USE OF LANGUAGE TECHNOLOGY TO IMPROVE MATCHING AND RETRIEVAL IN TRANSLATION MEMORY

ROHIT GUPTA

A thesis submitted in partial fulfilment of the requirements of the University of Wolverhampton for the degree of Doctor of Philosophy

2016

This work or any part thereof has not previously been presented in any form to the University or to any other body whether for the purposes of assessment, publication or for any other purpose (unless otherwise indicated). Save for any express acknowledgements, references and/or bibliographies cited in the work, I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

Signature: . . . . . .

Date: . . . . . .

ABSTRACT

Current Translation Memory (TM) tools lack semantic knowledge while matching. Most TM tools compute similarity at the string level, which does not take into account semantic aspects in matching. Therefore, semantically similar segments, which differ on the surface form, are often not retrieved. In this thesis, we present five novel and efficient approaches to incorporate advanced semantic knowledge in translation memory matching and retrieval.

Two efficient approaches which use a paraphrase database to improve translation memory matching and retrieval are presented. Both automatic and human evaluations are conducted. The results on both evaluations show that paraphrasing improves matching and retrieval.

An approach based on manually designed features extracted using NLP systems and resources is presented, where a Support Vector Machine (SVM) regression model is trained, which calculates the similarity between two segments. The approach based on manually designed features did not retrieve better matches than simple edit-distance.

Two approaches for retrieving segments from a TM using deep learning are investigated. The first one is based on Long Short Term Memory (LSTM) networks, while the other one is based on Tree Structured Long Short Term Memory (Tree-LSTM) networks. Eight different models using different datasets

and settings are trained. The results are comparable to a baseline which uses simple edit-distance.

CONTENTS

Abstract **iii**

List of Tables **xi**

List of Figures **xv**

Acknowledgements **xvii**

1 Introduction **1**

1.1 Translation Memory 2

1.2 Translation Memory Matching and Retrieval 3

1.3 Research Questions and Main Contributions 7

1.4 Thesis Structure 8

2 Related Work **11**

2.1 Research on Improving TM Matching and Retrieval 12

2.2 Combining TM with MT 21

2.3 Machine Translation Evaluation 24

2.4 Semantic Similarity and Textual Entailment using Traditional Approaches 31

2.4.1 Sentence Similarity as a Function of Words 31

2.4.2 Edit-distance Based Approaches 36

2.4.3 Combining Various Similarity Measures 38

2.5 Deep Learning for Semantic Similarity 39

2.6 Conclusion 43

3 Improving Matching and Retrieval using Paraphrases 45

3.1 Introduction 45

3.2 Our Approach 46

3.2.1 Paraphrase Corpus 47

3.2.2 Classification of Paraphrases 49

3.2.3 Filtering 50

3.2.4 Matching Steps 52

3.2.5 Edit-distance Computation 54

3.2.6 Dynamic Programming and Greedy Approximation (DPGA) 57

3.2.7 Dynamic Programming Only (DP) 60

3.2.8 Computational Considerations 64

3.3 Evaluation on the Europarl Corpus 65

3.3.1 Corpus Used 67

3.3.2 Results of Automatic Evaluations of Our Approaches (DPGA and DP) 67

3.3.3 Human Evaluation 71

3.3.3.1 Test sets, Tool and Translators' Expertise 72

3.3.3.2 Post-editing Time (PET) and Keystrokes (KS) . 73

3.3.3.3 Subjective Evaluation with Two Options (SE2) . 76

3.3.3.4 Subjective Evaluation with Three Options (SE3) 76

3.3.3.5 Subjective Evaluation on Exact Matches Only (SEM) 76
3.3.4 Results and Analysis of Human Evaluation 77
3.3.4.1 Results: Post-editing Time (PET) and Keystrokes (KS) 77
3.3.4.2 Results: Using Post-edited References 79
3.3.4.3 Results: Subjective Evaluations 79
3.3.4.4 Results: Segment-Wise Analysis 80
3.3.4.5 Results: Subjective Evaluation on Exact Matches Only (SEM) 83
3.4 Evaluation on the DGT-TM Corpus 85
3.5 Conclusion 95
4 Advanced Semantic Matching for TM **97**
4.1 Advanced Semantic Matching for TM using a Traditional Approach 98
4.1.1 Our Approach for Semantic Textual Similarity 99
4.1.2 STS System 100
4.1.2.1 System Description 100
4.1.2.2 Features 101
4.1.2.3 Corpus used for Training 105
4.1.3 Experiments and Results 105
4.1.4 Conclusion 109
4.2 Advanced Semantic Matching for TM using Recurrent Neural Networks 109

4.2.1 Recurrent Neural Networks 111
4.2.1.1 Basic RNN 111
4.2.1.2 LSTM 113
4.2.1.3 Tree Structured LSTM 115
4.2.2 A Neural Similarity Metric for Machine Translation Evaluation 118
4.2.3 Creating a Dataset from WMT Rankings 121
4.2.4 Results on Machine Translation Evaluation 124
4.2.5 ReVal for TM Matching and Retrieval 129
4.2.6 More Experiments using LSTM and Tree-LSTM for TM Matching 133
4.2.6.1 Training Data 134
4.2.6.2 Word Vectors 134
4.2.6.3 Deep Learning Models 136
4.2.6.4 Results and Analysis 138
4.2.7 Performance of Tree-LSTM on Other Language Pairs . . . 142
4.2.8 Further Analysis of Results 144
4.2.9 A Closer Look at Segment Representation using LSTM . . 147
4.2.10 Conclusion 150
4.3 Conclusion 150
5 Conclusion **153**
5.1 Comparing Paraphrasing and LSTM 155
5.2 Answers to Research Questions 157

5.3 Future Work 159

Bibliography **161**

A Related Publications **183**

LIST OF TABLES

1.1 Some Examples From Our Training Dataset 6

3.1 An Example Describing the Paraphrase Reduction Process 50

3.2 Paraphrases Used to Create TM Lattice in Figure 3.1 and Figure 3.2 50

3.3 Edit-distance Calculation using DPGA 59

3.4 Edit-distance Calculation using DP 63

3.5 Corpus Statistics 67

3.6 Results of Automatic Evaluation: Presented using Threshold Intervals 69

3.7 Results of Automatic Evaluation: Presented using Cutoff Thresholds 70

3.8 Test Sets for Experiments PET, KS, SE2 and SE3 72

3.9 Results of Human Evaluation on Set-1 (1-14) and Set-2 (15-30) . . 78

3.10 Results using Human Targeted References 80

3.11 Results of Human Evaluation on Exact Matches 83

3.12 DGT-TM Corpus Statistics 88

3.13 Results of Automatic Evaluation (English-German): Presented using Threshold Intervals 89

3.14 Results of Automatic Evaluation (English-German): Presented using Cutoff Thresholds 90

3.15 Results of Automatic Evaluation on DGT-TM (English-French): Presented using Threshold Intervals 91
3.16 Results of Automatic Evaluation on DGT-TM (English-French): Presented using Cutoff Thresholds 92
3.17 Results of Automatic Evaluation on DGT-TM (English-Spanish): Presented using Threshold Intervals 93
3.18 Results of Automatic Evaluation on DGT-TM (English-Spanish): Presented using Cutoff Thresholds 94

4.1 Test Set Statistics 106
4.2 Results Automatic Evaluation 106
4.3 Examples from Test-2 107
4.4 Variances Computed using Algorithm 6 123
4.5 Results: System-Level Correlations on WMT-14 (cs:czech, de:German, en:English, fr:French, hi:Hindi, ru:Russian) 125
4.6 Results: Segment-Level Correlations on WMT-14 127
4.7 Filtering Statistics for English-German 131
4.8 Tree-LSTM (ReVal) Results for English-German 133
4.9 Quality of Word Vectors 136
4.10 Eight LSTM Models 138
4.11 Comparing All LSTM models on English-German 141
4.12 Results using the 'WikiTree' Model on English-German (DE), English-Spanish (ES) and English-French (FR) 143

5.1 Comparing Paraphrasing and ‘WikiTree’ on English-German (All Matches) 156
5.2 Comparing Paraphrasing and ‘WikiTree’ on English-German (Differing Matches Only) 157

LIST OF FIGURES

3.1 (i) TM Segment, (ii) TM Lattice with Paraphrases (dashed area indicates Types 1 and 2 paraphrasing), (iii) TM Lattice after Edit-distance Calculation of First Five Tokens using DPGA (in bold), (iv) Input Test Segment 53

3.2 (i) TM Segment, (ii) TM Lattice with Paraphrases (dashed area indicates Types 1 and 2 paraphrasing), (iii) DP Approach Considering All Paraphrases Available (numbers show values of j as given in Table 3.4), (iv) Input Test Segment 61

3.3 Blank Screen 74

3.4 Editing is In Progress 74

4.1 An Unfolded Recurrent Neural Network 112

4.2 An Unfolded LSTM network 114

4.3 An Unfolded Tree LSTM network 117

4.4 Network Architecture for Similarity Prediction (h_{in} and h_{tm} are computed independently using S_{in} and S_{tm} respectively) 121

4.5 Examples Showing the Change in Segment Representation using the '840blstmSet1' Model for Each Token 149

ACKNOWLEDGEMENTS

My thesis is highly enriched from various discussions, guidance and help. There are several people who were there for me and provided much needed support and unconditional love all these years. I am grateful to everybody who directly or indirectly supported me.

My deepest gratitude goes to my director of studies, Dr. Constantin Orăsan, for providing guidance that made the challenging PhD achievable. Our brainstorming discussions made the things smoother, faster and possible. I am grateful for his extensive supervision, feedback and support in all situations. His advice always work wonders.

A massive thank you is reserved to my supervisor Prof. Josef van Genabith for his guidance and thorough feedback during my PhD. I am very much thankful for his continued support in Saarbrücken and afterwards.

I am very much thankful to my supervisor and head of RGCL, prof. Ruslan Mitkov for giving feedback and suggestions during my research as well as providing me the opportunity to interact with distingushed people and real users of translation memory in the Industry.

I am also thankful to Dr. Iustin Dornescu for his valuable supervision even though it was for a short period and Dr. Michael Oacks for providing access to various research papers related to my work.

I am grateful to Prof. Qun Liu for having many meetings, deep discussions and guidance during my secondment at the Dublin City University which positively influenced my PhD. I am also thankful to Prof. Khalil Sima'an and Milos Stanojevic for discussions during my secondment at the University of Amsterdam and Juan Jos Arevalillo Doval, Carla Parra and Manuel Arcedillo for providing me real experience on TM matching in the Industry.

Many thanks to Marcos Zampieri and Mihaela Vela for their support during human evaluations and other colleagues from Saarland University for their help during my stay at Saarbrücken.

Many thanks to my examiners Prof Lucia Specia, Dr Marcello Federico and Dr Kevan A. Buckley and chair Prof Matt Wyon for making the whole process of PhD defence a great discussion and making the process a wonderful experience.

The Research Group of Computation Linguistics (RGCL) is one of the best places in the world to study computational linguistics. I am glad that I got opportunity to work with brilliant people. I am grateful to the colleagues and friends from the RGCL, especially Hannah Bechara for proofreading the early draft of PhD thesis and having many discussions and collaborations during the EXPERT project. A big thank you to Yvonne Skalban for proofreading the PhD thesis. A special 'thank you' to Emma Franklin for proofreading various research papers and providing academic English sessions which immensely helped me in writing my PhD. Many thanks to Miguel Rios Gaona, Victoria Yavena, Georgiana Marsic, Vinita Nahar, Richard Evans, Le An Ha, Shiva Taslimipoor, Sanja Stajner, Sara Moje, Najah Albaqawi, Ismail El Maarouf, Mijail Kabadjov, Miranda

Chong, Wilker Aziz and visiting researchers for having relaxing and informative discussions and for organising much needed parties and events during this PhD journey.

I owe my most sincere gratitude to Iain Mansell for providing administrative support during this whole PhD marathon. His support made it possible for me to attend various conferences and the EXPERT project events and complete my secondments smoothly. I would also like to thank April Harper, Helen Williams, Katherine Shepherd and Stephanie Kyle for administrative support and proofreading papers at various points during my PhD.

Many thanks to different partners in the EXPERT project for providing guidance and support. I must not forget to mention ESRs and ERs with whom I discussed so many things. A special thanks goes to Liling Tan for having great and sometimes funny discussions and help during my stay in Saarbrücken and his stay in Wolverhampton. Many thanks to Chris Hokamp for his help and informative discussions during my stay in Dublin. I would also like to thank Anna Zaretskaya, Santanu Pal, Hernani Costa, Varvara Logacheva, Carolina Scarton, Liangyou Li, Joachim Daiber, Hoang Cuong, Eduard Barbu, Najeh Hajlaoui, Lianet Sepulveda Torres and Alexander Raginsky for your help, comments and discussions during the EXPERT project events, secondments, conferences and other meetings.

Any work is impossible without a financial support. I am thankful for the full support of the EXPERT project funding from the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no. 317471.

Finally, I would like to thank my mother Meera Gupta and father Pradeep Kumar Gupta for their unconditional love, blessings and support during the whole PhD.

CHAPTER 1

INTRODUCTION

Translation is a process in which text in one language is rewritten into another language while preserving the meaning. Depending on the approach used to translate a text, translation processes can be divided into three types: (1) Machine Translation, (2) Manual Translation, and (3) Computer-Aided Translation.

In Machine Translation (MT), a text in one language is converted into another language by an automatic system without the need for any human input during the translation process. Due to the absence of any human input, the translations may not necessarily be correct. However, the process is very fast and we can translate a large number of words in a few minutes.

In Manual Translation, a human translator translates an entire text manually. This process is very slow compared to machine translation and requires human experts proficient in both languages involved in the translation; the language of the source text and the language of the target text. The translation quality is generally good if human experts translate the text.

In Computer-Aided Translation (also referred to as "Computer Assisted Translation"), a translator translates with the help of a Computer-Aided Translation (CAT) tool and other tools and resources, such as a terminology

dictionary, a machine translation system and spell checkers. The main component of a Computer-Aided Translation tool is usually a Translation Memory (TM) tool, which speeds up the translation process by retrieving previous translations from a TM so that not every segment has to be translated from scratch. CAT is widely used in translation industries requiring good quality translations. Because of human intervention, it ensures that the quality of translations is up to the mark while benefitting from the technology available.

Our research is focused on the third type of translation process, computer-aided translation, and in particular translation memory matching and retrieval.

1.1 Translation Memory

A Translation Memory (TM) is simply an archive of previous translations. The concept of TM can be traced back to 1978 when Peter J. Arthern proposed the use of a translation archive (Arthern, 1978). The TM contains the source text and its corresponding translations in one or more languages. It also typically contains meta information, such as the origin of the source text, the identification of the translator and the date when it was translated.

A translation memory can be created by a translator or can be provided by a company. Translation memories are also created automatically by aligning segments from previously translated documents.

There are many sophisticated commercial and open source tools developed to use translation memories in an efficient manner. These tools are generally referred to as Translation Memory (TM) tools. A 'TM tool' is also referred to as 'TM' in

short.

The main functionality of a TM tool is to retrieve a translation for reuse from the TM. In this way, a TM tool helps translators by retrieving the relevant match for reuse if a match or partial match (usually called fuzzy match) is available in the TM, which also help translators in maintaining consistency with previous work.

TM tools are very popular among professional translators, reviewers and post-editors. In a recent survey conducted by Zaretskaya et al. (2015), 76% of replies by translators confirmed the use of a TM tool. Over the years, TM tools have been improved with the focus on providing a good graphical user interface to the translators, developing different filters to handle different file formats (e.g. pdf, xml, txt, html, word, xliff, subtitle etc), project management features and standardisation of the formats for storing and sharing of TMs. There are various exchange formats proposed for this storing and sharing purpose (TMX[1], TBX, GMX, XML:TM, XLIFF etc.). Current TM tools are also equipped with tools like terminology managers and plugins to support machine translation from MT service providers. Although extensive research has been done in NLP with emphasis on improving the performance of MT, there is not much research on improving the TM systems by using NLP techniques.

1.2 Translation Memory Matching and Retrieval

TM tools process an input file for translation and extract the segments to be translated. These segments are checked against previously stored segments in

[1]http://www.ttt.org/oscarstandards/

the TM. In a typical TM tool, three types of matches are displayed to the user: in context matches, exact matches, and fuzzy matches. Exact matches are those matches for which a 100% match is found in the TM. 'In context matches' generally refer to matches where previous and following segments are also exact matches. Different TM vendors refer to 'in context match' differently. For example, Memsource[2] call it 101% match and XTM[3] call it ICE match. When a TM tool is unable to find an exact match in the TM, it retrieves similar segments for post-editing in order to avoid translation from scratch. These similar segments are called fuzzy-matches. However, this retrieval process is largely limited to edit-distance based measures that work on surface form (or sometimes stem) matching.

Most of the commercial systems use edit-distance (Levenshtein, 1966) or some variation of it with some extra preprocessing to perform this matching. Preprocessing typically involves tokenisation, removing punctuations, removing stop words and stemming. Although these measures provide a strong baseline, they are not sufficient to capture the semantic similarity between the segments as judged by humans. Due to limited linguistic processing in TM tools, similar segments, which vary on surface forms, are often not retrieved. For example *What is the Commission's position on this issue?* and *What is the Commission's view on this matter?* have the same or very similar meaning, and *I would like to congratulate the rapporteur* and *I wish to congratulate the rapporteur* have the same meaning. Many more examples can be given. Because current TM systems

[2]https://www.memsource.com
[3]http://xtm-intl.com

work on surface form, they will not consider these segments similar enough to use one instead of the other. In this research, we try to mitigate such problems.

In this thesis we do not use a strict definition of the similarity and we do not define any manual linguistic rules which can decide the degree of similarity between a pair of sentences. Instead the notion of similarity captured by our approaches depends on the dataset used and is modeled using machine learning and data driven approaches, in this way removing the need for human intervention. In Chapter 3, we use a paraphrase database which contains paraphrases extracted using bilingual corpora. We use lexical and phrasal paraphrases. A paraphrase contains a maximum of 6 words. For example: *particularly concerned at the situation of* $\leftrightarrow$ *especially concerned about the situation of*.

In Chapter 4, we use a similarity corpus derived by us using data from the workshop on machine translation (WMT-2013) (Bojar et al., 2013), and the datasets available from the Semantic Textual Similarity (STS) shared tasks (Marelli et al., 2014b; Agirre et al., 2012, 2013, 2014, 2015). STS datasets are a mixture of various types including headlines, questions-answers from an online website, image captions and european parliament proceedings. Some pairs from the dataset are given in Table 1.1. The score is a number between 1 (least similar) and 5 (most similar), and Text1 and Text2 represent the pair of sentences in Table 1.1.

Whereas the similarity measures used in the thesis may be useful for other tasks, in this thesis they are applied specifically for improving the performance of TMs. In our research, we use paraphrasing with edit-distance and advanced

Text1	Prostate cancer screening: take the test or not?
Text2	Prostate cancer screening : the test should be done or not?
Score	4.83
Text1	Dog running towards camera with a ball in its mouth.
Text2	The black and white dog swims with a brown object in its mouth.
Score	2.6

Table 1.1: Some Examples From Our Training Dataset

semantic methods. The approaches presented in Chapter 3 use paraphrasing to improve upon a baseline edit-distance which is widely used in translation memory matching and retrieval.

Chapter 4 presents approaches which are related to standard sentence similarity measures. They compute semantic similarity at an advanced level, where the focus is on similar meanings being captured even when segments are not similar at string level. An approach which computes semantic similarity between a pair of sentences can get complex in terms of language dependent NLP libraries, time and computing resources needed. We designed our approaches simple enough for TM matching and retrieval. In addition, training data for standard semantic similarity task is limited, therefore we also derive a training data using the data from the WMT-2013 workshop (Bojar et al., 2013), which has much longer sentences and more appropriate for our TM matching task.

To summarise, our research focuses on improving TM matching and retrieval with the help of advanced language technology. This is achieved by:

- Using paraphrases in the TM matching process
- Using machine learning techniques for advanced semantic matching

1.3 Research Questions and Main Contributions

Our research focuses on investigating the following research questions:

Q1 How do we use paraphrases efficiently in the TM matching and retrieval process?

Q2 Does paraphrasing improve TM matching and retrieval?

Q3 Can advanced semantic matching techniques improve TM matching and retrieval?

The main contributions of this thesis are as follows:

1. In our literature survey, we cover various techniques to compute semantic similarity at the sentence level that can benefit TM matching and retrieval.

2. We propose two novel and efficient approaches to improve TM matching and retrieval using paraphrases. The approaches obtained better results compared to the baseline edit-distance. Our implementations are available on Github.[4]

3. We propose three approaches based on machine learning techniques. One approach is based on manually designed features and support vector machines (SVMs), and other two approaches are based on Recurrent Neural Networks (RNNs). Although all three approaches did not obtain better results using automatic evaluation metrics in comparison to the baseline

[4]https://github.com/rohitguptacs

edit-distance, our manual analysis indicates results comparable to the baseline edit-distance.

1.4 Thesis Structure

This thesis is divided into five chapters. The structure of the thesis is as follows:

In Chapter 2, we present related work in TM matching and retrieval, and semantic similarity matching in general. We explore various related NLP areas, which require semantic similarity computation at the sentence level, including machine translation evaluation, textual entailment, and semantic relatedness. We also give a brief overview of recently used deep learning techniques based on neural networks which can benefit translation memory matching and retrieval.

Chapter 3 presents two novel and efficient approaches to improve TM matching and retrieval using paraphrases. We present efficient ways to use existing paraphrase databases in the TM matching and retrieval process. Our approaches are based on 'dynamic programming and greedy approximation' and 'dynamic programming only'. The approaches incorporate paraphrasing with a word-based edit-distance procedure in polynomial time complexity. The approaches are simple and efficient enough to implement in a typical translation memory matching and retrieval pipeline. We have also conducted extensive automatic and human evaluations, and we concluded that translators save time and use less keystrokes when they use paraphrase enhanced TM matches compared to simple edit-distance matches. This chapter addresses the first two research questions (Q1 and Q2) stated in Section 1.3.

Chapter 4 presents three approaches to improve TM matching and retrieval using advanced semantics. This chapter is divided into two parts (Section 4.1 and Section 4.2). In the first part (Section 4.1), we present an approach based on manually designed features. We present techniques to extract features and use these features to train a semantic textual similarity (STS) system using SVM. We did not obtain better results than the baseline edit-distance. In the second part (Section 4.2), we present our work using deep learning techniques based on Recurrent Neural Networks to compute semantic similarity between segments. In particular, one approach is based on Long Short Term Memory (LSTM) networks and another approach is based on Tree Structured Long Short Term Memory (Tree-LSTM) networks. We train eight different models using different datasets and different settings. Our results do not show improvements for any of the eight models over the baseline edit-distance. This chapter addresses the third research question (Q3) stated in Section 1.3.

In Chapter 5, we compare our 'dynamic programming approach' to incorporate paraphrasing proposed in Chapter 3 and the 'WikiTree' model, which is based on Tree-LSTM networks, proposed in Chapter 4. We conclude our work and provide some future directions to extend our work.

Some of the work presented in this thesis has already been published in several peer-reviewed conference and workshop proceedings. Appendix A lists the related publications.

CHAPTER 2

RELATED WORK

In this chapter, we explore related work on TM matching and retrieval, and other natural language processing areas that can benefit TM matching and retrieval. The emphasis in this chapter is on presenting research that is relevant to this thesis.

In TM matching and retrieval, we need to compute similarity between segments so that a good matching segment can be retrieved (for post-editing, if required). A good match requires less or no post-editing whereas in the absence of a good match, a translator needs to translate from scratch.

In commercial TM systems, the similarity is generally computed using edit-distance or some variation of it. There is not much research on TM that specifically focuses on other measures of similarity for TM matching and retrieval, but there are various areas in NLP that require calculating some kind of similarity between two texts. Different similarity measures have different characteristics and their effectiveness depends on the type of text, length of text, domain and application. In speech recognition, similarity between two strings of phonemes is typically measured using an error rate measure. In machine translation evaluation, a sentence is typically evaluated against one or more available references and the final score over the test set is obtained by using the statistics obtained from all

sentences in the test set.

Research areas such as textual entailment and semantic similarity computation between texts relate to fuzzy match score computation for TM matching and retrieval. In these areas, the way similarity is computed differs substantially from the way fuzzy match score is computed in TM. However, as stated earlier, in TM we need to compute similarity between segments in order to retrieve a good match. TM segments generally correspond to sentences. Therefore, some of the techniques which work at the sentence level can be leveraged by TM matching and retrieval. Recent developments in the field of artificial neural networks also made it possible to represent segments as distributive dense vectors and compute the similarity between them.

Some previous research involves combining TM with machine translation, however, mostly to improve machine translation and rarely the other way round, i.e. to improve translation memory. We can divide the literature related to TM matching and retrieval into five categories: Research on improving TM matching and retrieval; combining TM with MT; machine translation evaluation; semantic similarity and textual entailment using traditional approaches; and deep learning for semantic similarity. We explore each of these categories in this chapter.

2.1 Research on Improving TM Matching and Retrieval

During the last decade, researchers have shown interest in improving fuzzy matching and retrieving better quality segments from TM. Several researchers

have used semantic or syntactic information in TM, but the evaluations they performed were small scale and in most cases limited to subjective evaluation by the authors. This makes it hard to judge how much a semantically informed TM matching system can benefit a translator. Existing research pointed out the need for similarity calculation in TM beyond the surface form comparison (Planas and Furuse, 1999; Macklovitch and Russell, 2000; Somers, 2003; Hodász and Pohl, 2005; Pekar and Mitkov, 2007; Mitkov, 2008). Macklovitch and Russell (2000) explained that using NLP techniques like named entity recognition and morphological processing can improve matching in TM. Somers (2003) highlighted the need for more sophisticated matching techniques that include linguistic knowledge like inflection paradigms, synonyms and grammatical alternations. Both, Planas and Furuse (1999) and Hodász and Pohl (2005) proposed to use lemma and parts of speech along with surface form comparison. Planas and Furuse (1999) proposed that considering different levels of representation of text in similarity computation is useful. The proposed structure has eight different levels:

1. Text characters: the characters in the sentence,

2. Surface Words: the surface forms of the words in the sentence,

3. Lemmas: lemmatised words in the sentence,

4. POS: parts of speech tags of the words,

5. XML content tags: The XML content tags, for example, layout attribute

tags used in an XML segment,

6. XML empty tags: These tags take care of objects inserted in the flow of text like images,

7. Glossary entries: glossary entries of company conventions (translations),

8. Linguistic analysis structures: This layer consists of "pivot schemata". The "pivot schemata" are simply patterns composed of pivot keywords and variables with a link between two schemata; source language pattern and target language pattern. For example, "A and B" in English mapping to "A et B" in French; A-A, and-et, and B-B. Here the pivot keyword is 'and' and the pivot variables are 'A' and 'B'.

Planas and Furuse (2000) proposed an algorithm that considers the above structure and computes the similarity between two segments. The algorithm to compute similarity is similar to the edit-distance algorithm, but it considers deletions and equality operations only and no insertions are allowed. This is an essential requirement for the algorithm implementation and it means the algorithm does not retrieve segments where insertions are required. A vector is formed after taking into consideration edit-distances calculated at each layer. The ranking of edit-distance vectors is performed based on a partial order defined on vectors. For example, assuming two layers with edit-distances a_i and b_i, and with vectors A=[a_1, b_1] and B=[a_2, b_2]. $A > B$ iff $(a_1 > a_2)$ or $(a_1 = a_2$ and $b_1 > b_2)$. This means that surface form characters are given more preference over words; words are given more preference over lemmas; and lemmas are given more preference

over POS tags and so on. The authors tested a prototype model on 50 sentences from the domain of software manuals and 75 sentences from a corpus containing online economical news with TM sizes of 7,192 sentences and 31,526 sentences, respectively. The authors concluded that the approach gives more usable results compared to Trados Workbench (a industry standard TM tool) used as a baseline. A fuzzy match retrieved was considered usable if less than half of the words required editing to obtain the input sentence.

Hodász and Pohl (2005) also included noun phrase (NP) detection and alignment in the matching process. The NPs are either tagged by a translator or by a heuristic NP aligner developed for English-Hungarian translation. The automatic NP detection and alignment is performed as follows:

- Dictionary: The method searches all stems of detected NPs in an English-Hungarian dictionary. The search starts in a greedy manner with the longest NP. If the longest NP is not found, sub parts of it are searched in the dictionary. A similar procedure is applied on both sides, English and Hungarian. Finally, the alignment is done based on whether at least one token on the Hungarian side matches the NP detected on English side.

- Cognate: Cognates are searched for among remaining NPs after the dictionary based search. Cognates are considered words that are longer than one character, have at least one capital letter, number or special character and either match or have the first four characters in common.

- POS: Parts of speech are matched when dictionary and cognate matches are

not possible.

The matching score is calculated by combining the above three matches, dictionary, cognate and POS. The matching score (MS) computation is given below:

$$MS = \frac{a \cdot \text{DMW} - b \cdot \text{DNMW} + c \cdot \text{CMW} + d \cdot \text{PMW} - e \cdot \text{RFW}}{W - \text{RFW}}$$

In the above equation, DMW represents the number of dictionary matched words, DNMW represents the non NP words, CMW represents the number of matched cognate words, PMW represents the number of words matched based on POS tags, RFW represents the number of function words and W is the total number of words. a, b, c, d and e are weights. The authors tested their approach on one sample sentence. They found that their system retrieves better matches compared to a baseline system which uses simple edit-distance without any linguistic processing. Hodász and Pohl (2005) claim that their approach matches using simplified patterns based on linguistic analysis and hence makes it more probable to find a match in TM.

Clark (2002) proposed to associate words or phrases to be translated with the previous translations of such words if available. The association can be performed based on commonality of identifying attributes (for example, formatting information and respective locations of words in a document). The identifying attributes are extracted from one or more source documents for the words to be translated and from one or more target documents for the previous stored translations.

Pekar and Mitkov (2007) presented an approach based on syntactic analysis. A segment is represented in a generalised form using this analysis. Three procedures are employed to carry out this generalisation: syntactic generalisation, lexico-syntactic generalisation and lexical generalisation. The goal of syntactic generalisation is to transform equivalent constructions like active and passive constructions to a single representation. Syntactic generalisation is performed using predefined syntactic transformation rules. Lexico-syntactic generalisation is used to account for the variability of syntactic constructions when using equivalent lexical expressions (For example, 'X likes Y' vs 'Y is appealing to X'). Lexical generalisation is used to generalise the lexical units to an equivalent class in a thesaurus. Lexical generalisation also recognises named entities and maps to an equivalent class. First-order logic representations of each predicate and its arguments found in the sentence are derived and compiled into a tree graph.

The authors tested the retrieval performance by using one test sentence obtained from the website proz.com[1] and an artificially created TM for this experiment which contains eight sentences obtained using the top eight documents retrieved through a Google search when using the test sentence as a query. Two sentences that would be the relevant matches are manually added; one with synonyms, and another with synonyms and syntactic variation. Three different similarity measures are compared in this evaluation: (1) cosine similarity; (2) tree edit-distance with WordNet similarity between matching nodes; (3) same as (2) with additional paraphrases. In (3), nine more variants of paraphrases are used

[1]http://www.proz.com is a website for freelance translators and companies.

to extend the query. Two verbs available in the query sentence are replaced by three paraphrases for each verb, making nine more combinations ($3 \times 3 = 9$). Two verbs are *purchase* (paraphrase used: *buy*, *sell*, *take delivery of*) and *receive* (paraphrases used: *notify*, *consult*, *send*).

The query segment used (Q) and top segment retrieved using all three approaches (1, 2 and 3) are given below:

Q: The company must purchase materials for release of goods, usually before any money is received from its customers.

1: It acknowledges receipt of the goods for shipment and is a certificate of ownership which must be produced before goods may be released for delivery.

2: The business has to acquire raw materials for production of commodities, typically prior to any payment is made by its clients.

3: The business has to take delivery of raw materials for production of commodities, typically before clients send any money.

The authors have found that method (1) retrieved one of the segments added using the Google search document; method (2) retrieved the manually added segment which contains synonyms; and method (3) retrieved the manually added segment which contains synonyms and syntactic variation.

One of the things to note is that the authors extended the query with nine more variants. This is one of the major problems when using paraphrasing. Generating

additional segments using paraphrasing increases the size of TM exponentially. In Chapter 3, we propose an algorithm to efficiently handle this issue.

Recently, Utiyama et al. (2011) presented an approach which uses paraphrasing in TM matching and retrieval. The authors proposed a method using a finite state transducer where they treat the TM matching and retrieval procedure as a search process and they do not consider the fuzzy matching aspect. The authors acquire a paraphrase list using the approach proposed in Bannard and Callison-Burch (2005). The method proposed by Utiyama et al. (2011) searches the best path in the composition of four weighted finite state transducers, given below:

1. InputFST: Accepts an input sentence and outputs the same sentence.

2. ParaFST: Accepts an input sentence and outputs its paraphrases. This FST consists of all paraphrases and all words in the source language vocabulary.

3. LMFST: Language model FST created using the Kyoto language modelling toolkit[2].

4. TMFST: Accepts an input sentence and outputs its index.

These four FSTs are composed and the best path is calculated to obtain the match as follows:

$$BestPath(\textit{InputFST} \circ \textit{ParaFST} \circ \textit{LMFST} \circ \textit{TMFST})$$

[2]http://www.phontron.com/kylm/

The InputFST, ParaFST, LMFST, TMFST transducers are compiled and processed using the OpenFST library[3]. Composition of all four FSTs and best path calculation is performed using the Kyoto FST Decoder[4].

Texts from the health-care domain are used for testing. From this data, two sets are constructed: set1 consisting of 41,712 sentences which occurred more than once in the dataset; and set2 consisting of 90,498 sentences which occurred only once in the dataset. Set1 is used to create 266,519 paraphrase pairs. Furthermore, sentences in set2 are divided into a development and test set consisting of 44,817 and 44,975 sentences, respectively. The test set is used as input to retrieve paraphrased sentences. The method retrieved 4.87% (2189) of input segments. The authors created a sample of 100 sentences from these retrieved translations to check the precision. A Japanese-English translator judged the accuracy of the translations. 91 out of the 100 sentences in the sample set are evaluated to be correct, resulting in a precision of 91%. The authors considered the word accuracy based method as a baseline. Word accuracy is calculated as follows: count the number of insertions, deletions and substitutions required to turn the reference into the input and divide this count by the number of words in the reference. The authors extracted the same number (2189) of top sentences retrieved using the word accuracy method. Similarly, 100 sentences were sampled from the retrieved translations and judged by the translator. 76 out of 100 translations were found to be correct, making it 76% precision. The downside of this approach is that their

[3]http://www.openfst.org/
[4]http://www.phontron.com/kyfd/

approach limits TM matching to exact matches only, which means fuzzy matches are not retrieved.

Jaworski (2013) presented a approach based on suffix arrays to speed up the search process in TM. The approach used a preprocessing stage where punctuations and stop words were removed. From the resulting string, all suffix strings are formed. For example for the string "I am fine", suffixes will be "I am fine", "am fine" and "fine". These suffixes are stored in an array called suffix array. The suffix array also stores the index of the sentence from which suffixes are formed and the offset of the suffix within the sentence. An algorithm based on prefix and subsequence matches is used to retrieve the matches using the suffix array. In the evaluation, the author reports that their approach retrieves matches four times faster than MemoQ (an industry standard CAT tool) and the quality of matches is somewhat comparable to MemoQ. However, MemoQ retrieved 20% more matches.

Timonera and Mitkov (2015) proposed splitting sentences into clauses as a preprocessing stage to improve matching and retrieval. The authors used a paraphrase corpus (Cohn et al., 2008) instead of a real TM for experiments and report that splitting sentences into clauses significantly improves matching and retrieval of the TM.

2.2 Combining TM with MT

Early research on improving translation memory focused on example-based machine translation (EBMT) techniques. Carl and Hansen (1999) concluded

that the linking of TM and EBMT seems better for the translation task. The hypothesis was that decomposition and generalisation, like that provided by an EBMT system, leads to better coverage while surface form comparison, like the one used by TM systems, leads to better quality. The authors reported that example-based machine translation performs better compared to a string-based translation memory matching for 80% or less fuzzy match score. This means that a translation produced by the EBMT system, which translates by decomposing and combining the segments in the TM, is more usable than low fuzzy matches obtained from the TM using string matching.

Recent techniques have been proposed to exploit SMT in various ways. Some techniques use SMT as an alternative when no match is found in TM (Simard and Isabelle, 2009; Dandapat, 2012) whereas some techniques use SMT to complete the fuzzy match extracted from the TM (Koehn and Senellart, 2010; Tezcan and Vandeghinste, 2011).

Simard and Isabelle (2009) used a quality estimation system to select the better translation between TM and SMT. The quality estimation system was trained on features like source and target segment lengths, source and target language model probabilities, language model probabilities ratios, machine translation probabilities from IBM model 2 (Brown et al., 1993) and various similarity measures (Levenshtein edit-distance, Longest common subsequences etc.). A Support Vector Machine (SVM) regression model was used to combine these features and predict the quality of translation. The authors also used the features from the TM technology to re-rank the n-best translations provided by the SMT

system. The features consist of Levenshtein edit-distance, unigram and bigram precision. The overall system was obtained by combining the SMT system and TM. For any input sentence the selection between TM or MT output was made using the quality estimation system. The combined system was checked on three different datasets viz Europarl (Koehn, 2005), Hansard (Roukos et al., 1995) and JRC-Acquis (Steinberger et al., 2006) corpora. When compared to the baseline SMT system, for the Hansard and JRC-Acquis datasets, the combined system obtained good improvements of 1.8 and 2.6 BLEU points respectively, whereas for Europarl, a slight decrease of 0.3 BLEU points was observed.

Koehn and Senellart (2010) proposed an approach which completes the fuzzy matches and shows that for more than 70% fuzzy match score the approach performs better than a single SMT or TM system. The testing was done using the MT evaluation metric BLEU with the baseline as either of MT or TM alone.

Tezcan and Vandeghinste (2011) pointed out various issues when integrating SMT and TM, mainly focusing on XML and markup issues. They propose a generalisation of XML tags before SMT training at the time of tokenisation and replacing the tags back after the translation. The testing was done on TM from the automotive domain on two language pairs, English-Spanish and English-French. 400K segments were used to train the SMT system for each language pair, 912 segments were used as test set for English-Spanish and 871 segments were used as test set for English-French. The approach achieved 0.7 BLEU points improvement for the English-Spanish language pair, and 0.88 BLEU points improvement on English-French.

There are recent trends in the adaptation of SMT systems using a TM system (Bertoldi et al., 2013; Wang et al., 2014). Bertoldi et al. (2013) built a small local (dynamic) translation model using the translations created by translators along with the typical static SMT model. This local translation model is updated every time the translator post-edits an MT output. Both the local as well as the global static SMT models perform the overall translation. It was observed that this technique can be used to improve machine translation systems if the text is very repetitive in nature.

Wang et al. (2014) proposed an approach to dynamically incorporate TM generated segments in SMT systems. This approach adds parameters from a TM system to improve decoding in SMT as well as update the SMT model with additional phrases from TM. For the above 50% fuzzy match score, the authors observed significant improvements in the machine translation. Furthermore, the proposed approach retrieves better results than either TM or the SMT system. Li et al. (2014) extends the work of Wang et al. (2014) by adding more features computed using multiple fuzzy matches retrieved from the TM. The authors obtained results comparable to Wang et al. (2014).

2.3 Machine Translation Evaluation

MT evaluation metrics evaluate a machine translation output based on one or more available references. In general, an MT evaluation metric measures the similarity between a machine translation output and the reference. MT evaluation metrics generally compute statistics at the sentence level and combine the statistics

computed at the sentence level for the whole test set to obtain the evaluation score of a machine translation system. The metrics, which calculate a score on the basis of individual sentence scores, can be leveraged for TM matching and retrieval. Often, MT evaluation metric scores are also used as features for the systems used to compute similarity between texts (Malandrakis et al., 2012; Gupta et al., 2014; Tan et al., 2015; Xu et al., 2015; Vu et al., 2015).

Several machine translation evaluation metrics are proposed every year in the machine translation evaluation tasks (Callison-Burch et al., 2012; Bojar et al., 2013, 2014; Stanojević et al., 2015). Some of the most popular and widely used metrics are BLEU (Papineni et al., 2002), Meteor (Denkowski and Lavie, 2014), NIST (Doddington, 2002), TER (Snover et al., 2006) and WER. Below we present some of the most used metrics in MT and TM evaluations.

BLEU (Papineni et al., 2002) is the most popular metric. The metric is based on N-gram counts. N-grams collected over the hypothesis translations are matched against the N-grams collected from the references. The BLEU metric is computed as follows:

$$\begin{aligned} BLEU &= BP \cdot \exp\left(\sum_{n=1}^{N} \log p_n\right) \\ BP &= min\left(1, e^{1-r/c}\right) \end{aligned} \tag{2.1}$$

where p_n represents the precision of N-grams of length n, c represents the length of test set and r represents the length of reference translation. The value of N is usually 4. BP represents the brevity penalty and it is used to compensate

high-precision hypotheses which are too short.

The NIST metric (Doddington, 2002) is similar to BLEU as it is also based on N-gram counts. In BLEU, each N-gram has an equal contribution whereas in NIST, N-grams are weighted. The importance of an N-gram is decided by the use of this N-gram in the test corpus. The rarer the n-gram is, the more importance it will be given.

Both BLEU and NIST match the given hypothesis translation and reference on the surface form and do not use any linguistic resource like WordNet or paraphrase databases to perform a semantic matching.

The METEOR (Denkowski and Lavie, 2014) metric evaluates translation on the basis of a phrase to phrase alignment between the given hypothesis translation and reference. Words and stems are considered as phrases of length 1. In the case where multiple references are available, the hypothesis is scored against each reference and the maximum scoring reference is used. The algorithm performs alignment on the basis of four matchers: exact words matcher, stem matcher, synonyms matcher and paraphrase matcher. The aligner constructs a search space by applying these matchers sequentially to get all possible matches between the hypothesis and the reference. Once all possible matches are identified, the aligner obtains the largest subset of these matches, fulfilling the following criteria in the order of preference given below:

1. Each word in each sentence is covered by zero or one matches

2. Maximize the number of words covered across both sentences

3. Minimize the number of chunks, where a chunk is no more than a series of contiguously matched and identically ordered phrases

4. Minimize the sum of absolute differences between match start positions in both sentences

After getting the alignment, precision (P) and recall (R) are calculated as follows (and used in the calculation of the Meteor score):

$$P = \frac{\Sigma_i w_i \cdot m_i(t)}{|t|} \qquad R = \frac{\Sigma_i w_i \cdot m_i(r)}{|r|}$$

In the above equations, $|t|$ is the number of words in a hypothesis translation t, $|r|$ is the number of words in the reference r. $m_i(t)$ is the number of words covered by matcher m_i for hypothesis translation t and $m_i(r)$ is the number of words covered by the matcher m_i for reference. w_i is the weight for each matcher. Using precision and recall F_{mean} is calculated as follows:

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1 - \alpha) \cdot R}$$

Finally, the METEOR score is calculated as follows:

$$Score = (1 - Penalty) \cdot F_{mean} \tag{2.2}$$

Where, $Penalty$ is used to account for word order, given as follows:

$$Penalty = \gamma \cdot \left(\frac{ch}{m}\right)^{\beta}$$

In the equation above, ch is the minimum number of chunks and m is the number of matched phrases.

Recently, Huang and Chang (2014) and Tan et al. (2015) used Meteor to compute semantic similarity between texts and achieved good results.

Word Error Rate (WER[5]) is used as one of the basic metrics for machine translation evaluation. This metric computes a score on the basis of the number of insertions, deletions and substitutions required, divided by the number of words in the reference. WER is also widely used to calculate fuzzy match score in TM.

The Translation Edit Rate (TER) metric (Snover et al., 2006) improves the WER metric for MT evaluation. TER combines the knowledge from multiple references and penalises reordering of words and phrases less compared to WER. In TER, block movement of words called *shifts* have the same penalty (of 1) as inserting, deleting and substituting a single word.

The TERp metric (Snover et al., 2008) is an extension of TER. TERp also includes stemming, synonyms and paraphrases. TERp uses all the edit operations of TER viz *insertions*, *deletions*, *substitutions* and *shifts* along with three additional operations: *stem* matches, *synonym* matches and *phrase* substitutions. Instead of treating all substitutions as edits of cost one, the cost of substitution varies depending on whether two words share the same stem, are synonyms or are paraphrases of each other (for *phrase* substitutions). Stemming is implemented using the Porter stemming algorithm (Porter, 1980), synonym operation is implemented using WordNet and a paraphrase table is generated using the approach proposed by Bannard and Callison-Burch (2005). Sequences of words in the reference are searched to get the paraphrases from this table and used

[5]http://en.wikipedia.org/wiki/Word_error_rate

for phrase substitution operations. The cost of phrase substitution is computed using the number of edits required to align the two phrases according to TERp (without phrase substitution) and the probability of the paraphrase.

The recently proposed ReVal metric (Gupta et al., 2015b) is based on dense vector spaces and recurrent neural networks. In particular, the metric uses Tree Structured Long Short Term Memory networks (Tai et al., 2015) and GloVe word vectors (Pennington et al., 2014). The training data is computed automatically from the WMT-13 (Bojar et al., 2013) human evaluation rankings. The rankings are converted into similarity scores between the reference and the translation. The metric has been tested on WMT-12 and WMT-14 test sets and has participated in the WMT-15 metric task. On the WMT-12, WMT-14 and WMT-15 test sets, the metric performed either best or second best overall using different correlation measures, which compute correlation of ReVal metric scores with official system level scores obtained using human evaluations in the WMT taks. The metric obtained an average 0.976 Pearson correlation over all languages for the WMT-15 task. More details about the metric are given in Chapter 4.

Guzmán et al. (2015) presented a metric based on word embeddings and neural networks. A neural network classifier is trained to predict a better system between the two systems. The metric uses human evaluation rankings available from the WMT workshops for training. This metric is limited to ranking the available systems and does not provide an absolute score.

Recently, Simard and Fujita (2012) presented an interesting paper where the MT evaluation metrics are considered as TM similarity functions. The

authors used English-French, English-German, English-Spanish language pairs (both ways) of the different corpora; Europarl, ECB, EMEA and JRC-Acquis (Tiedemann, 2009). 1000 segments from each corpus are selected for testing and the rest of them are used as TM. The quality of the retrieved matches is evaluated using the same MT evaluation metrics which are used as the similarity functions for the TM matching and retrieval. The authors find that a metric gives the best scores to the matches retrieved by that metric itself, except for the NIST metric. Also, two versions of Meteor are used: without-paraphrases and with-paraphrases. The authors observed that the matches retrieved using the without-paraphrases metric score better most of the time over matches retrieved using with-paraphrases. They pointed out that the semantic processing and matching performed on the source side (English) does not reflect on the target side because of the lack of necessary resources (e.g. lack of paraphrases and unavailability of WordNet on the target side) to evaluate on the target side. However, we have not observed such issues with our proposed edit-distance with paraphrasing approaches (in Chapter 3). Our approach even improves the BLEU score, which does not consider any linguistic resource.

We use BLEU, Meteor and TER to evaluate the quality of the retrieved segments in our research and ReVal for some of our experiments in Chapter 4. We also use features computed using BLEU for our semantic similarity system using SVMs in Chapter 4.

2.4 Semantic Similarity and Textual Entailment using Traditional Approaches

Apart from semantic textual similarity measures presented in various journals and conferences, there are Semantic Textual Similarity (STS) shared tasks organised every year where several similarity measures are proposed which compute semantic similarity between texts (Agirre et al., 2012, 2013, 2014, 2015).

We discuss here some of the techniques used in NLP for measuring similarity between texts, particularly at the sentence level using *traditional approaches*. These approaches differ from the recent deep learning research (presented in Section 2.5) in that they do not require neural networks for training and the features used to compute similarity between sentences are generally decided by human experts. For example, to calculate the similarity of a sentence with another sentence, some of the features can be: both sentences are of similar lengths; they share common words; or have similar syntactic structures.

We divide *traditional approaches* into three categories: Sentence Similarity as a Function of Words; Edit-distance Based Approaches; and Combining Various Similarity Measures.

2.4.1 Sentence Similarity as a Function of Words

Most of the techniques view sentence similarity as a function of words or function of words and word positions in the sentence (Li et al., 2006; Tsatsaronis et al., 2010; Islam and Inkpen, 2008; Gu et al., 2012). Li et al. (2006) and Tsatsaronis et al. (2010) used WordNet to obtain similarity between words and model sentence

similarity as a function of words and their positions within the sentence. Gu et al. (2012) and Islam and Inkpen (2008) used point-wise mutual information (PMI) (and some variants) computed using large corpora to obtain similarity between words and sentence similarity was modeled as a function of words. Apart from the PMI, Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997; Landauer et al., 1998), Hyperspace Analogues to Language (HAL) (Burgess et al., 1998) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003) are also generally used to compute semantic similarity using large corpora.

We briefly describe the approaches proposed by Li et al. (2006) and Islam and Inkpen (2008) below.

Li et al. (2006) presented a semantic similarity metric based on the semantic similarity of words in a sentence. Word similarity and word order similarity were used to calculate sentence similarity. Word order similarity was calculated by considering the index of the words appearing in a sentence as explained by the example below taken from (Li et al., 2006). The pair of sentences is as follows:

T1: RAM keeps things being worked with

T2: The CPU uses RAM as a short-term memory store

For the given pair of sentences, a joint word vector is formed by assigning unique words from both the sentences and preserving the order wherever possible. For the above pair of sentences T1 and T2, the joint vector T is given below:

T= {RAM keeps things being worked with The CPU uses as a short-term memory store}

Using the joint word vector with each of the sentences, corresponding word order vectors are formed respectively for each of the sentences. The method assigns a unique index number for each word in T1 and T2. The index numbers are nothing more than the order numbers in which the words appear in the sentence. The order vector is created using each sentence vector and joint word vector as follows, taking T1 as an example. For each word w_i in T, this method selects either the same or the most similar word above a predefined threshold in T1. If an identical or similar word is found, the index number of the corresponding word is given; otherwise 0 is given for the corresponding entry. For the above example, word order vectors for T1 and T2 are r1 and r2, respectively, given below:

$$r_1 = \{\ 1\ 2\ 3\ 4\ 5\ 6\ 0\ 3\ 3\ 0\ 0\ 0\ 1\ 1\ \} \qquad r_2 = \{\ 4\ 0\ 3\ 0\ 0\ 0\ 1\ 2\ 3\ 5\ 6\ 7\ 8\ 9\ \}$$

The word order similarity was determined by the normalised difference of word order. The word order similarity (S_r) was defined as follows:

$$S_r = 1 - \frac{||r_1 - r_2||}{||r_1 + r_2||} \tag{2.3}$$

A similarity measure to calculate similarity between individual words using WordNet was proposed as follows:

$$s(w1, w2) = e^{-\alpha l}\frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{2.4}$$

where $s(w1, w2)$ is the similarity between words $w1$ and $w2$, l is the shortest path length in WordNet between the synsets containing compared words $w1$ and $w2$, h is the depth of the subsumer in the hierarchical semantic nets (subsumer refers

to the synset that represents a hypernym of both words $w1$ and $w2$ in hierarchical semantic nets, for example, the synset of *person* is called the subsumer for words *girl* and *boy*), $\alpha \in [0, 1]$ and $\beta \in (0, 1]$ are parameters scaling the contribution of shortest path length and depth, respectively. Using the word similarity, semantic similarity (S_s) is calculated as follows: the joint vector (T) was represented as a row and the sentence (T1) was represented as a column. Each column of this matrix was filled with the value either 1 if both word are same or with the similarity for the pair of words. This matrix was used to create a lexical vector. Taking the maximum value from each column forms this lexical vector. Furthermore, the values in the lexical vector are weighted using information gain calculated using the Brown Corpus for the pair of words whose scores are used in lexical vector. For the example given above the lexical vector corresponding to T1 is as follows:

$$s_1 = \{0.390\ 0.330\ 0.179\ 0.146\ 0.239\ 0.074\ 0\ 0.082\ 0.1\ 0\ 0\ 0\ 0.263\ 0.288\}$$

Similarly, for the other sentence T2, a different lexical vector is formed.

$$s_2 = \{0.390\ 0\ 0.1\ 0\ 0\ 0\ 0.023\ 0.479\ 0.285\ 0.075\ 0.043\ 0.354\ 0.267\ 0.321\}$$

Using these lexical vectors the semantic similarity is calculated as follows:

$$S_s = \frac{s_1 \cdot s_2}{||s_1|| \cdot ||s_2||} \tag{2.5}$$

Sentence similarity was defined as follows:

$$S(T1, T2) = \delta S_s + (1 - \delta) S_r \tag{2.6}$$

where $S(T1, T2)$ is the similarity between sentences $T1$ and $T2$, S_s denotes semantic similarity calculated using WordNet by taking words occurring in sentences $T1$ and $T2$, S_r denotes word order similarity between sentences $T1$ and $T2$ and δ is a weighing parameter. For the example case above, the semantic similarity between sentences is $S_s = 0.6139$ and order similarity $S_r = 0.2023$. Taking a value of $\delta = 0.85$, which was empirically determined, the similarity between the sentences is 0.5522. For evaluation purposes, a human annotated dataset consisting 30 sentence pairs was created. The system performed well with 0.816 Pearson correlation coefficient against the human rated dataset.

Islam and Inkpen (2008) also use context and statistical information from a corpus. The authors proposed a method which uses string similarity between words, semantic similarity between words and a word order similarity. The string similarity is used to capture minute differences like misspelled words and it was calculated using a modified longest common subsequence algorithm. Semantic similarity between words is calculated using a method inspired by point-wise mutual information (PMI) using the British National Corpus. The word order similarity is calculated using the normalised difference of common-word order between the two sentences. The word order similarity is used to incorporate syntactic information.

The overall sentence similarity is calculated as follows. Suppose two sentences having m and n words ($m \geq n$) with c tokens in common. The algorithm creates a joint matrix of $(m - c) \times (n - c)$ size using the string similarity matrix and semantic word similarity matrix. The string similarity

and semantic word similarity matrices are created by calculating the respective similarity measures among the uncommon words between the two sentences ($(m - c)$ and $(n - c)$ words). The algorithm selects the biggest positive element (> 0) and stores it in ρ vector and removes the corresponding row and column elements from the matrix. The process proceeds until either the matrix contains no positive element or the matrix is empty. The whole sentence similarity between sentences P and R is computed as follows:

$$S(P, R) = \frac{(c(1 - w_f + w_f S_0) + \sum_{i=1}^{|\rho|} \rho_i) \times (m + n)}{2mn}$$

In the equation above, S_0 is the common word order similarity score and w_f is the weight deciding the contribution of the common word order similarity score. The evaluation was carried out on two different datasets; the Microsoft paraphrase corpus (Dolan et al., 2004) and the dataset created by Li et al. (2006), which contains 30 sentence pairs annotated with similarity scores. The system performed reasonably well with a Pearson correlation coefficient of 0.853 compared to 0.816 Pearson correlation coefficient obtained by Li et al. (2006) on the same dataset. On the test set from the Microsoft paraphrase corpus having 1725 sentence pairs, the method obtains 81.3 F-measure.

2.4.2 Edit-distance Based Approaches

Tree edit-distance is one of the prominent measures used to calculate similarity and textual entailment between two sentences. Basic tree edit-distance signifies the minimum cost of transforming a source sentence tree into a target sentence

tree. Transformation is done by a sequence of edit operations on nodes and the cost is calculated by summing the associated cost with each of these operations. Bille (2005) surveyed various tree edit-distance metrics. The standard algorithm uses a dynamic programming technique (Zhang and Shasha, 1989). The basic tree edit-distance calculation involves three operations: insertion of a node; deletion of a node; and substitution of a node. The three operations are given below:

1. Substitution (or Change): This operation involves relabeling a node.

2. Deletion (or Remove): This operation involves deletion of a node. All children of the removed node are assigned to the parent node.

3. Insertion: This operation involves insertion of a new node. Insertion of a node A under the parent node B, makes A the parent of a consecutive subsequence of the children of B.

Tree edit-distance is extensively used in computing textual entailment. Heilman and Smith (2010) extend tree edit-distance by adding more operations. The method implements nine operations involving edges, sub-trees, siblings and root. The authors presented a greedy best first search to find the edit sequence which may not be the minimal edit sequence. They used the Microsoft paraphrase corpus to train a classifier using the edit sequences. They employed tree edit models for three tasks: textual entailment, paraphrasing, and an answer selection task for a question answering system. The method obtained 61.8% accuracy on the textual entailment task, 73.2% accuracy for the paraphrase identification task and 0.6091

mean average precision and 0.6917 mean reciprocal rank for the question selection task.

Recently, Alabbas and Ramsay (2013) presented a modified tree edit-distance approach, which extends tree edit-distance to the level of subtrees. The approach extends Zhang-Shasha's algorithm (Zhang and Shasha, 1989) by allowing costs of operations that insert, delete, and exchange subtrees. The costs of operations were derived by an appropriate function of the costs of the operations on their parts. This method was used to check entailment between two Arabic text snippets. The method obtained 0.636 F-score compared to 0.578 F-score using a bag-of-words approach and 0.597 F-score using Levenshtein edit-distance.

Wang and Cer (2012) presented an approach that uses probabilistic edit-distance as a measure of semantic textual similarity. The approach uses probabilistic finite state automata and pushdown automata to model weighted-edit-distance where state transitions correspond to edit-operations. The system obtained an overall Pearson correlation coefficient of 0.5589, which was calculated using all datasets (and models) of the task.

2.4.3 Combining Various Similarity Measures

In recent Semantic Textual Similarity (STS) tasks (Agirre et al., 2012, 2013, 2014, 2015) various approaches are proposed which combine different text similarity scores.[6] Bär et al. (2012) and Marsi et al. (2013) presented approaches which require a corpus where parallel sentences are annotated with similarity scores.

[6] http://ixa2.si.ehu.es/stswiki/index.php/Main_Page

A general approach is to extract features from parallel sentences using various similarity measures and train a supervised machine learning system to predict the similarity between the sentences. Bär et al. (2012) participated in the STS task at SemEval-2012 (Agirre et al., 2012). The system obtained an overall Pearson correlation coefficient of 0.823 with the human annotated dataset used in the task. Marsi et al. (2013) extend the system by Bär et al. (2012) with additional features. The semantic textual similarity system (Gupta et al., 2014) used in our SVM based approach presented in Chapter 4 is similar to the systems by Bär et al. (2012) and Marsi et al. (2013).

2.5 Deep Learning for Semantic Similarity

Deep Learning is a state-of-the art technique in machine learning. Deep learning is a family of methods used to capture an abstract representation of data and have turned out to be successful in many NLP applications such as language modelling (Bengio et al., 2003), paraphrasing (Mikolov et al., 2013c; Socher et al., 2011a), sentiment analysis (Socher et al., 2013b; Tai et al., 2015), parsing (Socher et al., 2013a), machine translation (Mikolov et al., 2013b; Hermann and Blunsom, 2014; Bahdanau et al., 2014), machine translation evaluation (Gupta et al., 2015b,a) and semantic similarity between segments (Socher et al., 2011a) and documents (Le and Mikolov, 2014).

One of the advantages of these approaches is that features are extracted automatically and do not need manually designed features for training. Training can be performed using a large amount of data automatically without the need

of human experts. For example, constructing a WordNet for a language may require several years of human experts' time and effort. Dense vector space representations such as those obtained through Deep Neural Networks (DNNs) are able to capture semantic similarity between words (Mikolov et al., 2013c; Pennington et al., 2014), segments (Sutskever et al., 2014; Socher et al., 2011a) and documents (Le and Mikolov, 2014) naturally.

Recent deep learning developments are an extension of earlier neural networks. Earlier neural networks can perform well for up to two layers (Erhan et al., 2009). As the number of layers increase, the representation gets difficult. Hinton et al. (2006) presented a breakthrough technique to train a deep architecture of multiple layers. They presented an approach based on Restrictive Boltzmann Machines (Smolensky, 1986) and contrastive divergence learning. The approach learns the lower layer first using the unsupervised data. The system can also be trained afterwards in a supervised fashion using training labels. Subsequently, there are other techniques developed in this area (Bengio, 2009). These techniques efficiently learn neural networks of more than two layers.

There are several approaches to get distributed representations of sequences. These distributed representations implicitly capture some similarity between sequences related to the task. For example for the machine translation task, similar sentences in two different languages will be close to each other and for the sentiment analysis task texts having similar sentiments will be close to each other.

A basic distributed representation of phrases, sentences or documents can

be obtained using simple algebraic operations over word vectors (Mitchell and Lapata, 2008; Turney, 2012; Erk, 2012). Other approaches generally involve neural networks (Collobert and Weston, 2008; Socher et al., 2011a, 2013b; Blunsom et al., 2014; Tai et al., 2015) to get the phrase, sentence or document representations. Composition using neural networks generally obtains better performance than simple algebraic operations. For example, for sentiment prediction at the sentence level, Socher et al. (2013b) obtained 45.7% accuracy using a neural network based approach to compose sentence vectors and 32.7% accuracy using the average of word vectors as the sentence vector.

Neural network architectures which allow representation of variable length sequences can be useful for our research in translation memory matching and retrieval. A translation memory segment can be viewed as a sequence of words or characters. TM segments are of variable lengths so we need an architecture, which has ability to model variable length sequences.

Recurrent Neural Networks (RNNs) are one of the most widely used network architectures to model variable length sequences. RNNs are efficient and proved effective in representing a *sequence* for the many NLP tasks (Tai et al., 2015; Bahdanau et al., 2014; Luong et al., 2015). A *sequence* may refer to a question or a document in the question answering task, a hypothesis or a reference in the machine translation task, a sequence of words or characters in the language modelling task. For example, Tai et al. (2015) proposed tree-structured Long Short Term Memory (LSTM) networks, which are types of RNNs that use dependency and constituency parsing to model sentences for sentiment analysis

and semantic similarity prediction.

Recursive neural networks (Socher et al., 2011a, 2013b, 2011b) are also generally used to model sentences. They use tree structure of a sentence to compose representation of the sentence. A tree can be a constituency parse tree, a dependency parse tree or a binary tree. Socher et al. (2011a) modelled sentences using a recursive neural network based on the auto-encoder-decoder approach and predicted whether two sentences are paraphrases of each other using Euclidean distance. RNNs proposed by Tai et al. (2015) can also be classified as recursive neural networks as they use parse tree to model sentence representation.

In our research presented in Chapter 4, we further explain RNNs and tree-structured Long-Short-Term-Memory (LSTM) networks. We use these networks to improve translation memory matching and retrieval.

As we mentioned earlier, distributed representations of sequences are often obtained using word vectors. This is particularly helpful for the semantic similarity between segments. Generally, there is not enough data available to learn every semantic aspect from the labelled training data. Furthermore, word vectors themselves capture similarity between words. Therefore, even using a small amount of labelled training data, we can train a system which models sentences and predicts the similarity. This aspect is explored in Chapter 4.

Rumelhart et al. (1986) performed some of the earlier work to learn word vectors. The approach was slow, however. Recent advances in training word vectors (Mikolov et al., 2013c; Pennington et al., 2014) made a significant step towards learning word vectors efficiently.

Mikolov et al. (2013c) presented efficient techniques for learning distributed word vectors called Skip-gram and Continuous Bag of Words (CBOW) models. In the Skip-gram model, the training objective is to obtain word vectors that are good at predicting the context words. For a given sequence of words $w_1, w_2, w_3, ..., w_N$ used for training, the objective is to maximize the average log probability as follows:

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{-c\leq j\leq +c, j\neq 0}\log p(w_{n+j}|w_n) \tag{2.7}$$

where c is the size of training context.

In the CBOW model, the training objective is similar but instead of learning to predict context words, the model learns to predict a word given its context. The size of context generally varies between 5 to 30 words. An extension of the Skip-gram model called GloVe is proposed by Pennington et al. (2014). GloVe also use a co-occurrence matrix computed using a training corpus to get more information from the global context. We use GloVe word vectors for our work in Chapter 4.

2.6 Conclusion

In this chapter we presented related research in TM matching and retrieval. We also covered related areas that can benefit TM matching and retrieval. In translation memory matching and retrieval, most of the research focuses on string based similarity measures or some variation involving stemming, stop words removal or parts of speech tagging (Planas and Furuse, 1999; Hodász and Pohl, 2005; Jaworski, 2013). Some research focuses on deep linguistic processing, like parsing (Pekar and Mitkov, 2007; Mitkov, 2008) but it is not clear whether the

approaches are scalable. The other approach (Utiyama et al., 2011) incorporates semantic information in the form of paraphrasing but was limited to retrieving exact matches only.

Some machine translation evaluation techniques (Denkowski and Lavie, 2014; Snover et al., 2008) also include paraphrasing to match the reference with the translation but they do not implement edit-distance, which is widely used in translation memory matching and retrieval. Other related areas we covered are "semantic similarity and textual entailment using traditional approaches" and "deep learning for semantic similarity". Traditional approaches typically compute similarity between sentences using: a function over words present in sentences and their positions (Li et al., 2006; Tsatsaronis et al., 2010; Islam and Inkpen, 2008; Gu et al., 2012); edit-distance between the sentences (Bille, 2005; Heilman and Smith, 2010; Wang and Cer, 2012); or machine learning based approaches (Bär et al., 2012; Marsi et al., 2013; Gupta et al., 2014).

In Section 2.5, we presented related research in deep learning. Deep learning approaches have recently been developed and can successfully be applied to compute semantic similarity between sentences using distributive representations of sentences (Tai et al., 2015; Socher et al., 2011a).

CHAPTER 3

IMPROVING MATCHING AND RETRIEVAL USING PARAPHRASES

3.1 Introduction

Apart from retrieving exact matches, one of the core features of a TM system is the retrieval of previously translated similar segments for post-editing in order to avoid translation from scratch when an exact match is not available. However, this matching and retrieval process is generally limited to edit-distance based measures operating on surface form (or sometimes stem) matching. Most commercial systems use edit-distance (Levenshtein, 1966) or some variation of it, e.g. the open-source TM OmegaT[1] uses word-based Levenshtein edit-distance with some extra preprocessing. The preprocessing typically involves tokenisation, removing punctuation, removing stop words and stemming. Although these measures provide a strong baseline, they are not sufficient to capture semantic similarity between segments as judged by humans. This may result in uneven post-editing time by translators for the same fuzzy match scored segments and non-retrieval of semantically similar segments. For example, even though segments like *the period laid down in article 4(3)* and *the duration set forth in article 4(3)* have the same

[1]OmegaT is an open source TM available form http://www.omegat.org.

meaning, one segment may not be retrieved for the other in current TM systems as they have only 57% similarity when using word based Levenshtein edit-distance as implemented in OmegaT, even though one segment is a paraphrase of the other segment. To mitigate this limitation of TM, we propose two novel and efficient approaches to incorporating paraphrasing in TM matching without compromising the ease and flexibility of edit-distance which has been trusted by TM developers, translators and translation service providers over the years.

Section 3.2 presents both approaches, Sections 3.3 and 3.4 evaluates each of them.

3.2 Our Approach

A trivial approach to implementing paraphrasing along with edit-distance is to generate all the additional segments based on the paraphrases available and store these additional segments in the TM. This approach leads to combinatorial explosion and is highly inefficient both in terms of time necessary to perform matching and storage space for the extra segments generated. For a TM segment which has n different phrases where each phrase can be paraphrased in m more possible ways, we get $(m + 1)^n - 1$ additional segments (without considering that these phrases may contain paraphrases as well). For example, a TM segment which has four different phrases where each phrase can be paraphrased in five more possible ways, we get 1295 (6^4 -1) additional segments to store in the TM, which is inefficient even for small TMs. To handle this problem efficiently, we use dynamic programming and greedy approximation techniques.

Using our approaches, the fuzzy match score between segments can be calculated in polynomial time despite the inclusion of paraphrases. For example, if the translation memory used has a segment *What is the actual aim of this practice ?* and the paraphrase database has paraphrases *the actual* $\Rightarrow$ *the real* and *aim of this* $\Rightarrow$*goal of this*, for the input sentence *What is the real goal of this mission ?*, our approaches will give a 89.89% fuzzy match score (only one word, *practice*, needs substitution with *mission*) rather than 66.66% using simple word-based edit-distance.

3.2.1 Paraphrase Corpus

We have used the PPDB 1.0 paraphrases database (Ganitkevitch et al., 2013) for our work. This database contains lexical, phrasal and syntactic paraphrases automatically extracted using a large collection of parallel corpora, which include Europarl v7 (Koehn, 2005), the 10^9 French-English corpus (Callison-Burch et al., 2009), the Czech, German, Spanish and French portions of the News Commentary data (Koehn and Schroeder, 2007), the United Nations French- and Spanish-English parallel corpora (Eisele and Chen, 2010), the JRC Acquis corpus (Steinberger et al., 2006), Chinese and Arabic newswire corpora used for the GALE machine translation campaign,[2] parallel Urdu-English data from the NIST translation task,[3] the French portion of the Open Subtitles corpus (Tiedemann, 2009), and a collection of Spanish-English translation memories from TAUS[4].

[2]http://projects.ldc.upenn.edu/gale/data/Catalog.html
[3]LDC Catalog No. LDC2010T23
[4]http://www.translationautomation.com/

The paraphrases in this database are constructed using a bilingual pivoting method. It is based on the hypothesis that if two different English language phrases are translated to an identical foreign language phrase, the two English phrases are paraphrases of each other. Because of the automatic extraction, not all the paraphrases are accurate. This database contains syntactic, lexical and phrasal paraphrases. We use lexical and phrasal paraphrases. Some examples of lexical and phrasal paraphrases are given in Section 3.2.2.

The paraphrase database is available in six sizes (S, M, L, XL, XXL, XXXL) where S is the smallest and XXXL is the largest. The smaller packages contain only high precision paraphrases, while the larger ones aim to provide more coverage. The smallest package (S) contains 0.6 million while the largest package (XXXL) contains 68 million lexical and phrasal paraphrases. We have used lexical and phrasal paraphrases of L size, which contains 3 million paraphrases. The reason for choosing L size was to retain the quality of segments retrieved using paraphrasing and at the same time gain some coverage. Our empirical analysis suggested that paraphrases that have punctuation and numbers are noisy. Therefore, we removed paraphrases with punctuation and numbers and retained the remaining 2 million paraphrases for our work. The average, maximum and minimum lengths (numbers of words) of a paraphrase are 2.8, 6 and 1, respectively.

3.2.2 Classification of Paraphrases

We have classified paraphrases obtained from PPDB 1.0 into four types on the basis of the number of words in the source and target phrases. This was necessary in order to have efficient implementations. The four types are:

1. Paraphrases having one word in both the source and target sides, e.g. *period* ⇔*duration*

2. Paraphrases having multiple words on both sides but differing in one word only, e.g. *in the period* ⇔ *during the period*

3. Paraphrases having multiple words, but the same number of words on both sides, e.g. *laid down in article* ⇔ *set forth in article*

4. Paraphrases in which the number of words in the source and target sides differs, e.g. *a reasonable period of time to* ⇔ *a reasonable period to*

We store only the smallest substring which can capture all non matching words instead of the whole paraphrase. In other words, we remove matching words from both sides of strings. An example describing the reduction process is given in Table 3.1. The paraphrases are reduced before computing edit-distance because after capturing paraphrases for a particular segment we have already considered the context, and there is no need for it to be considered again while calculating edit-distance. Therefore, after creating a paraphrasing lattice for a TM segment, we reduce the lattice. This reduction step is a vital step to improve the efficiency of both our approaches.

Type	Paraphrases	Reduced form
Type 1	period ⇔ duration	period ⇔ duration
Type 2	in the period ⇔ during the period	in ⇔ during
Type 3	laid down in article ⇔ set forth in article	laid down ⇔ set forth
Type 4	a reasonable period of time to ⇔ a reasonable period to	of time to ⇔ to

Table 3.1: An Example Describing the Paraphrase Reduction Process

In our classification, Type 1 are one-word paraphrases and Type 2 can be reduced to one-word paraphrases after considering the context when storing in the TM lattice. Paraphrases of Type 3 and Type 4 remain multiword paraphrases after reduction (see Table 3.1). Suppose we have paraphrases as given in Table 3.2, a TM segment and the TM lattice with paraphrases stored in the reduced form are given in Figure 3.1 ((i) TM Segment , (ii) TM Lattice with Paraphrases).

the period laid down in	the period referred to in
laid down in article	provided for by article
the period	the time
the period	the duration
in article	under article

Table 3.2: Paraphrases Used to Create TM Lattice in Figure 3.1 and Figure 3.2

3.2.3 Filtering

Before processing begins, several filtering steps are applied for each input segment. The purpose of this filtering process is to remove unnecessary candidates from participating in the paraphrasing process and speed up the processing. Our filtering steps for obtaining potential candidates for paraphrasing are as follows:

1. LENFILTER: We first filter out the segments based on length because if segments differ considerably in length, the edit-distance will also differ correspondingly. In our case, the threshold for length was 39%. TM segments are discarded if the TM segments are shorter than 39% of the input or vice-versa.

2. SIMFILTER: Next, we filter the segments based on baseline edit-distance similarity. TM segments which have a similarity below a certain threshold will be removed. In our case, the threshold was 39%.

3. MAXFILTER: Next, after filtering the candidates with the above two steps we sort the remaining segments in decreasing order of baseline edit-distance similarity and pick the top 100 segments.

4. BEAMFILTER: Finally segments within a certain range of similarity with the most similar segment were selected for paraphrasing. In our case, the range is 35%. This means that if the most similar segment has 95% similarity, segments with a similarity below 60% will be discarded.

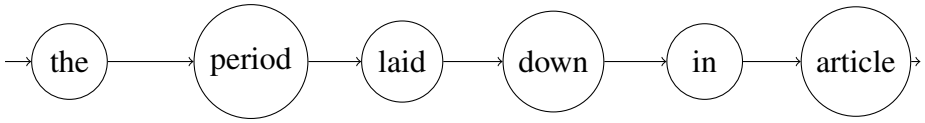
The detailed experiments are described in Gupta and Orăsan (2014) where the filtering thresholds were determined empirically by running the proposed method (given in Section 3.2.6) with various settings on the DGT-TM (Steinberger et al., 2012) English-French corpus. The values determined for the filtering thresholds in Gupta and Orăsan (2014) were, LENFILTER: 49%, SIMFILTER: 49%, MAXFILTER: 100 and BEAMFILTER: 35%. In the experiments that used the Europarl corpus, we observed that the number of retrieved segments is low.

Therefore, we decided to lower the LENFILTER and SIMFILTER to 39%. We kept the same filtering thresholds for all the experiments presented in this chapter.

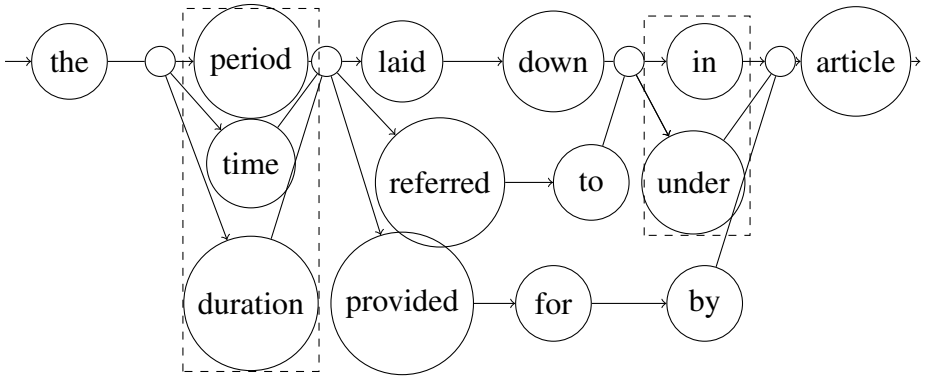
3.2.4 Matching Steps

There are two options for incorporating paraphrasing in a typical TM matching pipeline: paraphrase the input or paraphrase the TM. For our approach we have chosen to paraphrase the TM. There are a number of reasons for this. First, once a system is set up, the user can get the retrieved matches in real time; second, TMs can be stored on company servers and all processing can be done offline; third, the TM system does need not be installed on the user computer and can be provided as a service. However, we should emphasise that our method to paraphrase the TM does not generate all the possible segments. Instead this is achieved by our matching algorithms in an efficient way.

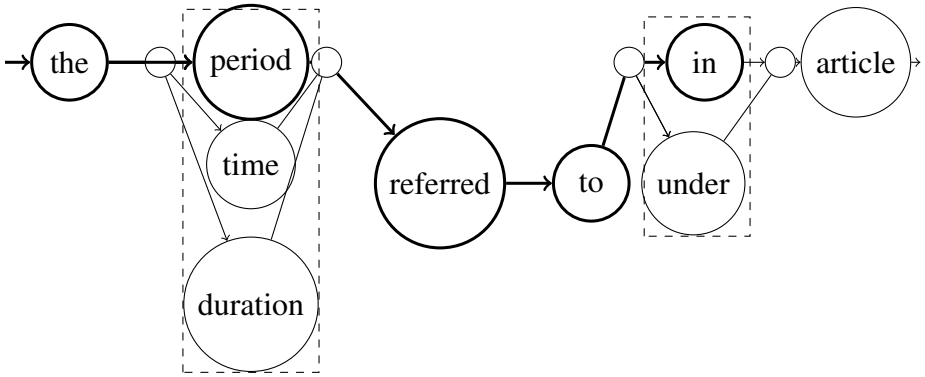
Paraphrasing the input has its own advantages. In general, input files are much smaller than TMs. Therefore, paraphrasing the input instead of the TM can save space.



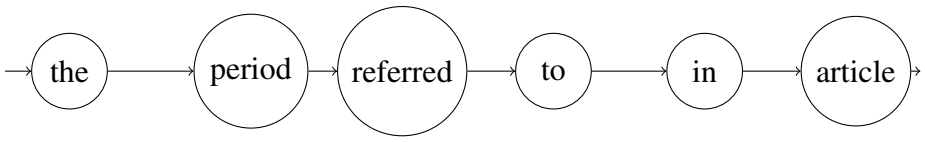
(i) TM Segment



(ii) TM Lattice with Paraphrases



(iii) TM Lattice after Edit-distance Calculation of First Five Tokens



(iv) Input Test Segment

Figure 3.1: (i) TM Segment, (ii) TM Lattice with Paraphrases (dashed area indicates Types 1 and 2 paraphrasing), (iii) TM Lattice after Edit-distance Calculation of First Five Tokens using DPGA (in bold), (iv) Input Test Segment

Our approach is based on the following steps:

1. Read the Translation Memories available
2. Classify paraphrases according to the four types presented in Section 3.2.2 and store paraphrase lattice for each segment in the TM in the reduced form
3. Read the file that needs to be translated
4. For each segment in the input file
   (a) retrieve the potential segments for paraphrasing in the TM according to the filtering steps of Section 3.2.3
   (b) search for the most similar segment based on Algorithm 4 or Algorithm 5 given in Section 3.2.6 and Section 3.2.7, respectively
   (c) retrieve the most similar segment if it is above a predefined threshold

3.2.5 Edit-distance Computation

For our implementation, we use basic Levenshtein (1966) edit-distance, which is a word-based edit-distance with cost 1 for insertion, deletion and substitution. Algorithm 1 describes the basic edit-distance procedure. The similarity is calculated by normalising edit-distance with the length of the larger segment.

We have employed this edit-distance as a baseline and adapted it to incorporate paraphrasing. When edit-distance is calculated, matching of paraphrases of Types 1 and 2 can be implemented in a more efficient manner compared to paraphrases of Types 3 and 4. The paraphrases of the Types 1 and 2 have a unique property that

Algorithm 1 Basic Edit-distance

procedure EDIT-DISTANCE(*InputSegment*,*TMS*)
  $M \leftarrow$ length of *TMS* ▷ Initialise M with length of TM segment
  $N \leftarrow$ length of *InputSegment* ▷ Initialise N with length of Input segment
  $D[i, 0] \leftarrow i$ for $0 \leq i \leq N$ ▷ initialisation
  $D[0, j] \leftarrow j$ for $0 \leq j \leq M$ ▷ initialisation
  for $j \leftarrow 1...M$ **do**
    TMToken $\leftarrow$ *TMS*$_j$ ▷ get Token of TM segment
    for $i \leftarrow 1...N$ **do**
      InputToken $\leftarrow$ *InputSegment*$_i$ ▷ get Token of Input segment
      $cost \leftarrow$ GETCOST(*InputToken*, *TMToken*) ▷ GETCOST procedure is defined in Algorithm 2
      $D[i, j] \leftarrow minimum(D[i-1, j]+1, D[i, j-1]+1, D[i-1, j-1]+cost)$ ▷ store minimum of insertion, substitution and deletion
    end for
  end for
  Return $D[N, M]$ ▷ Return minimum edit-distance
end procedure

Algorithm 2 Compute cost for basic edit-distance

procedure GETCOST(*InputToken*,*TMToken*)
  $cost \leftarrow 1$ ▷ Substitution cost if not matches
  if *InputToken* $=$ *TMToken* **then** ▷ match *InputToken* with *TMToken*
    $cost \leftarrow 0$ ▷ Substitution cost if matches
  end if
  Return $cost$ ▷ Return minimum edit-distance
end procedure

Algorithm 3 Compute cost using Type 1 and Type 2 paraphrases

procedure GETCOSTPARAPHRASE12(*InputToken*,*TMToken*)
  $cost \leftarrow 1$
  $OneWordPP \leftarrow$ get Type 1 and Type 2 paraphrases associated with *TMToken* including *TMToken* itself ▷ get one word paraphrases
  if *InputToken* $\in OneWordPP$ **then** ▷ applying type 1 and type 2 paraphrasing
    $cost \leftarrow 0$
  end if
  Return $cost$ ▷ Return minimum edit-distance
end procedure

they can be reduced to single word paraphrases by removing the other matching words.

The basic procedure as given in Algorithm 1 works by comparing each token one by one in the input segment with each token in the TM segment. This procedure makes use of previous edit-distance computations to optimise the edit-distance globally (for the whole sentence). To implement the paraphrases of Types 1 and 2, we can extend this procedure by searching (a successful search indicates a match) in a list of paraphrases (reduced single tokens) associated with the TM token in addition to comparing the TM token with the input token.

For the example given in Figure 3.1, if a word from the input segment matches any of the words *period, time or duration* (see Figure 3.1(ii) for TM lattice), the cost of substitution will be 0. The basic edit-distance procedure can be extended to incorporate Type 1 and Type 2 paraphrases by using a new GETCOSTPARAPHRASE12 procedure given in Algorithm 3 instead of GETCOST procedure of Algorithm 2. This procedure has the advantage that the complexity of the algorithm is only increased by the additional searching in the list ($log(n)$, where n is the number of Type 1 and Type 2 paraphrases associated with the TM token). For these two types, the edit-distance procedure is optimised globally as this is a simple case of matching one of these paraphrases when calculating the cost of substitution.

For Types 3 and 4, we propose two techniques: Using both dynamic programming and greedy approximation (DPGA) and using dynamic programming only (DP). There is no difference in both approaches if we only consider Type 1 and

Type 2 paraphrases. However, both techniques significantly differ when we use Type 3 and Type 4 paraphrases as well. We will elaborate on the difference later in this chapter. The DPGA procedure is described in Section 3.2.6 and the DP procedure is described in Section 3.2.7.

3.2.6 Dynamic Programming and Greedy Approximation (DPGA)

In this approach, both dynamic programming and greedy approximation techniques are used. Similarity is calculated with the potential segments for paraphrasing extracted as per Section 3.2.3.

The basic edit-distance calculation procedure is given in Algorithm 1. The algorithm implementing the DPGA approach is given in Algorithm 4. In Algorithm 4, *InputSegment* is the segment that we want to translate and *TMLattice* is the TM lattice with paraphrases.

In Algorithm 4, *lines 11 to 16* execute when Type 3 and Type 4 paraphrases are not available (e.g. edit-distance calculation of the second token "period"). Table 3.3 illustrates the edit-distance calculation of the first five tokens of the Input and TM segment with paraphrasing of the example given in Figure 3.1 (Figure 3.1(iv) shows the input test segment and Figure 3.1(ii) shows the TM lattice). The second column of the table represents the input segment and the second row represents the TM segment along with the paraphrases. In Table 3.3, if a word from the input segment matches any of the words "period", "time" or "duration", the cost of substitution will be 0.

Algorithm 4 Edit-distance with Paraphrasing Procedure

procedure EDIT-DISTANCEPP(*InputSegment*, *TMLattice*)
  $M \leftarrow$ length of *TM Segment* ▷ number of tokens in the TM segment
  $N \leftarrow$ length of *InputSegment* ▷ number of tokens in the Input segment
  $D[i, 0] \leftarrow i$ for $0 \leq i \leq N$ ▷ initialise two dimensional matrix D
  $D[0, j] \leftarrow j$ for $0 \leq j \leq (M + p')$ where p' accounts for increase in the TM segment length because of paraphrasing
  $decisionPoint \leftarrow 0$, $j \leftarrow 1$
  $cost \leftarrow 1$ ▷ initialisation of the substitution cost
  while $j \leq M$ **do**
    $TMToken \leftarrow$ *TMLattice*$_j$ ▷ getting current TM token to process, e.g. 3^{rd} token "laid"
    if there are no paraphrases of type 3 and type 4 starting from $TMToken$ **or** $decisionPoint \geq N$ **then**
      $decisionPoint \leftarrow decisionPoint + 1$, $j \leftarrow j + 1$
      for $i \leftarrow 1...N$ **do**
        $InputToken \leftarrow Input_i$
        $cost \leftarrow$ GETCOSTPARAPHRASE12$(InputToken, TMToken)$ ▷ GETCOSTPARAPHRASE12 procedure is defined in Algorithm 3
        $D[i, decisionPoint] \leftarrow minimum(D[i, decisionPoint - 1] + 1, D[i - 1, decisionPoint] + 1, D[i - 1, decisionPoint - 1] + cost)$
      end for
    else
      $prevDistance \leftarrow D[decisionPoint, decisionPoint]$
      $DP \leftarrow$ calculate edit-distance of each paraphrase and longest source phrase with $Input$ using D ▷ uses D for first word, consider Type 1 and Type 2 paraphrases for the source phrase
      selectedPhrase $\leftarrow$ select a minimum edit-distance paraphrase or a source phrase ▷ source phrase is preferred in case of a tie between a paraphrase and the corresponding source
      $curDistance \leftarrow$ edit-distance of the *selectedPhrase*
      if *selectedPhrase* is a paraphrase **then**
        $j \leftarrow j +$ length of the source phrase corresponding to *selectedPhrase*
        $decisionPoint \leftarrow decisionPoint +$ length of *selectedPhrase*
        update D using DP
      else if *selectedPhrase* is a source phrase **and** $curDistance = prevDistance$ **then** ▷ $true$ if the source phrase is exactly matching
        $j \leftarrow j +$ length of *selectedPhrase*
        $decisionPoint \leftarrow decisionPoint +$ length of *selectedPhrase*
        update D using DP
      else
        $j \leftarrow j + 1$, $decisionPoint \leftarrow decisionPoint + 1$
        update D using DP
      end if
    end if
  end while Return $D[N, decisionPoint]$
end procedure

	j	0	1	2					3	4				5
i		#	the	period duration time	laid	down	in under		referred	to	provided	for	by	in under
0	#	0	1	2	3	4	5		3	4	3	4	5	5
1	the	1	0	1	2	3	4		2	3	2	3	4	4
2	period	2	1	0	1	2	3		1	2	1	2	3	3
3	referred	3	2	1	1	2	3		0	1	1	2	3	2
4	to	4	3	2	2	**2**	3		1	**0**	2	2	3	1
5	in	5	4	3	3	3	**2**		2	1	3	3	**3**	0

Table 3.3: Edit-distance Calculation using DPGA

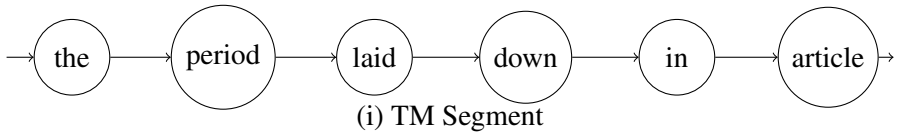
In Algorithm 4, *lines 18 to 32* account for the case when Type 3 and Type 4 paraphrases are available. For paraphrases of Types 3 and 4 the algorithm takes the decision locally at the point where all paraphrases finish. *Lines 23, 27 and 31* account for updating the value of j to reflect the current position for further calculation of edit-distance (e.g. $j = 5$ after selecting "referred to") and *lines 25, 29 and 32* update the matrix D.

As we can see in Table 3.3, starting from the third token of the TM, "laid", three separate edit-distances are calculated, two for the two paraphrases "referred to" and "provided for by" and one for the corresponding longest source phrase "laid down in", and the paraphrase "referred to" is selected as it gives a minimum edit-distance of 0. The last column of Table 3.3 ($j = 5$) shows the edit-distance calculation of the next token "in" (and paraphrase "under") after selecting "referred to". Because the algorithm has selected "referred to" as the preferred paraphrase and "referred to" is a paraphrase of "laid down", we will update the edit-distance values of column $j = 5$ using "to" ($j = 4$) as a previous column.

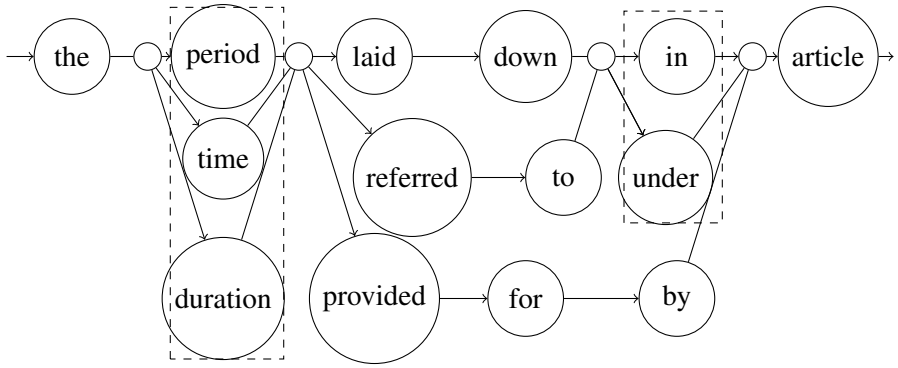
Figure 3.1 (iii) shows the preferred path in bold after considering paraphrases.

3.2.7 Dynamic Programming Only (DP)

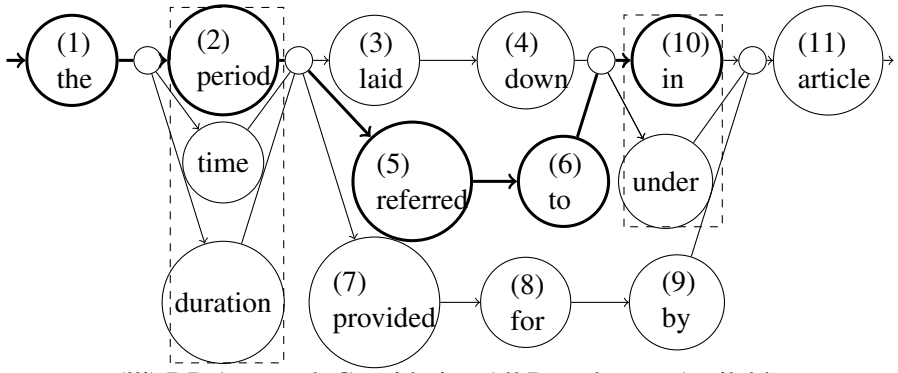
In the previous section (Section 3.2.6), we presented the approach based on greedy approximation and dynamic programming. In this section we will present another approach based on dynamic programming only. We will further discuss the difference in the both approaches.



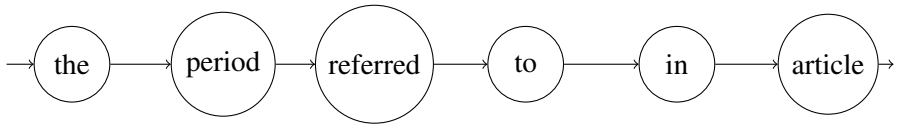
(i) TM Segment



(ii) TM Lattice with Paraphrases



(iii) DP Approach Considering All Paraphrases Available



(iv) Input Test Segment

Figure 3.2: (i) TM Segment, (ii) TM Lattice with Paraphrases (dashed area indicates Types 1 and 2 paraphrasing), (iii) DP Approach Considering All Paraphrases Available (numbers show values of j as given in Table 3.4), (iv) Input Test Segment

Algorithm 5 Edit-distance with Paraphrasing using Dynamic Programming

```
1: procedure EDIT-DISTANCE(InputSegment,TMLattice)
2:     Initialise D                                              ▷ a two dimensional array
3:     for j ← 1...number of nodes in TMLattice do
4:         for i ← 1...number of tokens in a InputSegment do
5:             if TMLattice_j is a TM token then                 ▷ condition to avoid paraphrasing the paraphrase
6:                 cost ← GETCOSTPARAPHRASE12(InputSegment_i, TMLattice_j)
7:             else
8:                 cost ← GETCOST(InputSegment_i, TMLattice_j)
9:             end if
10:            P ← previous indices of paths at TMLattice_j   ▷ get indices of previous nodes connecting to the present node TMLattice_j
11:            D[i, j] ← minimum(D[i − 1, j] + 1, D[i, k] + 1, D[i − 1, k] + cost for all k ∈ P)   ▷ store minimum edit-distance path by considering insertion, deletion or substitution for all paths
12:        end for
13:    end for
14:    m ← index of minimum edit-distance path at last node
15:    N ← length of InputSegment                                ▷ number of tokens in a input segment
16:    Return D[m, N]                                            ▷ Return minimum edit-distance
17: end procedure
```

In the Dynamic Programming (DP) approach, at every step we consider the best matching path. Table 3.4 illustrates the edit-distance calculation of the first five tokens of the Input and TM segment with paraphrasing of the example given in Figure 3.2. In Table 3.4, the second column represents the input segment and the second row represents the TM segment along with the paraphrases. Figure 3.2 (iv) shows the input test segment and Figure 3.2 (ii) shows the TM lattice. The basic edit-distance calculation procedure is given in Algorithm 1.

The DP algorithm is given in Algorithm 5. *InputSegment* is the segment to be translated and *TMLattice* is the TM lattice (TM segment with paraphrases). As in Table 3.3, in Table 3.4, if a word from the input segment matches any of the words "period", "time" or "duration", the cost of substitution will be 0.

The DP algorithm considers the minimum edit-distance path at every step. In Algorithm 5, *line 10* gets the previous indices at every step. For example, in Table 3.4, when executing the token 'in' the algorithm will consider previous indices of

	j	0	1	2	3	4	5	6	10	7	8	9	11
i		#	the	period duration time	laid	down	referred	to	in under	provided	for	by	article
0	#	0	1	2	3	4	3	4	5	3	4	5	6
1	the	1	0	1	2	3	2	3	4	2	3	4	5
2	period	2	1	0	1	2	1	2	3	1	2	3	4
3	referred	3	2	1	1	2	0	1	2	1	2	3	3
4	to	4	3	2	2	2	1	0	1	2	2	3	2
5	in	5	4	3	3	3	2	1	0	3	3	3	1
6	article	6	5	4	4	4	3	2	1	4	4	4	0

Table 3.4: Edit-distance Calculation using DP

'down' and 'to' (see lattice in Figure 3.2 (ii)) and store the minimum edit-distance path in *line 11*.

Table 3.4 shows that the column $j = 10$ is updated after considering both column $j = 4$ and $j = 6$ and column $j = 11$ is updated after considering column $j = 10$ and $j = 9$. In contrast to the DP approach, the DPGA approach described in Section 3.2.6 makes a greedy selection for Type 3 or Type 4 paraphrases. The DPGA approach will update only on the basis of the selected paraphrase as shown in Figure 3.1, whereas the DP approach will consider all connecting paths as shown in Figure 3.2.

Figure 3.2(iii) shows the preferred path in *bold* after considering paraphrases. Figure 3.2(iii) also shows that the DP approach retains all paraphrases for further edit-distance calculation; instead of retaining the best path in the DPGA approach (given in Section 3.2.6).

3.2.8 Computational Considerations

The time complexity of the basic edit-distance procedure is $O(m\,n)$ where m and n are lengths of source and target segments, respectively. After employing paraphrases of Type 1 and Type 2 the complexity increases to $O(m\,n\,log(p))$, where p is the number of paraphrases of Type 1 and Type 2 per token of TM segment. When employing paraphrases of Type 3 and Type 4, the complexity of the DP approach increases to $O(m\,n(log(p) + q\,l))$ and the complexity of the DPGA approach increases to $O(l\,m\,n(log(p)+q))$, where q is the number of Type 3 and Type 4 paraphrases stored per token and l is the average length of a Type 3

and Type 4 paraphrase.

The complexity of the DPGA approach is $O(l\,m\,n(log(p) + q))$, which is slightly higher than the $O(m\,n(log(p) + q\,l))$ complexity of the DP approach. However, in practice we have not observed much difference in the speed of both approaches.

If we consider Type 1 and Type 2 paraphrases in the same manner (comparing sequentially instead of searching in a list) as Type 3 and Type 4, the time complexity of the DP approach can simply be written as $O(k\,n)$, where k is the number of nodes in the TM lattice and n is the number of nodes in the input segment.

3.3 Evaluation on the Europarl Corpus

In TM, the performance of retrieval from a TM can be measured by counting the number of segments or words retrieved. However, NLP techniques are not 100% accurate and most of the time, there is a tradeoff between the quality of the retrieved segments and the number of segments retrieved. This is also one of the reasons why TM developers shy away from using semantic matching. One cannot measure the gain unless retrieval benefits the translator.

When we use paraphrasing in the matching and retrieval process, the fuzzy match score of a paraphrased segment is increased, which results in the retrieval of more segments at a particular threshold. This increment in retrieval can be classified into two types: without changing the top rank and by changing the top rank. For example, for a particular input segment, we have two segments: A and B

in the TM. Using simple edit-distance, A has a 65% and B has a 60% fuzzy score; the fuzzy score of A is better than that of B. As a result of using paraphrasing we notice two types of score changes:

1. the score of A is still better than or equal to that of B, for example, A has 85% and B has 70% fuzzy score;

2. the score of A is less than that of B, for example, A has 75% and B has 80% fuzzy score.

In the first case, paraphrasing does not supersede the existing model and just facilitates it by improving the fuzzy score so that the top segment ranked using edit-distance gets retrieved. However, in the second case, paraphrasing changes the ranking and now the top-ranked segment is different. In this case, the paraphrasing model supersedes the existing simple edit-distance model. This second case also gives a different reference with which to compare. In the experiments reported below, we take the top segment retrieved using simple edit-distance and the top segment retrieved using paraphrasing and compare to see which one is better. The next section (Section 3.3.1) describes our choice of corpus for experiments. In Section 3.3.2 we present the results using automatic evaluation. In Section 3.3.3, we present the settings for human evaluation and in Section 3.3.4, we present the results of the human evaluation.

3.3.1 Corpus Used

As a TM and test data, we have used English-German pairs of the Europarl V7.0 (Koehn, 2005) corpus with English as the source language and German as the target language. From this corpus we have filtered out segments of fewer than seven words and greater than 40 words to create the TM and test datasets. In our experiments, we have not paraphrased any capitalised words (but we lowercase them for both baseline and paraphrasing similarity calculation). This is to avoid paraphrasing any named entities. Table 3.5 shows our corpus statistics. Tokenization of the English data was done using the Berkeley Tokenizer (Petrov et al., 2006).

	TM	**Test Set**
Segments	1565194	9981
Source words	37824634	240916
Target words	36267909	230620

Table 3.5: Corpus Statistics

3.3.2 Results of Automatic Evaluations of Our Approaches (DPGA and DP)

This section presents the results using automatic evaluation. We compare the retrieval of both paraphrasing approaches with the baseline edit-distance. In automatic evaluation, we used the BLEU and Meteor machine translation evaluation metrics to check the quality of the retrieved segments. Table 3.6 presents the interval-wise results and Table 3.7 presents the results using a cutoff

threshold.

Tables 3.6 and 3.7 show similarity thresholds (TH) for TM, the total number of segments retrieved using the baseline approach (EditRetrieved), the additional number of segments retrieved using the paraphrasing approach (+ParaRetrieved), the percentage improvement in retrieval obtained over the baseline (%Improve), and the number of segments that changed their ranking and rose to the top because of paraphrasing (RankCh). BLEU-ParaRankCh and METEOR-ParaRankCh represent the BLEU score and the Meteor score over translations retrieved by our approach for segments which changed their ranking and come up to the top because of paraphrasing. BLEU-EditRankCh and METEOR-EditRankCh represent the BLEU score and the Meteor score on corresponding translations retrieved by the baseline approach. DP and DPGA represent the results of the DP and DPGA approaches, respectively.

Table 3.6 also shows BLEU-ParaAll and METEOR-ParaAll which represent the BLEU score and the Meteor score of the additional number of segments retrieved using our paraphrasing approach, and BLEU-EditAll and METEOR-EditAll which represent the BLEU score and the Meteor score on corresponding translations retrieved by our approach. The BLEU-ParaAll score is different from the BLEU-ParaRankCh score because BLEU-ParaAll also include the segments for which both edit-distance and our paraphrasing retrieve the same match. We compute BLEU-ParaAll and BLEU-EditAll, to see the overall difference in the BLEU score (or the Meteor score using METEOR-ParaAll and METEOR-EditAll) of additional segments retrieved using paraphrasing and the

corresponding top most match retrieved using edit-distance. Numbers in *bold* represents where our paraphrasing approach obtains better Meteor or BLEU than the baseline edit-distance.

	TH	**100**	**[85, 100)**	**[70, 85)**	**[55, 70)**
	EditRetrieved	117	98	226	704
DP	+ParaRetrieved	21	36	164	567
	%Improve	17.94	36.73	72.88	80.65
	RankCh	13	17	99	369
	BLEU-EditRankCh	24.55	14.87	9.14	5.51
	BLEU-ParaRankCh	**41.91**	**18.53**	8.43	**5.65**
	METEOR-EditRankCh	41.17	38.93	28.20	20.14
	METEOR-ParaRankCh	**59.25**	**39.52**	27.49	**21.68**
	BLEU-EditAll	24.19	28.44	14.40	8.27
	BLEU-ParaAll	**32.39**	**29.28**	14.19	**8.41**
	METEOR-EditAll	44.20	45.61	33.02	23.50
	METEOR-ParaAll	**53.24**	**45.79**	32.62	**24.41**
DPGA	+ParaRetrieved	17	29	97	311
	%Improve	14.52	29.59	42.92	44.17
	RankCh	10	13	54	202
	BLEU-EditRankCh	26.35	14.35	6.89	5.47
	BLEU-ParaRankCh	**51.56**	**15.46**	**8.45**	**5.83**
	METEOR-EditRankCh	43.40	35.13	25.96	19.99
	METEOR-ParaRankCh	**67.68**	**38.35**	**26.40**	**21.63**
	BLEU-EditAll	25.10	26.28	13.66	8.94
	BLEU-ParaAll	**37.31**	**26.45**	**14.32**	**9.18**
	METEOR-EditAll	44.94	43.88	31.63	23.83
	METEOR-ParaAll	**57.39**	**45.02**	**31.87**	**24.76**

Table 3.6: Results of Automatic Evaluation: Presented using Threshold Intervals

Tables 3.6 and 3.7 show that we obtain increase in retrieval when using paraphrases on each threshold level and on all the intervals. Table 3.6 shows that when using DPGA we obtain more than 14% increase in retrieval for exact matches and around 30% and 43% increase in the intervals [85, 100) and [70, 85),

	TH	100	95	90	85	80	75	70
	EditRetrieved	117	127	163	215	257	337	440
DP	+ParaRetrieved	21	22	30	42	66	115	170
	%Improve	17.94	17.32	18.40	19.53	25.68	34.12	38.64
	RankCh	13	14	21	32	49	95	155
	BLEU-EditRankCh	24.55	24.58	25.48	20.72	18.33	14.36	12.23
	BLEU-ParaRankCh	**41.91**	**39.89**	**35.89**	**28.84**	**21.66**	**16.56**	**13.24**
	METEOR-EditRankCh	41.17	42.32	42.03	40.19	38.09	34.73	31.53
	METEOR-ParaRankCh	**59.25**	**59.22**	**52.81**	**47.18**	**41.04**	**36.25**	**32.37**
DPGA	+ParaRetrieved	17	16	22	33	50	80	101
	%Improve	14.53	12.5	13.5	15.35	19.46	23.74	22.9
	RankCh	10	11	16	25	38	68	100
	BLEU-EditRankCh	26.35	26.14	27.70	21.71	22.37	17.43	13.85
	BLEU-ParaRankCh	**51.56**	**47.81**	**43.90**	**31.76**	**26.50**	**20.67**	**16.05**
	METEOR-EditRankCh	43.40	44.52	45.59	39.24	39.99	36.03	32.41
	METEOR-ParaRankCh	**67.68**	**66.75**	**61.09**	**50.07**	**45.37**	**39.31**	**34.51**

Table 3.7: Results of Automatic Evaluation: Presented using Cutoff Thresholds

respectively. Using DP we obtain more than 17% increase in retrieval for exact matches and more than 36% and 72% increase in the intervals [85, 100) and [70, 85), respectively. Table 3.7 shows that when using DPGA we obtain around 23% and using DP we obtain around 39% increase in retrieval for a threshold of 70%.

Table 3.6 and Table 3.7 also show that for the DPGA approach we obtained improvements in BLEU as well as Meteor scores. This suggests that the quality of the retrieved segments, for which fuzzy match scores get increased and come up to the top because of paraphrasing, is better compared to the corresponding segments retrieved using the baseline approach.

For the DP approach, we observe a small decrease in both BLEU and Meteor for the interval [70, 85). We can also observe that the DP approach retrieves more segments compared to the DPGA approach. The DP approach optimises globally which helps in bringing the more candidates. The DPGA approach takes

the local decision whenever Type 3 and Type 4 paraphrases are applied. For the Europarl corpus, the results suggest that this local decision helps in obtaining a better monotonic mapping between the two strings when calculating edit-distance, resulting in retrieving good quality candidates and discarding others.

3.3.3 Human Evaluation

In this section, we describe the human evaluation. We conducted a human evaluation of our DPGA approach. Human evaluations are expensive, so we selected the DPGA approach for evaluation as it retrieves better quality segments than the DP approach. As in the case of automatic evaluation, we consider the segments that changed their rankings using paraphrasing and supersede the existing simple edit-distance model. We take the top segment retrieved using paraphrasing and the top segment retrieved using simple edit-distance and compare to see which is better for a human translator to work with. We measure post-editing time (PET), keystrokes (KS) and two subjective evaluations (subjective evaluation with two options (SE2) and subjective evaluation with three options (SE3)) to compare.

We do not impose any penalty on paraphrasing; this means that if we obtain an exact match (100% fuzzy match) after considering paraphrasing we consider it an exact match. However, we prefer a match with simple edit-distance over paraphrasing in case of ties. The question arises whether we need to impose any penalty for paraphrasing. Is an exact match retrieved using paraphrasing really an exact match? We carried out another human evaluation to assess whether exact

matches retrieved after considering paraphrases are really exact matches.

3.3.3.1 Test sets, Tool and Translators' Expertise

To get the TM matches, we used the same corpus that was used in the automatic evaluation. Table 3.5 shows our corpus statistics. The sets' distribution for human evaluation is given in Table 3.8. The sets contain randomly selected segments from the additionally retrieved segments using paraphrasing which changed their top ranking. We have chosen the threshold intervals so as to select the segments from each range for the human evaluations. Two sets were created to facilitate the evaluation based on post-editing time and keystrokes (See Section 3.3.3.2). For this evaluation, each translator post-edited only one set.

TH	**100**	**[85, 100)**	**[70, 85)**	**Total**
Set-1	2	6	6	14
Set-2	5	4	7	16
Total	7	10	13	30

Table 3.8: Test Sets for Experiments PET, KS, SE2 and SE3

The translators involved in our experiments were third-year bachelor or masters translation students who were native speakers of German with English language level C1, in the age group of 21 to 40 years with a majority of female students. Our translators were not experts in any specific technical or legal field, or had previous professional experience working with TMs. However, they had previous experience with TMs as part of the classes they attended during their study of translation. For human evaluation, we used Europarl corpus which

contains parliamentary discussions (often scripted) and does not contain a specific technical or legal field text. Therefore, to a certain extent, we avoided the need of expert professional translators for human evaluation and also avoided any bias from unfamiliarity or familiarity with domain specific terms.

We used the PET tool (Aziz et al., 2012) for all our human experiments. However, settings were changed depending on the experiment. We carried out a pilot experiment before the actual experiment with the Europarl corpus. The aim of this pilot experiment was to familiarise translators with the PET tool. This experiment was done on a corpus (Vela et al., 2007) different from Europarl. 18 segments are used in the pilot experiment. While the findings are not included in this thesis, they informed the design of our main experiments.

3.3.3.2 Post-editing Time (PET) and Keystrokes (KS)

In this evaluation, the translators were presented with fuzzy matches and their task was to post-edit the segment in order to obtain a correct translation. The translators were presented with an English input segment, a German segment retrieved from the TM for post-editing and the English segment used for matching in the TM.

In this task, we recorded post-editing time (PET) and keystrokes (KS). The post-editing time taken for the entire file was calculated by summing up the time spent on each segment. Only one segment was visible on screen. The segment was only visible after clicking on the screen and the time was recorded from when the segment becomes visible until the translator finishes post-editing and goes to

the next screen. The next screen was a blank screen so that the translator can have a rest after post-editing a segment. Figure 3.3 shows the blank screen and Figure 3.4 shows the screen when a translator is editing.

Figure 3.3: Blank Screen

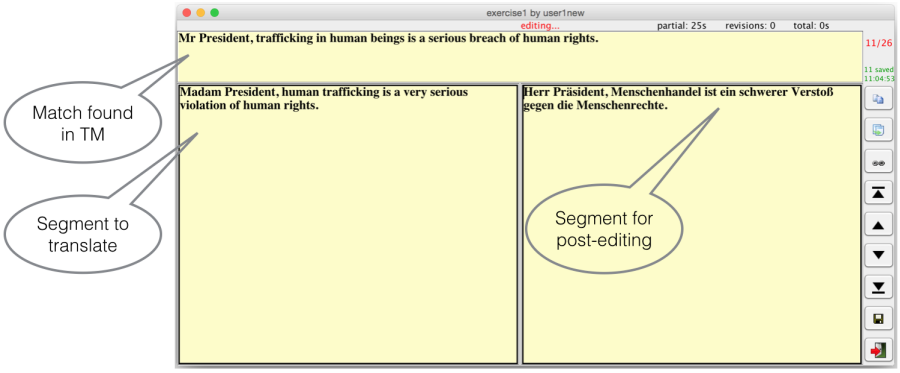


Figure 3.4: Editing is In Progress

The translators were aware that the time is being recorded. Each translator post-edited half of the segments retrieved using simple edit-distance (ED) and half

of the segments retrieved using paraphrasing (PP). The ED and PP matches were presented one after the other (ED at odd positions and PP at even positions or vice versa). However, the same translator did not post-edit the match retrieved using PP and ED for the same segment; instead five different translators post-edited the segment retrieved using PP and another five different translators post-edited the match retrieved using ED.

Post-editing time (PET) for each segment is the mean of the normalised time (N) taken by all translators on this segment. Normalised time (N) is calculated by multiplying the actual time taken by a translator on a segment by the average time taken on that file by all translators, divided by the time taken by the translator on this file. Normalisation is applied to account for both slow and fast translators.

$$\text{PET}_j = \frac{\sum_{i=1}^{n} N_{ij}}{n} \tag{3.1}$$

$$N_{ij} = T_{ij} \times \frac{\text{Avg time on this file by all translators}}{\sum_{j=1}^{m} T_{ij}} \tag{3.2}$$

In the equations 3.1 and 3.2 above, PET_j is the post-editing time for each segment j, n is the number of translators, N_{ij} is the normalised time of translator i on segment j, m is the number of segments in the file, and T_{ij} is the actual time taken by a translator i on a segment j.

Along with the post-editing time, we also recorded all printable keystrokes, white-space and erase keys pressed. For our analysis, we considered the average keystrokes pressed by all translators for each segment.

3.3.3.3 Subjective Evaluation with Two Options (SE2)

In this evaluation, we carried out a subjective evaluation with two options (SE2). We presented fuzzy matches retrieved using both paraphrasing (PP) and simple edit-distance (ED) to the translators and they had to choose which one is better. The translators were unaware of the details of how the fuzzy matches were obtained (ED or PP). To neutralise any bias, half of the ED matches were tagged as A and the other half as B, with the same applied to PP matches. 17 translators participated in this experiment. Finally, the decision of whether 'ED is better' or 'PP is better' was made on the basis of how many translators choose one over the other.

3.3.3.4 Subjective Evaluation with Three Options (SE3)

This evaluation was similar to Evaluation SE2 except that we provided one more option to translators. Translators could choose among three options: whether A is better; B is better; or both are equal. 7 translators participated in this experiment.

3.3.3.5 Subjective Evaluation on Exact Matches Only (SEM)

In this evaluation, our objective is to check whether an exact match after paraphrasing is really an exact match. We presented only exact matches retrieved using paraphrasing, which are not exact matches using simple edit-distance. 14 segments were presented to eleven translators. The translators task was to correct the segment and select an option from two options presented: can not be accepted as it is (post-edting was required); correct translation (no post-editing

was required).

3.3.4 Results and Analysis of Human Evaluation

Results for human evaluations (PET, KS, SE2 and SE3) on both sets (Set-1 and Set-2) are given in Table 3.9. Here 'Seg #' represents the segment number, 'ED' represents the match retrieved using simple edit-distance and 'PP' represents the match retrieved after incorporating paraphrasing. 'EDB', 'PPB' and 'BEQ' in Subjective Evaluations represent the number of translators choosing the 'ED is better', 'PP is better' and 'Both are equal' options respectively.

3.3.4.1 Results: Post-editing Time (PET) and Keystrokes (KS)

As we can see in Table 3.9, improvements were obtained for both sets. ↑ demonstrates cases in which PP performed better than ED and ↓ shows where ED performed better than PP. Entries in bold for PET, KS and SE2 indicate where the results are statistically significant.[5]

For Set-1, translators made 356.20 keystrokes and 532.60 keystrokes when editing PP and ED matches, respectively. Translators took 466.44 seconds for PP as opposed to 520.02 seconds for ED matches. This means that by using PP matches, translators edit 33.12% less, which saves 10.3% time.

For Set-2, translators made 468.59 keystrokes and 570.6 keystrokes when editing PP and ED matches respectively. Translators took 603.17 seconds for

[5] $p<0.05$, one tailed Welch's t-test for PET and KS, χ^2 test for SE2 and SE3. Because of the small sample size for SE3, no significance test was performed on individual segment basis. Because segments are different and each segment will take different post-editing time and keystrokes. We can not apply t-test on all 30 segments as a whole because it represents 30 different tasks. However, we applied chi square test for subjective evaluations.

	Post-editing				Subjective Evaluations				
	PET		KS		SE2 (2 Options)		SE3 (3 options)		
Seg #	**ED**	**PP**	**ED**	**PP**	**EDB**	**PPB**	**EDB**	**PPB**	**BEQ**
1	42.98	41.30↑	42.4	**0.4**↑	1	**16**↑	0	7↑	0
2!+	13.72	10.65↑	2.8	2.4↑	10	7↓	2	2	3
3*!	13.88	12.62↑	2.0	3.6↓	12	5↓	4	1↓	2
4	37.97	**17.64**↑	26.2	**6.2**↑	1	**16**↑	0	6↑	1
5!+	21.52	17.69↑	22.4	13.2↑	**13**	4↓	2	3↑	2
6!+	41.14	42.74↓	13.2	34.4↓	4	**13**↑	2	0	5
7!+	33.69	31.59↑	34.0	33.4↑	10	7↓	1	0	6
8	47.14	**23.41**↑	61.6	**6.4**↑	0	**17**↑	0	7↑	0
9	22.89	**14.20**↑	37.2	**2.2**↑	0	**17**↑	0	6↑	1
10	46.89	38.20↑	77.6	65.6↑	1	**16**↑	0	1	6
11	58.25	53.65↑	82.8	58.8↑	0	**17**↑	0	3	4
12!+	34.04	45.03↓	36.8	39.6↓	2	**15**↑	0	6↑	1
13	30.34	**21.12**↑	54.8	39.2↑	7	10↑	1	1	5
14!+	75.50	96.54↓	38.8	50.8↓	5	12↑	0	3	4
Set-1-subtotal	520.02	466.44↑	532.60	356.20↑	66	**172**↑	12	**46**↑	40
15	24.14	**9.18**↑	24.0	**0.0**↑	5	12↑	1	5↑	1
16*+	28.30	29.20↓	23.4	15.4↑	11	6↓	2	2	3
17*!	65.64	53.49↑	6.2	22.4↓	10	7↓	2	3↑	2
18	41.91	**20.98**↑	28.0	**2.0**↑	1	**16**↑	0	6↑	1
19	29.81	19.71↑	23.8	**6.8**↑	7	10↑	2	3↑	2
20	41.25	**15.42**↑	39.0	**3.8**↑	0	**17**↑	1	5↑	1
21*!	**42.04**	65.44↓	39.4	36.0↑	7	10↑	1	2	4
22	29.28	35.87↓	17.0	33.4↓	12	5↓	5	0↓	2
23	**32.64**	49.49↓	**11.4**	50.8↓	11	6↓	2	2	3
24!+	59.35	54.54↑	79.6	79.2↑	**17**	0↓	5	0↓	2
25	62.51	61.30↑	71.0	54.0↑	2	**15**↑	0	3	4
26*!	36.82	41.06↓	55.0	**23.4**↑	1	**16**↑	0	6↑	1
27!+	**27.21**	44.02↓	**24.4**	48.8↓	4	**13**↑	1	5↑	1
28	40.99	**33.08**↑	39.6	**24.6**↑	5	12↑	3	4↑	0
29	52.01	**31.55**↑	50.6	**23.4**↑	2	**15**↑	0	6↑	1
30*!	43.76	38.76↑	38.2	44.6↓	**15**	2↓	1	1	5
Set-2-subtotal	657.75	603.17↑	570.6	468.59↑	110	**162**↑	26	**53**↑	33
Total	1177.77	1069.61↑	1103.2	824.79↑	176	**334**↑	38	**99**↑	73

Table 3.9: Results of Human Evaluation on Set-1 (1-14) and Set-2 (15-30)

PP as opposed to 657.75 seconds for ED matches. This means that by using PP matches, translators edit 17.87% less, which saves 8.29% time.

In total, combining both sets, translators made 824.79 keystrokes and 1103.2 keystrokes when editing PP and ED matches, respectively. Translators took 1069.61 seconds for PP as opposed to 1177.77 seconds for ED matches. Therefore, by using PP matches, translators edit 25.23% less, which saves time by 9.18%. We observe that the percentage improvement obtained by keystroke

analysis is smaller compared to the improvement obtained by post-editing time. One of the reasons for this is that the translator spends a fair amount of time reading a segment before starting editing.

3.3.4.2 Results: Using Post-edited References

We also calculated the human-targeted translation error rate (HTER) (Snover et al., 2006) and human-targeted Meteor (HMETEOR) (Denkowski and Lavie, 2014). HTER and HMETEOR were calculated between ED and PP matches presented for post-editing and references generated by editing the corresponding ED and PP match. Table 3.10 lists HTER5 and HMETEOR5, which use five corresponding ED or PP references only and HTER10 and HMETEOR10, which use all ten references generated using ED and PP.

Table 3.10 shows improvements in both the HTER5 and HMETEOR5 scores. For Set-1, HMETEOR5 improved from 59.82 to 81.44 and HTER5 improved from 39.72 to 17.63.[6] For Set-2, HMETEOR5 improved from 69.81 to 80.60 and HTER5 improved from 27.81 to 18.71. We also observe that while ED scores of Set-1 and Set-2 differ substantially (59.82 vs 69.81 and 39.72 vs 27.81), PP scores are nearly the same (81.44 vs 80.60 and 17.63 vs 18.71). This suggests that paraphrasing not only brings improvement but may also improve consistency.

3.3.4.3 Results: Subjective Evaluations

The subjective evaluations also show significant improvements.

In subjective evaluation with two options (SE2) as given in Table 3.9, from a

[6]For HMETEOR, higher is better and for HTER lower is better.

	Set-1		Set-2	
	ED	**PP**	**ED**	**PP**
HMETEOR5	59.82	81.44	69.81	80.60
HTER5	39.72	17.63	27.81	18.71
HMETEOR10	59.82	81.44	69.81	80.61
HTER10	36.93	18.46	27.26	18.40

Table 3.10: Results using Human Targeted References

total of 510 (30×17) replies for 30 segments from both sets by 17 translators, 334 replies tagged 'PP is better' and 176 replies tagged 'ED is better'. [7]

In the subjective evaluation with three options (SE3), from a total of 210 (30×7) replies for 30 segments from both sets by 7 translators, 99 replies tagged 'PP is better', 73 replies tagged 'both are equal' and 38 replies tagged 'ED is better'.[8]

3.3.4.4 Results: Segment-Wise Analysis

A segment-wise analysis of 30 segments from both sets shows that 21 segments extracted using PP were found to be better according to PET evaluation and 20 segments using PP were found to be better according to KS evaluation. In subjective evaluations, 20 segments extracted using PP were found to be better according to SE2 evaluation, whereas 27 segments extracted using PP were found to be better or equally good according to SE3 evaluation (15 segments were found to be better and 12 segments were found to be equally good).

We have also observed that not all evaluations correlate with each other on a segment-by-segment basis. '!, '+ and '* next to each segment number in Table

[7]Statistically significant, χ^2 test, $p < 0.001$.
[8]Statistically significant, χ^2 test, $p < 0.001$.

3.9 indicate conflicting evaluations: '!' denotes that PET and SE2 contradict each other, '+' denotes that KS and SE2 contradict each other and '*' denotes that PET and KS contradict each other. In twelve segments where KS evaluation or PET evaluation show PP as statistically significantly better, except for two cases all the evaluations also show them better.[9] For Seg #13 SE3 shows 'Both are equal' and for Seg #26, PET is better for ED, however for these two sentences also all the other evaluations show PP as better.

In three segments (Seg #'s 21, 23, 27) KS evaluation or PET evaluation show ED as statistically significantly better, but none of the segments are tagged better by all the evaluations. In Seg #21 all the evaluations with the exception of PET show PP as better. In Seg #23, SE3 shows 'both are equal'. Seg #23 is given as follows:

> **Input:** The next item is the Commission declaration on Belarus .
>
> **ED:** The next item is the Commission Statement on AIDS .//Als nächster Punkt folgt die Erklärung der Kommission zu AIDS.
>
> **PP:** The next item is the Commission statement on Haiti .//Nach der Tagesordnung folgt die Erklärung der Kommission zu Haiti.

In Seg #23, apart from "AIDS" and "Haiti" the source side does not differ but the German side differs. The reason for PP match retrieval was that "statement on" in lower case was paraphrased as "declaration on" while in the other segment "Statement" was capitalised and hence was not paraphrased. If we look at the

[9] In this section all evaluations refer to all four evaluations viz PET, KS, SE2 and SE3.

German side of both ED and PP, “Nach der Tagesordnung” requires a broader context to accept it as a translation of “The next item” whereas “Als nächster Punkt” does not require much context.

In Seg #27, we observe contradictions between post-editing evaluations and subjective evaluations. Seg #27 is given below (EDPE and PPPE are post-edited translations of ED and PP match respectively):

Input: That would be an incredibly important signal for the whole region .

ED: That could be an important signal for the future .//Dies könnte ein wichtiges Signal für die Zukunft sein.

PP: That really would be extremely important for the whole region .//Und das wäre wirklich für die ganze Region extrem wichtig.

EDPE: Dies könnte ein unglaublich wichtiges Signal für die gesamte Region sein.

PPPE: Das wäre ein unglaublich wichtiges Signal für die ganze Region.

In the subjective evaluations, translators tagged PP as better than ED. But, post-editing suggests that it takes more time and keystrokes to post-edit the PP compared with ED.

There is one segment, Seg #22, on which all the evaluations show that ED is better. Seg #22 is given below:

Input: I would just like to comment on one point.

ED: I would just like to emphasise one point.//Ich möchte nur eine Sache betonen.

PP: I would just like to concentrate on one issue.//Ich möchte mich nur auf einen Punkt konzentrieren.

In segment 22, the ED match is clearly closer to the input than the PP match. Paraphrasing "on one point" as "on one issue" does not improve the result. Also, "konzentrieren" being a long word takes more time and keystrokes in post-editing.

3.3.4.5 Results: Subjective Evaluation on Exact Matches Only (SEM)

The results of the subjective evaluation on exact matches (SEM) are given in Table 3.11.[10] On 10 out of 14 segments, seven or more (two thirds) of the translators

Seg #	Yes	No	No Post-editing
1	11	0	Yes
2	10	1	Yes
3	10	1	Yes
4	9	2	Yes
5	8	3	Yes
6	9	2	Yes
7	2	9	**No**
8	10	1	Yes
9	1	9	**No**
10	11	0	Yes
11	11	0	Yes
12	6	5	indecisive
13	7	4	Yes
14	5	6	indecisive
Total	110	43	-

Table 3.11: Results of Human Evaluation on Exact Matches

[10]The Seg # 9 was skipped by one of the translator. Therefore, we have 10 evaluators for this segment instead of 11 evaluators for other segments.

agree that the segment does not require any post-editing. In rest of the cases, for two segments (Seg #13 and Seg #15) the judgements were contradictory, with half of the translators agreeing and half disagreeing whether the segment needs post-editing. In other two cases (Seg #7 and Seg #9) most of the translators chosen to post-edit the segments. The Seg #7 is given below (PPPE represents the most preferred post-edited translation):

EN The vote will take place immediately following the ongoing debates.

PP The vote will take place immediately after the ongoing debates. // Die Abstimmung findet unverzüglich im Anschluss an die laufenden Aussprachen statt.

PPPE Die Abstimmung findet unverzüglich im Anschluss an die laufenden Debatten statt.

We can see that the source segment match is accurate. Most of the translators edited 'Aussprachen' to 'Debatten'.

The Seg #9 is given below:

EN (The sitting was suspended at 11.25 p.m.)

PP (The sitting was closed at 11.25 p.m.) // (Die Sitzung wird um 23.25 geschlossen)

PPPE (Die Sitzung wurde um 23:25 geschlossen)

In segment 9, ‘closed’ and ‘suspended’ differ but this does not impact the target side. Translators changed the auxiliary verb ‘wird’ to ‘wurde’.

For, most of the segments translators agree to accept them as is. This suggests that for a majority of segments paraphrasing match can be presented as an exact match.

3.4 Evaluation on the DGT-TM Corpus

In this section, we conduct experiments to check the performance of both our DPGA and DP approaches on a different corpus.

For our experiments, we have used English-German, English-French and English-Spanish data from the 2014 release of the DGT-TM corpus (Steinberger et al., 2012). We conduct only automatic evaluations as human evaluations are expensive. In addition, because the DGT-TM corpus contains texts of legal genre, such an evaluation is even more difficult.

From this corpus, we filtered out segments of fewer than seven words and more than 40 words; the remaining pairs were used to create the TM and Test dataset. The test sets for all language pairs contain 20,000 randomly selected unique segments and the rest are used as the TM. The statistics of the datasets are given in Table 3.12.

We performed experiments using both the DP approach and the DPGA approach. The DGT-TM corpus is of legal genre and contains many punctuation marks. Therefore, we decided to additionally conduct experiments after removing punctuation marks. For both the DP and DPGA approaches, we conduct

experiments in two different preprocessing settings: S1 and S2. In setting S1, we do not remove any punctuation marks and in the preprocessing step we perform only tokenization; in setting S2, in addition to tokenization, we also remove punctuation marks in the preprocessing stage. In setting S2, punctuation marks are also removed from the target side. We used the Stanford tokenizer on the English side and the tokenizer provided with Moses (Koehn et al., 2007) on the target side. The source (English) tokenization is used for matching and target language tokenization is used when calculating the BLEU score.

Tables 3.13 and 3.14 present the results on the English-German language pair data. The results are presented in two different ways, using threshold intervals and using cutoff thresholds, in the same way as presented in Section 3.3. Table 3.13 present results using threshold intervals and Table 3.14 using cutoff thresholds.

Table 3.13 shows that we obtained 9.56% increase in retrieval for setting S1 and 11.41% for setting S2 in the [85, 100) threshold interval for the DP approach. For the cutoff thresholds, we observe between 0.7% to 2.4% increase in the number of segments retrieved. The reason for the low retrieval is that we have a very high retrieval of the exact matches using the baseline edit-distance. This impacts the percentage improvements obtained over exact and fuzzy match retrieval. Table 3.13 shows that the baseline edit-distance retrieves 6629 matches for the 100 threshold (exact match) and 1193 matches are retrieved for the [85, 100) threshold interval.

For the DPGA approach, we observe that in both settings (S1 and S2), we get increase in retrieval, however, in setting S1, for some cases the BLEU and Meteor

scores decrease for the DPGA approach. But, when we remove punctuation marks in setting S2, we obtain better BLEU and Meteor scores, for all the intervals.

If we compare the DP and DPGA approaches, the DP approach retrieves more matches compared to the DPGA approach.

Table 3.13 and Table 3.14, which present results on the DGT-TM dataset, show that we obtain better Meteor scores compared to the baseline edit-distance in both settings S1 and S2 for the DP approach. However, in the previous experiments, Table 3.6 shows that we do not obtain better Meteor scores for the [85, 75) threshold interval for the DP approach.

Table 3.6, which presents results on the Europarl dataset, shows that for the DPGA approach we obtain better Meteor scores for every threshold interval. Table 3.13, which presents results on the DGT-TM dataset, shows that the DPGA approach does not obtain better Meteor scores compared to the baseline edit-distance for the [85, 100) and [55, 70) threshold intervals.

Tables 3.13 and Tables 3.14 show that removing punctuation marks in the preprocessing step helps in retrieving more and better matches using both the DP and DPGA approaches. We have observed that removing punctuation marks in the preprocessing stage not only increases the retrieval but also increases the improvement in retrieval using paraphrases. We can also see that using the setting S2, for both approaches, we obtain improvements across all threshold intervals (Tables 3.13) or cutoff thresholds (Tables 3.14), except Table 3.14 shows that for using the DPGA approach, for the 80 threshold, we do not obtain increase in the BLEU score.

We obtain similar results on English-French and English-Spanish language pairs. Tables 3.15 and 3.16 show results on the English-French language pair and Tables 3.17 and 3.18 on the English-Spanish language pair.

Our results suggest that both approaches improve retrieval. In the cases where there are more chances of paraphrasing to be applied like the Europarl dataset, which is of spoken genre, we get better quality matches using the DPGA compared to the DP approach, whereas for the DGT-TM dataset, which is of legal genre, we obtain better quality matches using the DP approach.

	English-German		English-French		English-Spanish	
	TM	**Test Set**	**TM**	**Test Set**	**TM**	**Test Set**
Segments	204,776	20,000	204,713	20,000	202,700	20,000
Source words	4,179,007	382,793	4,177,332	382,358	4,140,473	383,694
Target words	3,833,088	343,274	4,666,196	407,495	4,783,178	433,450

Table 3.12: DGT-TM Corpus Statistics

		TH	**100**	**[85, 100)**	**[70, 85)**	**[55, 70)**
DP	S1	EditRetrieved	6629	1193	1029	1259
		+ParaRetrieved	54	114	146	445
		%Improve	0.81	9.56	14.19	35.35
		RankCh	7	8	27	217
		METEOR-EditRankCh	83.82	57.42	24.05	26.47
		METEOR-ParaRankCh	**90.76**	**64.15**	**33.88**	**26.89**
		BLEU-EditRankCh	72.79	38.21	12.13	15.38
		BLEU-ParaRankCh	**84.15**	**49.11**	**14.44**	14.40
	S2	EditRetrieved	6767	1078	889	1070
		+ParaRetrieved	60	123	150	380
		%Improve	0.89	11.41	16.87	35.51
		RankCh	8	15	36	176
		METEOR-EditRankCh	80.01	57.79	37.99	24.90
		METEOR-ParaRankCh	**90.27**	**67.34**	**43.95**	**27.59**
		BLEU-EditRankCh	64.97	50.22	25.20	13.37
		BLEU-ParaRankCh	**84.90**	**54.43**	**29.67**	**14.51**
DPGA	S1	EditRetrieved	6629	1193	1029	1259
		+ParaRetrieved	44	88	89	244
		%Improve	0.66	7.38	8.65	19.38
		RankCh	4	5	21	115
		METEOR-EditRankCh	75.69	63.78	25.11	27.74
		METEOR-ParaRankCh	**89.93**	49.97	**28.53**	27.58
		BLEU-EditRankCh	65.26	50.27	6.67	17.69
		BLEU-ParaRankCh	**73.79**	23.68	**8.08**	15.61
	S2	EditRetrieved	6767	1078	889	1070
		+ParaRetrieved	49	98	99	224
		%Improve	0.72	9.09	11.14	20.93
		RankCh	5	9	30	118
		METEOR-EditRankCh	69.20	49.19	38.58	25.00
		METEOR-ParaRankCh	**88.21**	**54.42**	**38.68**	**28.23**
		BLEU-EditRankCh	49.34	36.06	23.06	13.02
		BLEU-ParaRankCh	**77.85**	**36.38**	**23.48**	**14.99**

Table 3.13: Results of Automatic Evaluation (English-German): Presented using Threshold Intervals

		TH	**100**	**95**	**90**	**85**	**80**	**75**	**70**
DP	S1	EditRetrieved	6629	6877	7444	7822	8205	8498	8851
		+ParaRetrieved	54	163	128	119	113	136	150
		%Improve	0.81	2.37	1.72	1.52	1.38	1.6	1.69
		RankCh	7	10	16	22	31	53	70
		METEOR-EditRankCh	83.82	83.02	77.71	76.42	68.28	60.71	53.93
		METEOR-ParaRankCh	**90.76**	**88.19**	**80.73**	**81.38**	**72.65**	**65.89**	**60.92**
		BLEU-EditRankCh	72.79	73.00	68.61	69.66	58.76	51.85	46.20
		BLEU-ParaRankCh	**84.15**	**82.32**	**74.91**	**74.22**	**64.10**	**56.41**	**50.03**
	S2	EditRetrieved	6767	6959	7425	7845	8200	8503	8734
		+ParaRetrieved	60	165	149	130	115	137	156
		%Improve	0.89	2.37	2.01	1.66	1.4	1.61	1.79
		RankCh	8	10	16	29	35	57	76
		METEOR-EditRankCh	80.01	80.49	68.09	68.67	63.58	55.45	53.16
		METEOR-ParaRankCh	**90.27**	**87.98**	**76.22**	**73.63**	**68.75**	**61.37**	**58.09**
		BLEU-EditRankCh	64.97	67.95	58.36	59.83	53.17	44.15	42.42
		BLEU-ParaRankCh	**84.90**	**82.68**	**69.55**	**64.56**	**58.37**	**49.33**	**46.47**
DPGA	S1	EditRetrieved	6629	6877	7444	7822	8205	8498	8851
		+ParaRetrieved	44	155	107	93	82	97	92
		%Improve	0.66	2.25	1.44	1.19	1.0	1.14	1.04
		RankCh	4	5	11	13	22	41	53
		METEOR-EditRankCh	75.69	78.27	73.06	69.58	58.58	47.96	46.56
		METEOR-ParaRankCh	**89.93**	**89.77**	**74.46**	**69.93**	**60.49**	**49.98**	**48.84**
		BLEU-EditRankCh	65.26	66.23	63.10	60.54	43.24	33.60	32.10
		BLEU-ParaRankCh	**73.79**	**76.26**	**62.92**	**57.18**	**45.09**	**35.21**	**33.39**
	S2	EditRetrieved	6767	6959	7425	7845	8200	8503	8734
		+ParaRetrieved	49	153	132	105	83	93	103
		%Improve	0.72	2.2	1.78	1.34	1.01	1.09	1.18
		RankCh	5	5	11	20	25	45	59
		METEOR-EditRankCh	69.20	69.20	53.90	62.06	56.28	48.18	49.23
		METEOR-ParaRankCh	**88.21**	**88.21**	**67.87**	**63.34**	**57.62**	**53.06**	**50.96**
		BLEU-EditRankCh	49.34	49.34	40.65	49.43	41.87	34.29	34.00
		BLEU-ParaRankCh	**77.85**	**77.85**	**57.14**	**47.09**	39.35	**37.25**	**35.32**

Table 3.14: Results of Automatic Evaluation (English-German): Presented using Cutoff Thresholds

		TH	**100**	**[85, 100)**	**[70, 85)**	**[55, 70)**
DP	S1	EditRetrieved	6611	1197	1040	1257
		+ParaRetrieved	54	116	141	443
		%Improve	0.82	9.69	13.56	35.24
		RankCh	7	8	26	217
		METEOR-EditRankCh	94.88	45.92	34.91	32.26
		METEOR-ParaRankCh	**96.17**	**70.23**	**42.92**	**32.70**
		BLEU-EditRankCh	78.21	17.81	20.06	19.75
		BLEU-ParaRankCh	**87.56**	**46.58**	**24.67**	18.15
	S2	EditRetrieved	6750	1082	894	1078
		+ParaRetrieved	60	125	141	377
		%Improve	0.89	11.55	15.77	34.97
		RankCh	8	15	33	177
		METEOR-EditRankCh	81.73	50.35	49.02	31.45
		METEOR-ParaRankCh	**92.87**	**66.37**	**58.60**	**33.99**
		BLEU-EditRankCh	63.44	33.50	32.84	19.56
		BLEU-ParaRankCh	**84.80**	**47.38**	**39.70**	**20.02**
DPGA	S1	EditRetrieved	6611	1197	1040	1257
		+ParaRetrieved	44	90	87	241
		%Improve	0.67	7.52	8.37	19.17
		RankCh	4	5	20	115
		METEOR-EditRankCh	92.26	54.38	39.96	33.47
		METEOR-ParaRankCh	91.44	**61.28**	39.37	33.35
		BLEU-EditRankCh	79.60	28.87	18.76	20.87
		BLEU-ParaRankCh	69.32	27.42	17.19	17.50
	S2	EditRetrieved	6750	1082	894	1078
		+ParaRetrieved	49	100	91	223
		%Improve	0.73	9.24	10.18	20.69
		RankCh	5	9	28	118
		METEOR-EditRankCh	71.18	41.02	50.27	31.17
		METEOR-ParaRankCh	**85.27**	**52.54**	**58.60**	**35.77**
		BLEU-EditRankCh	48.88	15.76	32.25	18.18
		BLEU-ParaRankCh	**68.68**	**28.31**	**37.85**	**18.77**

Table 3.15: Results of Automatic Evaluation on DGT-TM (English-French): Presented using Threshold Intervals

		TH	**100**	**95**	**90**	**85**	**80**	**75**	**70**
DP	S1	EditRetrieved	6611	6859	7432	7808	8189	8492	8848
		+ParaRetrieved	54	163	128	121	113	131	145
		%Improve	0.82	2.38	1.72	1.55	1.38	1.54	1.64
		RankCh	7	10	16	22	32	52	69
		METEOR-EditRankCh	94.88	94.53	87.03	80.47	73.66	67.23	60.33
		METEOR-ParaRankCh	**96.17**	**95.42**	**90.46**	**88.26**	**80.25**	**72.56**	**67.10**
		BLEU-EditRankCh	78.21	80.78	71.28	65.14	57.08	52.58	47.18
		BLEU-ParaRankCh	**87.56**	**86.98**	**77.09**	**75.00**	**66.14**	**56.47**	**50.94**
	S2	EditRetrieved	6750	6939	7409	7832	8189	8490	8726
		+ParaRetrieved	60	165	149	132	115	134	147
		%Improve	0.89	2.38	2.01	1.69	1.40	1.58	1.68
		RankCh	8	10	16	29	36	56	74
		METEOR-EditRankCh	81.73	85.89	74.89	68.45	65.95	59.96	59.19
		METEOR-ParaRankCh	**92.87**	**92.58**	**81.72**	**80.00**	**77.34**	**68.71**	**67.99**
		BLEU-EditRankCh	63.44	70.27	61.09	52.75	49.17	44.10	44.54
		BLEU-ParaRankCh	**84.80**	**83.30**	**70.01**	**66.81**	**63.52**	**53.03**	**52.89**
DPGA	S1	EditRetrieved	6611	6859	7432	7808	8189	8492	8848
		+ParaRetrieved	44	155	107	95	84	94	90
		%Improve	0.67	2.26	1.44	1.22	1.03	1.11	1.02
		RankCh	4	5	11	13	22	40	52
		METEOR-EditRankCh	92.26	87.88	80.74	73.88	65.50	57.53	54.92
		METEOR-ParaRankCh	91.44	**91.29**	**83.15**	**78.98**	**68.10**	**58.80**	**56.90**
		BLEU-EditRankCh	79.60	75.47	63.30	55.00	43.62	37.22	36.17
		BLEU-ParaRankCh	69.32	**78.43**	59.92	**58.81**	**46.44**	**37.86**	**36.19**
	S2	EditRetrieved	6750	6939	7409	7832	8189	8490	8726
		+ParaRetrieved	49	153	132	107	85	91	95
		%Improve	0.73	2.20	1.78	1.37	1.04	1.07	1.09
		RankCh	5	5	11	20	25	44	57
		METEOR-EditRankCh	71.18	71.18	58.98	60.81	58.32	51.76	54.57
		METEOR-ParaRankCh	**85.27**	**85.27**	**68.77**	**71.88**	**68.30**	**61.04**	**63.37**
		BLEU-EditRankCh	48.88	48.88	39.89	40.72	38.40	32.05	35.33
		BLEU-ParaRankCh	**68.68**	**68.68**	**49.77**	**54.29**	**50.03**	**41.85**	**43.40**

Table 3.16: Results of Automatic Evaluation on DGT-TM (English-French): Presented using Cutoff Thresholds

		TH	**100**	**[85, 100)**	**[70, 85)**	**[55, 70)**
DP	S1	EditRetrieved	6620	1187	1028	1233
		+ParaRetrieved	54	115	141	433
		%Improve	0.82	9.69	13.72	35.12
		RankCh	7	8	29	209
		METEOR-EditRankCh	85.85	52.10	36.62	33.80
		METEOR-ParaRankCh	**88.45**	**64.83**	**44.48**	**34.58**
		BLEU-EditRankCh	72.79	24.74	17.49	19.20
		BLEU-ParaRankCh	**78.09**	**44.48**	**24.22**	19.09
	S2	EditRetrieved	6757	1076	885	1050
		+ParaRetrieved	60	125	142	374
		%Improve	0.89	11.62	16.05	35.62
		RankCh	8	15	34	179
		METEOR-EditRankCh	82.11	58.01	49.32	35.13
		METEOR-ParaRankCh	**87.12**	**67.94**	**60.42**	**36.07**
		BLEU-EditRankCh	61.84	37.02	33.42	18.84
		BLEU-ParaRankCh	**73.83**	**47.07**	**44.08**	**19.82**
DPGA	S1	EditRetrieved	6620	1187	1028	1233
		+ParaRetrieved	44	89	86	234
		%Improve	0.66	7.50	8.37	18.98
		RankCh	4	5	21	110
		METEOR-EditRankCh	73.62	51.61	35.72	34.85
		METEOR-ParaRankCh	**83.36**	49.02	**44.94**	**35.93**
		BLEU-EditRankCh	55.75	38.20	14.49	20.11
		BLEU-ParaRankCh	**75.49**	18.61	**24.13**	19.48
	S2	EditRetrieved	6757	1076	885	1050
		+ParaRetrieved	49	100	92	221
		%Improve	0.73	9.29	10.40	21.05
		RankCh	5	9	28	118
		METEOR-EditRankCh	69.89	51.35	49.58	35.46
		METEOR-ParaRankCh	**82.52**	**66.11**	**60.39**	**37.14**
		BLEU-EditRankCh	37.04	29.67	29.29	18.37
		BLEU-ParaRankCh	**69.89**	**44.80**	**42.02**	**19.83**

Table 3.17: Results of Automatic Evaluation on DGT-TM (English-Spanish): Presented using Threshold Intervals

		TH	**100**	**95**	**90**	**85**	**80**	**75**	**70**
DP	S1	EditRetrieved	6620	6867	7433	7807	8190	8489	8835
		+ParaRetrieved	54	162	129	120	112	130	145
		%Improve	0.82	2.36	1.74	1.54	1.37	1.53	1.64
		RankCh	7	10	16	22	32	55	72
		METEOR-EditRankCh	85.85	86.89	80.73	75.47	69.47	63.91	58.83
		METEOR-ParaRankCh	**88.45**	**88.39**	**83.28**	**79.27**	**73.86**	**70.51**	**65.57**
		BLEU-EditRankCh	72.79	73.42	66.05	61.39	54.19	49.74	44.30
		BLEU-ParaRankCh	**78.09**	**76.50**	**68.61**	**64.00**	**58.55**	**54.11**	**49.61**
	S2	EditRetrieved	6757	6947	7413	7833	8186	8489	8718
		+ParaRetrieved	60	165	149	132	114	133	148
		%Improve	0.89	2.38	2.01	1.69	1.39	1.57	1.70
		RankCh	8	10	16	29	36	57	76
		METEOR-EditRankCh	82.11	85.49	77.19	71.83	68.70	62.20	61.38
		METEOR-ParaRankCh	**87.12**	**86.77**	**80.87**	**76.95**	**76.12**	**70.78**	**69.40**
		BLEU-EditRankCh	61.84	69.58	58.69	54.14	49.59	45.26	45.25
		BLEU-ParaRankCh	**73.83**	**73.53**	**64.65**	**59.11**	**58.32**	**53.29**	**53.49**
DPGA	S1	EditRetrieved	6620	6867	7433	7807	8190	8489	8835
		+ParaRetrieved	44	154	108	94	83	92	89
		%Improve	0.66	2.24	1.45	1.20	1.01	1.08	1.01
		RankCh	4	5	11	13	22	40	52
		METEOR-EditRankCh	73.62	74.49	73.09	66.83	63.09	55.65	53.48
		METEOR-ParaRankCh	**83.36**	**86.73**	**77.91**	**71.88**	**66.62**	**61.02**	**57.85**
		BLEU-EditRankCh	55.75	49.33	53.48	49.00	43.27	36.41	33.85
		BLEU-ParaRankCh	**75.49**	**75.67**	**58.95**	**54.22**	**48.79**	**42.53**	**39.31**
	S2	EditRetrieved	6757	6947	7413	7833	8186	8489	8718
		+ParaRetrieved	49	153	132	107	83	90	96
		%Improve	0.73	2.20	1.78	1.37	1.01	1.06	1.10
		RankCh	5	5	11	20	25	44	57
		METEOR-EditRankCh	69.89	69.89	64.96	62.93	60.58	55.24	56.77
		METEOR-ParaRankCh	**82.52**	**82.52**	**77.59**	**71.36**	**69.85**	**66.21**	**66.04**
		BLEU-EditRankCh	37.04	37.04	38.09	40.16	38.72	35.14	35.77
		BLEU-ParaRankCh	**69.89**	**69.89**	**58.23**	**50.86**	**49.75**	**47.28**	**47.21**

Table 3.18: Results of Automatic Evaluation on DGT-TM (English-Spanish): Presented using Cutoff Thresholds

3.5 Conclusion

In this chapter, we proposed two novel and efficient approaches to improve matching and retrieval using paraphrasing. We conclude that paraphrasing significantly increases TM retrieval. For the Europarl dataset, we observe around 30% and 43% increase for the threshold intervals [85, 100) and [70, 85), respectively and around 23% increase over 70 or 75 cutoff threshold for our DPGA approach. The quality of the retrieved segments is also better, which is evident from all our human translation evaluations. On average on both sets used for evaluation, compared to paraphrasing, simple edit-distance takes 33.75% more keystrokes and 10.11% more time when evaluating the segments that changed their top rank and come up in the threshold intervals because of paraphrasing. For the DGT-TM dataset, the improvements are moderate because of the legal genre of the corpus. For our DPGA approach, depending on the preprocessing settings, we obtained around 7% to 11% improvements in retrieval for threshold intervals [85, 100) or [70, 85) on the English-German language pair. For our DP approach, depending on the preprocessing settings, we obtained around 9% to 16% improvements in retrieval for threshold intervals [85, 100) or [70, 85) on the English-German language pair. Similar improvements are obtained for English-Spanish and English-French language pairs of the DGT-TM dataset.

CHAPTER 4

ADVANCED SEMANTIC MATCHING FOR TM

In Chapter 3, we proposed two approaches which include paraphrasing in TM matching and retrieval, and improve over the baseline edit-distance based matching. However, the approaches still use a modified edit-distance with limited semantic information available from paraphrases. The same meaning can be represented using a different set of words or structures. When segments vary more on the surface form or structure, the approaches presented in Chapter 3, which largely rely on string matching, may miss such segments.

In this chapter, instead of computing similarity based on string matching, we present three approaches to improve TM matching and retrieval using advanced semantic matching techniques.

In the first section (Section 4.1), we present a system which uses manually designed features. A dataset annotated with similarity scores and the manually designed features are used to train a supervised machine learning system based on support vector machine (SVM) regression. The system predicts the similarity score between a pair of segments using the features extracted from the pair of segments. The features are computed using dependency parsing, paraphrasing, machine translation evaluation, pos-tagging, lemmatization and corpus pattern

analysis (Hanks, 2013).

In the second section (Section 4.2), we present two approaches based on Recurrent Neural Networks and in particular Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Tree Structured Long Short Term Memory (Tree-LSTM) networks. The LSTM and Tree-LSTM networks are trained to represent segments as vectors and compute similarity between a pair of vectors. Both the LSTM and the Tree-LSTM networks use word vectors (Pennington et al., 2014) and a dataset annotated with similarity scores for training. The Tree-LSTM network also requires dependency parsing to parse a segment. In the LSTM network, a segment representation is computed sequentially, whereas in the Tree-LSTM network, a segment representation is computed recursively using the dependency parsing.

4.1 Advanced Semantic Matching for TM using a Traditional Approach

This section presents our approach based on manually designed features. We extract features from the pair of segments which can help in determining the semantic similarity between segments. We train a supervised machine learning system, which predicts the similarity between segments based on the features. We extract features using advanced semantics (such as parsing and paraphrasing), machine translation evaluation, pos-tagging, lemmatization and Corpus Pattern Analysis (CPA). The assumption is that using this technique we can obtain more semantically similar matches than a surface form matching like edit-distance.

4.1.1 Our Approach for Semantic Textual Similarity

We build a semantic textual similarity (STS) system using a dataset annotated with similarity scores. Section 4.1.2 provides details of our STS system. This STS system works as a similarity function between two segments. In the translation memory matching and retrieval process, we retrieve TM segments using the STS system as follows:

1. Read the Translation Memories available
2. Read the input file that needs to be translated
3. Parse segments in the Translation Memories and in the input file using the Stanford dependency parser
4. For each segment in the input file
   (a) Extract features with all segments in the TM
   (b) Compute similarity with all segments in the TM using the STS system based on features extracted in Step (a)
   (c) Retrieve the most similar segment

As we can see in Step 4, we need to extract features for all the pairs of segments. If there are n segments in the input file and m segments in the TM, we extract features for $n \times m$ pairs. Computing all these features is a computational expensive process. To avoid repeating the parsing of the TM segments, we parse the translation memories and the input file in Step 3.

4.1.2 STS System

This section describes our STS system. We build our STS system using a supervised machine learning approach based on support vector machines (SVM). The approach uses an SVM regression model trained on features extracted using NLP technology from a dataset annotated with similarity scores. Section 4.1.2.1 describes the system details, Section 4.1.2.2 describes the features and Section 4.1.2.3 provides details of training dataset.

4.1.2.1 System Description

Our system is inspired by the system presented in Gupta et al. (2014), which calculates the similarity and entailment between a pair of sentences and performed well in the SemEval-2014 task-1, Semantic Relatedness and Textual Entailment task (Marelli et al., 2014a). We adapted this system to measure the similarity between two TM segments. Given the amount of calculation involved for TM matching and retrieval, we kept a subset of features used in Gupta et al. (2014). We kept only those features, which can be quickly calculated and proved the most useful for the original system. After removing these features we observed only a small decrease in performance.

We used a Support Vector Machine (SVM) regression model with Radial Basis Function (RBF) kernel. For the actual implementation we used the scikit-learn[1] toolkit which implements LibSVM[2] (Chang and Lin, 2011). This model estimates a continuous score between 1 and 5 for each sentence. We considered a continuous

[1]http://scikit-learn.org/stable/
[2]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

score from 1 to 5 because the available training dataset is tagged with scores from 1 to 5.

The SVM used an RBF kernel with $C = 8$ and $\gamma = 0.125$. The values of C and γ have been optimised through a grid-search which uses a 5-fold cross-validation method.

4.1.2.2 Features

Our system uses 12 features for training the system. The feature descriptions are given below:

Core Language Technology Features: We used existing language processing tools to extract features. The Stanford CoreNLP[3] toolkit provides lemma, parts of speech (POS), named entities and dependency relations of words in each sentence.

We calculated Jaccard similarity between surface form, lemma, dependency relations, POS and named entities of both sentences to get the feature values. The Jaccard similarity computes sentence similarity by dividing the number of common tokens by the total number of tokens of both sentences.

$$Sim(s1, s2) = \frac{|s1 \cap s2|}{|s1 \cup s2|} \tag{4.1}$$

where in equation (4.1), $Sim(s1, s2)$ is the Jaccard similarity between sets of words $s1$ and $s2$.

We calculated two separate feature values for dependency relations: the first feature concatenated the words involved in a dependency relation and the second

[3]http://nlp.stanford.edu/software/corenlp.shtml

used grammatical relation tags. For example, for the sentence pair "the kids are playing outdoors" and "the students are playing outdoors" the Jaccard similarity is calculated based on concatenated words "kids::the, playing::kids, playing::are, ROOT::playing, playing::outdoors" and "students::the, playing::students, playing::are, ROOT::playing, playing::outdoors" to get the value for the first feature and "det, nsubj, aux, root, dobj" and "det, nsubj, aux, root, dobj" to get the value for the second feature.

These core language technology features try to capture the token-based similarity and grammatical similarity between a pair of sentences.

Paraphrasing Features: We used the PPDB paraphrase database (Ganitkevitch et al., 2013) to get the paraphrases. We used lexical and phrasal paraphrases of size "L" . For each sentence of the pair, we created two sets of bags of n-grams ($1 \leq n \leq$ length of the sentence). We extended each set with paraphrases for each n-gram available from the paraphrase database. We then calculated the Jaccard similarity (see Section 4.1.2.2) between these extended bag of n-grams to get the feature value. This feature captures cases where one sentence is a paraphrase of the other.

Negation Feature: Our system does not attempt to model similarity with negation, but since negation is an important feature for contradiction, we designed a non-similarity feature. The system checks for the presence of a negation word such as 'no', 'never' and 'not' in the pair of sentences and returns "1" ("0"

otherwise) if both or none of the sentences contain any of these words.

Machine Translation Evaluation Features: We also used BLEU (Papineni et al., 2002), a very popular machine translation evaluation metric, as a feature. BLEU is based on n-gram counts and is meant to capture the similarity between translated text and references for machine translation evaluation. The BLEU score was calculated over surface, lemma and POS to get three feature values. In a pair of sentences, one side was treated as a translation and another as a reference. We applied it at the sentence level to capture the similarity between two sentences. We use regular BLEU and not smoothed BLEU. Regular BLEU scores zero if there is no matching four-gram between the hypothesis and the reference. We used regular BLEU because our paraphrasing feature already extract the feature based on common n-grams between two sentences, which is somewhat similar to smoothed BLEU.

Corpus Pattern Analysis Feature: Corpus Pattern Analysis (CPA) (Hanks, 2013) is a procedure in corpus linguistics that associates word meaning with word use by means of semantic patterns. CPA is a new technique for mapping meaning onto words in text. It is currently being used to build a "Pattern Dictionary of English Verbs" (PDEV[4]). It is based on the Theory of Norms and Exploitations (Hanks, 2013). PDEV is being created manually and more than 1700 verbs are already tagged.

There is one feature extracted from PDEV. The feature makes use of a derived

[4]http://pdev.org.uk

resource called the CPA network (Bradbury and El Maarouf, 2013). The CPA network links verbs according to similar semantic patterns (e.g. both 'pour' and 'trickle' share an intransitive use where the subject is "liquid"). One of the examples of a CPA pattern is given below with the possible set of verbs following this pattern:

[Human] [Verb] [Rule]

Verb: follow abrogate abolish activate disregard simplify obey break

We can see that all the verbs given above can fit into the pattern. For example, a human can *follow, abrogate, abolish, activate, disregard, simplify, obey,* or *break* a *rule*.

The feature value is calculated as follows. It compares the main verbs in both sentences. When both verbs are the same or both verbs occur in a same pattern, the system returns a value of "1". When the verb in one segment occurs in a pattern but the verb in the other segment does not share the same pattern, the system returns 0. In all other cases, system returns 0.5. We do not detect the pattern but check whether both verbs belong to the same pattern irrespective of the actual pattern used in the text.

As mentioned earlier that we do not use all the features of the original system (Gupta et al., 2014). The features which we did not use include seventeen Machine Translation Quality Estimation (QE) features, one coreference resolution feature and one featured based on corpus pattern analysis. Machine Translation Quality Estimation (QE) features are based on the work of Specia et al. (2009) and used as a baseline in recent QE tasks (such as Callison-Burch et al. (2012)). Some of

the QE features are the number of punctuation marks, the average length of words, the number of words, n-gram frequencies and language model probabilities. A full list of the QE features is provided in the documentation of the QE system[5] (Specia et al., 2009). The coreference resolution feature was calculated using clusters of coreferential entities obtained by treating pair of sentences as a document. Details about these features is available in (Gupta et al., 2014).

4.1.2.3 Corpus used for Training

We do not have access to any sufficiently large dataset in which segments are retrieved from a TM and labelled with scores to the segments on the basis of the quality of the retrieved segments. Therefore, we used the SICK dataset (Marelli et al., 2014b) to train our STS system. The dataset consists of simple sentences extracted mostly from image captions. The dataset has on average 9.6 words per sentence. This dataset is annotated with similarity scores by human annotators. The SICK dataset is developed for evaluating semantic similarity. We used 4934 parallel sentences to train our STS system.

4.1.3 Experiments and Results

We carried out evaluations on English-French data from the DGT-TM corpus . The test set was generated by a random selection of segments. As a baseline, we used the same measure as in Chapter 3 (the word based edit-distance measure). The statistics for our test set is given in the Table 4.1.[6]

[5]https://github.com/lspecia/quest

[6]We selected a small test set because it is complex to take big TM. For our test set, we need to compare Input$\times$TM $(25,00 \times 10,000)$ pairs.

	# segments
Input	2,500
TM	10,000

Table 4.1: Test Set Statistics

We performed both a manual and an automatic evaluation. For our automatic evaluation, we used the machine translation evaluation metrics Meteor (Denkowski and Lavie, 2014) and BLEU (Papineni et al., 2002). For each input segment, we retrieved the most similar sentence (and their proposed translation into French) as indicated by the baseline edit-distance and our STS system. Table 4.2 presents the results of automatic evaluation when having a threshold of 70% over the edit-distance and ignoring exact matches. BLEU-ED represents BLEU score using the baseline edit-distance, BLEU-SS represents BLEU score using our approach, METEOR-ED represents Meteor score using the baseline edit-distance, and METEOR-SS represents Meteor score using our approach. We observe that the proposed method does not yields better results.

Threshold	[70, 100)
BLEU-ED	69.15
BLEU-SS	63.57
METEOR-ED	79.27
METEOR-SS	74.46

Table 4.2: Results Automatic Evaluation

To gain a deeper understanding of our system's performance, we also performed a manual evaluation. We considered the source side (English) of

the segments for this evaluation. A native speaker of English performed the manual evaluation. Three different options were given to the evaluator: Semantic similarity is better, Edit-distance is better, or both are similar. When keeping the 70% threshold and ignoring exact matches, we retrieved 266 fuzzy matched segments. In these 266 segments, 258 segments were tagged as similar, for 6 segments, edit-distance retrieved better and for 2, our semantic similarity approach retrieved better. Some of the examples from Test-2 are given in Table 4.3.

1	Input	For the purposes of this Regulation :
	ED	For the purpose of this demonstration :
	SS	For the purposes of this Regulation the following definitions shall apply :
2	Input	This Decision shall enter into force on the date of its publication in the Official Journal of the European Union .
	ED	This Decision shall enter into force on the day of its publication in the Official Journal of the European Union .
	SS	This Decision shall enter into force on the date of its adoption .
3	Input	The Commission sought and verified all information deemed necessary for the determination of dumping .
	ED	The Commission sought and verified all the information deemed necessary for the purposes of the review .
	SS	The Commission sought and verified all the information provided by interested parties and deemed necessary for the determination of dumping , resulting injury and Union interest .

Table 4.3: Examples from Test-2

In Table 4.3, example 1 shows our approach (SS) performed better, while examples 2 and 3 show edit-distance (ED) performed better.

Although the approach does not perform better overall, there are several

factors, which should be taken into consideration. The genre of the training set and test set were very different. The SICK dataset consists of simple sentences extracted mostly from image captions while the DGT-TM corpus has much larger and complex sentences mainly from the legal domain. The average number of words per segment for TM is 27.9 and for input it is 32.54 for the test set, whereas for the SICK training dataset the average of words per sentence is only 9.63. Furthermore, we do not use many features. For example, in the SemEval 2014 task-1, the best performing systems used more features. Bjerva et al. (2014) used 32 features and Zhao et al. (2014) used 72 features. They also extracted features from WordNet and distributional semantics. Distributional semantic features are derived using distributional vectors of words. Bjerva et al. (2014) used the word2vec model (Mikolov et al., 2013c) and Zhao et al. (2014) used latent semantic analysis to train the word vectors. All these factors suggest substantial scope for improvement in our system. However, we do not extend our system with more features, as extending the system with more features will slow down the system and may not be practical enough to be used in the TM matching and retrieval. Furthermore, the system presented in this section significantly differs in performance compared to the systems presented in the next section.

The system that computes similarity between two segments presented in this section obtains 0.69 Pearson correlation on the SICK dataset, while simple LSTM presented in the next section obtains 0.84 Pearson correlation on the same dataset. Pearson correlation coefficient is a measure of linear correlation between two variables and was used as one of the measures in the SemEval task (Marelli et al.,

2014a). The value of coefficient lies between +1 and -1, where +1 indicates total positive correlation, 0 no correlation and -1 total negative correlation.

4.1.4 Conclusion

In this section, we developed an approach to employ a semantic similarity system in a TM framework. Our experiment shows results slightly comparable to the baseline edit-distance. However, possibly due to some limitations, limited availability of labelled corpora for our task and fewer number of features used, we did not obtain results better than the baseline edit-distance.

In the next section, we extend our work on using semantic matching which tries to compute similarity between segments better than surface form. We present two new approaches based on Recurrent Neural Networks which do not need hand designed features. The approaches presented in the next section makes efficient use of word vectors. Furthermore, we need a better and larger training set. Therefore, we also derive a new training corpus containing more complex sentences compared to the SICK dataset.

4.2 Advanced Semantic Matching for TM using Recurrent Neural Networks

Deep learning is a machine learning technique used in many areas like image processing and natural language processing, and achieved state-of-the-art results in many cases.

In this research, we would like to explore whether deep learning techniques

can provide an efficient way to model segments and to compute the similarity between segments. This is motivated by recent research that used deep learning to calculate similarity between sentences (Gupta et al., 2015b,a; Socher et al., 2011a; Le and Mikolov, 2014). As we stated earlier, TM systems often compute similarity based on surface form which does not capture similarity as judged by humans. In this section, using a distributive representation of segments through deep learning, we compute similarity which implicitly models semantic information like synonymy and paraphrasing and hope to compute better similarity compared to edit-distance.

This section presents our research to improve translation memory matching and retrieval using deep learning and in particular Recurrent Neural Networks (RNNs). We used Long Short Term Memory (LSTM) and Tree structured LSTM (Tree-LSTM) networks for our work. We also present a brief analysis of LSTM and Tree-LSTM architectures and how they can be useful for translation memory matching and retrieval. In the following section (Section 4.2.1) we present general RNN architectures. Section 4.2.2 provides the details of a similarity metric based on LSTMs, Section 4.2.3 presents our approach to automatically derive a similarity training dataset using WMT-13 rankings, Section 4.2.4 presents machine translation evaluation results, Section 4.2.5 presents results of TM matching and retrieval, followed by more experiments, results and analysis in Sections 4.2.6, 4.2.7, 4.2.8 and 4.2.9.

4.2.1 Recurrent Neural Networks

Artificial Neural Networks (ANNs) are mathematical models inspired by the information processing capabilities of biological brains. The basic structure of an ANN is a network consisting of small processing units called nodes, which are joined to each other by weighted connections. In terms of the original biological model, the nodes represent neurons, and the connection weights represent the strength of the synapses between the neurons (Graves, 2012). Connections between nodes may or may not form a cycle.

Recurrent Neural Networks (RNNs) are a type of ANNs which allow cyclic connections. ANNs which do not allow cyclic connections are referred to as feed forward neural networks.

RNNs can recursively model sequences of arbitrary length into a fixed dimension vector. This section explains a basic RNN, LSTM and Tree Structured LSTM network.

4.2.1.1 Basic RNN

The basic RNN obtains a representation by processing the current input and the previous representation in a recursive manner.

Equation 4.2 below defines a RNN.

$$\begin{aligned} h_t &= f(Wx_t + Uh_{t-1} + b) \\ y_t &= \sigma(Vh_t) \end{aligned} \tag{4.2}$$

where f represents a non linear function (generally sigmoid or $\tanh$), h represents

the hidden state representation, x represents the input, y_t represents the output and t represents the time step. W, U and V represent input, update and output weights, respectively. Equation 4.2 shows that a hidden state h_t is computed recursively using a previous hidden state h_{t-1}. A hidden state representation h_t at a given time t can be interpreted as a representation of the sequence observed up to time t. h_t is called hidden state because h_t is not the actual output we usually need for a particular task. Generally a hidden state is processed further by another neural network or the output layer to obtain the output of a particular task. For example, to compute the semantic similarity between two sentences, another neural network may process hidden state representations of both sentences to predict the similarity between them.

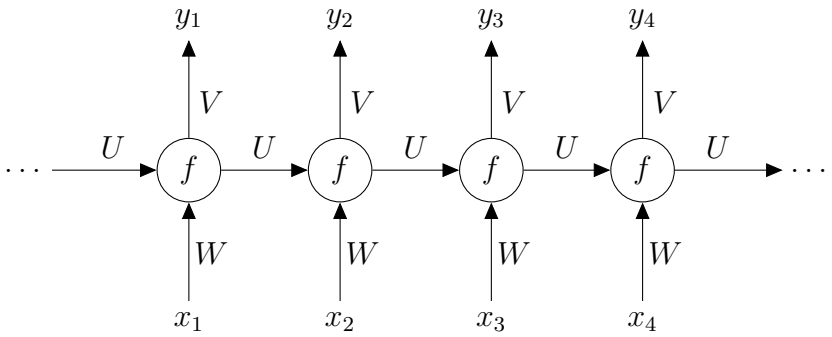


Figure 4.1: An Unfolded Recurrent Neural Network

Figure 4.1 shows an unfolded RNN. The figure is formed by unfolding the network along the input sequence. In Figure 4.1 W, U and V indicate weights. This is a general representation of a RNN. To represent segments and compute the similarity between segments, we typically consider the last hidden state

representation (the representation obtained after processing the whole segment).

4.2.1.2 LSTM

The basic RNN has the problem of vanishing and exploding gradients which makes training difficult (Bengio et al., 1994). In our research, we use LSTM and Tree-LSTM architectures. This section explains the LSTM network architecture and the next section (Section 4.2.1.3) explains the Tree LSTM network architecture. LSTM is an extension of simple recurrent neural networks which tackles the problem of vanishing and exploding gradients by introducing a memory cell with multiplicative input and output gate units. Input gates protect against irrelevant inputs and output gates against current irrelevant memory content. This enables distributed representations of longer sequences. After the initial architecture proposed by Hochreiter and Schmidhuber (1997), several alternative architectures have been proposed. We use a recent architecture as used in Zaremba and Sutskever (2014) and Tai et al. (2015).

The LSTM network used in our work is given as follows:

$$
\begin{aligned}
i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right) \\
f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right) \\
o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right) \\
u_t &= \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right) \\
c_t &= i_t \odot u_t + f_t \odot c_{t-1} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
\tag{4.3}
$$

where i, f and o represent the input, forget and output gates. h represents the hidden state, c represents the cell contents and t represents the time step. In Equation 4.3, the LSTM network updates its cell contents (c_t) based on the previous cell contents (c_{t-1}) and current input update (u_t) which are obtained after applying the forget gate and input gate respectively. Furthermore, the hidden state (h_t) is obtained after applying the output gate (o_t) to the non-linear mapping of the current cell activation ($\tanh(c_t)$).

Figure 4.2 further shows an unfolded LSTM network with the application of input, forget and output gates to represent a segment. For simplicity, Figure 4.2 does not show how input, forget and output gates are updated. As we can see, input gates block any irrelevant input, the forget gate is applied in between the time steps to preserve or delete the information as needed and the output gate helps in getting the desired output by blocking the irrelevant memory contents.

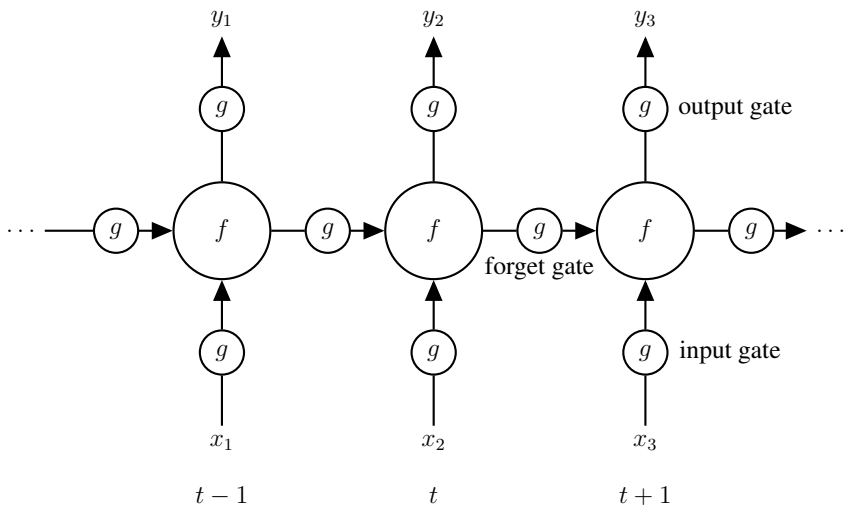


Figure 4.2: An Unfolded LSTM network

For our task of similarity computation between segments in translation memory matching and retrieval, the LSTM network implicitly provides the ability to model the relevance of tokens in calculating similarity by using gates. Furthermore, these gates can be partially closed or open. For example, for a less information carrying word the input gate can be almost closed but not completely closed.

4.2.1.3 Tree Structured LSTM

Tree-LSTM is an extension of the simple LSTM. The simple LSTM processes sequences sequentially (typically left to right) as shown in Figure 4.2. Tree-LSTM processes the sequence recursively with the help of a tree structure. This is particularly helpful for representing sentences and has shown to perform better for various NLP tasks requiring modelling of sentences, including machine translation evaluation, sentiment analysis and semantic relatedness between sentences (Gupta et al., 2015b,a; Tai et al., 2015). A tree structure is formed by parsing the sentence using a constituency or a dependency parser. In our work, we use the Stanford dependency parser and the child sum Tree-LSTM model developed by Tai et al. (2015). The hypothesis of using Tree-LSTM is that we can get a better representation of a sentence by processing it via a tree structure instead of sequentially. The representation is obtained at the root node of the tree instead of last node of the sequence. Furthermore, the child sum Tree-LSTM model applies a separate forget gate between each child and the parent, which provides the network with the ability to partially or fully forget the whole clause,

depending on the relevance to judge similarity between two sentences.

The Tree-LSTM used in our work is defined as follows:

$$
\begin{aligned}
\tilde{h}_j &= \sum_{k \in C(j)} h_k \\
i_t &= \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right) \\
f_{jk} &= \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right) \\
o_t &= \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right) \\
u_t &= \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right) \\
c_j &= i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \\
h_j &= o_j \odot \tanh\left(c_j\right)
\end{aligned}
\tag{4.4}
$$

where $\tilde{h}_j$ represents the sum of the hidden states of the children at node j. i, f and o represent the input, output and forget gates respectively. We can see that the cell content c_j is updated by summing up the cell contents of all children obtained after applying a separate forget gate at each child (f_{jk}) and current input update after applying the input gate.

Figure 4.3 shows an unfolded Tree-LSTM. As we can see in Figure 4.3, the forget gate is applied between each child and parent, and the input gate is applied for input at each node.

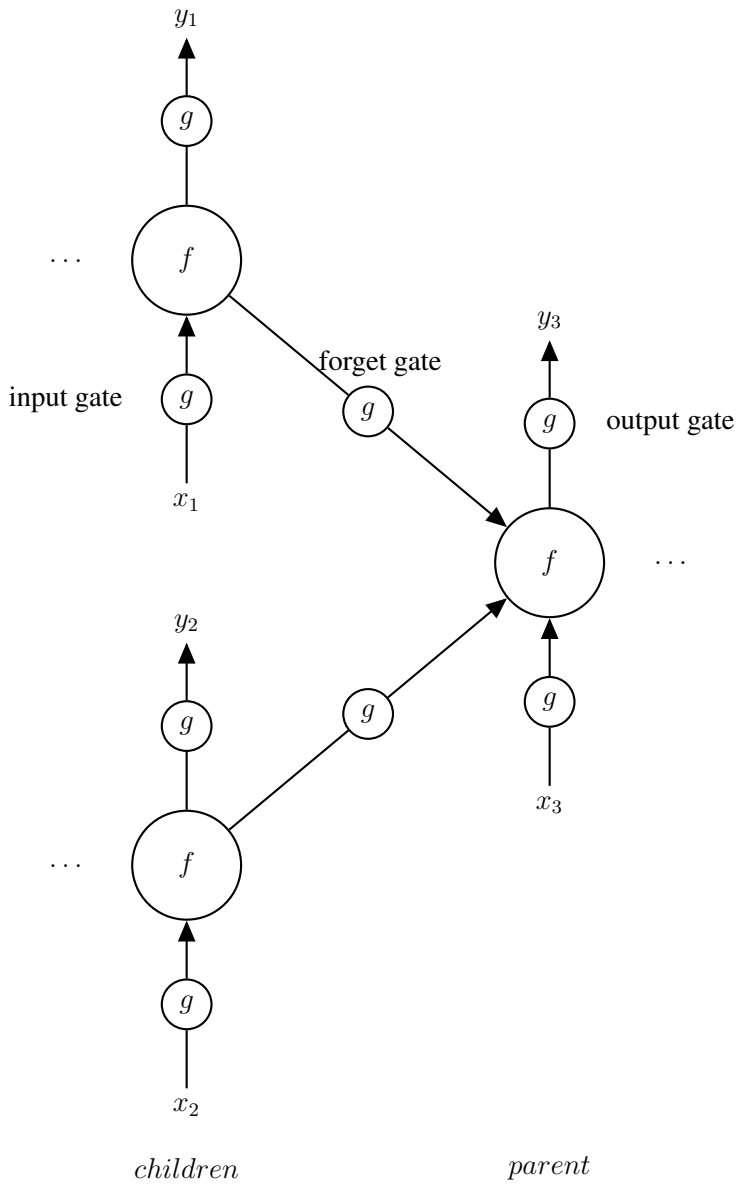


Figure 4.3: An Unfolded Tree LSTM network

4.2.2 A Neural Similarity Metric for Machine Translation Evaluation

In this section, we present our machine translation evaluation metric. We will use the same metric for TM matching and retrieval. For machine translation evaluation, the metric computes similarity between the reference and the hypothesis and for TM matching and retrieval, the metric computes the similarity between the input segment and the TM segment.

We represent both the reference (h_{ref}) and the translation (h_{tra}) using an LSTM or Tree-LSTM and predict the similarity score $\hat{y}$ based on a neural network which considers both distance and Hadamard product between h_{ref} and h_{tra}:

$$\begin{aligned} h_+ &= |h_{ref} - h_{tra}| \\ h_\times &= h_{ref} \odot h_{tra} \\ h_s &= \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right) \\ \hat{p}_\theta &= \text{softmax}\left(W^{(p)} h_s + b^{(p)}\right) \\ \hat{y} &= r^T \hat{p}_\theta \end{aligned} \tag{4.5}$$

where, h_+ represents the distance between h_{ref} and h_{tra}, $h_\times$ represents the Hadamard product, h_s represents the hidden state vector, σ is a sigmoid function, $\hat{p}_\theta$ is the estimated probability distribution vector and $r^T = [1\ 2 ... K]$. $W^{(\times)}$, $W^{(+)}$ and $W^{(p)}$ represents the weights and, $b^{(h)}$ and $b^{(p)}$ represents the biases.

The cost function $J(\theta)$ is defined over probability distributions p and $\hat{p}_\theta$ using regularised Kullback Leibler (KL) divergence.

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{KL}\left(p^{(i)} \middle|\middle| \hat{p}_\theta^{(i)}\right) + \frac{\lambda}{2} ||\theta||_2^2 \tag{4.6}$$

In Equation 4.6, i represents the index of each training pair, n is the number of training pairs and p is the sparse target distribution such that $y = r^T p$ is defined as follows:

$$p_j = \begin{cases} y - \lfloor y \rfloor, & j = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & j = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}$$

for $1 \leq j \leq K$, where, $y \in [1, K]$ is the similarity score of a training pair. For our training we keep $K = 5$ which gives us score between 1 and 5. For example, for $y = 2.7$, $p^T = [0\ 0.3\ 0.7\ 0\ 0]$. We map the score between 1 and 5 to 0 and 1 by the equation 4.7 below:

$$Score0\_1 = \frac{Score1\_5 - 1}{4} \tag{4.7}$$

For our work, we use *GloVe* word vectors (Pennington et al., 2014) and the simple LSTM, the dependency Tree-LSTM and neural network implementations by Tai et al. (2015). Our work is similar to (Tai et al., 2015), where they applied Tree-LSTM for semantic similarity between sentences and sentiment analysis.

Similar to (Tai et al., 2015), we also generate random vectors for unknown words. However, we use unique random-seed specific to a unknown word, which makes sure that we always generate the same vector for the same word but a different vector for a different word. This makes our results on a test set stable and can be exactly replicated, which is not possible with random vectors for unknown words used in (Tai et al., 2015).

(Tai et al., 2015) implementation requires both training and testing data at the same time. We removed such dependency so that we can test the same trained model on different sets.[7]

The system uses the scientific computing framework Torch.[8] For machine translation evaluation metric, training is performed on the data computed in Section 4.2.3. The system uses a mini batch size of 25 with a learning rate of 0.05 and a regularization strength of 0.0001. The LSTM dimension is 150 and the similarity module hidden vector dimension is 50. The training is performed for 10 epochs. System-level scores are computed by aggregating and normalising segment-level scores.

Figure 4.4 summarises the procedure for similarity prediction. The LSTM or Tree-LSTM network takes sentences as input and produces the vector representation of these sentences. These vector representations further feed into the feed forward neural network which considers both the absolute difference and the Hadamard product between the two vectors and predicts the target probability distribution $\hat{p}$.

[7]The updated code for MT evaluation is available at https://github.com/rohitguptacs/ReVal.
[8]http://torch.ch

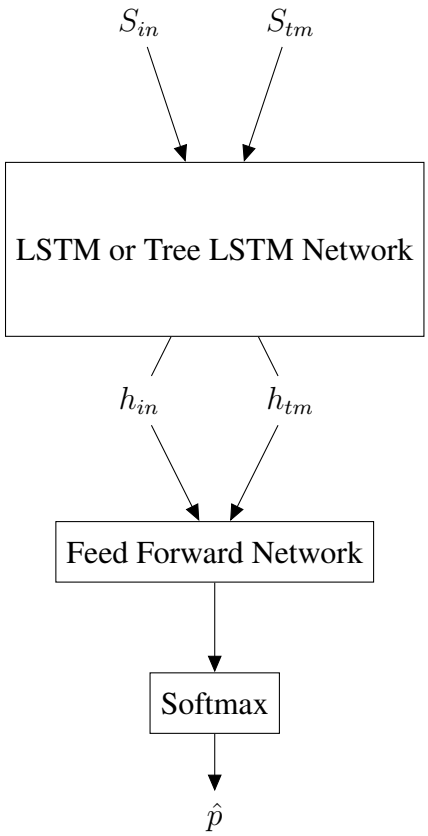


Figure 4.4: Network Architecture for Similarity Prediction (h_{in} and h_{tm} are computed independently using S_{in} and S_{tm} respectively)

4.2.3 Creating a Dataset from WMT Rankings

As we do not have access to any dataset which provides scores to segments on the basis of how well they were translated, we used the WMT-13 ranks corpus (Bojar et al., 2013) to automatically derive training data. This corpus is a by-product of the manual system evaluation carried out in the WMT-13 evaluation. In the evaluation, the annotators are presented with a source segment, the output of five systems and a reference translation. The annotators are given the following instructions: "*You are shown a source sentence followed by several candidate*

translations. Your task is to rank the translations from best to worst (ties are allowed)". Using the WMT-13 ranked corpus, we derived a corpus where the reference and corresponding translations are assigned similarity scores. The fact that *ties are allowed* makes it more suitable to generate similarity scores. If all translations are bad, annotators can mark all as rank 5 and if all translations are accurate, annotators can mark all as rank 1. The selection of the WMT-13 corpus over the corpora released as a part of other WMT workshops is motivated by the fact that it is the largest among them. It contains ten times more rankings than WMT-12 and three to four times more rankings than WMT-14 (Callison-Burch et al., 2012; Bojar et al., 2014). This makes it possible to obtain enough reference-translation pairs which are evaluated several times.

Our hypothesis is that if a translation is given similar ranks many times, the average of the ranks reflects its similarity score with the reference. A better ranked translation among many systems will be close to the reference whereas a worse ranked translation among many systems will be dissimilar from the reference. If a translation is given similar ranks many times, in other words, if the variance in ranks is low, we can be more sure that the average of the ranks will reflect the similarity score between the reference and the translation. Therefore, we compute variance in the ranks to filter noisy candidates and collect reference-translation pairs below a certain variance only. We determined appropriate variance values using Algorithm 6 below for $n = 3, 4, 5, 6, 7$ and ≥ 8, separately. The computed variance values are given in Table 4.4.

In Algorithm 6, the $kendall$ function calculates the Kendall tau correlation

n	3	4	5	6	7	≥ 8
Var	0.65	1.0	1.2	1.2	1.3	0.85

Table 4.4: Variances Computed using Algorithm 6

Algorithm 6 Variance Computation

```
1: procedure GETVARIANCE(judgements)
2:     V, v ← −1, 0.25                                  ▷ Initialise N
3:     for v ≤ max do
4:         pairs ← pairs with variance below v
5:         score ← kendall(pairs, judgements)
6:         if score ≥ 0.78 then
7:             V ← v
8:             v ← v + 0.05
9:         else
10:            break
11:        end if
12:    end for
13:    Return V                                         ▷ Return variance
14: end procedure
```

using the WMT-13 human judgements. We select a set for which the correlation coefficient is greater than 0.78.[9] The correlation is computed using the annotations for which scores are available in the corpus ($pairs$). In other words, the corpus acts as a scoring function for the available reference-translation pairs, which gives a similarity score between a reference and a translation. We select pairs below the variance values obtained for $n = 4, 5, 6, 7$ and ≥ 8. Finally, all the pairs are merged to obtain the corpus tagged with similarity scores (WMT-13). We obtained 11,559 sentence pairs using this technique. We kept 9,559 for training, 1,000 for development and 1,000 for test.[10] We also merged 4,500 sentence pairs of the SICK data (Marelli et al., 2014b) with the training set and 500 sentence pairs to the development set. In total, 14,059 pairs are used for training (9,559

[9]The score was decided so that we obtain around 10K pairs which are annotated at least four times.

[10]1,000 pairs are kept as a test set. However, we use WMT-12 and WMT-14 rankings instead of this test set for testing.

from WMT-13 + 4,500 from SICK) and 1,500 pairs are used for development (1,000 from WMT-13 + 500 from SICK).

4.2.4 Results on Machine Translation Evaluation

In this section, we evaluate our metric ReVal, which is based on Tree-LSTM. We also compare ReVal with the simple LSTM. We evaluated our metrics using the submissions of the metrics participated in the WMT-14 metric task and official scores (which are obtained after human evaluations) of the machine translation systems participated in the WMT-14 machine translation task. In the WMT-14 metric task, the performance of a metric at the system-level was evaluated by computing Pearson and Spearman correlations between the scores produced by a metric against the official scores of the machine translation task.

Table 4.5 shows system-level Pearson correlation obtained on five different language pairs as well as average Pearson correlation (PAvg) over all language pairs. The last column of the table also shows average Spearman correlation (SAvg). The 95% confidence level scores are obtained using bootstrap resampling as used in the WMT-2014 metric task evaluation. The scores in bold show best scores overall and the scores in italic show best scores between Simple LSTM and ReVal.

In Table 4.5, the first part shows the results obtained using Simple LSTM and ReVal. The second part shows the best three overall systems from the WMT-14 metric task. The third part shows the systems from the WMT-14 task which obtained best results for certain languages but do not preform well overall. The

Test	**cs-en**	**de-en**	**fr-en**	**hi-en**	**ru-en**	**PAvg**	**SAvg**
Simple LSTM	.922 ± .051	.882 ± .028	.974 ± .009	.898 ± .011	.863 ± .023	.908 ± .024	.872 ± .060
ReVal	**.993** ± .017	.904 ± .025	.978 ± .008	.908 ± .010	**.881** ± .022	.933 ± .016	**.915** ± .042
DISCOTK-PARTY-TUNED	.975 ± .031	**.943** ± .020	.977 ± .009	.956 ± .007	.870 ± .022	**.944** ± .018	.912 ± .043
LAYERED	.941 ± .045	.893 ± .026	.973 ± .009	**.976** ± .006	.854 ± .023	.927 ± .022	.894 ± .047
DISCOTK-PARTY	.983 ± .025	.921 ± .024	.970 ± .010	.862 ± .015	.856 ± .023	.918 ± .019	.856 ± .046
REDSYS	.989 ± .021	.898 ± .026	**.981** ± .008	.676 ± .022	.814 ± .026	.872 ± .021	.786 ± .047
REDSYSSENT	**.993** ± .018	.910 ± .024	.980 ± .008	.644 ± .023	.807 ± .027	.867 ± .020	.771 ± .043
BLEU	.909 ± 0.54	.832 ± .034	.952 ± .012	.956 ± .007	.789 ± .027	.888 ± .027	.833 ± .058
Meteor	.980 ± .029	.927 ± .022	.975 ± .009	.457 ± .027	.805 ± .026	.829 ± .023	.788 ± .046

Table 4.5: Results: System-Level Correlations on WMT-14 (cs:czech, de:German, en:English, fr:French, hi:Hindi, ru:Russian)

last part shows systems implementing BLEU and Meteor in the WMT-14 metric task.

Tables 4.5 shows that ReVal significantly outperforms the simple LSTM model. Compared with the other participating metrics, ReVal is fully competitive with the best of the current complex approaches that combine many different metrics, substantial external resources and may require a significant amount of feature engineering and tuning.

For example, DISKOTK-PARTY-TUNED (Joty et al., 2014) uses five different discourse metrics and twelve different metrics from the ASIYA MT evaluation toolkit (Giménez and Màrquez, 2010). The metric computes the number of common sub-trees between a reference and a translation using a convolution tree kernel (Collins and Duffy, 2001). The basic version of the metric does not perform well but in combination with the other 12 metrics from the ASIYA toolkit obtained the best results for the WMT-14 metric shared task. LAYERED (Gautam and Bhattacharyya, 2014), uses linear interpolation of different metrics. LAYERED uses BLEU and TER to capture lexical similarity, Hamming score and Kendall Tau Distance (Birch and Osborne, 2011) to identify syntactic similarity, and dependency parsing (De Marneffe et al., 2006) and the Universal Networking Language[11] for semantic similarity.

Table 4.6 shows segment-level results on the WMT-14 task dataset. Segment-level results are computed using Kendall tau correlations with human judgements obtained in the WMT-14 task. In Table 4.6, 'Avg wmt12' represents the average

[11] http://www.undl.org/unlsys/unl/unl2005/UW.htm

Test	**cs-en**	**de-en**	**fr-en**	**hi-en**	**ru-en**	**Average**	**Avg wmt12**
Simple LSTM	$.204 \pm .015$	$.232 \pm .014$	$.289 \pm .013$	$.319 \pm .013$	$.236 \pm .012$	$.256 \pm .013$	$.254 \pm .013$
ReVal	$.243 \pm .016$	$.274 \pm .013$	$.333 \pm .013$	$.360 \pm .014$	$.278 \pm .011$	$.298 \pm .013$	$.295 \pm .014$
DISCOTK-PARTY-TUNED	$\mathbf{.328} \pm .014$	$\mathbf{.380} \pm .014$	$\mathbf{.433} \pm .013$	$.434 \pm .013$	$\mathbf{.355} \pm .010$	$\mathbf{.386} \pm .013$	$\mathbf{.386} \pm .013$
BEER	$.284 \pm .015$	$.337 \pm .014$	$.417 \pm .013$	$\mathbf{.438} \pm .014$	$.333 \pm .011$	$.362 \pm .013$	$.358 \pm .013$
REDCOMBSENT	$.284 \pm .015$	$.338 \pm .013$	$.406 \pm .012$	$.417 \pm .014$	$.336 \pm .011$	$.356 \pm .013$	$.346 \pm .013$
METEOR	$.282 \pm .015$	$.334 \pm .014$	$.406 \pm .012$	$..420 \pm .013$	$.329 \pm .010$	$.354 \pm .013$	$.341 \pm .013$
BLEU_NRC	$.226 \pm .014$	$.272 \pm .014$	$.382 \pm .013$	$.322 \pm .013$	$.269 \pm .011$	$.294 \pm .013$	$.267 \pm .013$
SENTBLEU	$.213 \pm .016$	$.271 \pm .014$	$.378 \pm .013$	$.300 \pm .013$	$.263 \pm .011$	$.285 \pm .013$	$.258 \pm .014$

Table 4.6: Results: Segment-Level Correlations on WMT-14

performance across all language pairs when evaluating based on the Kendall tau variant used in the WMT-12 metric task. For the segment-level, our metric outperforms BLEU based approaches and the other three systems[12] but lags behind some other approaches.

We also evaluated ReVal on the WMT-12 task dataset. Our metric performed best for two out of four language pairs and best overall at the system level with 0.950 and 0.926 Pearson and Spearman correlation coefficient, respectively. At the segment level, we obtained 0.222 Kendall Tau correlation which was better than seven out of the total ten metrics in the WMT-12 task.

One of the reasons for the difference in segment-level and system-level correlations is that Kendall Tau segment-level correlation is calculated based on rankings and does not consider the amount of difference between scores. Here is an example similar to that given in Hopkins and May (2013). Suppose four systems produce the translations T0, T1, T2 and T3. Suppose we have two metrics M1 and M2 and they produce scores and rankings as follows. GS represents the correct ranking and scores; Scores are in a scale [0, 1] with a higher score indicating a better translation:

M1: T0 (0.10), T3 (0.71), T1 (0.72), T2 (0.73)

M2: T1 (0.71), T0 (0.72), T2 (0.73), T3 (0.74)

GS: T0 (0.10), T1 (0.71), T2 (0.72), T3 (0.73)

Certainly, M1 produces better scores and ranking than M2. But, Kendall Tau

[12]These three systems are not given in this thesis. See Machácek and Bojar (2014) for results of these systems.

segment-level correlation is higher for M2. (There are four concordant pairs in the M1 rank and five in the M2 rank.) Therefore, if a metric does not scale well as per the quality of translations, it may still obtain a good Kendall Tau segment-level correlation and a better metric may end up getting a low correlation. Another reason for the discrepancy between segment and system-level scores may be a low agreement on annotations. For the WMT-14 dataset, inter-annotator and intra-annotator agreement were 0.367 and 0.522. These problems should not occur with Pearson correlation at the system level because system-level scores are calculated using more sophisticated approaches (Koehn, 2012; Hopkins and May, 2013; Sakaguchi et al., 2014). For example, Hopkins and May (2013) model the differences among annotators by adding random Gaussian noise.

Our metric ReVal also participated in the WMT-15 and WMT-16 tasks (Stanojević et al., 2015; Bojar et al., 2016). In the WMT-16 task, for each language pair, 10,000 artificial systems using sampling from participated systems were created to improve the evaluations. For both years, our metric performed well and obtained second best results at the system level.[13]

4.2.5 ReVal for TM Matching and Retrieval

In this section, we present our experiments using ReVal for TM matching and retrieval. Instead of computing the similarity between the reference and the translation for MT evaluation, the metric is used to compute similarity between the input segment and the TM segment.

[13]Detailed results in the WMT-15 and WMT-16 tasks are given in Gupta et al. (2015a) and Bojar et al. (2016).

For our experiments, we do not process each segment of the TM using ReVal; instead we use filtering steps to get the potential segments as used in Chapter 3 for paraphrasing (see Section 3.2.3). The filtering steps for obtaining potential candidates are as follows:

1. LENFILTER: TM segments are discarded if the TM segments are shorter than 39% of the input or vice-versa.

2. SIMFILTER: Next, we filter the segments based on baseline edit-distance similarity. TM segments which have a similarity below 39% are removed.

3. MAXFILTER: Next, after filtering the candidates with the above two steps we sort the remaining segments in decreasing order of baseline edit-distance similarity and pick the top 100 segments.

4. BEAMFILTER: Finally segments within a certain range of similarity with the most similar segment are selected to be processed using an LSTM. In our case, the range is 35%. This means that if the most similar segment has 95% similarity, segments with a similarity below 60% are discarded.

The statistics of the segments retrieved after filtering steps are given in Table 4.7, which shows the number of segments in the specified interval (#Segments), the number of segments for which we retrieve at least two segments from the TM in the specified interval (#SegmentsMin2), and the average number of segments retrieved from the TM in the specified interval (Average).

S1 and S2 refer to the two different preprocessing settings. In S1, only

tokenization is performed as a preprocessing stage, and in S2, punctuation marks are removed in both the source and target sides. The DGT-TM dataset is of legal genre and contains many punctuation marks. We deleted the punctuation marks to see how it affects the baseline edit-distance and ReVal scores. As Table 4.7 shows for threshold interval [70, 85) we get around 13 segments to rerank in both settings S1 and S2. For the [85, 100) interval we get around two segments for reranking in S1 and S2.

		[85, 100)	**[70, 85)**	**[55, 70)**
S1	#SegmentsMin2	705	1805	1668
	#Segments	1084	1984	1800
	Average	2.07	13.57	10.55
S2	#SegmentsMin2	716	1847	1548
	#Segments	1032	2047	1708
	Average	2.19	13.41	11.02

Table 4.7: Filtering Statistics for English-German

Table 4.8 shows the results of using ReVal on data from the English-German language pair. It is difficult to judge how many more segments can be retrieved when using the similarity score based on ReVal. Therefore, in this chapter we compared the quality of retrieved segments only. We hypothesize that if the quality of the retrieved segments is improved, we can lower the threshold to get more matches.

To compare ReVal performance with the baseline edit-distance, we compared segments retrieved across threshold intervals [85, 100), [70, 85) and [55, 70). These threshold intervals are decided by the baseline edit-distance so for example

a [85, 100) interval contains segments having 85% or more but less than 100% fuzzy match score using the baseline edit-distance. We do not consider segments with 100% fuzzy match score because they are exact matches.

Table 4.8 show similarity threshold intervals (Threshold Interval), and the number of segments retrieved using the baseline edit-distance in a Threshold Interval (#Segments). METEOR-ed, BLEU-ed and TER-ed represent the Meteor scores, the BLEU scores and the TER scores over translations retrieved by the baseline edit-distance, while METEOR-ReVal , BLEU-ReVal and TER-ReVal represent the Meteor scores, the BLEU scores and the TER scores for the corresponding match retrieved by ReVal. For Meteor and BLEU, higher scores are better and for TER, lower scores are better.

Table 4.8 shows that we do not obtain improvements on all three measures, Meteor, BLEU and TER. For the [85, 100) threshold interval, we observe 0.6 absolute points decrease in the Meteor score in setting S1, and 0.9 absolute points decrease in the Meteor score in setting S2. For the [70, 85) threshold interval, we observe 4.5 absolute points decrease in the Meteor score in setting S1, and 2.9 absolute points decrease in the Meteor score in setting S2.

	Threshold Interval	**[85, 100)**	**[70, 85)**	**[55, 70)**
S1	#Segments	1193	1029	1259
	METEOR-ed	83.8	65.7	48.1
	METEOR-ReVal	83.2	61.2	44.1
	BLEU-ed	78.3	57.6	36.9
	BLEU-ReVal	77.4	51.9	32.4
	TER-ed	13.7	32.3	49.7
	TER-ReVal	14.4	38.1	57.1
S2	#Segments	1078	889	1070
	METEOR-ed	82.3	67.3	48.6
	METEOR-ReVal	81.4	64.4	45.2
	BLEU-ed	76.5	60.3	37.9
	BLEU-ReVal	75.0	56.7	33.8
	TER-ed	14.9	32.3	51.3
	TER-ReVal	16.2	36.1	56.2

Table 4.8: Tree-LSTM (ReVal) Results for English-German

4.2.6 More Experiments using LSTM and Tree-LSTM for TM Matching

In the previous section (Section 4.2.5), we do not observe improvements using ReVal as a similarity metric for TM matching and retrieval. In an attempt to obtain better results, in this section, we extend our work and compare eight different models as given in Section 4.2.6.3.

Our aim is to compare all eight models with the baseline edit-distance. We also aim to compare different LSTM models with each other to access: the performance of simple LSTM models compared to Tree LSTM models; the performance of the models which include TM as a part of the word vectors training corpus compared to the models which do not include TM as a part of the word vectors training corpus.

4.2.6.1 Training Data

To obtain more training data, apart from the dataset derived in Section 4.2.3, we added the similarity tagged data available from the SemEval 2012, 2013, 2014 and 2015 workshops (Agirre et al., 2012, 2013, 2014, 2015). The data contains various types including headlines, questions-answers from an online website, image captions and Europarl. We obtained 10,592 sentences for training and 3,000 for development. Using these datasets we created two training sets, Set-1 and Set-2. Set-1 is the same as used in Section 4.2.5. Set-2 contains additional pairs from the SemEval 2012, 2013, 2014 and 2015 workshops. The details of both datasets are given below:

Set-1: 14,059 pairs for training (9,559 from WMT-13 + 4,500 from SICK) and 1,500 pairs for development (1,000 from WMT-13 + 500 from SICK)

Set-2: 24,651 pairs for training (9,559 from WMT-13 + 4,500 from SICK + 10,592 from SemEval 2012-2014) and 4,500 pairs for development (1,000 from WMT-13 + 500 from SICK + 3,000 from SemEval 2015).

4.2.6.2 Word Vectors

We use three sets of word vectors to train our different models. Details of these word vectors are given below:

840b: Pre trained GloVe word vectors, which are trained on the 840 billion token Common Crawl corpus (Pennington et al., 2014). The GloVe vectors

are obtained from the web.[14]

Wiki: Word vectors trained on English Wikipedia containing 1.9 billion tokens using GloVe (Pennington et al., 2014). Training is performed for 100 iterations with window size of 15 words.

Wiki+TM: Added TM used in our experiments to English Wikipedia, while the rest of the settings are same as 'Wiki'. Our hypothesis is that by adding TM as a part of word vectors training, we may make word vectors more appropriate for our task.

We also tested the models to observe the performance of word vectors themselves. We have used the analogy dataset (Mikolov et al., 2013a) which is typically used to evaluate the quality of word vectors. The vectors are tested on analogy questions like, sister : brother :: woman : ?. The dataset contains 19,544 questions classified into two subsets: the semantic subset and the syntactic subset. The semantic questions are generally about people or places, and the syntactic questions are generally about adjectives and verb forms.

On the analogy dataset, word vector performances are as given in Table 4.9. In Table 4.9, "Semantic" and "Syntactic" represent the results on semantic and syntactic subsets, respectively. Table 4.9 shows that, overall, '840b' which is trained on 840 billion common crawl corpus performs best. Wiki and Wiki+TM have similar quality. However, Wiki is slightly better compared to Wiki+TM, which shows that adding the TM does not improve the word vector performance

[14]http://nlp.stanford.edu/projects/glove/

on the test data. One of the reasons is that the DGT-TM corpus has automatically aligned segments and the segments are not coherent like Wikipedia text, which has large paragraphs, and the sentences are placed coherently. As we know, the core part of word vector training is learning from the context, this may be one of the reasons for a slight decrease in the quality of word vectors after adding TM as part of the word vector training.

	Semantic	**Syntactic**	**Total**
840b	78.39	75.67	76.90
Wiki	79.91	66.44	72.55
Wiki+TM	79.40	66.06	72.11

Table 4.9: Quality of Word Vectors

4.2.6.3 Deep Learning Models

This section gives details of our different models. We have two different LSTM networks (simple LSTM and Tree-LSTM), two different training datasets as given in Section 4.2.6.1 and three different word vectors as given in Section 4.2.6.2. Using these networks and datasets, we trained eight different models given below:

ReVal: The first model is trained on Set-1, 840b with Tree-LSTM. This is the same model used for machine translation evaluation and used in the experiments given in Section 4.2.5.

WikiTMTree: The second model is trained on Set-2, Wiki+TM with Tree-LSTM. This model and the 'WikiTree' model given below are trained to see whether including TM as a part of training word vectors improves TM

matching performance.

WikiTree: The third model is trained on Set-2, Wiki with Tree-LSTM. This model is a baseline for 'WikiTMTree' as it does not include TM in the training of word vectors. The rest of the settings are the same as WikiTMTree.

WikiTMlstm: The fourth model is trained on Set-2, Wiki+TM with simple LSTM. This model and the 'Wikilstm' model given below are the same as 'WikiTMTree' and 'WikiTree' respectively, except that we use simple LSTM instead of Tree-LSTM. Using this model, we want to compare Tree-LSTM and simple LSTM for the TM matching task.

Wikilstm: The fifth model is trained on Set-2, Wiki with simple LSTM. This model is a baseline for 'WikiTMlstm' as it does not include TM in the training of word vectors. The rest of the settings are the same as WikiTMlstm.

840blstmSet1: The sixth model is trained on Set-1, 840b with simple LSTM. We trained this model to compare simple LSTM with Tree-LSTM.

840blstmSet2: The seventh model is trained on Set-2, 840b with simple LSTM. We trained this model to compare simple LSTM trained on Set-1 with simple LSTM trained on Set2.

840bTreeSet2: The eighth model is trained on Set-2, 840b with Tree-LSTM. We trained this model to compare Tree-LSTM trained on Set-1 with

Tree-LSTM trained on Set2.

Table 4.10 summarises our all eight models.

Model	**Word Vectors**	**Training Set**	**Type of LSTM**
ReVal	840b	Set-1	Tree-LSTM
WikiTMTree	Wiki+TM	Set-2	Tree-LSTM
WikiTree	Wiki	Set-2	Tree-LSTM
WikiTMlstm	Wiki+TM	Set-2	simple LSTM
Wikilstm	Wiki	Set-2	simple LSTM
840blstmSet1	840b	Set-1	simple LSTM
840blstmSet2	840b	Set-2	simple LSTM
840bTreeSet2	840b	Set-2	Tree-LSTM

Table 4.10: Eight LSTM Models

4.2.6.4 Results and Analysis

In this section, we present the results obtained using all eight models as given in Section 4.2.6.3. For our experiments, we use the same TM and Test datasets, the same filtering steps, and the same preprocessing settings S1 and S2 as used in Section 4.2.5, where we evaluated TM matching and retrieval using ReVal.

Similar to Table 4.8, Table 4.11 shows results in fuzzy match threshold intervals (Threshold Interval) and these threshold intervals are decided by the baseline edit-distance. We also add intervals [70, 100) and [55, 100) to obtain the overall performance of a model. #Segments represents the number of segments in a Threshold Interval. We use the Meteor, BLEU and TER scores to compare the quality of retrieved segments. In Table 4.11, bold font indicates the best and bold-italic font indicates the best in eight LSTM models.

Table 4.11 shows that we do not observe improvements compared to the baseline edit-distance using any of the eight LSTM modes in both settings (S1 and S2). However, we observe a slightly lower decrease in the scores in setting S2 compared to setting S1.

We observe that Tree-LSTM models retrieve better results compared to the simple LSTM models. If we compare 'WikiTMTree' and 'WikiTMlstm', in setting S1, we observe that the 'WikiTMTree' model obtains 63.3 Meteor score compared to 61.7 Meteor score obtained by the 'WikiTMlstm' model in the [55, 100) threshold interval.

Table 4.11 shows that the addition of training data marginally improves the results. For the [70, 100) and [55, 100) threshold intervals, '840blstmSet2' obtains better Meteor, BLEU and TER scores compared to '840blstmSet1' for setting S2. Similarly, for the [70, 100) and [55, 100) threshold intervals, the '840bTreeSet2' model obtains the better Meteor, BLEU and TER scores, except for the marginal increase in TER for the threshold interval [55, 100), compared to 'ReVal' in settings S2. Both models differ only in the use of the training data: 'ReVal' uses Set-1 for training whereas, '840bTreeSet2' uses Set-2. The word vectors and the other settings are the same.

We also observe that a slight difference in the quality of word vectors does not impact the retrieval results. We can see that for the [70, 100) and [55, 100) threshold intervals, in setting S2, 'WikiTree' obtains similar Meteor, BLEU and TER scores compared to '840bTreeSet2'.

To see the impact of using TM as part of word vectors training, we compare

'WikiTMTree' with 'WikiTree', and 'WikiTMlstm' with 'Wikilstm'. We have not observed significant differences when we include TM as part of word vectors training. Table 4.11 shows that for the [55, 100) threshold interval, 'WikiTMTree' and 'WikiTree' have the same 63.3 Meteor score in setting S1 and, 64.6 and 64.5 Meteor scores respectively in setting S2. 'WikiTMlstm' and 'Wikilstm' obtain 61.7 and 62.3 Meteor scores respectively in setting S1, and 64.2 and 63.8 Meteor scores respectively, in setting S2, for the [55, 100) threshold interval.

It is difficult to distinguish the best LSTM model, but the 'WikiTree' model obtains best results in setting S1 and close to the best in setting S2 for the [55, 100) and [70, 100) threshold intervals.

In this section, we presented results on the baseline edit-distance and eight LSTM models on data from the English-German language pair. In the next section, we conduct more experiments and analyse results on two more language pairs, English-Spanish and English-French.

	Threshold Interval	**[85, 100)**	**[70, 85)**	**[55, 70)**	**[70, 100)**	**[55, 100)**
S1	#Segments	1193	1029	1259	2222	3481
	METEOR-ed	**83.8**	**65.7**	**48.1**	**75.8**	**65.7**
	METEOR-WikiTMTree	83.0	62.7	***44.6***	74.0	***63.3***
	METEOR-WikiTree	82.8	***63.1***	44.5	***74.1***	***63.3***
	METEOR-WikiTMlstm	82.5	59.7	43.0	72.4	61.7
	METEOR-Wikilstm	81.9	61.4	43.8	72.8	62.3
	METEOR-ReVal	***83.2***	61.2	44.1	73.4	62.8
	METEOR-840blstmSet1	82.0	58.5	42.0	71.6	60.9
	METEOR-840blstmSet2	82.7	58.6	41.0	72.1	60.8
	METEOR-840bTreeSet2	82.9	61.3	44.1	73.3	62.7
	BLEU-ed	**78.3**	**57.6**	**36.9**	**69.6**	**58.0**
	BLEU-WikiTMTree	77.3	54.0	32.4	67.2	54.8
	BLEU-WikiTree	***77.4***	***54.3***	***32.9***	***67.4***	***55.2***
	BLEU-WikiTMlstm	76.5	50.8	31.8	65.7	53.8
	BLEU-Wikilstm	76.2	52.2	32.3	65.7	53.9
	BLEU-ReVal	***77.4***	51.9	32.4	66.5	54.5
	BLEU-840blstmSet1	76.1	49.5	30.7	64.8	52.8
	BLEU-840blstmSet2	76.5	49.9	30.6	65.6	53.2
	BLEU-840bTreeSet2	77.0	52.5	***32.9***	66.5	54.6
	TER-ed	**13.7**	**32.3**	**49.7**	**21.9**	**32.0**
	TER-WikiTMTree	14.7	38.4	58.4	25.2	37.2
	TER-WikiTree	***14.6***	***37.4***	***55.3***	***24.7***	***35.7***
	TER-WikiTMlstm	15.7	40.6	58.4	26.7	38.2
	TER-Wikilstm	15.6	39.9	59.1	26.3	38.2
	TER-ReVal	14.4	38.1	57.1	24.9	36.5
	TER-840blstmSet1	15.9	41.7	59.2	27.3	38.8
	TER-840blstmSet2	15.6	41.6	61.6	27.1	39.6
	TER-840bTreeSet2	***14.6***	39.2	57.4	25.5	37.0
S2	#Segments	1078	889	1070	1967	3037
	METEOR-ed	**82.3**	**67.3**	**48.6**	**75.6**	**66.5**
	METEOR-WikiTMTree	***81.8***	64.9	***45.5***	74.2	***64.6***
	METEOR-WikiTree	***81.8***	64.8	45.3	74.2	64.5
	METEOR-WikiTMlstm	***81.8***	64.2	44.9	73.9	64.2
	METEOR-Wikilstm	81.0	64.2	44.8	73.4	63.8
	METEOR-ReVal	81.4	64.4	45.2	73.8	64.2
	METEOR-840blstmSet1	80.5	62.3	43.7	72.3	62.7
	METEOR-840blstmSet2	80.9	63.7	44.3	73.2	63.5
	METEOR-840bTreeSet2	81.7	***65.3***	45.3	***74.3***	64.5
	BLEU-ed	**76.5**	**60.3**	**37.9**	**69.3**	**59.0**
	BLEU-WikiTMTree	***75.7***	57.1	34.1	67.4	***56.6***
	BLEU-WikiTree	75.5	56.8	***34.2***	67.2	56.4
	BLEU-WikiTMlstm	***75.7***	56.3	34.0	67.1	56.3
	BLEU-Wikilstm	74.6	56.3	34.1	66.5	56.0
	BLEU-ReVal	75.0	56.7	33.8	66.9	56.2
	BLEU-840blstmSet1	73.1	54.9	33.0	65.4	54.8
	BLEU-840blstmSet2	74.4	55.7	33.9	66.3	55.7
	BLEU-840bTreeSet2	75.2	***57.4***	33.9	***67.5***	***56.6***
	TER-ed	**14.9**	**32.3**	**51.3**	**22.7**	**32.3**
	TER-WikiTMTree	***15.8***	36.8	56.8	25.2	35.8
	TER-WikiTree	***15.8***	36.6	57.6	25.1	36.0
	TER-WikiTMlstm	15.9	37.7	57.6	25.7	36.4
	TER-Wikilstm	16.6	37.9	58.8	26.1	37.1
	TER-ReVal	16.2	36.1	***56.2***	25.1	***35.5***
	TER-840blstmSet1	17.8	39.2	59.0	27.4	38.0
	TER-840blstmSet2	17.3	38.3	59.1	26.7	37.5
	TER-840bTreeSet2	16.2	***35.5***	57.2	***24.8***	35.7

Table 4.11: Comparing All LSTM models on English-German

4.2.7 Performance of Tree-LSTM on Other Language Pairs

In this section, we present the experiments on three language pairs: English-German, English-Spanish and English-French. We use the DGT-TM input test and TM datasets as used in Chapter 3 (Section 3.4) and use the same filtering steps as used in Section 4.2.5. We compare the baseline edit-distance and the 'WikiTree' model, which obtains overall better results compared to the other LSTM models proposed in Section 4.2.6.

Similar to Section 4.2.6, Table 4.12 shows the results over five threshold ranges. We calculate the Meteor, BLEU and TER scores for the segments in the threshold intervals.

Table 4.12 shows that we obtained similar results for all three language pairs in both settings S1 and S2. We observe that Meteor, BLEU and TER for edit-distance is always better compared to the baseline edit-distance.

The methods developed in this chapter use RNNs for calculating the similarity between segments. The hope was that by using this approach, we can obtain the excellent results like we obtained for MT evaluation. However, our evaluation shows that the baseline edit-distance retrieves better segments than the proposed methods.

In the next section, we present human evaluation and perform further analysis of the results.

		Threshold Interval	**[85, 100)**	**[70, 85)**	**[55, 70)**	**[70, 100)**	**[55, 100)**
S1	DE	#Segments	1193	1029	1259	2222	3481
		METEOR-ed	83.8	65.7	48.1	75.8	65.7
		METEOR-WikiTree	82.8	63.1	44.5	74.1	63.3
		BLEU-ed	78.3	57.6	36.9	69.6	58.0
		BLEU-WikiTree	77.4	54.3	32.9	67.4	55.2
		TER-ed	13.7	32.3	49.7	21.9	32.0
		TER-WikiTree	14.6	37.4	55.3	24.7	35.7
	ES	#Segments	1187	1028	1233	2215	3448
		METEOR-ed	85.1	70.6	51.2	78.7	68.9
		METEOR-WikiTree	83.6	66.8	47.2	76.2	65.9
		BLEU-ed	79.4	61.3	40.9	71.4	60.7
		BLEU-WikiTree	77.7	57.2	36.4	68.6	57.3
		TER-ed	14.2	29.9	46.8	21.2	30.4
		TER-WikiTree	15.7	35.1	52.8	24.4	34.6
	FR	#Segments	1197	1040	1257	2237	3494
		METEOR-ed	85.3	69.3	51.4	78.3	68.8
		METEOR-WikiTree	84.4	66.2	47.1	76.4	66.1
		BLEU-ed	78.1	58.3	39.2	69.4	58.9
		BLEU-WikiTree	77.3	55.3	34.7	67.7	56.3
		TER-ed	14.7	32.6	50.4	22.7	32.5
		TER-WikiTree	15.7	37.7	56.4	25.5	36.4
S2	DE	#Segments	1078	889	1070	1967	3037
		METEOR-ed	82.3	67.3	48.6	75.6	66.5
		METEOR-WikiTree	81.8	64.8	45.3	74.2	64.5
		BLEU-ed	76.5	60.3	37.9	69.3	59.0
		BLEU-WikiTree	75.5	56.8	34.2	67.2	56.4
		TER-ed	14.9	32.3	51.3	22.7	32.3
		TER-WikiTree	15.8	36.6	57.6	25.1	36.0
	ES	#Segments	1076	885	1050	1961	3011
		METEOR-ed	85.1	73.4	53.9	79.9	71.3
		METEOR-WikiTree	84.0	69.9	50.1	77.7	68.6
		BLEU-ed	78.8	64.9	43.0	72.5	62.9
		BLEU-WikiTree	77.6	60.9	39.1	70.1	60.0
		TER-ed	15.3	30.1	47.7	22.0	30.5
		TER-WikiTree	16.1	34.3	53.3	24.3	34.0
	FR	#Segments	1082	894	1078	1976	3054
		METEOR-ed	83.7	70.4	51.7	77.8	69.1
		METEOR-WikiTree	84.0	67.4	48.5	76.6	67.3
		BLEU-ed	76.3	61.0	40.4	69.5	60.0
		BLEU-WikiTree	77.2	57.7	37.3	68.6	58.3
		TER-ed	17.9	34.5	53.2	25.3	34.6
		TER-WikiTree	17.4	38.7	58.4	26.9	37.5

Table 4.12: Results using the 'WikiTree' Model on English-German (DE), English-Spanish (ES) and English-French (FR)

4.2.8 Further Analysis of Results

Our automatic evaluation metrics do not show improvements. Therefore, to better analyse the results we manually evaluated the results in the interval [70, 100) for the English-German language pair data. There are 679 segments for which a different fuzzy match is retrieved using our WikiTree model and the baseline edit-distance. We randomly selected half of the segments (339 segments) for the evaluation. We evaluated the English side which is used for matching instead of the German side due to the unavailability of a German annotator. Our assumption is that when the English side is better, the German side will also be better. Our annotator had English language level C1. Out of the total of 339 segments evaluated, the annotator tagged 275 segments as similar, for 45 segments the baseline edit-distance retrieved a better match and for 20 segments the WikiTree model retrieved a better match. The results show that most of the time both methods retrieve similar results but the baseline edit-distance is somewhat better. However, when analysisng the segments where the annotator tagged WikiTree model as better, we observe that the WikiTree model computes similarity at a deeper level.

In the examples presented below, Input represents the input segment, DEEP represents the match retrieved using our WikiTree model and EDIT represents the match retrieved using the baseline edit-distance.

In the following example, we can see our approach retrieves a better match than the baseline.

Input: together hereinafter referred to as the ‘ Parties ’ ,

DEEP: hereinafter jointly referred to as ‘ the Parties ’ ,

EDIT: Together hereinafter referred to as ‘ the Contracting Parties ’ ,

We can see strings *together hereinafter* and *hereinafter jointly* are the same in meaning, although they differ at surface level. Such candidates may be given a lower score than the baseline edit-distance by automatic evaluation metrics, even though our approach retrieves a better match. Furthermore, the paraphrase corpus used in Chapter 3 does not contain *together hereinafter* and *hereinafter jointly* as paraphrases, therefore this candidate will be missed by our paraphrasing approaches presented in Chapter 3.

Similarly, in the example given below the position of *therefore* is changed without making any difference in the meaning. If a translator looked at this pair, they may not change the segment retrieved by DEEP, whereas the segment retrieved by EDIT clearly needs post-editing because the message is different.

Input: The Convention should therefore be amended accordingly ,

DEEP: Therefore the Convention should be amended accordingly ,

EDIT: Annex I should therefore be amended accordingly ,

In some cases, we notice that the candidates differ at string level more when our approach is used, but the candidates are conceptually similar to the baseline edit-distance. Some examples are given below:

Input: Having regard to the initiative of the italian republic ,

DEEP: Having regard to the proposal of the Italian Government ,

EDIT: Having regard to the initiative of the European Commission ,

Similarly,

Input: This appropriation is intended to cover the costs of :

DEEP: This appropriation is also intended to cover the following expenditure :

EDIT: This appropriation is intended to cover the following :

We also present below some cases where the candidates retrieved by the baseline edit-distance are better. In the segment given below, we see that string *two* is ignored by our deep learning based approach.

Input: The two cuts shall be made at the joints ;

DEEP: The cuts must have been made at the joints ;

EDIT: The two cuts must have been made at the joints ;

In the segment given below, the edit-distance match is much closer (same month *May* in both Input and EDIT).

Input: It shall apply from 1 May 2004 .

DEEP: It shall apply from 1 July 2012 .

EDIT: It shall apply from 1 May 2015 .

Our analysis suggests that the WikiTree model and the baseline edit-distance retrieve similar matches in most of the cases. There are some cases where WikiTree model retrieves some good matches. This shows us there is scope to retrieve better matches compared to the baseline edit-distance. However, for our experimental settings, we can conclude that the baseline edit-distance is better compared to the WikiTree model overall.

4.2.9 A Closer Look at Segment Representation using LSTM

In this section we present some examples to see how LSTM represents the segment and takes into account the relevance of tokens. To see the impact of each token, we compute the cosine distance between the representation at time $t-1$ and t. This means we feed the tokens one by one and see how much impact the current token has on the overall representation.

Figure 4.5 shows two example segments from the DGT-TM corpus used in our experiments. The change for the first word 'concerning' or 'the' is 1 because there was no representation earlier to it. As we start constructing the segment we can observe that the change for stop words like 'the' and 'of' is less pronounced compared to the change for content words. We have also seen that the change for a stop word occurring for the first time is more pronounced compared to the second time (for example change for the first occurrence of 'the' compared to the second occurrence of 'the' in the same segment), which suggest that when model has seen a number of words, a stop word matters even less. We have also seen that

even when a segment has only one word, ‘concerning’ in the first segment and ‘the’ in the second segment, the representation changes more for ‘verification’ as a second token and less for ‘the’ as a second token. We can see ‘verification’ has around 0.58 distance from the previous representation, whereas ‘the’ has around 0.25 distance from the previous representation.

This shows that our system automatically learns to give less importance to words which carry less information like stop words ‘the’ and ‘of’. This analysis also indicates that if we train our system on a dataset which is more appropriate for our TM matching task, like a dataset annotated with the post-editing effort, we can train our system to give importance based on the post-editing effort needed. We can see in our second example that the change for ‘,’ is as pronounced as the word ‘personnel’. However, the post-editing time will be much less for writing a ‘,’ at the end of the segment.

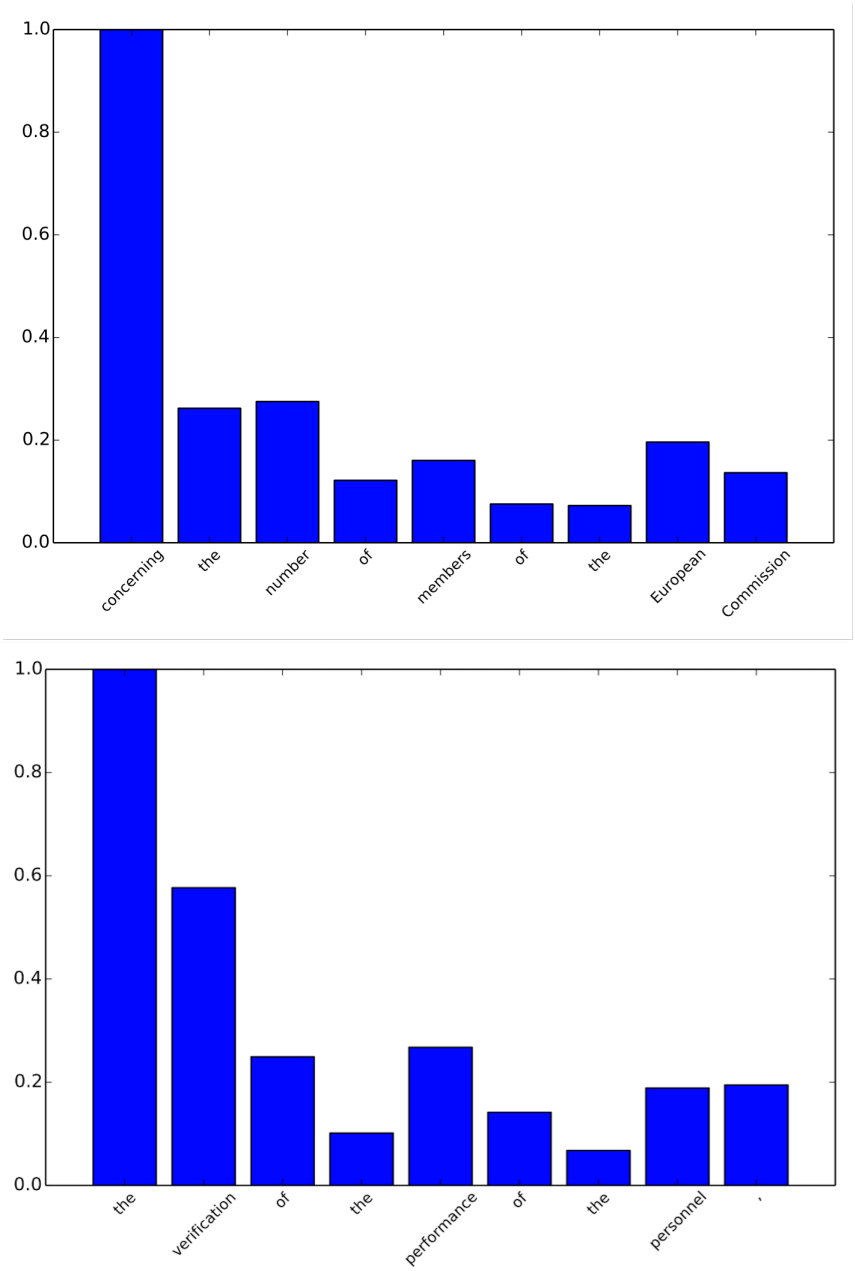


Figure 4.5: Examples Showing the Change in Segment Representation using the '840blstmSet1' Model for Each Token

4.2.10 Conclusion

In this section, we briefly described simple RNN, LSTM and Tree Structured LSTM. We presented our work on modelling segments and computing similarity between segments using RNNs. We trained eight different models. We have shown that for TM matching and retrieval Tree-LSTM models retrieve better results compared to simple LSTM models.

Our evaluation shows no improvements using the LSTM models compared to the baseline edit-distance when automatic evaluation metrics like BLEU, Meteror and TER are used. However, manual analysis of the segments retrieved by WikiTree model, the best performing LSTM model, shows that in some cases WikiTree model retrieve better results. However, overall we observe better matches using the baseline edit-distance compared to the WikiTree model.

4.3 Conclusion

In this chapter, we described three approaches to incorporate deep semantic information in TM matching and retrieval. We presented an approach based on support vectors machines and two approaches based on recurrent neural networks. Our approach based on manually designed features did not obtained better results than the baseline edit-distance.

We compared eight different LSTM models using different architectures and training datasets. We observed that Tree-LSTM is better compared to the simple LSTM. We also observe that using TM as a part of the corpus used to train word

vector representations does not improve the word vector quality or the quality of the retrieved segments. We evaluated our WikiTree model on three language pairs, English-German, English-French and English-Spanish and obtained similar results for all the three language pairs. For all three language pairs, we did not obtain better results compared to the baseline edit-distance when evaluating using Meteor, BLEU and TER metrics.

Our human evaluation of the WikiTree model shows that in some cases we obtain better matches, however, the baseline edit-distance performed better overall. In total, 339 segments were evaluated in the interval [70, 100). For 275 segments both approaches retrieved similar matches, for 45 segments edit-distance retrieved a better match and for 20 segments the WikiTree model retrieved a better match.

One of the reasons for not obtaining better results could be that we do not have a training corpus designed for the TM matching and retrieval task. Instead, we are using the training corpus derived from the machine translation evaluation task and the semantic similarity task. Therefore, it may be possible to obtain better results by training the RNNs using a corpus created for the TM matching (like a dataset annotated with post-editing effort).

CHAPTER 5

CONCLUSION

In this thesis, we presented various approaches to improve translation memory matching and retrieval using language technology. We proposed approaches which use existing paraphrase databases in an efficient manner. Our results show that taking into consideration paraphrasing during the matching leads to better results. However, these approaches have their limitations and depend on the availability of paraphrases. The segments for which there are no paraphrases available are still limited to simple edit-distance matching.

We extended our work on semantic matching and proposed advanced methods to compute semantic similarity for TM matching and retrieval. We proposed an approach based on manually designed features, which uses Support Vector Machines for training. Our approach relies on features extracted using dependency parsing, machine translation evaluation, named entity recognition, word overlaps, paraphrasing, and corpus pattern analysis. The approach did not obtain better results than the baseline edit-distance. We further proposed novel and efficient approaches to compute semantic similarity for TM matching and retrieval using LSTM and Tree-LSTM networks. Our results show that we do not retrieve better matches from TM using LSTM and Tree-LSTM networks compared to the

baseline edit-distance.

The remaining of this chapter presents the main conclusions of this thesis. We review each chapter and present the main findings for each of them. Given that two sets of approaches proposed in this thesis, in Section 5.1, we discuss which one is better. In Section 5.2, we answer our research questions, followed by future work in Section 5.3.

In Chapter 2, we reviewed previous work on TM matching and retrieval and work on combining TM with machine translation. We also reviewed similarity computation techniques at the sentence level in various NLP areas.

Chapter 3 presents two approaches which can efficiently use paraphrasing in translation memory matching and retrieval. Our first approach is based on 'dynamic programming and greedy approximation (DPGA)' and our second approach is based on 'dynamic programming only (DP)'. We evaluated our results on English-German, English-Spanish and English-French pairs of the DGT-TM corpus and English-German pairs of the Europarl corpus. The DPGA approach obtained between 7% and 11% increase in the segments retrieved on the DGT-TM dataset for the threshold intervals [85, 100) and [70, 85). The DP approach obtained between 9% to 16% increase in the segments retrieved on the DGT-TM dataset for the threshold intervals [85, 100) and [70, 85). The exact increase depends on the preprocessing setting. For the Europarl dataset, we observed more than 36% and 72% increase for the threshold intervals [85, 100) and [70, 85), respectively using our DP approach.

In the same chapter, we also conducted human evaluations of our DPGA

approach to evaluate the quality of the retrieved segments. On average, on both sets used for evaluation, we observe that translators made 824.79 keystrokes and 1103.2 keystrokes when editing paraphrasing and baseline edit-distance matches, respectively. Translators took 1069.61 seconds for paraphrasing as opposed to 1177.77 seconds for baseline edit-distance matches. Therefore, by using paraphrasing matches, translators edit 25.23% less, which saves time by 9.18% when evaluating the segments that changed their top ranking and come up in the threshold intervals because of paraphrasing. We can conclude that paraphrasing helps in retrieving more and better matches compared to the baseline edit-distance. The increase in retrieval depends on the genre of the corpus.

In Chapter 4, we presented three approaches based on advanced semantics. We presented an approach based on support vector machines, which uses features extracted using various language technologies. We presented two approaches based on RNNs and in particular based on LSTM and Tree-LSTM Networks. For all three approaches based on advanced semantics we do not obtain better results than the baseline edit-distance.

5.1 Comparing Paraphrasing and LSTM

In this section, we compare our DP approach from Chapter 3, which incorporate paraphrasing in TM matching and retrieval and the 'WikiTree' model from Chapter 4, which is based on Tree-LSTM networks, on the English-German language pair of the DGT-TM corpus. The DP approach performs better than the DPGA approach in Chapter 3 and the 'WikiTree' model obtains better results

than the other seven LSTM models in Chapter 4 for the DGT-TM corpus.

We use the preprocessing setting S1 and the same filtering steps as applied in Chapter 4. Tables 5.1 and 5.2 present the results. As in Chapter 4, the threshold intervals are determined based on the baseline edit-distance similarity. Table 5.1 presents the Meteor, BLEU and TER scores using all segments retrieved in a threshold interval. Table 5.2 presents the scores using only the segments for which either the match retrieved using the 'WikiTree' model or the DP approach differ with the baseline edit-distance, therefore, ignoring the segments for which all three approaches retrieve the same matches.

We also conducted significance testing of our results. The standard deviations of scores are calculated by bootstrap resampling and p-values are obtained by approximate randomization (Clark et al., 2011). In Tables 5.1 and 5.2, the scores with '*' and '**' indicate that the results are statistically significantly better than the baseline edit-distance with $p < 0.01$ and $p < 0.05$, respectively.

	Threshold Interval	**[85, 100)**	**[70, 85)**	**[55, 70)**	**[70, 100)**	**[55, 100)**
	#Segments	1193	1029	1259	2222	3481
	METEOR-ed	83.84 ± 0.44	65.66 ± 0.90	48.09 ± 0.48	75.78 ± 0.54	65.75 ± 0.45
	METEOR-WikiTree	82.84 ± 0.45	63.06 ± 0.90	44.48 ± 0.51	74.07 ± 0.56	63.35 ± 0.48
	METEOR-pp	83.87 ± 0.43**	65.82 ± 0.88*	48.14 ± 0.47	75.87 ± 0.54*	65.83 ± 0.45
	BLEU-ed	78.27 ± 0.59	57.56 ± 0.97	36.91 ± 0.62	69.56 ± 0.61	58.04 ± 0.53
	BLEU-WikiTree	77.43 ± 0.60	54.27 ± 0.98	32.88 ± 0.61	67.41 ± 0.63	55.25 ± 0.56
	BLEU-pp	78.30 ± 0.58	57.70 ± 0.97**	37.02 ± 0.61	69.63 ± 0.61*	58.12 ± 0.53
	TER-ed	13.74 ± 0.44	32.30 ± 0.94	49.67 ± 0.56	21.95 ± 0.55	31.95 ± 0.45
	TER-WikiTree	14.62 ± 0.46	37.40 ± 0.95	55.32 ± 0.64	24.70 ± 0.59	35.75 ± 0.51
	TER-pp	13.71 ± 0.43	32.16 ± 0.92**	49.72 ± 0.57	21.87 ± 0.55**	31.92 ± 0.46

Table 5.1: Comparing Paraphrasing and 'WikiTree' on English-German (All Matches)

Both tables show that the DP approach is better compared to the 'WikiTree'

	Threshold Interval	**[85, 100)**	**[70, 85)**	**[55, 70)**	**[70, 100)**	**[55, 100)**
	#Segments	272	418	801	690	1491
	METEOR-ed	79.49 ± 0.99	61.69 ± 1.82	44.75 ± 0.58	68.89 ± 1.29	55.68 ± 0.70
	METEOR-WikiTree	74.79 ± 1.00	55.07 ± 1.73	38.96 ± 0.56	63.03 ± 1.27	49.86 ± 0.71
	METEOR-pp	79.65 ± 1.00**	62.09 ± 1.84*	44.84 ± 0.57	69.19 ± 1.29*	55.87 ± 0.70
	BLEU-ed	70.83 ± 1.22	53.34 ± 1.88	34.03 ± 0.74	61.19 ± 1.32	46.53 ± 0.78
	BLEU-WikiTree	66.18 ± 1.30	44.83 ± 1.63	27.44 ± 0.64	53.56 ± 1.26	39.49 ± 0.77
	BLEU-pp	70.96 ± 1.20	53.70 ± 1.87**	34.21 ± 0.73	61.43 ± 1.31*	46.73 ± 0.78
	TER-ed	17.79 ± 0.95	34.61 ± 1.90	52.53 ± 0.71	27.81 ± 1.31	41.31 ± 0.73
	TER-WikiTree	21.95 ± 1.09	47.62 ± 1.69	61.61 ± 0.79	37.24 ± 1.32	50.56 ± 0.78
	TER-pp	17.64 ± 0.94	34.25 ± 1.90**	52.60 ± 0.71	27.53 ± 1.32**	41.23 ± 0.73

Table 5.2: Comparing Paraphrasing and 'WikiTree' on English-German (Differing Matches Only)

model. Table 5.1 shows that the DP approach obtains a small yet significant gain in the Meteor, BLEU and TER scores for the [70, 100) threshold interval. The scores are only slightly better because the majority of the segments retrieved by both the DP approach and the baseline edit-distance are the same. Table 5.2 shows somewhat more difference when we consider only the segments for which either the 'WikiTree' model or the DP approach differ with the baseline edit-distance. However, still a large number of the segments retrieved by the DP approach and the baseline edit-distance are the same. The 'WikiTree' model more often obtains a different match than the DP approach.

5.2 Answers to Research Questions

In this section, we revisit the research questions proposed in Chapter 1.

Q1 How do we use paraphrases efficiently in the TM matching and retrieval process?

We answer Q1 in Chapter 3 by proposing two approaches which efficiently incorporate paraphrases in edit-distance. Both our approaches have polynomial

time complexity and show that it is possible to use paraphrases efficiently in the translation memory matching and retrieval process.

Q2 Does paraphrasing improve TM matching and retrieval?

Our results obtained using both automatic and human evaluations in Chapter 3 show that paraphrasing improves TM matching and retrieval. Although we obtain improvements in retrieval, the baseline edit-distance and the methods which use paraphrasing retrieve the same segments in the large number of cases which reduces the overall impact on improvements.

Q3 Can advanced semantic matching techniques improve TM matching and retrieval?

To answer Q3, we presented three approaches which compute similarity based on advanced semantics. We did not obtain better results than the baseline edit-distance for all three approaches based on advanced semantics. We observed that the baseline edit-distance is a strong baseline measure which is to be expected given that it is widely used in commercial TM systems. Even using advanced deep learning techniques based on LSTM and Tree-LSTM networks, which model segments as dense vectors and compute semantic similarity between them, the evaluation metrics used suggest that the baseline edit-distance performed better compared to our models based on deep learning.

However, as discussed in Chapter 4, one of the reasons for not obtaining better results could be that we do not have a training corpus designed for the TM

matching and retrieval task. Instead, we are using the training corpus derived from the machine translation evaluation task and the semantic similarity task. Therefore, it may be possible to obtain better results by training the RNNs using a corpus created for the TM matching. Our manual evaluation shows that for some input segments, we obtain better matches using our deep learning model, however, the baseline edit-distance performed better overall.

5.3 Future Work

Our work in this thesis can be extended in various ways.

We have used LSTM and Tree Structured LSTM in Chapter 5. Bidirectional LSTMs are also used in various NLP and speech processing tasks. In particular, for machine translation and speech recognition, bidirectional LSTM networks are better compared to simple LSTM networks (Bahdanau et al., 2014; Graves et al., 2013). We would like to see whether using bidirectional LSTMs can improve matching and retrieval. One criticism of the methods proposed in Chapter 4 is that the training data was developed in the context of MT evaluation. On MT evaluation, we obtained state of the art results. Therefore, it may be possible to obtain better results if we have a large dataset created for the TM matching task and train the system using such dataset. Furthermore, a thorough human evaluation will be better for measuring the quality of retrieved segments.

It will be interesting to see whether using more features in our approach proposed in Chapter 3 brings improvements to TM matching and retrieval. In fact, word vectors used by us in Chapter 4 can also be used to derive features.

In Chapter 3, we have used paraphrases of size “L”. The PPDB paraphrases database is available in other sizes as well. Also, there are other paraphrase databases available. It will be interesting to see the impact of using smaller and larger sets of PPDB, and other paraphrase databases.

BIBLIOGRAPHY

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., Rigau, G., Uria, L., and Wiebe, J. (2015). Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Mihalcea, R., Rigau, G., and Wiebe, J. (2014). Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland.

Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., and Guo, W. (2013). Sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *In* SEM 2013: The Second Joint Conference on Lexical and Computational Semantics*, Atlanta, Georgia, USA.

Agirre, E., Diab, M., Cer, D., and Gonzalez-Agirre, A. (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation, in conjunction with the First*

Joint Conference on Lexical and Computational Semantics, pages 385–393, Montreal, Canada. Association for Computational Linguistics.

Alabbas, M. and Ramsay, A. (2013). Natural language inference for Arabic using extended tree edit distance with subtrees. *Journal of Artificial Intelligence Research*, 48:1–22.

Arthern, P. J. (1978). Machine Translation and Computerized Terminology Systems, A Translator's viewpoint. In *Translating and the Computer: Proceedings of a Seminar*, pages 77–108.

Aziz, W., Castilho, S., and Specia, L. (2012). PET: a Tool for Post-editing and Assessing Machine Translation. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, Istanbul, Turkey.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bannard, C. and Callison-Burch, C. (2005). Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604, Ann Arbor, Michigan, USA.

Bär, D., Biemann, C., Gurevych, I., and Zesch, T. (2012). Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *First Joint Conference on Lexical and Computational Semantics, Association for Computational Linguistics*, pages 435–440, Montreal, Canada.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bertoldi, N., Cettolo, M., and Federico, M. (2013). Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. In *Proceedings of the XIV Machine Translation Summit*, pages 35–42.

Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239.

Birch, A. and Osborne, M. (2011). Reordering metrics for MT. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1027–1035, Portland, Oregon. Association for Computational Linguistics.

Bjerva, J., Bos, J., van der Goot, R., and Nissim, M. (2014). The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic*

Evaluation (SemEval 2014), pages 642–646, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022.

Blunsom, P., Grefenstette, E., and Kalchbrenner, N. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, Baltimore, Maryland, USA.

Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013). Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Bojar, O., Graham, Y., Kamran, A., and Stanojević, M. (2016). Results of the WMT16 Metrics Shared Task. In *Proceedings of the First Conference*

on Machine Translation, pages 199–231, Berlin, Germany. Association for Computational Linguistics.

Bradbury, J. and El Maarouf, I. (2013). An empirical classification of verbs based on Semantic Types: the case of the 'poison' verbs. In *Proceedings of the Joint Symposium on Semantic Processing*, pages 70–74, Trento, Italy.

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

Burgess, C., Livesay, K., and Lund, K. (1998). Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25(2-3):211–257.

Callison-Burch, C., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2012). Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Callison-Burch, C., Koehn, P., Monz, C., and Schroeder, J. (2009). Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece. Association for Computational Linguistics.

Carl, M. and Hansen, S. (1999). Linking Translation Memories with Example-Based Machine Translation. In *Proceedings of 7th Machine Translation Summit*, pages 617–624, Singapore.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181, Portland, Oregon. Association for Computational Linguistics.

Clark, J. P. (2002). System, method, and product for dynamically aligning translations in a translation-memory system. US Patent 6,345,244.

Cohn, T., Callison-Burch, C., and Lapata, M. (2008). Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4):597–614.

Collins, M. and Duffy, N. (2001). Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press.

Collobert, R. and Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, Helsinki, Finland. ACM.

Dandapat, S. (2012). *Mitigating the Problems of SMT using EBMT Sandipan Dandapat*. PhD thesis, Dublin City University.

De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 449–454.

Denkowski, M. and Lavie, A. (2014). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350, Geneva, Switzerland.

Eisele, A. and Chen, Y. (2010). MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation(LREC'10)*, Valletta, Malta.

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. In *International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, pages 153–160, Clearwater Beach, Florida, USA.

Erk, K. (2012). Vector Space Models of Word Meaning and Phrase Meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Ganitkevitch, J., Benjamin, V. D., and Callison-Burch, C. (2013). PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia.

Gautam, S. and Bhattacharyya, P. (2014). LAYERED: Metric for Machine Translation Evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 387–393, Baltimore, Maryland, USA.

Giménez, J. and Màrquez, L. (2010). Linguistic measures for automatic machine translation evaluation. *Machine Translation*, 24(3-4):209–240.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385. Springer Science & Business Media.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Gu, Y., Yang, Z., Nakano, M., and Kitsuregawa, M. (2012). Towards Efficient

Similar Sentences Extraction. In *Proceedings of Intelligent Data Engineering and Automated Learning*, volume 7435, pages 270–277. Springer.

Gupta, R., Béchara, H., Maarouf, I. E., and Orăsan, C. (2014). UoW: NLP techniques developed at the University of Wolverhampton for Semantic Similarity and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Gupta, R. and Orăsan, C. (2014). Incorporating Paraphrasing in Translation Memory Matching and Retrieval. In *Proceedings of the European Association of Machine Translation (EAMT-2014)*, Dubrovnik, Croatia.

Gupta, R., Orăsan, C., and van Genabith, J. (2015a). Machine Translation Evaluation using Recurrent Neural Networks. In *Proceedings of the tenth Workshop on Statistical Machine Translation*, pages 380–384, Lisbon, Portugal.

Gupta, R., Orăsan, C., and van Genabith, J. (2015b). ReVal: A Simple and Effective Machine Translation Evaluation Metric Based on Recurrent Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1066–1072, Lisbon, Portugal.

Guzmán, F., Joty, S., Màrquez, L., and Nakov, P. (2015). Pairwise neural machine translation evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 1: Long Papers), pages 805–814, Beijing, China. Association for Computational Linguistics.

Hanks, P. (2013). *Lexical Analysis: Norms and Exploitations*. Mit Press, Cambridge, Massachusetts.

Heilman, M. and Smith, N. A. (2010). Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019, Los Angeles, California, USA.

Hermann, K. M. and Blunsom, P. (2014). Multilingual Models for Compositional Distributed Semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 58–68, Baltimore, Maryland, USA.

Hinton, G., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hodász, G. and Pohl, G. (2005). MetaMorpho TM: a linguistically enriched translation memory. In *International Workshop: Modern Approaches in Translation Technologies*, pages 26–30, Borovets, Bulgaria.

Hopkins, M. and May, J. (2013). Models of translation competitions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1416–1424, Sofia, Bulgaria.

Huang, P. and Chang, B. (2014). SSMT: A Machine Translation Evaluation View to Paragraph-to-Sentence Semantic Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 585–589, Dublin, Ireland.

Islam, A. and Inkpen, D. (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data*, 2(2):1–25.

Jaworski, R. (2013). Anubis-speeding up computer-aided translation. In *Computational Linguistics*, pages 263–280. Springer.

Joty, S., Guzmán, F., Màrquez, L., and Nakov, P. (2014). DiscoTK: Using Discourse Structure for Machine Translation Evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 402–408, Baltimore, Maryland, USA.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, volume 5, pages 79–86, Phuket , Thailand.

Koehn, P. (2012). Simulating human judgment in machine translation evaluation

campaigns. In *Proceedings of the Ninth International Workshop on Spoken Language Translation*, pages 179–184, Hong Kong.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180, Prague. Association for Computational Linguistics.

Koehn, P. and Schroeder, J. (2007). Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague. Association for Computational Linguistics.

Koehn, P. and Senellart, J. (2010). Convergence of Translation Memory and Statistical Machine Translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31, Denver, Colorado.

Landauer, T. K. and Dumais, S. T. (1997). A Solution to Plato's problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological review*, 104(2):211.

Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An Introduction to Latent Semantic Analysis. *Discourse processes*, 25(2-3):259–284.

Le, Q. and Mikolov, T. (2014). Distributed Representations of Sentences and

Documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196, Beijing, China.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Li, L., Way, A., and Liu, Q. (2014). A Discriminative Framework of Integrating Translation Memory Features into SMT. In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas*, volume 1, pages 249–260, Vancouver, BC Canada.

Li, Y., McLean, D., Bandar, Z. A., O'shea, J. D., and Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1138–1150.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Machácek, M. and Bojar, O. (2014). Results of the WMT-14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, Baltimore, Maryland, USA.

Macklovitch, E. and Russell, G. (2000). What's been forgotten in Translation Memory. In *Proceedings of the 6th Conference for Machine Translation in the Americas (AMTA)*, pages 137–146, Cuernavaca, Mexico.

Malandrakis, N., Iosif, E., and Potamianos, A. (2012). DeepPurple: Estimating Sentence Semantic Similarity using N-gram Regression Models and Web

Snippets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 565–570, Montreal, Canada. Association for Computational Linguistics.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. (2014a). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., and Zamparelli, R. (2014b). A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC 2014*, Reykjavik, Iceland.

Marsi, E., Moen, H., Bungum, L., Sizov, G., Gambäck, B., and Lynum, A. (2013). Ntnu-core: Combining strong features for semantic similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 66–73.

Mikolov, T., Corrado, G., Chen, K., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12, Scottsdale, Arizona.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting Similarities among Languages for Machine Translation. *CoRR*, pages 1–10.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013c).

Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, ke Tahoe Nevada.

Mitchell, J. and Lapata, M. (2008). Vector-based Models of Semantic Composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, USA.

Mitkov, R. (2008). Improving Third Generation Translation Memory systems through identification of rhetorical predicates. In *Proceedings of LangTech2008*.

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

Pekar, V. and Mitkov, R. (2007). New Generation Translation Memory: Content-Sensivite Matching. In *Proceedings of the 40th Anniversary Congress of the Swiss Association of Translators, Terminologists and Interpreters*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empiricial Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning Accurate,

Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440, Sydney, Australia.

Planas, E. and Furuse, O. (1999). Formalizing Translation Memories. In *Proceedings of the 7th Machine Translation Summit*, pages 331–339, Singapore.

Planas, E. and Furuse, O. (2000). Multi-level Similar Segment Matching Algorithm for Translation Memories and Example-Based Machine Translation. In *Proceedings of the 18th conference on Computational linguistics*, volume 2, pages 621–627. Association for Computational Linguistics.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Roukos, S., Graff, D., and Melamed, D. (1995). Hansard French/English. *Linguistic Data Consortium*.

Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1-2. *Cambridge, MA*.

Sakaguchi, K., Post, M., and Van Durme, B. (2014). Efficient Elicitation of Annotations for Human Evaluation of Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 1–11, Baltimore, Maryland, USA.

Simard, M. and Fujita, A. (2012). A Poor Man's Translation Memory Using Machine Translation Evaluation Metrics. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas*, California, USA.

Simard, M. and Isabelle, P. (2009). Phrase-based Machine Translation in a Computer-assisted Translation Environment. In *Proceedings of the Twelfth Machine Translation Summit*, pages 120–127, Ottawa, Ontario, Canada.

Smolensky, P. (1986). *Chapter 6:Information processing in dynamical systems: Foundations of harmony theory*. MIT Press.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for machine translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Snover, M., Madnani, N., Dorr, B., and Schwartz, R. (2008). TERp System Description. In *MetricsMATR workshop at AMTA*, Honolulu, Hawaii, USA.

Socher, R., Bauer, J., Manning, C. D., and Ng, A. Y. (2013a). Parsing with Compositional Vector Grammars. *In Proceedings of the ACL*, pages 455–465.

Socher, R., Huang, E., and Pennington, J. (2011a). Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems*, pages 801–809, Granada, Spain.

Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011b). Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, Bellevue, Washington, USA.

Socher, R., Perelygin, A., and Wu, J. (2013b). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of EMNLP*, pages 1631–1642, Seattle, USA.

Somers, H. (2003). Translation memory systems. *Computers and Translation: A Translator's Guide*, 35:31–48.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M., and Cristianini, N. (2009). Estimating the Sentence-Level Quality of Machine Translation Systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, pages 28–37, Barcelona, Spain.

Stanojević, M., Kamran, A., Koehn, P., and Bojar, O. (2015). Results of the WMT15 Metrics Shared Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 256–273, Lisbon, Portugal. Association for Computational Linguistics.

Steinberger, R., Eisele, A., Klocek, S., Pilos, S., and Schlüter, P. (2012). DGT-TM: A freely available Translation Memory in 22 languages. *LREC*, pages 454–459.

Steinberger, R., Pouliquen, B., and Widiger, A. (2006). The JRC-Acquis: A

multilingual aligned parallel corpus with 20+ languages. In *arXiv preprint cs*, volume 4.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, Montreal, Canada.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Tan, L., Scarton, C., Specia, L., and van Genabith, J. (2015). USAAR-SHEFFIELD: Semantic Textual Similarity with Deep Regression and Machine Translation Evaluation Metrics. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 85–89, Denver, Colorado. Association for Computational Linguistics.

Tezcan, A. and Vandeghinste, V. (2011). SMT-CAT integration in a Technical Domain : Handling XML Markup Using Pre & Post-processing Methods. In *Proceedings of the 15th Conference of the European Association for Machine Translation*, pages 55–62, Leuven, Belgium.

Tiedemann, J. (2009). News from OPUS - A Collection of Multilingual Parallel

Corpora with Tools and Interfaces. In Nicolov, N., Bontcheva, K., Angelova, G., and Mitkov, R., editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Timonera, K. and Mitkov, R. (2015). Improving Translation Memory Matching through Clause Splitting. In *Proceedings of the Workshop on Natural Language Processing for Translation Memories (NLP4TM)*, pages 17–23, Hissar, Bulgaria.

Tsatsaronis, G., Varlamis, I., and Vazirgiannis, M. (2010). Text relatedness based on a word thesaurus. *Journal of Artificial Intelligence Research*, 37(1):1–40.

Turney, P. D. (2012). Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, pages 533–585.

Utiyama, M., Neubig, G., Onishi, T., and Sumita, E. (2011). Searching Translation Memories for Paraphrases. In *Proceedings of the 13th Machine Translation Summit*, pages 325–331, Xiamen, China.

Vela, M., Neumann, S., and Hansen-Schirra, S. (2007). Querying multi-layer annotation and alignment in translation corpora. In *Proceedings of the Corpus Linguistics Conference CL.*

Vu, T. T., Tran, Q. H., and Pham, S. B. (2015). TATO: Leveraging on Multiple Strategies for Semantic Textual Similarity. In *Proceedings of the 9th*

International Workshop on Semantic Evaluation (SemEval 2015), pages 190–195, Denver, Colorado.

Wang, K., Zong, C., and Su, K.-Y. (2014). Dynamically Integrating Cross-Domain Translation Memory into Phrase-Based Machine Translation during Decoding. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 398–408, Dublin, Ireland.

Wang, M. and Cer, D. (2012). Stanford: Probabilistic Edit Distance Metrics for STS. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 648–654, Montreal, Canada.

Xu, W., Callison-Burch, C., and Dolan, W. B. (2015). SemEval-2015 Task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado.

Zaremba, W. and Sutskever, I. (2014). Learning to execute. *arXiv preprint arXiv:1410.4615*.

Zaretskaya, A., Pastor, G. C., and Seghiri, M. (2015). Translators requirements for translation technologies: Results of a user survey. In *Proceedings of the AIETI7 Conference New Horizons is Translation and Interpreting Studies*, Malaga, Spain.

Zhang, K. and Shasha, D. (1989). Simple Fast Algorithms for the Editing

Distance between Trees and Related Problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Zhao, J., Zhu, T., and Lan, M. (2014). ECNU: One Stone Two Birds: Ensemble of Heterogenous Measures for Semantic Relatedness and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

APPENDIX A

RELATED PUBLICATIONS

We classify publications as per the related chapter:

Publications related to Chapter 3:

- Gupta, R., and Orăsan, C. (2014). Incorporating Paraphrasing in Translation Memory Matching and Retrieval. In *Proceedings of the European Association of Machine Translation (EAMT-2014),* Dubrovnik, Croatia.
- Gupta R., Orăsan C., Zampieri M., Vela M., and van Genabith J. (2015). Can Translation Memories afford not to use paraphrasing? In *Proceedings of the European Association of Machine Translation (EAMT-2015),* Antalya, Turkey.
- Gupta R., Orăsan C., Liu Q. and Mitkov R. (2016) A Dynamic Programming Approach to Improving Translation Memory Matching and Retrieval using Paraphrases. In *Proceedings of the 19th International Conference on Text, Speech and Dialogue (TSD),*, pages 259-269, Brno, Czech Republic.

Publications related to Chapter 4:

- Gupta, R., Béchara, H., Maarouf, I. E., and Orăsan, C. (2014). UoW: NLP techniques developed at the University of Wolverhampton for Semantic Similarity and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.
- Gupta, R., Béchara, H., and Orăsan, C. (2014). Intelligent Translation Memory Matching and Retrieval Metric Exploiting Linguistic Technology. In *Proceedings of the Translating and Computer 36*, London, UK.
- Gupta, R., Orăsan, C., and van Genabith J. (2015). Machine Translation Evaluation using Recurrent Neural Networks. In *Proceedings of the tenth Workshop on Statistical Machine Translation*,pages 380-384, Lisbon, Portugal.
- Gupta, R., Orăsan, C., and van Genabith, J. (2015). Reval: A simple and effective machine translation evaluation metric based on recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*,pages 1066-1072, Lisbon, Portugal.