrams in quartile 1 of frequency (lower frequency words) in a corpus of the source language (SMT training corpus)
- Percentage of unigrams in quartile 4 of frequency (higher frequency words) in a corpus of the source language
- Percentage of bigrams in quartile 1 of frequency of source words in a corpus of the source language
- Percentage of bigrams in quartile 4 of frequency of source words in a corpus of the source language
- Percentage of trigrams in quartile 1 of frequency of source words in a corpus of the source language
- Percentage of trigrams in quartile 4 of frequency of source words in a corpus of the source language
- Number of punctuation marks in the source sentence
- Number of punctuation marks in the target sentence
- Depth of parse tree of source sentence

4.2.1.2 POSTECH Approaches

POSTECH’s participation was the winning system at the WMT17 shared task, which uses a predictor-estimator architecture, many variations of which have been studied and proposed by Kim et al. [22, 23] and Kim and Lee [21, 20]. We follow the architecture described by Kim et al. [22] for this work.

Kim et al. [22] describe a two-step end-to-end neural QE architecture, called predictor-estimator architecture. The predictor-estimator architecture consists of two types of neural network models: 1) word predictor, which is trained on parallel corpora, i.e. using source and reference translations. 2) quality estimator, a neural regressor, trained on QE data.

The first model, word predictor, tries to predict each word in the target sentence using the source sentence and the remaining target sentence as context. They propose an RNN encoder-decoder [11, 2] model based word predictor, which uses bidirectional RNN in encoder as well as decoder to use the source sentence information as well as the entire left and right context of target sentence to predict each word.

The estimator part, then, extracts a quality estimation feature vector (QEFV) for each word in MT sentence using internal network connections of the word predictor network. For sentence-level QE, the QEFVs are then passed to bidirectional RNN to obtain a summary vector, which, then, is passed to regression layer which generates quality score for sentences.

We define two variations of the model for our experiments: POSTECH.two-step and POSTECH.multi-task.

POSTECH.two-step trains the two models, word predictor and quality estimator separately as described by Kim et al. [22]. Input to the word prediction step is source and reference sentences, and the outputs are the predicted words. Whereas, the quality estimator takes source and MT sentence as input.
and outputs quality score for the sentence. No external parallel corpora have been used for pre-training the word predictor as it is not available for most of the language pairs we work with.

The main idea of POSTECH system proposed by Kim et al. [22] is to take advantage of pre-training of word predictor using large external parallel corpora. Since we do not use any external corpora, we propose a variation of this model, which jointly learns both, word predictor and quality estimator, in a multi-task setting. We call this model POSTECH.multi-task. The inputs to this model are the source and MT sentence, and the outputs are predicted words and quality score.

Recently, Kim et al. [23] proposed single-level and multi-level stack propagation based learning for the two steps. We experimented with single-level stack propagation, as we do not have necessary training data for all sentence, word and phrase level QE, which the multi-level model requires. In our experiments, we did not see any significant improvement across datasets between single-level stack propagation [23] and two-step learning [22].

4.2.1.3 SHEF/CNN Approach

![Architecture of the SHEF/CNN-C+F model](image)

Paetzold and Specia [36] propose an architecture that combines engineered features and character-level information using deep Convolutional Neural Networks (CNN) and Multi-Layered Perceptrons (MLP). The model SHEF/CNN-C+F has three parts, sentence encoders for source and MT sentence, MLP for engineered features and a final layer to combine both and generate quality scores.

The sentence encoder takes the sequence of characters as input, and converts it to a sequence of character embeddings. They stack four pairs of convolution and max-pooling for each window size. Each stack is applied to character embeddings in parallel, and later flattened and concatenated to get a
sentence vector. Two different encoders, each for source and MT sentences are created. The encoded source and MT sentence are then concatenated with the encoded features, which are obtained by applying MLP on engineered features. A final layer is applied on the concatenated vectors, which predicts the quality scores.

### 4.2.2 Proposed Models

In this section, we discuss our proposed neural architectures for QE. Section 4.2.2.1 describes two proposed RNN-based models: \textit{RNN} and \textit{RNN.summary-attention}. Section 4.2.2.2 describes the proposed CNN-based models: \textit{CNN.Siamese}, \textit{CNN.Combined} and \textit{CNN.Combined.no-features},

#### 4.2.2.1 Recurrent Neural Network (RNN) Approaches

The POSTECH architecture, described in Section 4.2.1.2, takes advantage of the pre-training of word predictor on large external parallel corpora. Since no such datasets are easily available for most language pairs in our case, we propose a simplified version of POSTECH removing the word prediction step, and simplifying the QEFV extraction.

The model takes source sentence and MT sentence as input. A bidirectional RNN encoder, is applied on the source sentence, which gives a fixed size representation, which in turn is used as the initial state for decoder. Decoder is also a bidirectional RNN, with attention over the encoder outputs for each word and predicts a QEFV for each word in MT sentence. The outputs of decoder, QEFVs, are then...
“summarised” by another bidirectional RNN, to generate a summary vector for the sentence pair. This summary vector is then passed to a regression layer, which outputs the predicted quality score. The predicted quality score is compared with the actual quality scores under the L2 loss function for training the network using back-propagation. Figure 4.2 shows the architecture of the RNN model.

![Figure 4.2: Architecture of the RNN model](image)

Figure 4.2: Architecture of the RNN model

We also propose a variation of this model, called RNN.summary-attention, in which the summary vectors are created using attention mechanism over bidirectional RNN outputs. The QEFVs obtained from decoder are passed to a bidirectional RNN, the outputs of which are then passed to a word attention mechanism, similar to Yang et al. [47], to get a fixed length summary vector. Attention allows the model to give more importance to certain words in the context while ignoring the others, effectively learning the focus points to better predict the quality score. Figure 4.3 shows the architecture of the RNN.summary-attention model.

![Figure 4.3: Architecture of the RNN.summary-attention model](image)

Figure 4.3: Architecture of the RNN.summary-attention model

4.2.2.2 Convolutional Neural Network (CNN) Approaches

In the basic CNN model, we encode both the source and MT sentence, using CNN-based sentence encoders, similar to one proposed by Kim [24] for the text classification task. The encoder takes a sentence as a list of word embeddings and applies multiple convolution filters with varying window
sizes and applies max-over-time pooling [14] operations for each filter, output of which is then passed to a dense layer, to obtain a sentence vector.

We create two independent encoders (weights are not shared), each for source and target language sentences. The source and MT sentences are encoded using encoder for their respective languages. Finally we take cosine similarity of the two encoded sentence vectors to obtain the quality score. We call this model \textit{CNN.Siamese}. Figure 4.4 shows the architecture of this model.

We also propose an extension of \textit{CNN.Siamese} model in which the model computes the quality scores in two different ways using the same encoded sentences. One path computes the cosine similarity between the two encoded sentences. The other path concatenates the sentence encodings, optionally along with feature embeddings, and applies a fully connected layer to produce quality scores, similar to \textit{SHEF/CNN-C+F} model described in Section 4.2.1.3. The final quality score is computed by averaging the two quality scores given by different paths. The architecture of this model is shown in Figure 4.5. We include two variations, with and without engineered features in our experiments, called \textit{CNN.Combined} and \textit{CNN.Combined.no-features} respectively.

For each CNN based model, we tried two initializations for word embeddings: 1) Random 2) Using the pre-trained models published by FastText\footnote{https://fasttext.cc/docs/en/pretrained-vectors.html} [3], which are trained on Wikipedia\footnote{https://www.wikipedia.org/} for corresponding languages. The experiments, which use the FastText embeddings are denoted by +\textit{fastText} suffix.
4.3 Experimental Settings

The code used for experiments has been made publicly available at https://github.com/nisargjhaveri/tqe.

SVR.baseline model is trained using scikit-learn library [39]. Keras [12], with Theano [43] is used to implement all the neural network models, including the baselines.

Development set was used for parameter tuning for SVR.baseline for each dataset. For neural models, development data was used as validation data while training models, to early stop the training to prevent overfitting.

GRU cells [11], with 500 hidden units, are used in RNNs in all the neural network models. Sentences are clipped to length of 100 words and padded with masking. Vocabulary size is limited to 40,000 words for all the experiments. Word embedding size is set to 300.

For all proposed CNN based models, 200 filters of sizes 3, 4 and 5 each were used in the sentence encoders. Sentence vector size was set to 500. The dense layers sizes were set to 50 in CNN.Combined model.

The parameters were changed when running CNN.Combined model for news.gu as the above-mentioned parameters yielded exceptionally bad results. We analyse this result also in Section 4.4. Only for news.gu dataset for the two variants of CNN.Combined model, both the number of filters and the sentence vector sizes were set to 100 and the dense layer sizes were set to 5.
4.4 Evaluation and Results

Two types of evaluation are performed for all experiments: 1) Using Pearson’s correlation coefficient between the predicted quality scores and the actual quality scores, to evaluate scoring. 2) Using Spearman’s correlation coefficient to evaluate the ranking of sentences according to quality.

We also report statistical significance of the results considering POSTECH\_two\_step as baseline, over ten different runs.

![Table 4.2: Results for the Scoring Task, Pearson’s Correlation (\(\ast\) and \(\dagger\) indicate statistically significantly better or worse (\(p < 0.05\)) compared to POSTECH\_two\_step respectively)](image)

<table>
<thead>
<tr>
<th>Model</th>
<th>wmt17</th>
<th>news_gu</th>
<th>ilci_gu</th>
<th>ilci_hi</th>
<th>ilci_te</th>
<th>ilci_bn</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR_baseline (original features)</td>
<td>39.98</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SVR_baseline</td>
<td>38.26</td>
<td>20.12</td>
<td>44.67</td>
<td>39.58</td>
<td>44.20</td>
<td>33.65</td>
</tr>
<tr>
<td>POSTECH_multi_task</td>
<td>42.44</td>
<td>38.85</td>
<td>45.63</td>
<td>46.51</td>
<td>45.21</td>
<td>38.66</td>
</tr>
<tr>
<td>POSTECH_two_step</td>
<td>50.40</td>
<td>30.14</td>
<td>49.47</td>
<td>50.23</td>
<td>46.18</td>
<td>44.43</td>
</tr>
<tr>
<td>SHEF/CNN_C+F (original features)</td>
<td>40.34\dagger</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SHEF/CNN_C+F</td>
<td>34.22\dagger</td>
<td>29.05</td>
<td>44.32\dagger</td>
<td>39.73\dagger</td>
<td>46.60</td>
<td>34.93\dagger</td>
</tr>
<tr>
<td>RNN</td>
<td>41.71\dagger</td>
<td>37.74\ast</td>
<td>48.56</td>
<td>50.58</td>
<td>49.07\ast</td>
<td>45.14\ast</td>
</tr>
<tr>
<td>RNN_summary_attention</td>
<td>39.68\dagger</td>
<td>37.30\ast</td>
<td>48.85</td>
<td>52.59\ast</td>
<td>49.42\ast</td>
<td>44.85</td>
</tr>
<tr>
<td>CNN_Siamese</td>
<td>44.22\dagger</td>
<td>43.75\ast</td>
<td>49.29</td>
<td>52.71\ast</td>
<td>49.56\ast</td>
<td>44.83</td>
</tr>
<tr>
<td>CNN_Siamese+fastText</td>
<td>47.39\dagger</td>
<td>48.60\ast</td>
<td>51.85\ast</td>
<td>53.06\ast</td>
<td>49.69\ast</td>
<td>45.40</td>
</tr>
<tr>
<td>CNN_Combined.no-features</td>
<td>45.83\dagger</td>
<td>43.43\ast</td>
<td>48.88</td>
<td>52.01\ast</td>
<td>49.31\ast</td>
<td>44.68</td>
</tr>
<tr>
<td>CNN_Combined.no-features+fastText</td>
<td>48.14\dagger</td>
<td>49.06\ast</td>
<td>52.12\ast</td>
<td>53.17\ast</td>
<td>49.35\ast</td>
<td>45.00</td>
</tr>
<tr>
<td>CNN_Combined</td>
<td>46.98\dagger</td>
<td>41.51\ast</td>
<td>52.46\ast</td>
<td>53.00\ast</td>
<td>51.14\ast</td>
<td>46.62\ast</td>
</tr>
<tr>
<td>CNN_Combined+fastText</td>
<td>48.96\dagger</td>
<td>46.11\ast</td>
<td>52.71\ast</td>
<td>53.51\ast</td>
<td>50.06\ast</td>
<td>46.08\ast</td>
</tr>
</tbody>
</table>

Table 4.2: Results for the Scoring Task, Pearson’s Correlation (\(\ast\) and \(\dagger\) indicate statistically significantly better or worse (\(p < 0.05\)) compared to POSTECH\_two\_step respectively)

<table>
<thead>
<tr>
<th>Model</th>
<th>wmt17</th>
<th>news_gu</th>
<th>ilci_gu</th>
<th>ilci_hi</th>
<th>ilci_te</th>
<th>ilci_bn</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR_baseline (original features)</td>
<td>43.16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SVR_baseline</td>
<td>40.65</td>
<td>7.06</td>
<td>42.15</td>
<td>38.44</td>
<td>41.20</td>
<td>31.62</td>
</tr>
<tr>
<td>POSTECH_multi_task</td>
<td>44.52</td>
<td>22.46</td>
<td>43.15</td>
<td>44.43</td>
<td>42.03</td>
<td>35.69</td>
</tr>
<tr>
<td>POSTECH_two_step</td>
<td>52.06</td>
<td>19.61</td>
<td>46.85</td>
<td>48.23</td>
<td>42.83</td>
<td>40.94</td>
</tr>
<tr>
<td>SHEF/CNN_C+F (original features)</td>
<td>43.37\dagger</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SHEF/CNN_C+F</td>
<td>37.98\dagger</td>
<td>14.89\dagger</td>
<td>42.97\dagger</td>
<td>39.09\dagger</td>
<td>44.39*</td>
<td>32.61\dagger</td>
</tr>
<tr>
<td>RNN</td>
<td>43.42\dagger</td>
<td>27.42\ast</td>
<td>46.00</td>
<td>48.77</td>
<td>46.33\ast</td>
<td>42.11\ast</td>
</tr>
<tr>
<td>RNN_summary_attention</td>
<td>41.74\dagger</td>
<td>23.21</td>
<td>46.07</td>
<td>50.48\ast</td>
<td>46.34\ast</td>
<td>41.90\ast</td>
</tr>
<tr>
<td>CNN_Siamese</td>
<td>46.20\dagger</td>
<td>31.98\ast</td>
<td>46.48</td>
<td>51.16\ast</td>
<td>46.50\ast</td>
<td>41.43</td>
</tr>
<tr>
<td>CNN_Siamese+fastText</td>
<td>49.49\dagger</td>
<td>41.87\ast</td>
<td>48.34\ast</td>
<td>51.67\ast</td>
<td>45.13\ast</td>
<td>41.27</td>
</tr>
<tr>
<td>CNN_Combined.no-features</td>
<td>47.90\dagger</td>
<td>29.81\ast</td>
<td>46.03</td>
<td>50.37\ast</td>
<td>45.77\ast</td>
<td>41.23</td>
</tr>
<tr>
<td>CNN_Combined.no-features+fastText</td>
<td>50.10\dagger</td>
<td>41.13\ast</td>
<td>49.08\ast</td>
<td>51.78\ast</td>
<td>45.13\ast</td>
<td>40.88</td>
</tr>
<tr>
<td>CNN_Combined</td>
<td>48.79\dagger</td>
<td>30.70\ast</td>
<td>50.21\ast</td>
<td>51.32\ast</td>
<td>47.58\ast</td>
<td>44.19\ast</td>
</tr>
<tr>
<td>CNN_Combined+fastText</td>
<td>51.06</td>
<td>38.20\ast</td>
<td>49.77\ast</td>
<td>52.28\ast</td>
<td>45.90\ast</td>
<td>42.39\ast</td>
</tr>
</tbody>
</table>

Table 4.3: Results for the Ranking Task, Spearman’s Correlation (\(\ast\) and \(\dagger\) indicate statistically significantly better or worse (\(p < 0.05\)) compared to POSTECH\_two\_step respectively)
Table 4.2 shows comparison of different models for the scoring task using Pearson's correlation. Table 4.3 shows comparison of different models for the ranking task using Spearman's correlation.

We find that POSTECH\_two\_step model works best for WMT17 en–de dataset for both the tasks, but fails to give best results for any other dataset, in the low-resource settings we explore. We also find that the proposed CNN-based models generally work better for Indian language datasets. The better performance of CNN-based models over RNN-based models for Indian languages might be because of the free word order property of Indian languages. CNN does not directly rely on entire sequence and order of words, rather it picks best phrases depending on filter sizes from the sentence without explicitly looking at the order.

Our final model CNN\_Combined, with or without the use of FastText embeddings works best for four out of five Indian language datasets for the scoring task. For news\_gu dataset, our combined CNN model, without engineered features, CNN\_Combined.no-features+fastText, gives the best results. On investigating the relatively low results of the two variants of CNN\_Combined model on news\_gu, we found that due to some engineered features and the relatively small size of train set, the combined CNN model with features was rapidly overfitting. The similar model without engineered features, CNN\_Combined.no-features works as expected and yields the best results on news\_gu.

Similarly, for the ranking task, the two variants of CNN\_Combined model outperform all the models for four out five datasets. For the remaining Indian language dataset, news\_gu, CNN\_Siamese+fastText model yields the best result.

We also notice that using fastText embeddings in CNN based models generally works better compared to using random embeddings. However, in some cases, especially for Telugu and Bengali datasets, random initialization of embeddings performs better.

Our word-level CNN encoder based Siamese architecture, CNN\_Siamese model outperforms the SHEF/CNN-C+F model, which is a character based deep CNN model, combined with engineered features. We also show that combining the Siamese architecture with MLP based architecture in SHEF/CNN-C+F, CNN\_Combined model, further improves the results.

The RNN based models work comparably or better for all Indian language datasets, but are much simpler and have much lower number of trainable parameters compared to POSTECH models. However, the difference between the two RNN based models, RNN and RNN\_summary-attention, across datasets is inconclusive.

In Table 4.4, we show some chosen examples of scores predicted by our proposed system, CNN\_Combined+fastText and the baseline (POSTECH\_two\_step) system, along with source, MT and reference sentences and actual quality scores. Note that across examples with low to high quality scores, our method can accurately predict the quality score much better than the baseline.
### Table 4.4: Example of output by baseline (POSTECH.two-step), compared with our proposed model (CNN.Combined+fastText), across all datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source sentence</th>
<th>MT sentence</th>
<th>Correct sentence</th>
<th>Baseline</th>
<th>Our model</th>
<th>Actual TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>news.gu</td>
<td>Every year, loud sound from firecrackers causes stress, terror and even death in strays and birds.</td>
<td>દર વષ, ફટાકડાથી ઘોડથાપાણા અંતર તણાવ, આતંક અને પથ્થરીઓની મૃત્યુ પાણ આવે છે.</td>
<td>દર વષ, ફટાકડાથી ઘોડથાપાણા અંતર તણાવ, આતંક અને પથ્થરીઓની મૃત્યુ પાણ આવે છે.</td>
<td>0.03</td>
<td>0.31</td>
<td>0.33</td>
</tr>
<tr>
<td>ilci.gu</td>
<td>The total distance of this route is 163 kilometers from Pathankot to Jogindernagar.</td>
<td>આ માગણી હું અંતર પઠાનકોટ થી ઍજગદરનગર સુધી કુમાર હોય છે 163 કિમી.</td>
<td>પઠાનકોટ થી ઍજગદરનગર સુધી કુમાર હોય છે 163 કિમી.</td>
<td>0.31</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td>ilci.te</td>
<td>People of Hindustan, Pakistan, Bangladesh, Egypt do business in Manama Souk.</td>
<td>(gcaחm) , 枨จળ , e comeback . , ڈાંગામના સુખામમાં SBH.x .</td>
<td>ગાંધામ ગ્રામેર અટિટ્ટિ કાઢ કાઢ કાઢામા રાખો.</td>
<td>0.89</td>
<td>0.53</td>
<td>0.50</td>
</tr>
<tr>
<td>ilci.bn</td>
<td>There are eight-ten houses of wood in Gejam village.</td>
<td>গাজাম ·ােমর আটিট কােঠর কাঠােমা রেয়েছ.</td>
<td>গাজাম বছরের আট দশটি কাঠ বাড়ি আছে।</td>
<td>0.58</td>
<td>0.85</td>
<td>0.89</td>
</tr>
</tbody>
</table>

### 4.5 Conclusions

In this chapter, we study the effectiveness of different neural network architectures for QE for Indian languages. We also introduce multiple datasets for the task, which can be used as benchmark for future work in the area. We observe that our proposed CNN.Combined model beats the state-of-the-art methods by a significant margin. These models are later used in Chapter 5 to experiment with the use of QE in CLS.
Chapter 5

Cross-lingual Summarization

In this chapter, we describe our experiments on cross-lingual summarization using different methods and techniques. We describe some existing methods for CLS. We also describe an extension of an existing mono-lingual summarization method for CLS along with different variations.

In this chapter, we also experiment with integration of QE, described in Chapter 4, to improve CLS. We show that QE can improve the quality of generated summaries in target language, in terms of readability. It may also help in improving the content of the summary, evaluated by ROUGE score (Section 5.3.1), by eliminating wrongly translated or bad quality sentences.

Additionally, we explore the usefulness of Sentence Simplification for CLS. Sentence simplification aims to map a complex sentence to one or more simple alternative sentences with similar meaning. Major type of simplification approaches are a) Sentence compression, with drops some parts of the sentence, b) Sentence splitting, which splits a complex sentence in several simpler sentences, c) Paraphrasing and word substitution. Compressive summarization is widely used in the area of summarization in general, to remove unnecessary parts of sentences included in the summary. For CLS, sentence compression or any other form of simplification has an additional benefit that it helps in obtaining better translation [18, 41]. We propose the use of sentence simplification to improve translation in cross-lingual summary generation.

The rest of the chapter is organized as follows: Section 5.1 describes different datasets used for our experiments while Section 5.2 explains different models used. Section 5.3 lists and describes different kinds of evaluations we perform for our experiemnts. Section 5.4 lists specifics of our experiments, results and analysis. At the end, we conclude the chapter in Section 5.5.

5.1 Datasets

This section describes two datasets we used for experiments with different CLS methods. These datasets have been derived from different sources with different specifications about the task, details of which are provided in their respective subsections.
5.1.1 DUC 2004 Gujarati

DUC 2004 Gujarati dataset is derived from the DUC 2004 dataset for the use in cross-lingual summarization evaluation. The details about the preparation of this dataset is described in Section 3.7.2. The dataset contains source documents in English from the original DUC 2004 corpus and summaries for all document sets in Gujarati.

The summarization task defined by DUC 2004 specifies that the generated summaries should contain at most 665 characters. We keep the same limit in our cross-lingual variant also and do not consider the language variations in favour of consistency.

5.1.2 TAC 2011 MultiLing Pilot Dataset

We propose the use of a dataset published as part of TAC 2011 language-independent multi-document summarization task for CLS evaluation.

MultiLing Pilot 2011 dataset\(^1\) was published as part of TAC 2011 Summarization Track, for language-independent or multi-lingual summarization task [17]. The data was prepared by sentence-by-sentence translation of document sets from English to six other languages: Arabic, Czech, French, Greek, Hebrew and Hindi. The model summaries were created by fluent speakers (generally, native speakers) of each corresponding language. As a result the dataset contains parallel documents in all seven languages and their respective summaries. The dataset contains ten document sets in seven languages, and three summaries for each document set.

We use the dataset in a cross-lingual setting where summaries are generated in the target language for the source documents in a different source language. The target language summaries are then evaluated using the summaries available in the dataset. Here, the source and target languages could be any two different languages selected from the set of seven languages for which the data is available.

We use this dataset to show comparative results for English to Hindi summarization task. In all our experiments for this dataset, we limit the generated summary to contain at most 250 words, as defined by the MultiLing Pilot 2011 task [17].

5.2 Methods

In this section we describe different methods we use for our experiments. Section 5.2.1 and Section 5.2.2 describes the two methods, SimFusion and CoRank, proposed for CLS by Wan [44] respectively. Section 5.2.3 describes the method, which we refer to as LinBilmes, described by Lin and Bilmes [28] and its use for CLS.

We experiment with different parameters and settings of all three methods. Details of the experiments and results are described in Section 5.4.

\(^1\)http://users.iit.demokritos.gr/~ggianna/TAC2011/MultiLing2011.html
5.2.1 SimFusion

This method, proposed by Wan [44], uses English-side information along with Chinese-side information for Chinese sentence ranking in a graph-based framework. All the sentences in the source documents are first translated automatically from the source language (English in their case) to the target language (Chinese in their case). The sentence similarities are computed for all the sentence pairs in both the languages and are fused to get final similarity score for a sentence pair in target language. The intuition behind the method is that the information from a single side is not very reliable in cross-lingual summarization. Finally, using the similarity scores for target sentence pairs, a graph is created where each node is a sentence. PageRank-like computations on this graph are then used to extract summary sentences.

5.2.2 CoRank

Similar to SimFusion method, described in Section 5.2.1, Wan [44] also proposed this method to leverage information from both languages. It assumes that a sentence would be salient if it is heavily linked with other salient sentences in the same language as well as if it is heavily linked with salient sentences from other language. In this method, the source sentences and the automatically translated sentences in target language are ranked simultaneously using a unified graph-based algorithm.

5.2.3 LinBilmes

We adapt the summarization framework and sub-modular functions described by Lin and Bilmes [27, 28] for CLS by adding different options for translation and simplification at different steps. Lin and Bilmes [28] models the summary quality as

\[ F(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S) \]  

In Equation 5.1, \( \mathcal{L}(S) \) and \( \mathcal{R}(S) \) are Coverage function and Diversity Reward function respectively. Coverage function \( \mathcal{L}(S) \) measures the coverage of summary set \( S \) to the document, while the diversity reward function \( \mathcal{R}(S) \) positively rewards diversity. \( \lambda \geq 0 \) is a trade-off coefficient.

The summary is obtained by greedily optimizing the summary quality function. In addition to this, we define a new objective function to take translation quality estimation into account while summarizing. We also extend the summary quality \( F(S) \) to include more than two objectives.

QE Objective Function

We propose a new monotone sub-modular objective function to incorporate QE scores while summarizing to get a better cross-lingual summaries. All the source sentences are automatically translated first to the target language and quality scores for each sentence pair is computed using one of the models
described in Chapter 4. The error scores given by the selected QE model is then converted to quality score using Equation 5.2. Here, \( q(s_i) \) is quality score for sentence \( s_i \) and \( Score_{HTER}(s_i) \) is HTER score predicted by the QE model for the given sentence.

\[
q(s_i) = (1 - Score_{HTER}(s_i))^4
\]  
\hfill (5.2)

The power, 4, in the Equation 5.2 is experimental with the intuition being high error should be penalized more than low error in translation quality.

The QE objective \( Q \) for a summary set \( S \), is simply defined as sum of quality scores for each sentence in the set.

\[
Q(S) = \sum_{s_i \in S} q(s_i)
\]  
\hfill (5.3)

Here, \( S \) is the summary set, i.e. a set containing a subset of sentences from the document. Clearly, the QE objective \( Q \) is monotone sub-modular if \( q(s_i) \geq 0, \forall s_i \).

**Redefined Summary Quality**

We extend Equation 5.1 to accommodate more than two objective functions. We redefine the summary quality as

\[
\mathcal{F}(S) = \lambda_{\text{coverage}} \mathcal{L}(S) + \lambda_{\text{diversity}} \mathcal{R}(S) + \lambda_{\text{quality}} Q(S)
\]  
\hfill (5.4)

The three coefficients, \( \lambda_{\text{coverage}} \), \( \lambda_{\text{diversity}} \) and \( \lambda_{\text{quality}} \) are are the trade-off coefficients or the weights of respective objective functions.

5.3 Evaluation Metrics

5.3.1 ROUGE Score

*Recall-Oriented Understudy for Gisting Evaluation*, commonly known as ROUGE score is an important and widely used metric for automated evaluation of summaries [26].

We have multiple reference summaries for each document set in each of the datasets. We report ROUGE-1 and ROUGE-2 scores for the generated summaries by comparing them to the reference summaries available.

The ROUGE scores are calculated using a Python implementation mentioned in Section 6.1.2.

5.3.2 Perplexity

We also report perplexity of the generated summaries in target language. The perplexity was calculated using the bi-gram language models trained on FIRE 2011 monolingual datasets\(^2\) for Gujarati and Hindi using kenlm\(^3\).

\(^2\)http://fire.irs.res.in/fire/static/data

\(^3\)https://github.com/kpu/kenlm
We assume that better perplexity on a language model trained on large corpora indicates better readability or naturalness of the sentences in the generated summary. The better readability refers to better perplexity in further text.

5.4 Experiments and Results

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>Perplexity</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>F-score</td>
<td>Recall</td>
<td>F-score</td>
<td>including OOVs</td>
</tr>
<tr>
<td>CoRank</td>
<td>21.52</td>
<td>21.17</td>
<td>3.92</td>
<td>3.87</td>
<td>2579.31</td>
</tr>
<tr>
<td>CoRank.earlySimplify</td>
<td>21.35</td>
<td>20.96</td>
<td>3.77</td>
<td>3.72</td>
<td>2426.25</td>
</tr>
<tr>
<td>SimFusion</td>
<td>22.32</td>
<td>22.08</td>
<td>4.06</td>
<td>4.03</td>
<td>3125.34</td>
</tr>
<tr>
<td>SimFusion.earlySimplify</td>
<td>22.71</td>
<td>22.45</td>
<td><strong>4.15</strong></td>
<td><strong>4.13</strong></td>
<td>3067.00</td>
</tr>
<tr>
<td>LinBilmes</td>
<td>21.17</td>
<td>21.41</td>
<td>3.58</td>
<td>3.62</td>
<td>5407.18</td>
</tr>
<tr>
<td>LinBilmes.qe</td>
<td>21.49</td>
<td>21.70</td>
<td>3.69</td>
<td>3.73</td>
<td>4265.37</td>
</tr>
<tr>
<td>LinBilmes.earlySimplify</td>
<td>20.67</td>
<td>20.74</td>
<td>3.37</td>
<td>3.39</td>
<td>4421.74</td>
</tr>
<tr>
<td>LinBilmes.earlyTranslate</td>
<td>22.11</td>
<td>21.32</td>
<td>3.80</td>
<td>3.69</td>
<td>1522.23</td>
</tr>
</tbody>
</table>

Table 5.1: Evaluation of different CLS methods on DUC 2004 Gujarati dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>Perplexity</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>F-score</td>
<td>Recall</td>
<td>F-score</td>
<td>including OOVs</td>
</tr>
<tr>
<td>CoRank</td>
<td>36.52</td>
<td>36.30</td>
<td>7.04</td>
<td>7.07</td>
<td>564.09</td>
</tr>
<tr>
<td>CoRank.earlySimplify</td>
<td>37.72</td>
<td>37.64</td>
<td>7.48</td>
<td>7.57</td>
<td>523.14</td>
</tr>
<tr>
<td>SimFusion</td>
<td>37.95</td>
<td>37.67</td>
<td>6.80</td>
<td>6.83</td>
<td>524.72</td>
</tr>
<tr>
<td>SimFusion.earlySimplify</td>
<td>38.40</td>
<td><strong>38.28</strong></td>
<td><strong>7.55</strong></td>
<td><strong>7.59</strong></td>
<td>492.03</td>
</tr>
<tr>
<td>LinBilmes</td>
<td>37.06</td>
<td>36.88</td>
<td>6.64</td>
<td>6.69</td>
<td>661.00</td>
</tr>
<tr>
<td>LinBilmes.qe</td>
<td>37.50</td>
<td>37.31</td>
<td>6.64</td>
<td>6.69</td>
<td>612.19</td>
</tr>
<tr>
<td>LinBilmes.earlySimplify</td>
<td>36.48</td>
<td>36.48</td>
<td>6.90</td>
<td>7.02</td>
<td>539.56</td>
</tr>
<tr>
<td>LinBilmes.earlyTranslate</td>
<td>36.25</td>
<td>36.29</td>
<td>5.68</td>
<td>5.77</td>
<td><strong>361.06</strong></td>
</tr>
</tbody>
</table>

Table 5.2: Evaluation of different CLS methods on MultiLing Pilot 2011 dataset for Hindi

We compare performance of different methods for English to Gujarati summarization using the proposed DUC 2004 Gujarati dataset described in Section 5.1.1 and for English to Hindi summarization using the MultiLing Pilot 2011 dataset described in Section 5.1.2.

Table 5.1 and Table 5.2 show the ROUGE scores and the perplexity of the summaries for DUC 2004 Gujarati and MultiLing Pilot 2011 dataset respectively for different variations of the methods CoRank, SimFusion and LinBilmes.

We note that, SimFusion works best for both of our datasets, contrary to the trend noted by Wan [44], which proposed and compared SimFusion and CoRank for English to Chinese summarization.
We also experiment with the other five languages which are a part of the MultiLing 2011 Dataset. Table 5.3 shows results obtained using various methods in terms of ROUGE scores. Note that for most datasets, SimFusion works best, while CoRank performs better in a few cases.

<table>
<thead>
<tr>
<th>Language</th>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Recall</td>
<td>F-score</td>
</tr>
<tr>
<td>Arabic</td>
<td>CoRank</td>
<td>29.03</td>
<td>29.75</td>
</tr>
<tr>
<td></td>
<td>SimFusion</td>
<td>29.81</td>
<td>30.49</td>
</tr>
<tr>
<td></td>
<td>LinBilmes</td>
<td>27.12</td>
<td>27.64</td>
</tr>
<tr>
<td>Czech</td>
<td>CoRank</td>
<td>31.55</td>
<td>31.49</td>
</tr>
<tr>
<td></td>
<td>SimFusion</td>
<td>33.24</td>
<td>33.16</td>
</tr>
<tr>
<td></td>
<td>LinBilmes</td>
<td>30.27</td>
<td>30.29</td>
</tr>
<tr>
<td>French</td>
<td>CoRank</td>
<td>47.75</td>
<td>46.84</td>
</tr>
<tr>
<td></td>
<td>SimFusion</td>
<td>48.88</td>
<td>48.00</td>
</tr>
<tr>
<td></td>
<td>LinBilmes</td>
<td>48.23</td>
<td>47.36</td>
</tr>
<tr>
<td>Greek</td>
<td>CoRank</td>
<td>34.68</td>
<td>34.71</td>
</tr>
<tr>
<td></td>
<td>SimFusion</td>
<td>36.23</td>
<td>36.16</td>
</tr>
<tr>
<td></td>
<td>LinBilmes</td>
<td>34.61</td>
<td>34.59</td>
</tr>
<tr>
<td>Hebrew</td>
<td>CoRank</td>
<td>21.87</td>
<td>22.06</td>
</tr>
<tr>
<td></td>
<td>SimFusion</td>
<td>22.69</td>
<td>22.98</td>
</tr>
<tr>
<td></td>
<td>LinBilmes</td>
<td>20.65</td>
<td>21.00</td>
</tr>
</tbody>
</table>

Table 5.3: Evaluation of different CLS methods on MultiLing Pilot 2011 dataset for different languages

In Section 5.4.1 we discuss our experiments and effect of QE in detail. Further, in Section 5.4.2 and Section 5.4.3 we discuss different configurations and its effect on summarization, specifically the effect of sentence simplification and early translation.

### 5.4.1 Quality Estimation

To evaluate the effectiveness of QE on CLS, we run different experiments described in this subsection.

We train two QE models, one each for English to Gujarati QE and English to Hindi QE using the CNN-based Siamese model described in Section 4.2.2.2. The English to Gujarati model was trained on the news.gu dataset (Section 4.1.2) and the English to Hindi model was trained on ilci.hi dataset (Section 4.1.3). We use these models to get quality scores for DUC 2004 Gujarati dataset and MultiLing Pilot 2011 Dataset respectively.

We run experiment with different values of $\lambda_{\text{quality}}$. The values of $\lambda_{\text{coverage}}$ and $\lambda_{\text{diversity}}$ were fixed to 1.0 and 6.0 respectively as these parameters were shown to work best by Lin and Bilmes [28].

We varied the value of $\lambda_{\text{quality}}$ from 1 to 10 and plot the resulting ROUGE-1 score, ROUGE-2 score and Perplexity on graphs. Figure 5.1 and Figure 5.2 shows these graphs for both the datasets. The best results for each dataset are also present in Table 5.1 and Table 5.2 with the model name LinBilmes.qe.
For DUC 2004 Gujarati dataset (Figure 5.1), we note that as the weight of the QE objective increases, ROUGE score first increases and then decreases. We get the best ROUGE score when $\lambda_{quality}$ is set to 3. We also note that as $\lambda_{quality}$ increases, the continuously perplexity decreases, as with high $\lambda_{quality}$, the optimizer will pick high quality sentences, but ignore the coverage and diversity aspects, leading to low ROUGE score.

On the other hand, for MultiLing Pilot 2011 Dataset (Figure 5.2), the effect of QE is inconclusive. An explanation for this may be that the train and test data had different domains for QE models. The QE model for English to Hindi was trained on ilci.hi, which contains text from health and tourism domains. While the test sentences, sentences from MultiLing Pilot 2011 Dataset, are from news domain. This might have affected the performance of the QE model and as a result produced inconclusive results for CLS.
The dataset used to train QE model for English to Gujarati was `news.gu`, which contains sentences from news domain, similar to the DUC 2004 Gujarati dataset.

5.4.2 Sentence Simplification

To see the effect of sentence simplification, we perform a simple experiment, in which we perform early-simplification, i.e. the source documents are first simplified and then the respective methods are applied to generate the summaries. We used the Neural Text Simplification system\(^4\) published by Nisioi et al. [34] to simplify the source sentences.

\(^4\)https://github.com/senisioi/NeuralTextSimplification
We note that using sentence simplification greatly helps in terms of readability. Table 5.1 and Table 5.2 show that for both the datasets, applying early-simplification leads to better perplexity for all the three algorithms.

### 5.4.3 Early-translation

Additionally, we also run LinBilmes with early-translation, in which the documents are translated first then the summarizer is run on the translated documents. The early-translation configuration is not included for the other two methods as they already use information from both the source and automatically translated sentences.

We note that early-translate leads to significantly better perplexity with minor decrease in ROUGE score.

### 5.5 Conclusions

We described different datasets and experiments on cross-lingual summarization in this chapter. We explored the usage of QE and sentence simplification in different settings along with experiments with several existing methods for CLS.

In our experiments, QE shows promising results on DUC 2004 Gujarati dataset, indicating that not only QE can help improve the readability in CLS, it can also improve the content of the summaries.

From our preliminary experiments with sentence simplification, we also show that the usage of sentence simplification can help significantly in improving the readability of the cross-lingual summary.

We make all the methods and variations described in this chapter available for use as a tool-kit. The tool-kit, clstk, is described in Chapter 6.
Chapter 6

clstk

Concluding our work on cross-lingual summarization, we publish a tool-kit for easy experimentation with CLS. In this chapter, we describe the tool-kit, clstk. The tool-kit is intended for the use of both, developers and researchers working on cross-lingual summarization. End-users wanting to use cross-lingual summarization in real-world applications can also benefit from the tool-kit. The goals of the tool-kit are as follows.

- Help researchers quickly implement new models as well as compare with existing models for cross-lingual summarization.
- Provide a unified platform and API for researchers to publish their models.
- Make different algorithms and models for cross-lingual summarization accessible for use in real-world end-user applications.

The proposed tool-kit contains a collection of several cross-lingual summarization methods which can be used as baselines in future work on CLS as well as bootstrap code to develop new methods for the problem. Additionally, the tool-kit contains a summary evaluation module, which can be helpful in evaluating and experimenting with cross-lingual summarization easily.

All the experiments described in Chapter 5 can be performed using the tool-kit without any extra effort. This shows the usefulness of the tool-kit.

The tool-kit, clstk, is freely available at https://github.com/nisargjhaveri/clstk. The complete documentation for clstk is available at https://clstk.readthedocs.io. clstk is primarily written in Python and published as free and open source software under the MIT licence.

The rest of the chapter is organized as follows: Section 6.1 describes different components and modules of the system. Section 6.2 describes companion tools and our roadmap for the tool-kit. Finally, we conclude with a brief summary in Section 6.3.

1https://github.com/nisargjhaveri/clstk/blob/master/LICENSE
6.1 Components

In this section, we describe the different logical components of the system, which widely correspond to the code structure of the tool-kit. Section 6.1.1 describes the core part of the system which glues different components together. Section 6.1.2 describes the component responsible for evaluation of cross-lingual summaries. Section 6.1.3, Section 6.1.4 and Section 6.1.5 describe automatic translation, sentence simplification and translation quality estimation modules respectively. At the end, Section 6.1.6 lists different summarization algorithms included in the system.

Figure 6.1 show high-level interactions between different components described in this section.

6.1.1 The Core

The core contains the bootstrap code for summarization needs. The core provides:

- A common standard structure for documents and summaries to ensure interoperability between different components.
- Utilities for loading document sets into the common structure.
- Common utilities on document sets, documents and sentences, for example sentence splitting, tokenization, etc.

Additionally, the core also provides command-line access to different summarizers included in the tool-kit.

6.1.2 Evaluation

This module contains utilities for evaluation of generated summaries.

ROUGE score is an important and widely used metric for automated evaluation of summaries. The tool-kit currently contains a Python implementation of the ROUGE score.
Additionally, the tool-kit integrates with the original ROUGE package [26]. We also publish and recommend a Unicode-aware version of the original ROUGE-1.5.5 package\(^2\) for evaluation of cross-lingual summaries as the target language may require use of Unicode characters.

A command-line script for evaluation is also provided which runs a selected summarization method with given parameters for all document sets in a directory and reports the ROUGE score given the reference summaries. This is quite useful when trying out multiple methods to get comparative scores.

### 6.1.3 Translation

Translation is an important module when working with CLS. The tool-kit includes a translation module to easily incorporate machine translation in the summarization process. The module is designed keeping in mind that different methods may want to use the translation at different stages and in different contexts. For example, the module allows for translating documents before summarization or after summarization or selectively translating sentences while summarizing.

There are a large number of machine translation (MT) systems available both commercially and non-commercially. We acknowledge the need of use of particular MT systems based on various reasons. Hence, the module is designed to allow easy integration with various third-party MT tools and APIs. Currently, the tool contains integration with the Google Translate API\(^3\).

### 6.1.4 Sentence Simplification

Compressive summarization is widely used in the area of summarization in general. We also show that sentence simplification can be helpful in cross-lingual summarization as well in Section 5.4.2.

We include a sentence simplification module in the tool. Similar to the translation module, the sentence simplification module in the tool-kit also allows for integration with third-party tools and APIs.

Currently, the tool contains integration with the Neural Text Simplification system\(^4\) published by Nisioi et al. [34].

### 6.1.5 Translation Quality Estimation

The use of Translation Quality as a measure while extracting sentences for CLS has been explored in the past [9, 45]. In Chapter 5, we also experimented with QE and argued about the usefulness of QE for CLS. We include a Translation Quality Estimation module in the tool which can be used to experiment with QE scores in different contexts and different stages while summarizing.

Currently, the tool integrates with the QE system\(^5\) we published as part of our work described in Chapter 4, which contains implementation of several state-of-the-art models for QE.

---

\(^2\)https://github.com/nisargjhaveri/ROUGE-1.5.5-unicode
\(^3\)https://translate.google.com/
\(^4\)https://github.com/senisioi/NeuralTextSimplification
\(^5\)https://github.com/nisargjhaveri/tqe
6.1.6 Summarizers

One of the major goals of the tool-kit is to make available multiple approaches and methods for CLS. The tool currently contains implementations of two models for CLS by Wan [44]: SimFusion and CoRank. Additionally, the tool contains an implementation of the popular sub-modular function maximization based summarization algorithm [28], and adapts it for use in the cross-lingual setting. All three methods are described in detail in Section 5.2.

The three summarizers already implemented act as examples of the intended use of the system. Other summarizers can be easily integrated into the tool.

6.2 Companion Tools and Roadmap

The CLS workbench described in Chapter 3 can be used as a companion tool with this system in the development of new methods for the task. clstk can be plugged into the workbench, to help rapidly generate CLS data for new language pairs, and later the data can be used to improve or implement new methods in clstk as also proposed in Chapter 3.

In future, we plan to implement more existing methods in the tool and encourage community to contribute and use the tool-kit. An easy web-interface can be provided for the implemented methods to enable people to play with different CLS methods easily.

6.3 Conclusions and Discussion

We contribute the only available tool-kit for CLS, clstk, containing different existing methods as well as bootstrap code to easily develop and experiment with new methods for the same. We explain the tool-kit in detail along with its design and different components included in this chapter.

As mentioned earlier, clstk can run all the experiments and variations described in Chapter 5 by setting simple command-line options. The tool-kit can also be used to experiment with different methods for CLS with different settings or to easily tweak existing methods. For example, the modular design of the tool-kit makes it easy to experiment with different MT systems without changing anything else in the pipeline.

With the use of the bootstrap code and modules included in the workbench, one can easily implement new CLS methods, as while implementing a new method, one doesn’t have to worry about things like loading data or integrating an MT system. The already implemented algorithms, SimFusion and CoRank, serve as apt examples of this.
Chapter 7

Conclusions and Future Work

In this thesis, we described our efforts on cross-lingual summarization and translation quality estimation for Indian languages.

We publish a web-based workbench to rapidly generate human-edited cross-lingual summaries as well as collect data about the translation and summarization process. We used the workbench to generate two datasets, one each for English to Gujarati translation quality estimation and English to Gujarati cross-lingual summarization evaluation. These datasets can be used as benchmark datasets for future work in the area.

We used the dataset generated using the workbench, along with several other new datasets for four Indian languages, to study translation quality estimation for these languages. We proposed several novel neural network architectures for QE. We experimented with multiple state-of-the-art models, along with the proposed architectures, and showed that one of our proposed architectures, CNN.Combined works best for most Indian language datasets beating the current state-of-the-art by a significant margin.

Later, we integrated our work on QE with cross-lingual summarization. We proposed a novel objective function to account for quality of translation while ranking sentences for summarization. We experimented with several methods and different settings to compare their performance using two datasets, one each for English to Gujarati summarization and English to Hindi summarization. We show that the usage of QE and sentence simplification has potential to have significant impact in improving the cross-lingual summarization.

At the end, we described a tool-kit for CLS, clstk, which is designed to help researchers by making bootstrap code available for use to easily implement new methods. clstk also contains several existing methods for CLS. Researchers and developers can make use of the tool-kit to develop new methods for CLS or end-user applications making use of existing CLS methods.

There exist several possibilities to extend our work in the future. The workbench can be improved further by including more features to improve the usability and efficiency. The possible extensions include the integration of bi-lingual or mono-lingual dictionaries, translation memory module or quality estimation and automatic post-editing tools as described in Section 3.8.

We limited ourselves to work on low-resource setting while exploring QE by not using any external datasets or resources. We believe that the use of such resources can help improve the performance of QE
systems further. Extensive work to explore easily available or collectable resources and their efficient use for QE can be taken up. In addition to this, a study of how these methods work in a more general setting for other language pairs can be taken up to move towards achieving truly language-agnostic methods, while our work which focused on translation from English to several Indian languages.

From our preliminary experiments with QE and sentence simplification for CLS, we strongly believe that these two techniques can have significant impact on improving cross-lingual summarization. In the future, more efficient techniques for using QE and sentence simplification in CLS should be explored. In addition, better QE techniques, which work across different domains need to explored to achieve much better translation quality estimations for summarization.

Though clstk is powerful and self-contained as of now, there is a great scope for improvement in the future. As a free and open source software (FOSS), clstk will greatly benefit from the community as and when the community starts using and contributing to the tool-kit. More methods for CLS should be implemented in the tool-kit in the future keeping it up-to date with new research in the area. Integration of the tool-kit with other exiting natural language processing (NLP) and information retrieval and extraction (IRE) tool-kits should be explored for better interoperability with other tools.
Related Publications


Bibliography


