

T. C.

İSTANBUL 29 MAYIS ÜNİVERSİTESİ

SOSYAL BİLİMLER ENSTİTÜSÜ

ÇEVİRİBİLİM ANABİLİM DALI

**CREATING A STATISTICAL BASED AUTOMOTIVE
ORIENTED MACHINE TRANSLATION ENGINE**

(YÜKSEK LİSANS TEZİ)

Alper ÇALIK

Danışman:

Prof. Dr. Işın ÖNER

İSTANBUL

2019

T. C.
İSTANBUL 29 MAYIS ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
ÇEVİRİBİLİM ANABİLİM DALI

**CREATING A STATISTICAL BASED AUTOMOTIVE
ORIENTED MACHINE TRANSLATION ENGINE**

(YÜKSEK LİSANS TEZİ)

Alper ÇALIK

Danışman:
Prof. Dr. Işın ÖNER

İSTANBUL
2019

T. C.

İSTANBUL 29 MAYIS ÜNİVERSİTESİ

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Çeviribilim Anabilim Dalı, Çeviribilim Bilim Dalı'nda 010516YL04 numaralı Alper ÇALIK'ın hazırladığı "*Creating a Statistical Based Automotive Oriented Machine Translation Engine*" konulu yüksek lisans tezi ile ilgili tez savunma sınavı, 13.09.2019 günü saatleri arasında yapılmış, sorulan sorulara alınan cevaplar sonunda adayın tezinin başarılı olduğuna oy birliği ile karar verilmiştir.

Prof. Dr. Işın ÖNER

İstanbul 29 Mayıs Üniversitesi

(Tez Danışmanı ve Sınav Komisyonu Başkanı)

Prof. Dr. Ayşe Banu KARADAĞ

Yıldız Teknik Üniversitesi

Dr. Öğr. Üyesi Nilüfer ALİMEN

İstanbul 29 Mayıs Üniversitesi

BEYAN

Bu tezin yazılmasında bilimsel ahlak kurallarını uyulduğunu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, tezin herhangi bir kısmının bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunulmadığını beyan ederim.

Alper ÇALIK

13.09.2019

ÖZ

Bu tezin, çeviribilim öğrencileri, çevirmen adayları veya çevirmenler ile makine çevirisi konusunda temel düzeyde bilgi edinmek isteyen kişiler için İngilizce – Türkçe dil çiftinde makine çevirisi çıktılarının gösterildiği bir kaynak olması amaçlanmaktadır. Tez kapsamında otomotiv odaklı istatistiksel tabanlı makine çevirisi motoru geliştirilerek, motorun oluşturulma adımları ve motoru oluştururken karşılaşılabilecek sorunlar açıklanmıştır. İstatistiksel Makine Çevirisi motorunun İngilizce – Türkçe dil çiftinde nasıl işlediğini görmek isteyen kişiler, sürecin aşamalarına ilişkin bilgiler ve görseller bulabileceklerdir. İstatistiksel Makine Çevirisi motorunun oluşturulması için gerekli teknik adımları içeren bu çalışmanın gelecekte iş bulma kaygısı olan öğrenciler için faydalı bir kaynak olması beklenmektedir. Tezde kullanılan veriler sınırlı olduğundan, yüksek kaliteli makine çevirisi çıktısı elde etme amacı güdülmemiştir.

Anahtar Sözcükler:

Teknoloji, Makine Çevirisi, İstatistiksel Makine Çevirisi, Nöral Makine Çevirisi

ABSTRACT

This thesis is intended to be a source showing the machine translation outputs in the English - Turkish language pair for translation studies students, candidate translators, translators, and those who wish to obtain basic knowledge of machine translation. Within the scope of the thesis, the development of the automotive-oriented statistical-based machine translation engine, the steps of creating the engine and the problems that may be encountered while creating the engine are explained. Those who want to see how the Statistical Machine Translation engine works in the English - Turkish language pair will be able to find both information and illustrations regarding the stages of the process. This study which covers the steps of creating the Statistical Machine Translation engine is expected to be a useful resource for students who are concerned about finding jobs in the future. Since the data used in the thesis is limited, it is not intended to obtain high quality machine translation output.

Key words:

Technology, Machine Translation, Statistical Machine Translation, Neural Machine Translation

ACKNOWLEDGEMENTS

First, I would like to express my deepest and sincerest appreciation to my supervisor Prof. Dr. Işın ÖNER, the Head of Translation Studies Department, for teaching me what translation studies means, and for always forcing me for the better. Her insistence to always get the best led me to update my thesis constantly and make it more understandable and acceptable.

I also would like to thank Prof. Dr. Ayşe Banu KARADAĞ for her academic and moral support and keen eye for detail. Her critics helped me to find new ideas regarding my thesis, and changed my perspective.

I also would like to thank Asst. Prof. Nilüfer ALİMEN, for helping to make the things easier in this process.

I owe my greatest thanks to my brother Bekir DİRİ for his support. Without him, I would not be able to study on my thesis with passion. He always encouraged me to go on studying, and helped with any kind of problem I encountered.

Also, I owe a debt of gratitude to dear Selçuk ÖZCAN, who is one of the best machine translation engine experts I have ever known. His support, even in his busiest days, made me feel relaxed when I descended into desperation.

TABLE OF CONTENTS

TEZ ONAY SAYFASI.....	ii
BEYAN.....	iii
ÖZ.....	iv
ABSTRACT.....	v
ACKNOWLEDGEMENTS.....	vi
LIST OF ABBREVIATIONS.....	vii
LIST OF FIGURES.....	viii
1 GENERAL SCOPE	1
1.1 Theoretical Framework.....	3
1.2 General Information on Computer Aided Translation (CAT) Tools.....	4
1.3 General Information on Machine Translation.....	5
2 THE TRANSFORMATION OF TRANSLATOR	7
2.1 Translator 1.0.....	9
2.2 Translator 2.0.....	10
2.3 Translator 3.0.....	10
2.4 Translator 4.0.....	11
3 MACHINE TRANSLATION	13
3.1 The History of Machine Translation.....	13
3.2 Rule Based Machine Translation.....	14
3.3 Statistical Machine Translation (SMT).....	15
3.4 Neural Machine Translation (NMT).....	15
3.5 Differences between SMT and NMT.....	16
3.6 Differences between SMT and RBMT.....	17
3.7 Corpus.....	18
3.7.1 Technical Translation.....	18
3.7.2 Automotive Texts.....	19
4 CREATING A STATISTICAL MACHINE TRANSLATION ENGINE	20
4.1 Ethical Considerations.....	20

4.2 Linux-Ubuntu Installation.....	20
4.3 Moses	21
4.3.1 Installing Moses	22
4.4 Preparing the Data	28
4.4.1 (Pre)Alignment	28
4.4.2 GIZA++	31
4.4.3 Running the GIZA++.....	33
4.4.4 Training the Engine	38
5 EVALUATION	41
6 CONCLUSION	47
REFERENCES	49
ÖZGEÇMİŞ.....	60

LIST OF ABBREVIATIONS

BLEU	(Bilingual Evaluation Understudy)
CAT	(Computer Assisted Translation)
HTS	(Human Translated Sentence)
MT	(Machine Translation)
MTS	(Machine Translated Sentence)
MTSBS	(Machine Translated Sentence BLEU Score)
NMT	(Neural Machine Translation)
QA	(Quality Assurance)
RBMT	(Rule-Based Machine Translation)
SMT	(Statistical Machine Translation)
ST	(Source Text)
TM	(Translation Memory)

LIST OF FIGURES

Figure 1: The Game Changers of 2016 (Source: <https://blog.taus.net/the-game-changers-of-2016?fbclid=IwAR0IagI9v0tDmgnE181Ai2YXdIQ7kXX76fWdwXYnPOuKifGV1gV3Qsz24sA>)

Figure 2: A screenshot from Linux terminal showing the installation command is working

Figure 3: A screenshot from Linux terminal to continue the process

Figure 4: Raw alignment example from AbbyAligner

Figure 5: Aligned segment examples from AbbyAligner

Figure 6: Variable adding example

Figure 7: Undefined character example

Figure 8: An example from Linux terminal showing the file creation is successful

Figure 9: An example from Linux terminal showing the command is working

Figure 10: An example from Linux terminal showing how to initiate the engine

Figure 11: An example from Linux terminal showing the best translation result

1 GENERAL SCOPE

The aim of this thesis is to provide information on the steps of creating statistical machine translation engine. With the technological developments, translation industry started to change and new terms like computer aided translation (CAT) tools, quality assurance (QA) tools, term-banks, machine translation, translation memory (TM), etc. came into prominence. Since these terms were unusual to the most of the people in the field who were used to do translation with traditional methods when mentioned for the first time, it took time for these people to get used to these new terms. We can still see translators who reject to use CAT tools or QA tools and claim that any CAT tool cannot do what a human translator does. But in this century, the requirements of the translation sector change rapidly in line with the skills of the translators. Instead of an ordinary human translator, translators who keep abreast of technological developments become more preferable. Machine translation engines are one of these technological developments because it is commonly discussed if the machine translation replaces human translators. Based on these discussions I decided to study on machine translation and create a statistical base machine translation engine. In this thesis, the technical steps of creating a domain specific SMT engine are showed. An open source statistical machine translation system, Moses, was used to create SMT engine. The codes are adapted, and explained accordingly. The reason why I chose to work on this topic is that resources on using Moses system are not enough. Because, the manual of Moses suppose that users have basic knowledge of coding. So, I explained the codes to make creating the engine easier for the user.

In the first chapter, information on the theoretical framework of the study will be outlined. The departure point is Antony Pym's article about the shifting from translation competence to translator competence. Then, general information on computer aided translation tools and machine translation will be summarized in the following parts.

In the second chapter, information about industrial revolutions will be given and the impact of these revolutions on translation industry will be clarified briefly. Translators will be named as Translator 1.0, Translator 2.0, etc. depending on the period they are in. What Industry 1.0, 2.0, etc. means for a translator will be explained briefly.

The third chapter is dedicated to general information on machine translation, the history of machine translation, machine translation engine types such as rule based, statistical and neural, the differences between these types was given to make the reader familiar with these types. Additionally, the corpus which was used to create the statistical machine translation engine will be demonstrated and general information on technical translation, and automotive texts which creates the corpus, will be given.

The fourth chapter will give an outline of creating a statistical machine translation engine. Moreover, the ethical considerations about the data usage will be discussed. Then, the installation process of Linux-Ubuntu, which is required to use Moses, the open source machine translation engine which is used to create our statistical machine translation engine, will be illustrated. In this core chapter, the steps of creating SMT engine including the preparation of data, for example alignment process will be exemplified.

In this study readers can find answers to the following questions: What is the machine translation? What are the types of machine translation engines? What are the differences between statistical machine translation, neural machine translation and rule-based machine translation? How can corpus be created to be used for statistical machine translation engine? What does BLEU score mean? What are the steps of creating a statistical machine translation engine?

First of all, I need to state the reason why I chose this topic. I am still a member of the Translation and Interpretation Students of Turkey Platform where we discuss on the topics related to translation studies, translators, the developments in translation technologies and whatever we wish to know on translation. We, there, share our ideas and organize events covering all kinds of issues. Recently, I have realized that the number of the questions about the future of translation, translators and machines have been increasing day by day. Then we have organized a meeting at Trakya Üniversitesi addressing the freshmen and sophomores. We discussed on these questions and I noticed that most of the students were afraid of the future of translation as they think that the machines will take care of everything. I felt that there was something wrong and most of the students know almost nothing about these kinds of developments, and their effects. Who knows maybe they are right but they still should stay up to date with these improvements. They should at least try to learn how the machine works and how they

can benefit from it instead of escaping from contributing the field. I decided to do something to encourage them and started to this study. The aim of this study is not to overcome the concerns of the students. The aim is to show the technical part of creating a statistical based machine translation engine. Additionally, the installation manual of Moses (the open source SMT toolkit) is not clear enough to be understood by everyone. So, I explained the steps of creating SMT engine in a simpler way.

1.1 Theoretical Framework

In this study my departure point in terms of theoretical framework will be Anthony Pym's "Redefining Translation Competence in an Electronic Age. In Defence of a Minimalist Approach" article. Pym states different notions of translation competencies in his article by saying that:

So "competence" cannot be confused with questions of professional qualifications, no matter how much teachers like myself might worry about training students for the workplace. This makes sense, since qualifications change with technology and social demands, bringing in bundles of history that are simply too big for the eternal generalities of a science. Then again, if the science is supposed to help train translators, and translators are going to be employed for whatever competence they acquire, surely we cannot just remain silent about what the market requires? (Pym 2003, 482).

As Pym stated in his article, there is a shifting from translation competence to translator's competence and the market requires so many new skills. These skills include mainly the use of technology, and subsequently CAT tools, Quality Assurance (QA) tools and MT. New job definitions have been started to given to translators. For example, translators are now asked to align the bilingual files in a way the data can be transferred to the MT engine and post-edit these data and also ensure performing quality checks on these data. The reason behind this is to create a project based engines and to reduce the number of people involved in this translation process and to make more profit for translation companies in short terms. This seems something disadvantageous for translators but it may turn into an advantage. In my thesis I will also try to find answers to following questions: Is it possible for translators to turn the technological

improvements, especially machine translation, into an advantage for themselves? Is it possible for a translator who knows very little about technology to create MT engines? What kind of data can be used to create an SMT engine? What are the limitations on the data usage (are there any limitations that prevent us from using each kind of data?) etc.

1.2 General Information on Computer Aided Translation (CAT) Tools

In 1990s, with the opportunity to access to cheaper and smaller computers, and proliferation of the internet usage, computers became one of the most important parts of our lives (Schumacher and Morahan 2001, 96). As mentioned in the previous chapter of this study, this situation has affected the translation studies and the sector. Since then, studies for developing aiding tools for translation have been one of the main concerns of the sector. People involved in the sector have been trying to make the translation process easier and cheaper for both customers and translators. To achieve this, the companies developed so many software called CAT (Computer Assisted Translation) tools. These CAT tools can be used both online and offline and can be evaluated under two headings, cloud-based and desktop tools.

Cloud-Based tools like SmartCAT, Nubuto, Memsource, etc. are the systems that provide the users an account by which they can see the text to be translated in segmented form, translation memory (if any), term list (if any) and mostly use integrated spell checker and also consistency checker in the cloud. Any user can access to his/her translation task via these kinds of tools by entering their usernames and passwords. These tools save the approved segments automatically and keep them in the cloud. Some of these tools have also desktop versions, for example Memsource desktop. The user can also work offline and do translation in these kinds of desktop tools, and these tools synchronize the translation task with the cloud version. All of these tools save the approved segments and keeps it in the respective translation memory for the future references. When the user uses this translation memory for any other projects in future s/he will be able to see the old translated sentences and if there is a match between the old and new texts the tool matches these automatically. This provides users a quick reference to the terms or words s/he used before for the translation of the

respective source text and gives a result depending on the matching percentage of two similar sentences so that the translator does not have to translate the sentences s/he translated before or just needs to edit a fuzzy-matching sentence.

Computer-Based CAT tools are mostly used offline but can be connected to clients' or companies' servers to be able to benefit from the translation memories which clients or companies provide. So, these tools can be classified both as hybrid and as offline. In case of any system breakdown, the saved data (translation memory, glossary, any unsaved segments, the final document) can be lost. However, hybrid computer-based CAT tools can send the data to the clients' or companies' servers so that hybrid solutions prevent data loss. These are the tools that user does not need internet connection to perform translation task. Users can work on it offline and these kinds of programs also provide users a segmented screen like cloud based systems. These tools are very similar to the cloud based ones but all of the translation works are stored in the computer and in case you have any problems with your computer it means you also have problems with your translation task and you cannot access to the translation files.

1.3 General Information on Machine Translation

Machine translation (MT) is an automated translation process in which human translators are not involved. In this process, computer software is used to translate a text from one natural language (such as English) to another (such as Turkish). Till the invention of the computers, it was not possible to mention the automation processes for translation. Translations were made manually by human translators since translators did not have the chance to use electronic devices to help them for their translation tasks but, with the technological developments, they also started to benefit from the aiding tools like CAT tools and also automated systems like MT. A translator using an MT engine just puts the source to be translated in the source text part of MT engine and gets a result without any intervention to the text. It is not what I try to say whether the result is correct or enough to be used but how the system works. As a contemporary topic, we, as translators, editors, academicians and translation studies students, all need to be aware of this automated process since it is shaking the translation industry.



2 THE TRANSFORMATION OF TRANSLATOR

With the technological developments, the world has been changing, and people feel these changes in every field of life. The century we are in now shows these changes' biggest ones since the technological developments have reached a level that almost every day new things come to our lives. This is because the communication with people all over the world is now quite easy and people can show their inventions or studies to the world in a few seconds via internet. From the invention of the steam engine to the invention of telephone and telegram; from the invention of first micro-computer to the wide-spread usage of internet, we see that technology has been developing day by day and all of these revolutions are named like Industry 1.0, Industry 2.0, Industry 3.0 and Industry 4.0.

Industry 1.0 dates back to 1800s, and in that century, with the development of water and steam powered machines to aid workers, new job titles like owners, managers and employees serving customers have come to the fore instead of working areas and narrower job titles.

Following the 1800s, in the 20th century, electricity became the main source of power since it is easier to use compared to water and steam-powered systems. This allowed machines to be able to be designed with their own power sources, and made them more portable. And also management programs were developed so that facilities had chance to increase their effectiveness and efficiency. So, the beginning of this century is called Industry 2.0 age.

Industry 3.0 is the age that electronic devices were invented and manufactured to make it possible for more fully automate individual machines to supplement or replace operators. In this period software systems were developed to benefit from electronic hardware. To enable humans to plan, schedule and track product flows through the enterprise, planning tools replaced material planning systems. Because of the will of cutting the costs, installation of products were conducted in low cost countries such as China and this created a supply chain. To manage this supply chain the requirement of using supply chain management software became a must. The last decades of 20th century involve these developments, and this period is called Industry 3.0 age.

Today, 21st century is called as internet of things (IOT) age. Industry 4.0 mainly covers artificial intelligence, robotic issues and fully automated systems where machines take care of most of the tasks instead of people. Industry 4.0 combines information technologies (IT) with manufacturing. The developments within the Industry 4.0 help people to track their manufacturing activities, project details, etc. in real time. It uses cyber-physical systems to manage processes without human intervention. (Source: <http://www.apics.org/apics-for-individuals/apics-magazine-home/magazine-detail-page/2017/09/20/industry-1.0-to-4.0-the-evolution-of-smart-factories> (accessed March 14, 2019)).

I, in my thesis, try to classify the technological developments in translation industry and reflections of these developments on translators. So I categorize translators as translator 1.0, 2.0, 3.0 and 4.0 depending on the requirements of a specified period for translators. It is not claimed that this classification for translators corresponds to the very period of Industry 1.0, 2.0, 3.0 and 4.0. It is just about the technological developments in translation industry and refers to the translators. In the illustration below, the technological developments in time can be seen.











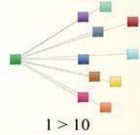

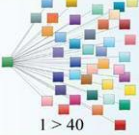

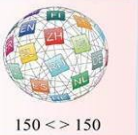





	1980 TRANSLATION	1990 LOCALIZATION	2000 GLOBALIZATION	2010 INTEGRATION	2020 CONVERGENCE
INDUSTRY FOCUS	 Documents	+  Software	+  Simship	+  Integration in enterprise systems	+  Embedded in every app, on every screen
TYPE OF CONTENT	 Paper	+  Digital	+  Static Web	+  Dynamic Web	+  Personalized
LANGUAGES	 1 > 10	 1 > 25	 1 > 40	 6 <> 60	 150 <> 150
	0%	10%	20%	30%	40%
TECHNOLOGY	None	TM and terminology software	Workflow (GMS)	MT and Advanced Leveraging	Real-time customized MT
DATA	Glossaries	Project TMs	Centralized TMs	Limited shared data	Unlimited big data
COMMUNICATIONS	 B2B G2C B2C	+  C2B	+  C2C	+  Social	+  M2M and IoT

Figure 1 The Game Changers of 2016 (Source: <https://blog.taus.net/the-game-changers-of-2016?fbclid=IwAR0IagI9v0tDmgnE181Ai2YXdIQ7kXX76fWdwXYnPOuKifGV1gV3Qsz24sA> (accessed January 15, 2019));

2.1 Translator 1.0

It was 1960s... An office room... A few tables... Fast typewriters... Typewriter erasers... Manila folders... In them, translation and original text files are filed... Dictionaries... Papers, placed between folios... On them, in addition to the dictionary, vocabulary entries, and their usages are written.

A few translators... College graduate or language proficient... Perhaps the not even a college graduate but loving the translation job, believe in developing his/her grammar through translation... Phone... Fax... Accounts held in a notebook...¹

In 1960s, not only in Turkey, in the world standard that I tried to describe a translation agency. (Öner 2006, 234)

¹ Unless otherwise stated, all translations are mine.

In her article, Öner summarizes the situation of translation agencies and translators in 1960s. As can be understood from the article, translators were not graduated from translation and interpretation departments, translation tasks were undertaken by college graduates or people speaking a foreign language, and also these translators were using typewriters. At that time, out of the ordinary, the translation process, from translating a text to hold accounts, seemed running manually, in non-digital media. Translators did not have any computers in their agency, they did not use CAT tools, they all were unaware of the systems or did not use any of them like MT, TM, electronic dictionaries, term-bases. I prefer to call these kinds of translators as Translator 1.0 since they are at the very beginning of the technological developments.

2.2 Translator 2.0

In 1980s there were still not ground-breaking developments in translation industry, and the main focus of the industry was on paper, and glossaries were used. The computers were not fully included in translation process. The language combinations were not that much. But in 1990s, the access to cheaper computers and the widespread usage of internet led translators somewhere else. With the entering of localization into our lives, digital platforms replaced the papers, and software replaced documents, TMs and terminology software started to be used, the number of language combinations increased. People who involved the translation process were obliged to keep pace with these developments and use the software, TMs and terminology software, etc. Then they turned into Translator 2.0, who uses technology for the translation tasks.

2.3 Translator 3.0

When it was 2000s with the effect of globalization, and other technological developments, the number of the language combinations was increased, and to cater the translation needs translation process started to transform into a faster process that electronic corpus, faster internet access, computers which are connected to network, term-banks, etc. came into prominence. Öner, summarizes the situation as follows:

2000s... A company building... Networks, dozens of computers are connected... Fast internet access via different channels... Electronic library... Consisting of dictionaries, encyclopedias... Data banks...

Computer aided translation tools... Project management tools... Breaking down the translations undertaken by the company, showing what phase the project is in, which translators are working on which projects...

Translators, all of them are graduates; moreover, most of them are translation department graduates, engineers. Working as a project manager, language engineer, quality control specialist... (Öner 2006, 234).

As can be understood from her article, people who were involved in translation process were using computer aided translation tools; project management tools, and new job definitions like project manager, language engineer, quality control specialist became a part of translation process. So, using CAT tools or other software required new skills for people who were involved the translation process, and translators had to gain these new skills. Then they turned into being Translator 3.0.

2.4 Translator 4.0

When we consider the transformation of the translator from past to now, today, it is at a level that even the discussions about the machines replacing human translators have come to the scene. Translators need to gain new skills beyond translating or using CAT tools. They need to learn how to post-edit a translation retrieved from a machine translation engine, or translation agencies are developing customer oriented machine translation engines for the big volume projects so that they can make profit by decreasing the number of translators involved in the translation process. These agencies try to make the machine translation engines handle at least 30-40% of the translation task, and pay translators for just 60-70% of the document to be translated. When a one-million- words project is considered, it means 300-400 thousands of words that the machine translation engine is expected to handle. This seems of course disadvantageous for translators since the need for translators decreases. But when we consider the machine translation engine creation phases, these translators can handle different tasks. So as to create the data to train an engine, translation agencies still need human

translators. Translators who improve themselves in terms of the requirements to create data, and learn the background of an MT engine can still work under a different job definition. This definition might be language engineer, post-editor, etc.



3 MACHINE TRANSLATION

Machine translation systems date back to World War II. For military purposes machine translation engine was built during that time. Then, it has been developed, and different machine translation systems have been used. The examples of machine translation engine systems are rule based, statistical and neural. Each of them has different way of learning and working.

3.1 The History of Machine Translation

The technology has always been the determinant for people and also for governments in the history. From invention of the wheel to contemporary advances, the technology has become the most important part of our lives. The technological developments have a big effect on each discipline from religion to health and so on. One of these disciplines is Translation Studies, and as the participants of this discipline we feel the effect of these developments deeply.

Translation is as old as the history itself and all kinds of developments in the history have been felt in translation too. From the invention of the computer to the neural-based systems, translation has felt the effects of these developments. These effects started to be felt in translation industry with the first computer's, ENIAC's, invention. It was built during World War II for military purposes and became the departure point of so many advances. After the building of this general purpose computer, the developments on the computer science got faster and for the military's intelligence purposes the translation was tried to be made a part of the computational systems as people thought that providing intelligence from the opponent will be possible via computers with the integration of translation systems to computers. From that time the researchers shifted their focus to find a way to make the translation possible by using computers. With this object in mind the Statistical Machine Translation idea was put forward. Studies on creating a system that understands a foreign language and decodes its structure to give meaningful words or sentences had started.

After these historical developments, in 1980s, people felt the impact of technology on translation industry more deeply. After that, the aiding tools like

Translation Memories (TM), still widely used in translation industry, have been created. Beside this, the CAT (Computer Assisted Translation) tools were started to be developed. In 1990s, globalization and prevalence of the internet provided people to access cheap and powerful computers and subsequently the developments in the field of Translation Technologies increased. We can give the developments in SMT (Statistical Machine Translation), firstly mentioned in 1949, and also NMT (Neural Machine Translation) as examples. We felt the impact of these developments the most in 2016 with the introduction of the Google Translate's NMT engine. After that time people started to think if the machine replaces the human translator or not (cf. Öner Bulut, 2019). In the next chapters, I will give basic information on SMT and NMT, and also try to explain the possible effects of MT on human translators. And finally I will explain the steps of creating an SMT engine. (Source: <https://kantanmtblog.com/2013/07/12/the-history-of-mt-pt-1/>, (accessed February 14, 2019)).

3.2 Rule Based Machine Translation

The Rule Based Machine Translation is a system which contains monolingual, bilingual texts and so many grammar rules, lexicon and software programs for processing of these rules, and gives a translation suggestion in line with these rules, lexicon and texts given to engine. Linguistic rules are collected and processed in analysis, transfer and generation stages. The machine is trained via so many grammatical rules of the respective source and target languages. Then, machine processes these rules and puts the given words in order and gives a result. When big volume of data is used, the result retrieved from these machines are generally not useful when compared to SMT since there are countless grammar rules of a language. Of course it is possible to teach the engine all of these rules but its costs are really high and it is a long process. To create a RBMT engine the data needed is less than SMT or NMT. By teaching linguistic rules and using dictionaries a small scale RBMT can be created, but this may not be enough to give fluent results. Creating a RBMT requires linguistic expertise to be able to

introduce the linguistic rules to the engine. (Source: <https://kantanmtblog.com/2014/02/13/rbmt-vs-smt/>, (accessed February 14, 2019)).

3.3 Statistical Machine Translation (SMT)

Statistical Machine Translation is a system that finds the most probable translation of a foreign language that was introduced to the machine via monolingual and bilingual parallel corpus. We give so many parallel texts to the engine to get the translation of a sentence or a word; and the engine provides us the most probable translation in relation with the given text. While training the engine the bilingual corpus needs to be segmented and aligned to ensure that engine can learn the translation of the source text. This seems similar with the Translation Memories which are widely used in translation industry. But the main difference between the TM and engine is that TM gives us the existing matches and when there is a big change in the source we get no result, but the engine gives us the most probable result by using n-gram method. N-gram method is a probability of guessing the word sequence according to the data given to the engine (cf. Clark, n.d.). So as to n-gram model to function, the text given to the machine must be aligned and cleaned. This process is explained in the alignment part. To create an SMT engine there is no need to be a language expert since there is no need to teach the engine any rule to get a translation result. A well-trained SMT engine provides fluent translation results and is better in terms of terminological consistency. When it does not provide fluent translation result it is easier for post-editor to detect the error and correct it than NMT. NMT engines provide logical translation results even if it is not correct. For post-editors dealing with the NMT engine output, it is not possible to see the error without checking the source text. This is the main reason why I work on SMT engine.

3.4 Neural Machine Translation (NMT)

NMT is a system that learns and performs translations via encoding – decoding systems (cf. Bahdanau and et al., 2014) and gives logical translation results depending on the bilingual data given to it. Apart from SMT there is no need to parsing the sentence in NMT as NMT systems examine the full context of a sentence and try to create more

fluent and meaningful sentences. It converts the source words into numeric representations and creates target words. In NMT there is no need to give monolingual data to the engine but parallel data must be given in higher amounts for higher quality. NMT outputs are meaningful words or sentences that they give very good results in general but in specific domains well-trained SMT engines still work better when compared to SMT (with the same amount of data). To create an NMT engine the data needed are millions of segments to start. It is of course possible to collect such a big volume of data and make it ready for NMT engine but this is a really long lasting process when compared to SMT engine. In addition, NMT engine outputs are meaningful sentences and it is hard to find out the errors without checking the source text. So, a post-editor needs to check the source text because mechanical QA tools are not enough to identify the errors. When we check on Google Translate, after the launching of it in 2016 the speculations that claimed that NMT translates everything arose since it makes great job especially in English – Turkish language combination, the results we get are mostly meaningful and linguistically correct. This makes it difficult to realize if there is an error in the target.

3.5 Differences between SMT and NMT

To mention the basic differences between SMT and NMT engines, it can be said that the main difference between SMT and NMT is the readability and fluency of the output. The NMT engines give more logical outputs as they deal with the full context although it is not correct. There is no need to give monolingual data to NMT but SMT needs monolingual data. For more specific domains, SMT can be considered as better than NMT as NMT needs much more data than SMT. When the training data is not similar with the text to be translated, NMT works better as it will give logical outputs (of course when enough data is given). Although giving logical, linguistically correct outputs seems one of the biggest advantages for NMT, it is going to be harder to detect the errors when the training stage of the engine is considered. Although QA tools can be used to detect the errors of an SMT engine output, it is not possible for NMT engine

outputs. For training of an NMT engine the data needed is at least 20 times higher than SMT. Omniscien Technologies² states that:

Typical SMT engine will range in data size from 1-8 million segments, with a large engine as much as 20 million segments. On the other hand, a typical NMT engine starts at 20 million segments of in-domain data and goes up into hundreds of millions of segments. Few client companies have ever had anywhere near these volumes of data. In 10+ years of business, we have only encountered perhaps 3 or 4 companies with such volumes. (Source: <https://omniscien.com/migrate-from-smt-to-nmt/> (accessed April 25, 2019)).

Due to these restrictions I have opted to create an SMT engine and show how it can be created.

3.6 Differences between SMT and RBMT

Rule Based Machine Translation engine can provide high quality outputs even in small scale projects (in terms of the volume of data given to the engine) when all the rules of the languages are taught well, but the time needed to teach all the rules to the engine is quite long and the expenses are really high. When it is considered that languages are living things, the rules are continuously changing and for each change the engine needs to be updated when it is necessary. To update the engine, even to teach the engine the basic rules, a language expert is needed. SMT engines learn from the data given to it and give similar outputs to the training data without learning any of linguistic rules. Due to the long-lasting and costly training stage of RBMT engines, SMT engines are much more preferred because bilingual data can be retrieved from internet although it is still long-lasting process to make it ready to give the engine. To sum up due to the high costs, long time requirement and need for a language expert to create a RBMT engine, the SMT is one step forward when compared to RBMT. Moreover, getting the data for SMT engines is possible thanks to the fact that internet secures SMT engines' position against RBMT.

² Omniscien Technologies is a leading global supplier of high-performance and secure high-quality Language Processing, Machine Translation (MT) and Machine Learning technologies and services for content intensive applications (Source: <https://omniscien.com/more/about/>, (Accessed April 25, 2019)).

3.7 Corpus

The documents I used to create an SMT engine are user manuals which should be dealt with under the title of technical translation. I have been working as a freelance translator for 7 years and my specialization area is technical translation. So I chose user manuals as the data, and also due to the restricted volume of data I can use for creating an SMT, it will be hard to get a result from the engine for other text types like literary texts. Before starting to create a SMT engine let's check briefly what technical translation is, and automotive texts mean.

3.7.1 Technical Translation

“In the early 1990s, the growth of the internet has made it much easier for software publishers to distribute and market their products in other countries” (Esselink 2000, 5). This was, of course, not only the case for software publishers but also for the other producers who wanted to sell their products in different countries. The producers knew that they should reach to the target customers somehow to market their products. To do this, they had to make their products attractive to the customers in a way that this would not be possible without making the purpose, function, manual of the respective product understandable for the customers who did not speak the same language with them. To achieve this, they localized the respective content of the products such as user manuals, advertising texts, etc. All of these kinds of texts can be regarded under the title of technical translation. Briefly, what is technical translation?

A technical translation is a type of specialized translation involving the translation of documents produced by technical writers, or more specifically, texts that relate to technological subject areas or texts that deal with the practical application of scientific and technological information (Source: <https://www.technitrad.com/what-is-technical-translation/>, (accessed 23 May, 2019)).

3.7.2 Automotive Texts

Automotive sector has been growing day by day and the need for the vehicles increases with it. With the globalization impact, companies have been forced to open new facilities in different countries to meet the requirements of their customers from all over the world. This spill-over effect also brings the translation need to address different language speaking customers into the forefront. From the beginnings of 2000s, automotive companies have been asking the translation companies for creating machine translation engines to reduce the translation costs. Yamagata, a localization company founded in Japan, revealed a case with Honda in which the company was asked for developing an MT system in order to help Honda for reducing translation costs, and giving quick responses to the customers' needs. The Honda also asked the company for the system to be fully integrated with their IT systems (Source: <https://www.youtube.com/watch?v=aIxI8mGbNuY>, (accessed, June 3, 2019)). This is just a simple example that proves the need for MT systems for huge volumes of texts and demand of the customers. Increasing number of automotive and new systems, integrated with autos, will keep pushing the need for new translations up. So, companies tend to use MT systems instead of human translations. By taking this factor into consideration as well as the other factors like my field of interest, I have chosen to use automotive texts in my study.

4 CREATING A STATISTICAL MACHINE TRANSLATION ENGINE

In order to create a machine translation engine, we need a big corpus. To create this corpus, online sources can be used but to me, without the permission of the owner of data, it is not ethical to benefit from the online sources. In this chapter, ethical considerations, Linux-Ubuntu installation, Moses installation, preparing the data, GIZA++ and engine training steps are explained.

4.1 Ethical Considerations

As mentioned before the main requirement to create an SMT engine is data. But first of all we have to know if we can use any data related to domain we will work on. There are millions of words and their translations in the internet which can be accessible by anyone. Can we use these data as we wish? I reckon, the answer to this question is absolutely no. Because the data's being open to everyone in the internet does not mean that the owner of the data allow everyone to use his/her words as they want. So it will not be ethical for us to use any data even if it is open source. Before we use them, we need to get permission from the owner of the data.

4.2 Linux-Ubuntu Installation

As mentioned in the previous pages, we need Linux operating system to use Moses for creating an engine. In this part, the steps of installing Linux operating system are explained. First of all, I need to state that we may use various operating systems such as Windows, Linux, MacOS. We can install these operating systems to our computers from beginning but if we use any of these and do not want to remove it from our computer and install a new operating system we can prefer to install a virtual machine tool so that we can use two different operating systems in our computer at the same time. The operating system of my computer is Windows 10 and I need to install Linux-Ubuntu to setup Moses on. As I did not want to remove Windows from my computer I installed Virtual Tool Box and setup the Ubuntu. In this part I will explain how to install Ubuntu and the figures can also be seen at the Appendix part.

First of all, download the virtual machine tool from <https://www.oracle.com/tr/virtualization/virtualbox/> website. Then, download the Linux- Ubuntu setup file (ISO file) from <https://ubuntu-tr.net/> website. Click on the virtual machine setup file to install. Select the type of operating system as Ubuntu by typing Ubuntu in the operating system type field and click next. Give a name to virtual machine and click next. Select the RAM size depending on the features of the computer, and click next. Select the VDI (Virtual Disk Image) from the virtual disk creation wizard screen. Then virtual disk storage details screen will be opened and select the dynamically allocated option from the screen. Then, click on the create option to create the virtual machine. Now virtual machine is installed and ready to install Ubuntu. Click on the devices option from the upper part of the screen and select, CD/DVD devices option and upload the ISO file which was downloaded before. From the opened screen, select the language, and click on install Ubuntu option then, next. Designate the time-zone and click next. Now Ubuntu will be installed on the virtual machine and Moses can be downloaded and installed.

4.3 Moses

There are many systems, both cloud and desktop, to help creating an SMT engine like KantanMT, Mtradumatica, etc. but the most foremost of these systems is Moses. KantanMT, MTradumatica are the systems which were created by using Moses as base. Since all of the systems are reproduced by using Moses as the base, I have chosen to work on the main system (Moses) to create an SMT engine. Moses is an open source SMT system that ensures individual users to create their own SMT engines just by preparing their parallel texts (source and translation) (Koehn 2018, 11). It provides a substructure to create an engine and help users to train their engines in line with the data they have. Moses system is a project³ developed by Philipp Koehn, computer scientist, academician in Edinburgh University and machine translation researcher, and his fellows. They also wrote a manual on how to use Moses system and gave the necessary

³ Koehn, Philipp. 2018. "The Moses decoder was supported by the European Framework 6 projects EuroMatrix, TC-Star, the European Framework 7 projects EuroMatrixPlus, Let's MT, META-NET and MosesCore and the DARPA GALE project, as well as several universities such as the University of Edinburgh, the University of Maryland, ITC-irst, Massachusetts Institute of Technology, and others."

codes to install and start Moses. These codes can be used to create individual SMT engine by using our own parallel corpus. In the next section I will share these codes and show how the individual translation model can be created. The initial step is to create an SMT engine by using our own parallel corpus (automotive texts). Before installing Moses, I must specify that all of the codes to install Moses are obtained from the manual of Moses, and <http://achrafothman.net/site/how-to-install-moses-statistical-machine-translation-in-ubuntu/>, (accessed June 15, 2019) web site, and adapted accordingly.

4.3.1 Installing Moses

First of all, in order to download and install Moses, the terminal needs to be opened. Then, the necessary codes need to be entered to start to create the SMT engine on the computer. Necessary codes will be given one by one. Firstly, a workplace needs to be created on the computer to ensure all the necessary files and packages are saved in it. To do this, “mkdir smt” code needs be entered. This will create a folder named “smt” on the home page. Then “cd smt” code needs to be entered so that all the required packages will be saved in the file. These packages are Ubuntu packages required to create Moses engine.

To install these packages, the code that needs to be written is “sudo apt-get install build-essential git-core pkg-config automake libtool wget zlib1g-dev python-dev libbz2-dev”. After writing this code the system will ask for password. The password here is what is designated to start to the operating system, Ubuntu. Note that due to the security issues of the system, what is written to enter the password cannot be seen in the system. So that, when the password is written it is needed to press the enter button although there is nothing on the screen. Then the system will install the required packages for Ubuntu. The screen will be as below.

```

alper@alper-VirtualBox:~$ sudo apt-get install build-essential git-core pkg-config automake libtool wget
[sudo] password for alper:
Sorry, try again.
[sudo] password for alper:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'git' instead of 'git-core'
The following additional packages will be installed:
  autoconf autotools-dev cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7
  gcc-7-base gcc-8-base git-man libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1
  libc-dev-bin libc6-dev libcc1-0 libckit5 libdpkg-perl liberror-perl
  libfakeroot libgcc-7-dev libgcc1 libgomp1 libitm1 liblsan0 libltdl-dev
  libmpx2 libquadmath0 libsigsegv2 libstdc++-7-dev libstdc++6 libtsan0
  libubsan0 linux-libc-dev m4 make manpages-dev
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc cpp-doc gcc-7-locales
  debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg
  gcc-multilib flex bison gcc-doc gcc-7-multilib libgcc1-dbg libgomp1-dbg
  libitm1-dbg libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg
  libubsan0-dbg libckit5-dbg libmpx2-dbg libquadmath0-dbg git-daemon-run
  | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn glibc-doc bzip2 libtool-doc libstdc++-7-doc gfortran
  | fortran95-compiler gcj-jdk m4-doc make-doc
The following NEW packages will be installed:
  autoconf automake autotools-dev build-essential dpkg-dev fakeroot g++ g++-7
  gcc gcc-7 git git-man libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libc-dev-bin libc6-dev
  libckit5 liberror-perl libfakeroot libgcc-7-dev libitm1 liblsan0
  libltdl-dev libmpx2 libquadmath0 libsigsegv2 libstdc++-7-dev libtool
  libtsan0 libubsan0 linux-libc-dev m4 make manpages-dev pkg-config

```

Figure 2 A screenshot from Linux terminal showing the installation command is working

While the system is installing the required packages it will stop and ask for permission to use disk space. To allow it, Y, should be written as the answer to the question “Do you want to continue? [Y/N]”. The figure is below. Once “Y” is written the system will go on downloading and installing the packages.

```

File Edit View Search Terminal Help
Note, selecting 'git' instead of 'git-core'
The following additional packages will be installed:
  autoconf autotools-dev cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7
  gcc-7-base gcc-8-base git-man libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1
  libc-dev-bin libc6-dev libcc1-0 libckit5 libdpkg-perl liberror-perl
  libfakeroot libgcc-7-dev libgcc1 libgomp1 libitm1 liblsan0 libltdl-dev
  libmpx2 libquadmath0 libsigsegv2 libstdc++-7-dev libstdc++6 libtsan0
  libubsan0 linux-libc-dev m4 make manpages-dev
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc cpp-doc gcc-7-locales
  debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg
  gcc-multilib flex bison gcc-doc gcc-7-multilib libgcc1-dbg libgomp1-dbg
  libitm1-dbg libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg
  libubsan0-dbg libckit5-dbg libmpx2-dbg libquadmath0-dbg git-daemon-run
  | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn glibc-doc bzip2 libtool-doc libstdc++-7-doc gfortran
  | fortran95-compiler gcj-jdk m4-doc make-doc
The following NEW packages will be installed:
  autoconf automake autotools-dev build-essential dpkg-dev fakeroot g++ g++-7
  gcc gcc-7 git git-man libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libc-dev-bin libc6-dev
  libckit5 liberror-perl libfakeroot libgcc-7-dev libitm1 liblsan0
  libltdl-dev libmpx2 libquadmath0 libsigsegv2 libstdc++-7-dev libtool
  libtsan0 libubsan0 linux-libc-dev m4 make manpages-dev pkg-config
The following packages will be upgraded:
  cpp cpp-7 gcc-7-base gcc-8-base libcc1-0 libdpkg-perl libgcc1 libgomp1
  libstdc++6 wget
10 upgraded, 38 newly installed, 0 to remove and 570 not upgraded.
Need to get 40.0 MB/40.9 MB of archives.
After this operation, 157 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```

Figure 3 A screenshot from Linux terminal to continue the process

From now on the terminal will remember and not ask for password again. Then, necessary packages need to be downloaded by typing the code “sudo apt-get install (here, the name of the package is written)”, for example “sudo apt-get install g++”. The

necessary packages are “g++”, “subversion”, “git”, “git automake”, “git libtool”, “git libtool zlib1g-dev”, “git libtool libboost-all-dev”, “git libtool libbz2-dev”, “git libtool liblzma-dev”, “git libtool python-dev”, “git libtool graphviz”, “git libtool imagemagick”, “git libtool make”, “git libtool cmake”, “git libtool autoconf”, “git libtool doxygen” and “libsoap-lite-perl”.

The necessary packages like GIZA++⁴ and IRSTLM⁵ should be downloaded as shown in the next steps. The necessary codes to download GIZA++ are “git clone <https://github.com/moses-smt/giza-pp.git>”, “cd giza-pp”, “make” respectively. After downloading and installing processes are completed, GIZA++ binaries need to be copied to Moses Decoder. The necessary codes to do this are “cd ../mosesdecoder”, “mkdir tools”, “cp ../giza-pp/GIZA++-v2/GIZA++ ../giza-pp/GIZA++-v2/snt2cooc.out ../giza-pp/mkcls-v2/mkcls tools”, “cd ..”, respectively. In the next step, the language model should be installed. To do this, firstly, the IRSTLM should be downloaded and unzipped to the home directory. The IRSTLM can be downloaded from the website <https://sourceforge.net/projects/irstlm/files/irstlm/irstlm-5.80/irstlm-5.80.08.tgz/download>. After it is downloaded and unzipped to the home directory, it needs to be compiled. To do this, the necessary codes are, “mkdir irstlm”, “cd irstlm-5.80.08”, “cd trunk”, “./regenerate-makefiles.sh”, “./configure --prefix=/smt/irstlm” (here, the smt is the folder that is created in the beginning, if any other folder name was given, then that should be written instead of smt). After entering this code, the necessary codes to install the IRSTLM are “make install”, “cd ..”, “cd ..”. Now it is time to install the boost⁶ manually. “Boost provides free peer-reviewed portable C++ source libraries” (Source: <https://www.boost.org/>, (accessed May 15, 2019)). In order to prevent any Moses compilation failure, the boost needs to be installed manually.

⁴ GIZA++ is an automatic word alignment tool that trains the parallel corpus several iterations from two directions (source to target language and vice-versa) (Word Alignment Using GIZA++ on Windows, Liang Tian, Fai Wong, Sam Chao, 369, 2015).

⁵ Federico, M., N. Bertoldi and M. Cettolo. 2008. IRSTLM (IRST Language Modeling) is a toolkit that features algorithms and data structures suitable to estimate, store, and access very large LMs, IRST Language Modeling ToolkitVersion 5.20.00USER MANUAL.

⁶ Boost is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing, (Source: https://www.ace-net.ca/wiki/Boost_C%2B%2B, (Accessed April, 15, 2019)).

The necessary codes to install the boost manually are “wget”, “https://dl.bintray.com/boostorg/release/1.64.0/source/boost_1_64_0.tar.gz”, “tar zxvf boost_1_64_0.tar.gz”, “cd boost_1_64_0/” , “./bootstrap.sh”, “./b2 –layout=system link=static install || echo FAILURE”, “cd ..”. This lasts a little bit long depending on the features of the computer. Once this step is completed, the CMPH2.0⁷ needs to be installed. “wget”, “http://www.achrafothman.net/aslsmt/tools/cmph_2.0.orig.tar.gz”, “tar zxvf cmph_2.0.orig.tar.gz”, “cd cmph-2.0/”, “./configure”, “make”, “make install” are the codes necessary to install the CMPH2.0. Then, XML-RPC needs to be downloaded via entering the codes, “wget http://www.achrafothman.net/aslsmt/tools/xmlrpc-c_1.33.14.orig.tar.gz”, “tar zxvf xmlrpc-c_1.33.14.orig.tar.gz”, “cd xmlrpc-c-1.33.14/”, “./configure”, “make”, “make install”, “cd ..”. Last but not least, to install Moses, “cd mosesdecoder”, “make -f contrib/Makefiles/install-dependencies.gmake”, “./bjam –with-boost=../boost_1_64_0 –with-cmph=../cmph-2.0 –with-irstlm=../irstlm” are the necessary codes. This will last approximately 1 hour, of course depending on the features of the computer used, and internet speed. After the completion of installation, it is time to prepare the data and train the SMT engine. Note that Moses is installed in smt folder. To sum up, the codes below are used respectively:

To create a folder and save the files in it, enter:

- `mkdir smt`
- `cd smt`

To install the respective Ubuntu packages, enter:

- `sudo apt-get install build-essential git-core pkg-config automake libtool wget zlib1g-dev python-dev libbz2-dev`

To install the necessary packages, enter:

- `sudo apt-get install (the name of the package is written here without bracket)`

Packages are:

- `g++`
- `subversion`

⁷ Cmph is a free minimal perfect hash C library, providing several algorithms in the literature in a consistent, ease to use, API, (Source: <https://sourceforge.net/projects/cmph/>, (Accessed April 15, 2019)).

- `git`
- `git automake`
- `git libtool`
- `git libtool zlib1g-dev`
- `git libtool libboost-all-dev`
- `git libtool libbz2-dev`
- `git libtool liblzma-dev`
- `git libtool python-dev`
- `git libtool graphviz`
- `git libtool imagemagick`
- `git libtool make`
- `git libtool gmake`
- `git libtool autoconf`
- `git libtool doxygen`
- `libsoap-lite-perl`

To install GIZA++, enter:

- `git clone https://github.com/moses-smt/giza-pp.git`
- `cd giza-pp`
- `make`

To copy the binaries, enter:

- `cd ../mosesdecoder`
- `mkdir tools`
- `cp ../giza-pp/GIZA++-v2/GIZA++../giza-pp/GIZA++-v2/snt2cooc.out../giza-pp/mkcls-v2/mkcls tools`
- `cd..`

To download the IRSTLM, go to:

- <https://sourceforge.net/projects/irstlm/files/irstlm/irstlm-5.80/irstlm-5.80.08.tgz/download>

To compile the IRSTLM, enter:

- `mkdir irstlm`
- `cd irstlm- 5.80.08`

- `cd trunk`
- `./regenerate-makefiles.sh`
- `./configure --prefix=/smt/irstlm`
- `make install`
- `cd..`

To install and compile the boost, enter:

- `wget https://dl.bintray.com/boostorg/release/1.64.0/source/boost_1_64_0.tar.gz`
- `tar zxvf boost_1_64_0.tar.gz`
- `cd boost_1_64_0/`
- `./bootstrap.sh`
- `./b2 --layout=system link=static install || echo FAILURE`

To install and compile CMPH, enter:

- `cd..`
- `wget http://www.achrafothman.net/aslsmt/tools/cmph_2.0.orig.tar.gz`
- `tar zxvf cmph_2.0.orig.tar.gz", "cd cmph-2.0/`
- `./configure`
- `make`
- `make install`

To install and compile XML-RPC, enter:

- `wget http://www.achrafothman.net/aslsmt/tools/xmlrpc-c_1.33.14.orig.tar.gz`
- `tar zxvf xmlrpc-c_1.33.14.orig.tar.gz`
- `cd xmlrpc-c-1.33.14/`
- `./configure`
- `make`
- `make install`
- `cd ..`

To install Moses, enter:

- `cd mosesdecoder`

- `make -f contrib/Makefiles/install-dependencies.gmake`
- `./bjam --with-boost=../boost_1_64_0 --with-cmph=../cmph-2.0 --with-irstlm=../irstlm`

This will take a little bit longer depending on the internet speed and features of the computer used. Once the installation is completed, Moses will be ready to be trained.

4.4 Preparing the Data

This section will provide information on how to do (pre-) alignment, GIZA++ and how to train the engine in the following subsections

4.4.1 (Pre)Alignment

The main requirement to create an SMT engine is of course bilingual data. The data used to create an engine must be aligned before given to the engine otherwise the engine cannot recognize the source text and its meaning, target text. This alignment process is just to be sure that the source and target sentences are in the same lines. After the alignment process is completed, the source and target texts should be saved in .txt format, separately (one .txt file will include the English sentences, and the other will include Turkish sentences). The actual alignment will be conducted by GIZA++ in the training phase after the necessary codes are entered. The detailed information on what GIZA++ is and how it can be installed is given in the Installing Moses part.

The engine will recognize the source and its translation and align them according to its own algorithms. In order to ensure that the machine can recognize the corresponding sentences, and align the sentences by itself via GIZA++, variables are added to the segments and its corresponding translation (target). Variables are the tags added to the beginning of the source and target texts. The same variables are added to the beginning of two corresponding sentences and the built-in alignment tool recognizes the source sentence and its translation via these tags. These variables specify what the translation of the source sentence is in the target .txt file. In short, this alignment step

can be considered as preparation phase for actual alignment step. Once the necessary variables are added to the segments, the machine can recognize the bilingual segments.

For this study, I have approximately 600,000 words of source text (English) and the Turkish translation of the source. To align the source and target texts sentence by sentence, different alignment tools can be used. The tool aligns the source and target according to its own algorithms but before we add variables to the files, the (pre)aligned sentences must be checked by a human since there may be some misalignments as we can see in the figure below. In addition, we need to clean the data to be used for the engine. So that, following the alignment step, the data is checked for mechanical errors like punctuation, spelling, missing numbers, etc. After that, the necessary corrections are made by a human to ensure the correct data is given to the engine. The alignment examples can be seen in the figures below. In the first figure, the raw source and target data that was not aligned and was not cleaned is shown. In the second one, the aligned sentences can be seen. It is the machine translation engine creators' responsibility to ensure the correctness of the lines to be able to make it possible for the engine to learn in a correct way and to get higher quality results. So, lines need to be checked one by one until we make sure that target line (translation) corresponds the source line.

No	English	Turkish
458	Environment to save money and the environment by reducing fuel consumption.	Önceden planlama ile aracı kinetik enerjisini daha iyi kullanarak yapılan ekonomik sürüş, yakıt tüketimini %10'a kadar azaltır.¶
459	The main factors affecting the fuel consumption are your driving and the condition of the vehicle.¶	•
460		Tekerlek frenlerinin aşınmaması için retarder ve egzoz frenini kullanın.¶
461	Your driving •Plan ahead when driving so that you avoid large variations in speed.¶	•
462	•	Önceden ısıtılmış bir motor, soğuk motorla çalıştırmaya göre daha az aşınacaktır.¶
463	Economical driving with good forward planning and greater use of the vehicle's kinetic energy can reduce the fuel consumption by 10%.¶	•
464		Sürüşün ardından motor durdurulmadan önce yaklaşık 1 dakika kadar rölantide çalıştırın.
465		Aksi takdirde, turbo şarjın zarar görmeye tehlikesi vardır.¶
466	•	Aracın durumu •Bakımı iyi yapılmış bir motor ve yakıt sistemi daha verimli çalışır.
467	Use the retarder and exhaust brake to save the wheel brakes.¶	Uzun dönemde, tavsiye edilen servis aralıklarına uymak yerine araç bakımını geciktirmek daha pahalıya mal olacaktır.
468	•	İyi bakılan bir araç, önemsenmeyen ve bakımı kötü yapılan bir araçta göre her zaman daha ekonomik olacaktır.¶
469	A preheated engine is subject to less wear when starting than a cold engine.¶	Tamam!7¶
470	•	Çevre •Araç iyi durumda tutulduğunda, hizmet ömrü ve yakıt tasarrufu artacaktır.¶
471	Run the engine at idling speed for approximately 1 minute after driving before switching off the engine.	Not!
472	Otherwise there is a risk of the turbocharger being damaged.¶	Temiz bir hava filtresi, yakıt tüketimini önemli oranda düşürür.¶
473	Vehicle condition •A well-serviced engine and fuel system give good efficiency.	Ekonomik dışı oran Aracınızın ekonomik dışı oran varsa, hız denetiminde yaklaşık 1.200 dev/dak motor devrine sahip olacaktır.
474	In the long run it is always more expensive to neglect vehicle maintenance than to observe the recommended service intervals. A vehicle that is well looked after is always more economical in operation than one that is neglected and poorly maintained.¶	Düşük motor devri, yakıtın tasarruf etmesini anlamına gelir.
475	Complete!6¶	Daha fazla vites değiştirmeyi gerektirebilir.¶
476	Environment •The vehicle's service life and fuel economy are improved if the vehicle is kept in good condition.¶	Aracınızda aşırı hız (over drive) donanımı varsa sabit hızda ağır bir yükte (örn. yokuşlu bir arazide veya araç tam yükliken) 11. vitesle sürülebilirsiniz.
477	Note!	Daha iyi çekiş gücüne sahip olursunuz ve çok fazla vites değiştirmenize gerek kalmaz.¶
478	A clean air filter considerably reduces the fuel consumption.¶ Economy gear ratio If your vehicle has an economy gear ratio, it will have an engine speed of approximately 1,200 rpm at cruising speed.	Aracınızda Opticruise donanımı varsa en iyi yakıt ekonomisi ve sürüş konforunu elde etmek için otomatik modda sürmeniz gerekir.¶
479	The low engine speed means that you save fuel.	Lastikler •Lastiklerin basıncının yetersiz olması sürüşü yavaşlatır ve dolayısıyla da yakıt tüketimini artırır.
	It may require rather more gear changes.¶	Ayrıca, lastikler çabuk aşınır ve bu da sürüş sırasında güvenliği azaltır.

Figure 4 Raw alignment example from AbbyAligner

The Figure 4 shows the output that the alignment tool gives according to its own algorithms. So the segments need to be checked if there is any misalignment. When the segments are checked it is seen that some of the source and target segments do not correspond. For example, in 460th segment, it is seen that the source is empty but the target is “Tekerlek frenlerinin aşınmaması için retarder ve egzoz frenini kullanın”. But, the source needs to be “Use retarder and exhaust brake to save the wheel brakes”, the 467th segment. These two segments should be aligned by a human to provide correct parallel data. There is another example in the figure below that all of the segments are aligned by a human. Additionally, the empty lines need to be deleted before given to the engine.

No	English	Turkish
401	Please read these instructions carefully and pass on the information to other possible users of the fire extinguisher.	Lütfen bu talimatları dikkatle okuyunuz ve emniyetli kişilere de aktarınız.†
403	This should be done before it is placed/mounted in position, or before it is used for the first time.	Bu cihaz yerine yerleştirilmeden takılmadan ya da ilk kullanımdan önce gerçekleştirilmelidir.
404	The fire extinguisher needs to be turned every three months so that the powder in the fire extinguisher does not shrink.	Yangın söndürme cihazındaki tozun çekmesini için yangın söndürme cihazın üç ayda bir çevrilmesi gerekir.
405	This has a negative effect on the capacity of the fire extinguisher.†	Bu işlem, yangın söndürme cihazının kapasitesi üzerinde olumsuz bir etkisi vardır.†
406	•If possible, provide users with information about how to use the fire extinguisher and explain the risks of misuse.	Mümkünse, kullanıcılara yangın söndürme cihazının nasıl kullanılacağı hakkında bilgi veriniz ve yanlış kullanımın doğuracağı riskleri açıklayınız.
407	Keep children away from fire extinguishers.†	Küçük çocukları, yangın söndürme cihazlarından uzak tutunuz.†
408	The instruction text and illustration on the fire extinguisher's label describe the correct method of use.†	Yangın söndürme cihazının üstündeki etikette yazan talimatlar ve resimler, cihazın doğru kullanım yöntemini göstermektedir.†
409	†	†
410	Do not aim the jet of extinguishing medium directly at someone.	Cihazdan püskürtülen söndürme maddesini doğrudan kimsenin üzerine tutmayınız.
411	Keep a safety distance of at least 1 metre when extinguishing burning clothes on a person.†	Birinin üzerindeki yanan giysileri söndürürken en azından 1 metrelik güvenli bir mesafede kalınız.†
412	•	•
413	Only skilled service technicians trained in accordance with local legislation may open and carry out maintenance on the fire extinguisher.†	Ancak yerel mevzuata uygun şekilde eğitilmiş deneyimli servis teknisyenleri yangın söndürme cihazını açabilir ve bakımını yapabilirler.†
414	•	•
415	The fire extinguisher may be pressurised.	Yangın söndürücü basınçlı olabilir.
416	Do not expose it to force from outside and do not use force to open it.†	Açmak için dışardan güç uygulamayın ve güç kullanmayın.†
417	•	•
418	Allow a skilled service engineer to depressurise damaged or rusty fire extinguishers or fittings.	Hasar görmüş ya da paslanmış yangın söndürücülerini ya da bağlantı parçalarını basınçlı bir servis mühendisi tarafından boşaltılmasına sağlayınız.
419	This must be done before the fire extinguisher is taken to a waste disposal company.†	Bu işlem, yangın söndürücü bir atık bertaraf şirketine gönderilmeden önce gerçekleştirilmelidir.†
420	†	†
421	It is not permitted to make any modifications, e.g. welding or soldering, to the fire extinguisher.†	Yangın söndürücü üzerinde, örneğin kaynak ya da lehim gibi herhangi bir değişiklik yapmak yasaktır.†
422	•	•
423	Keep the fire extinguisher clean.	Yangın söndürme cihazını temiz tutunuz.
424	Do not use any aggressive cleaning agents, but just clean it with a damp cloth.†	Aşınmaya neden olabilecek temizlik maddeleri kullanmayın, sadece ıslak bir bezle siliniz.†
425	•	•
426	The fire extinguisher should only be used as a fire extinguisher container.†	Yangın söndürücü, sadece yangın söndürme cihazı kabı olarak kullanılmalıdır.†

Figure 5 Aligned segment examples from AbbyAligner

Once this (pre)alignment step is completed, GIZA++ is needed to align the segments for Moses. This (pre)alignment step is for cleaning and aligning the raw data.

4.4.2 GIZA++

GIZA++ is an alignment tool that creates bilingual files that are compatible with Moses. Once the GIZA++ is installed, the data should be in a format that GIZA++ can recognize. To ensure this, variables must be added to the beginning of each segment. An MS Excel file can provide adding variables accordingly. To do this, \$1, \$2 and more (as many as needed) variables should be added until the end of the segments, for example \$50,000 for the segment 50,000. When the first a few variables are added to the left column of the source segments, they can be drawn to the end of the MS Excel document, and MS Excel will add the rest of the variables by itself. The same procedure should be done for the target segments. After the completion of adding variable step for both source and target, the MS Excel document should be as figure below.

ID	Description	ID	Description
54059	Have replaced the wizards Manual temperature control, Troubleshooting the manual functions of the climate system and heater control for the climate system.	54059	Manuel sıcaklık kontrolü, klima sisteminin manuel işlevlerinde sorun giderme ve klima sistemi için ısıtıcı kontrolü işlevleri değişmiştir.
54060	The wizards for the calibration of the end position of the windows have been moved to the function view.	54060	Pencerelerin son konumunun kalibrasyonunun işlevi, işlev ekranına taşınmıştır.
54061	Version information for Scania Diagnos & Programmer 3, Update for version 2.32.	54061	Scania Diagnos & Programmer 3 için sürüm bilgisi.
54062	Clarification of the functions Own torque curve – Torque limiter 2 (mode 2) and Own torque curve – Torque limiter 3 (mode 3):	54062	2.32 sürümü için güncelleme
54063	Due to legal requirements, these cannot be set using SDP3 for Tier 3 industrial and marine engines.	54062	Özel tork eğrisi – Tork sınırlayıcı 2 (mod 2) ve Özel tork eğrisi – Tork sınırlayıcı 3 (mod 3) işlevlerine yönelik açıklama:
54064	The Individual adaptation of gearbox to engine wizard can now be run on vehicles with Opticruise but without an electrical clutch actuator.	54063	Yasal gereksinimler nedeniyle bunlar, Tier 3 endüstriyel ve denizcilik motorlarına yönelik SDP3 kullanılarak ayarlanamaz.
54065	BMS, EBS5	54064	Şanzımanın motora ayrı olarak adaptasyonu şiribazı, artık Opticruise olan ancak elektrikli kavrama kumandası olmayan araçlarda çalıştırılabilir.
54066	Improvements to the ESP function for 3-axle trucks.	54065	BMS, EBS5
54067	Preliminary support to ensure an adaptation of the sensitivity of the ESP function on trucks with a particularly low centre of gravity.	54066	3 dingilli kamyonlar için ESP işlevine yönelik iyileştirmeler
54069	A deviation in the Reset and calibration of the EGR circuit wizard has been corrected in order to carry out calibration of the EGR valve.	54067	BMS, EBS5 ve EBS7
54070	The Individual adaptation of gearbox to engine wizard now automatically checks that the clutch pedal is not depressed.	54068	Özellikle düşük ağırlık merkezi olan kamyonlarda ESP işlevinin hassasiyetinin adaptasyonunu sağlamaya yönelik başlangıç desteği.
54071	Version information for Scania Diagnos & Programmer 3, Update 2 for version 2.32.	54069	EGR devresinin sıfırlanması ve kalibrasyonu şiribazındaki bir sapma, EGR valfi kalibrasyonunu gerçekleştirmek için düzeltilmiştir.
54072	This version is connected to the following FQ case:	54070	Şanzımanın motora ayrı olarak adaptasyonu şiribazı debriyaj pedalına basılıp basılmadığını artık otomatik olarak kontrol edecektir.
54073	When trying to update EMS S8 to the latest software version (60.51.10), spare part programming does not work.	54071	Scania Diagnos & Programmer 3 için sürüm bilgisi.
54074	This and the fact that certain error code texts are not displayed have been rectified.	54072	2.32 sürümü için 2. güncelleme
54075		54073	Bu sürüm, aşağıdaki FQ durumu ile ilişkilidir:
54076		54074	EMS S8'in son yazılım sürümüne (60.51.10) güncellemeye çalışırken yedek parça programlaması çalışmaz.
54077			
54078			

Figure 6 Variable adding example

By the way, undefined characters should be spotted and cleaned to achieve higher quality MT output. An example to the undefined characters is shown in the figure below.

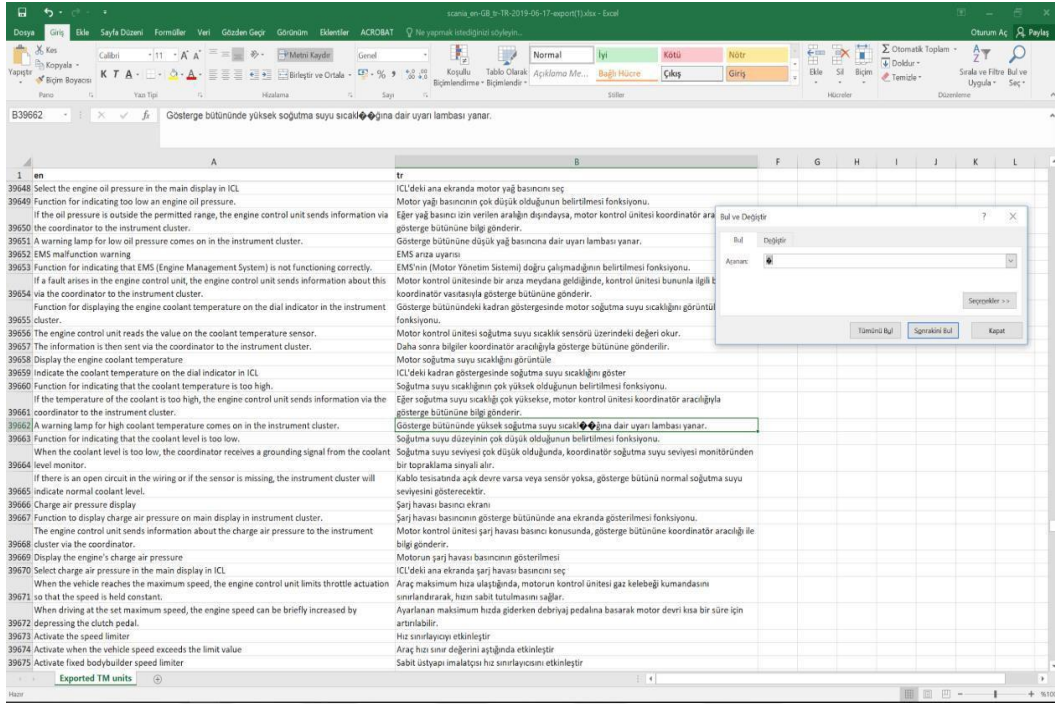


Figure 7 Undefined character example

These characters should be detected and changed accordingly. Ctrl + find option will help to see all of these characters. After this step is completed, the document needs to be made prepared to be recognized by GIZA++. Notepad++ can be used to make necessary formatting operations on the file. The segments should be justified left. To be able to ensure this, once the file is opened via Notepad++, the tab needs to be deleted. First, ctrl + find should be opened and then, regular expression option needs to be selected. In the next step, the necessary code to delete the tab needs to be written and replaced with \$1. $^{\wedge}([\backslasht]*?)\t$ is the necessary code to delete the tab. Afterwards, the source and target text need to be saved as plain text separately. I named the source file as EN and the target as TR. If any other file name is used, the codes to run GIZA++ should be changed accordingly. Now, the data is ready to be recognized by GIZA++.

4.4.3 Running the GIZA++

As mentioned before, to get aligned corpus, GIZA++ is needed. There are a few steps to run the GIZA++ that first of all, the terminal is opened and user needs to navigate to the .\giza-pp file and then GIZA++-v2 file. To do this, “cd smt”, “cd .\giza-pp”, “cd

GIZA++-v2” codes are written. When a cd code is written before a folder name, it means that we are working on that respective folder. After the navigation to the GIZA++-v2 folder via terminal, “./plain2snt.out [source_language_corpus] [target_language_corpus]” code should be written. Since the source language corpus is EN and the target language corpus is TR, this code should be written as “./plain2snt.out EN TR” (without brackets). This code will generate vocabulary and sentence files in GIZA++-v2 folder named after EN.vcb, TR.vcb and also EN_TR.snt, TR_EN.snt. Now user needs to navigate to .\mkcls-v2\ folder. In order to achieve this user needs to get back to the first folder. Writing just “cd” on the terminal will navigate user to the previous folder. Then “cd .\mkcls-v2\” needs to be written. Then “./mkcls -pEN -VEN.vcb.classes” and “./mkcls -pTR -VTR.vcb.classes” codes need to be written. This will create EN.vcb.classes, EN.vcb.classes.casts, TR.vcb.classes and TR.vcb.classes.casts files in the respective folder. Then we get back to GIZA++v2 folder via terminal as described above. Finally, “./GIZA++ -S EN.vcb -T TR.vcb -C EN_TR.snt” code needs to be written. This code will finally create an actual.ti.ini file in the directory and the screen will be as shown in the below figure (Source: https://okapiframework.org/wiki/index.php/GIZA%2B%2B_Installation_and_Running_Tutorial, (accessed, July 04, 2019))⁸.

```

alper@alper-X555LNB: ~/GIZA++/GIZA++-v2
40000
50000
Reading more sentence pairs into memory ...
(WARNING:(a)truncated sentence 51772)(WARNING:(a)truncated sentence 52028)Reading more sentence pairs into memory ...
#centers(pre/hillclimb/real): 1 1 1 #al: 166.261 #alsophisticatedcountcollection: 6.38579 #hsteps: 1.33721
#peggingimprovements: 0
D4 table contains 406 parameters.
A/D table contains 66792 parameters.
A/D table contains 52714 parameters.
Ntable contains 732720 parameter.
p0_count is 416621 and p1 is 21991.6; p0 is 0.949861 p1: 0.0501391
Model4: TRAIN CROSS-ENTROPY 4.79913 PERPLEXITY 27.8408
Model4: (10) TRAIN VITERBI CROSS-ENTROPY 4.84298 PERPLEXITY 28.7001
Dumping alignment table (a) to file:119-07-04.140642.alper.a3.final
Dumping distortion table (d) to file:119-07-04.140642.alper.d3.final
Dumping nTable to: 119-07-04.140642.alper.n3.final

Model4 Viterbi Iteration : 10 took: 8 seconds
H3333344444 Training Finished at: Thu Jul 4 14:08:05 2019

Entire Viterbi H3333344444 Training took: 49 seconds
=====
writing Final tables to Disk
Dumping the t table inverse to file: 119-07-04.140642.alper.tl.final
Dumping the t table inverse to file: 119-07-04.140642.alper.actual.ti.final
Writing PERPLEXITY report to: 119-07-04.140642.alper.perp
Writing source vocabulary list to : 119-07-04.140642.alper.trn.src.vcb
Writing source vocabulary list to : 119-07-04.140642.alper.trn.trg.vcb
Writing source vocabulary list to : 119-07-04.140642.alper.tst.src.vcb
Writing source vocabulary list to : 119-07-04.140642.alper.tst.trg.vcb
writing decoder configuration file to 119-07-04.140642.alper.Decoder.config

Entire Training took: 83 seconds
Program Finished at: Thu Jul 4 14:08:05 2019
=====
alper@alper-X555LNB:~/GIZA++/GIZA++-v2$

```

Figure 8 An example from Linux terminal showing the file creation is successful

⁸ The codes are adapted by me accordingly.

To run GIZA++ enter:

- `cd smt`
- `cd .\giza-pp`
- `cd GIZA++-v2`
- `./plain2snt.out EN TR`
- `cd .\mkcls-v2\`
- `mkcls -pEN -VEN.vcb.classes`
- `./mkcls -pTR -VTR.vcb.classes`
- `./GIZA++ -S EN.vcb -T TR.vcb -C EN_TR.snt`

Now a corpus needs to be created in the working directory. To do this, first of all a corpus folder should be created to save respective files in it. To create a corpus folder, “cd”, “mkdir corpus” are the necessary codes. Then, “cd corpus” code needs to be typed to make it possible to save the files to save in it. If the data to be used is stored on the web, “wget http://(the name of the website or wherever the files are stored)” should be typed. If the files are stored in the computer memory then, the files need to be copied to the corpus file. Alternatively, there are sample files which can be downloaded from <http://www.statmt.org/moses/download/sample-models.tgz> website. These are the prepared files for engine training. Once these sample files are extracted into the corpus folder, the prepared source text can be copied into the already prepared news-commentary-v8.fr-en.fr file and the target text can be copied into the already prepared news-commentary-v8.fr-en.en file. Then the name of the files can be changed as alper.tr-en.tr and alper.tr-en.en. In the next steps, tokenisation⁹, truecasing¹⁰ and cleaning¹¹ need to be performed before entering the training commands. “~/smt/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en < ~/corpus/alper.tr- en.tr

⁹ Tokenisation means that spaces have to be inserted between (e.g.) words and punctuation. (Koehn 2019, 36).

¹⁰ Truecasing means that the initial words in each sentence are converted to their most probable casing. This helps reduce data sparsity (Koehn 2019, 36).

¹¹ Cleaning is for removing long sentences and empty sentences as they can cause problems with the training pipeline, and obviously mis-aligned sentences are removed (Koehn 2019,36).

`> ~/corpus/alper.tr-en.tok.tr"` are the codes that should be typed together to perform tokenisation. Here, `alper.tr-en.en` is the name of the source corpus file that is going to be trained. This code will create a file in corpus folder named `alper.tr-en.tok.tr`. Then, `"~/smt/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en < ~/corpus/alper.tr-en.en > ~/corpus/alper.tr-en.tok.en"` codes need to be typed. This code will create an `alper.tr-en.tok.en` file in the corpus folder. In the second step, to perform truecasing, the necessary codes are `"~/smt/mosesdecoder/scripts/recaser/train-truecaser.perl --model ~/corpus/truecase-model.en -corpus ~/corpus/alper.tr-en.tok.en"` and then, `"~/smt/mosesdecoder/scripts/recaser/train-truecaser.perl --model ~/corpus/truecase-model.tr -corpus ~/corpus/alper.tr-en.tok.tr"`. These codes will create `alper.tr-en.true.tr` and `alper.tr-en.true.en` files in corpus folder. In the next step `"~/mosesdecoder/scripts/recaser/truecase.perl \ --model ~/corpus/truecase-model.en \ < ~/corpus/alper.tr-en.tok.en \ > ~/corpus/alper.tr-en.true.en"` and `"~/mosesdecoder/scripts/recaser/truecase.perl \ --model ~/corpus/truecase-model.tr \ < ~/corpus/alper.tr-en.tok.tr \ > ~/corpus/alper/tr-en.true.tr"` codes are needed. These codes will be typed one by one and will create two files in the corpus file named `truecase-model.en` and `truecase-model.tr`. Lastly, to ensure that Moses to perform better the length of the sentences should be limited. To limit the sentences, cleaning should be performed. `"~/mosesdecoder/scripts/training/clean-corpus-n.perl \ ~/corpus/alper.tr-en.true fr en \ ~/corpus/alper/tr-en.clean 1 80"` is the code to clean the corpus. Now our corpus is ready to use but we need a language model. Language model is necessary to obtain fluent output so it is built with the target language (in this study, it is Turkish language). To create a language model `"wget -O - https://kheafeld.com/code/kenlm.tar.gz |tar xz"`, `"mkdir kenlm/build"`, `"cd kenlm/build"`, `"cmake.."`, `"make -j2"` are the necessary codes. <https://kheafeld.com/code/kenlm/>, (accessed July 9, 2019). The files downloaded from the website should be moved to `mosesdecoder/bin` folder. Now type `cd ..` and then, `"mkdir ~/lm"`, `"cd ~/lm"`. Now we need an `.arpa` file, `"~/smt/mosesdecoder/bin/lmplz -o 3 <~/corpus/alper.tr-en.true.fr > alper.tr-en.arpa.fr"` is the necessary code. Then, to binarise the `.arpa.tr` file the necessary code is `"~/smt/mosesdecoder/bin/build_binary alper.tr-en.arpa.en alper.tr-en.blm.en"` and `"~/smt/mosesdecoder/bin/build_binary alper.tr-en.arpa.fr alper.tr-en.blm.fr"` are the necessary codes. In order to check the

language model, “echo "Çıkış sinyali verilen değerden yüksek" | ~/kenlm/build/bin/query alper.tr-en.blm” code can be entered, or any sentences related to the domain (the corpus that is created to be used for MT engine) can be entered. Once this code is typed, the below screen will be appeared in the terminal.

```
.tr
This binary file contains probing hash tables.
Çıkış=2828 1 -5.3950863 sinyali=118 1 -3.2100174 verilen=51 1 -3.8236697d
eğerden=3716 1 -4.125833 yüksek=53 2 -0.94842553 </s>=2 2 -1.8149598 T
total: -19.317991 OOV: 0
Perplexity including OOVs: 1658.3080518892118
Perplexity excluding OOVs: 1658.3080518892118
OOVs: 0
Tokens: 6
Name:query VmPeak:38476 kB VmRSS:4752 kB RSSMax:14744 kB user:0 sys:0.00
28 CPU:0.00284502 real:0.00105339
```

Figure 9 An example from Linux terminal showing the command is working

To create a corpus folder in the working directory and save the training data in it, enter:

- `mkdir corpus`
- `cd corpus`

To tokenize the files, enter:

- `~/smt/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en < ~/corpus/training/alper.tr-en.en > ~/corpus/alper.tr-en.tok.en`
- `~/smt/mosesdecoder/scripts/tokenizer/tokenizer.perl -l fr < ~/corpus/training/alper.tr-en.tr > ~/corpus/alper.tr-en.tok.tr`

To truecase the files, enter:

- `~/smt/mosesdecoder/scripts/recaser/train-truecaser.perl --model ~/corpus/truecase-model.en -- corpus ~/corpus/alper.tr-en.tok.en`
- `~/smt/mosesdecoder/scripts/recaser/train-truecaser.perl --model ~/corpus/truecase-model.en -- corpus ~/corpus/alper.tr-en.tok.tr`
- `~/smt/mosesdecoder/scripts/recaser/train-truecaser.perl --model ~/corpus/truecase-model.fr -- corpus ~/corpus/alper.tr-en.tok.tr`
- `~/smt/mosesdecoder/scripts/recaser/truecase.perl --model ~/corpus/truecase-model.en\< ~/corpus/alper.tr-en.tok.en \> ~/corpus/alper.tr-en.true.en`
- `~/smt/mosesdecoder/scripts/recaser/truecase.perl --model ~/corpus/truecase-model.fr\< ~/corpus/alper.tr-en.tok.tr \> ~/corpus/alper.tr-en.true.tr`

- **To clean the data by limiting the sentence length to 80, enter:**
`~/mosesdecoder/scripts/training/clean-corpus-n.perl ~/corpus/alper.tr-en.true fr
en ~/corpus/alper.tr-en.clean 1 80`
- **To create language model, enter:**
- `wget -O - https://kheafield.com/code/kenlm.tar.gz |tar xz`
- `mkdir kenlm/build`
- `cd kenlm/build`
- `cmake ..`
- `make -j2`
- `cd ..`
- `mkdir lm`
- `cd lm`
- `~/smt/mosesdecoder/bin/lmplz -o 3 <~/corpus/alper.tr-en.true.fr > alper.tr
en.arpa.fr ~/smt/mosesdecoder/bin/build_binary alper.tr-en.arpa.en alper.tr-
en.blm.en ~/smt/mosesdecoder/bin/build_binary alper.tr-en.arpa.fr alper.tr-
en.blm.fr`

4.4.4 Training the Engine

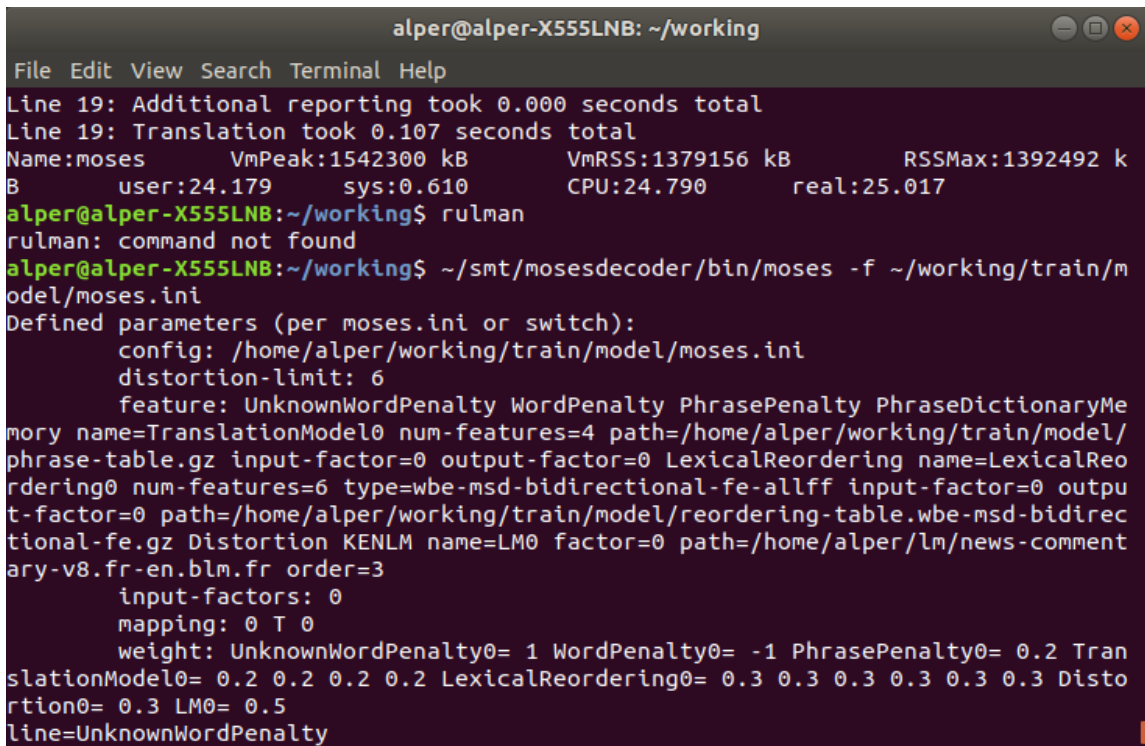
Now it is time to train the engine. First of all we need to create a working folder by typing “`mkdir ~/working`” and then, to save the working files in it, “`cd ~/working`”. In order to start the training, “`nohup nice ~/smt/mosesdecoder/scripts/training/train-model.perl -root-dir train -corpus ~/corpus/alper.tr-en.clean -e en -f fr -alignment grow-diag-final-and -reordering msd-bidirectional-fe -lm 0:3:$HOME/lm/alper.tr-en.blm.fr:8 -external-bin-dir ~/smt/mosesdecoder/tools >& training.out &`” code needs to be typed. This code will create a `moses.ini` file in `working/train/model` folder. Now the system can be checked. With the “`~/smt/mosesdecoder/bin/moses -f ~/working/train/model/moses.ini`” code the system can be initiated. Once this code is typed, a few seconds later the text to be translated can be written to the terminal. Without writing this code to initiate the system, it is not possible to obtain a result. The below figures show us that without writing this code how the screen will be and after writing this code how the system translates. To sum up, the codes below are used respectively:

To train the engine, enter:

- `mkdir ~/working`
- `cd ~/working`
- `nohup nice ~/smt/mosesdecoder/scripts/training/train-model.perl -root-dir train -corpus ~/corpus/alper.tr-en.clean -e en -f fr -alignment grow-diag-final-and -reordering msd-bidirectional-fe -lm 0:3:$HOME/lm/alper.tr-en.blm.fr:8 -external-bin-dir ~/smt/mosesdecoder/tools >& training.out &`

To initiate the engine, enter:

- `~/smt/mosesdecoder/bin/moses -f ~/working/train/model/moses.ini`



```
alper@alper-X555LNB: ~/working
File Edit View Search Terminal Help
Line 19: Additional reporting took 0.000 seconds total
Line 19: Translation took 0.107 seconds total
Name:moses      VmPeak:1542300 kB      VmRSS:1379156 kB      RSSMax:1392492 kB
B      user:24.179      sys:0.610      CPU:24.790      real:25.017
alper@alper-X555LNB:~/working$ rulman
rulman: command not found
alper@alper-X555LNB:~/working$ ~/smt/mosesdecoder/bin/moses -f ~/working/train/m
odel/moses.ini
Defined parameters (per moses.ini or switch):
  config: /home/alper/working/train/model/moses.ini
  distortion-limit: 6
  feature: UnknownWordPenalty WordPenalty PhrasePenalty PhraseDictionaryMemory
name=TranslationModel0 num-features=4 path=/home/alper/working/train/model/phrase-table.gz
input-factor=0 output-factor=0 LexicalReordering name=LexicalReordering0 num-features=6
type=wbe-msd-bidirectional-fe-allff input-factor=0 output-factor=0 path=/home/alper/working/train/model/reordering-table.wbe-msd-bidirectional-fe.gz
Distortion KENLM name=LM0 factor=0 path=/home/alper/lm/news-commentary-v8.fr-en.blm.fr order=3
  input-factors: 0
  mapping: 0 T 0
  weight: UnknownWordPenalty0= 1 WordPenalty0= -1 PhrasePenalty0= 0.2 TranslationModel0= 0.2 0.2 0.2 0.2 LexicalReordering0= 0.3 0.3 0.3 0.3 0.3 0.3 Distortion0= 0.3 LM0= 0.5
line=UnknownWordPenalty
```

Figure 10 An example from Linux terminal showing how to initiate the engine

```
alper@alper-X555LNB: ~/working
File Edit View Search Terminal Help
Line 10: Translation took 0.000 seconds total
rulman
Translating: rulman
Line 11: Initialize search took 0.000 seconds total
Line 11: Collecting options took 0.000 seconds at moses/Manager.cpp Line 141
Line 11: Search took 0.000 seconds
bearing
BEST TRANSLATION: bearing [1] [total=-10.349] core=(0.000,-1.000,1.000,-3.135,-
4.533,0.000,-3.611,-0.511,0.000,0.000,0.000,0.000,0.000,0.000,-18.279)
Line 11: Decision rule took 0.000 seconds total
Line 11: Additional reporting took 0.000 seconds total
Line 11: Translation took 0.000 seconds total
motor
Translating: motor
Line 12: Initialize search took 0.000 seconds total
Line 12: Collecting options took 0.000 seconds at moses/Manager.cpp Line 141
Line 12: Search took 0.000 seconds
engine
BEST TRANSLATION: engine [1] [total=-9.204] core=(0.000,-1.000,1.000,-0.903,-1.
194,-2.007,-1.136,-0.722,0.000,0.000,0.000,0.000,0.000,0.000,-18.279)
Line 12: Decision rule took 0.000 seconds total
Line 12: Additional reporting took 0.000 seconds total
Line 12: Translation took 0.000 seconds total
```

Figure 11 An example from Linux terminal showing the best translation result

Since I have no commercial concern I do not expect to get higher quality results. Here, my aim is to show how the MT engine works.

5 EVALUATION

This chapter will give information on the evaluations of the MT outputs. The evaluation has been conducted via BLEU metrics, which will be briefed in the following section.

BLEU is an automated evaluation system that evaluates the MT output instead of human. It evaluates the MT output according to the reference human translated text. “The central idea behind the BLEU score is that the closer a machine translation is to a professional translator, the better it is” (Papineni and et al., 2002, 311). Since the evaluation of MT is long term task and it is expensive Papieni and et al. developed the BLEU to contribute MT system developers. The higher scores mean the higher quality of MT output compared to reference translation. So, the BLEU score just shows the correlation between human translation and machine translation. Below I give examples showing the human translation of a text, MT output and BLEU scores. The first sentence is source text, the second one human translated sentence, the third one is machine translated sentence and the last one is the machine translated sentence BLEU score.

Examples:

ST	HTS	MTS	MTSBS
safety precautions to detect gas leakage in a gas-fuelled vehicle is a requirement from the authorities that is stated in ece r110.	Güvenlik önlemleri Gaz yakıtlı araçtaki gaz kaçağını tespit etmek ECE R110'te belirtilen yetkililerin getirdiği bir gerekliliktir.	on güvenlik relates yakıtlı scania kaçağını to the detected ece r110'te specified yetkililerin getirdiği a gerekliliktir.	2.377053
the purpose of the requirement is to prevent safety-critical situations as far as possible.	Gerekliliğin amacı güvenliği tehlikeye düşürecek durumları mümkün olduğunca önlemektir.	gerekliliğin defect güvenliği tehlikeye düşürecek durumları possible önlemektir.	1.304298
gas leakage is	Gaz kaçağı gaz yakıt	on leaks accelerator the	2.487097

detected by using 2 characteristics in the gas fuel system: gas pressure and mass flow rate.	sistemdeki 2 özellik kullanılarak tespit edilir: gaz basıncı ve debi.	2 characteristic - optimum edilir: accelerator and debi.	
if a leakage occurs, the pressure will drop and the system will warn for a leakage	Kaçak meydana gelirse basınç düşer ve sistem kaçak uyarısı verir.	kaçak occurs pressure % 50 verir. system .	2.415965
visit a scania workshop to have the fuel system repaired if the symbol is displayed.	Sembol görüntülenirse yakıt sisteminin onarılması için bir Scania servisini ziyaret edin.	.sembol previously the onarılması, scania times ziyaret edin.	2.910968
warning for gas leakage can also be displayed due to: •the manual cock has been opened.	Gaz kaçağı uyarısı şu nedenlerle görüntülenebilir: •manuel musluk açılmıştır.	on leaks warning default nedenlerle görüntülenebilir: •manuel musluk açılmıştır.	2.190408
manual cocks are closed at the same time as the engine consumes gas.	manuel musluklar tam motor gaz tüketirken kapanmıştır.	manual musluklar full engine accelerator tüketirken kapanmıştır.	3.314288
trying to start the engine at the same time as manual cocks are closed.	manuel musluklar kapalı olduğunda motoru çalıştırmaya çalışılmıştır.	manual musluklar if engine çalışılmıştır.	2.099844
filling fuel while the	motor çalışırken yakıt	engine fuel doldurma.	

engine is running.	doldurma.		11.909345
fitting external components that are not compatible with scania's systems.	Scania'nın sistemleriyle uyumlu olmayan harici bileşenler takma.	scania'nın systems legal) takma.)	2.083626
if the warning is displayed after one of the above activities has been carried out, turn the starter key to the lock position and wait for more than 20 seconds so that the vehicle is switched off completely and then try to start it again.	Yukarıdaki eylemlerden biri gerçekleştirildikten sonra uyarı görüntülenirse, marş anahtarını kilit konumuna çevirin ve araç tamamen kapalı konuma gelene dek 20 saniyeden fazla bekleyip yeniden çalıştırmayı deneyin.	yukarıdaki eylemlerden of the warning görüntülenirse, 1020 the of the off gelene until 20 on bekleyip , çalıştırmayı deneyin.	1.676151
if the fuel system is fault-free, the warning should no longer be shown.	Yakıt sisteminde arıza yoksa uyarı artık gösterilmeyecektir.	fuel system fault warning gösterilmeyecektir. :	5.594423
safety valves for pressure drop in the fuel system if the fuel system rapidly loses pressure, for example when a fuel pipe breaks, the safety valves operate.	Yakıt sistemindeki basınç düşüşü için emniyet valfları Yakıt sistemi hızlı bir şekilde basınç kaybederse, örneğin bir yakıt borusu kırılırsa, emniyet valfi çalışır.	fuel system drop , relief valves if a pressure kaybederse, a fuel kırılırsa, the çalışır. .	4.259881
safety precautions switches off the flow of fuel from	Güvenlik önlemleri gaz tüplerinden yakıt akışını keser.	güvenlik relates tüplerinden fuel do keser.	2.985966

the gas bottles.			
tow the vehicle to a scania workshop to have the fuel system repaired if this occurs.	Bunun meydana gelmesi durumunda, yakıt sisteminin onarılması için aracı bir Scania servisine çekin.	bunun of the if durumunda, onarılması the scania servisine çekin.	3.256759
blown fuse b350919 symbol which indicates that the fuse for the gas tank solenoid valve has blown.	Atmış sigorta b350919 Gaz deposu solenoid valfi sigortasının attığını gösteren sembol.	atmış fuse b350919 on tank start to the attığını sembol.	4.955971
renew the fuse if the symbol is displayed in icl.	ICL'de sembol görüntüleniyorsa sigortayı yenileyin.	icl'de symbol görüntüleniyorsa fuse yenileyin.	4.67329
vehicle gas and safety action in the event of fire warning!	Araç gazı ve güvenlik Yangın durumunda yapılacaklar UYARI!	vehicle gas and safety yangın yapılacaklar uyarı! 1	23.761019
in the event of fire, switch off the engine and immediately notify the fire brigade that the vehicle contains vehicle gas and what type of gas it is.	Yangın durumunda, motoru kapatın ve aracın gaz içerdiğini ve bunun hangi tip gaz olduğunu yangın ekibine hemen bildirin.	yangın durumunda, start ; accelerator for and of the = d fire ekibine almost bildirin.	1.562232
each gas tank has safety valves that open when the pressure in the tank gets too high.	Her bir gaz deposunun depodaki basınç çok yüksek seviyeye eriştiğinde açılan emniyet valfleri vardır.	limitation accelerator tank tank is eriştiğinde radio to the valve vardır.	2.858684

As can be seen MT output, and BLEU scores differs for each sentence depending on the data given to the engine and correlation of the reference translation (human translation) to the corpus. When the higher amount of data is given to the engine, the higher quality of results will be obtained. In this BLEU score metrics, the score range is between 0-1 and this result score is presented as multiplied by 100. The perfect match results are closer to 1 whereas the perfect mismatch results are closer to 0. The higher BLEU scores show that they have higher quality compared to the reference text. BLEU score uses n-grams to for both candidate translation (MT output) and reference text (human translation). N-gram is a sequence of N words (Kumar, 2017, Source: <https://blog.xrds.acm.org/2017/10/introduction-n-grams-need/>), (accessed, June 16, 2019)). N words indicates the number of words, for example, fuel tank (is a 2-gram), tapered roller bearing (is a 3-gram), filling the fuel tank (4-gram). Depending on the frequency of these words in the respective corpus, the score changes. Papieni and et al. states in their paper titled as “BLEU: a Method for Automatic Evaluation of Machine Translation” that:

The primary programming task for a BLEU implementor is to compare n-grams of the candidate with the n-grams of the reference translation and count the number of matches. These matches are position-independent. The more the matches, the better the candidate translation is. (Papieni, et al. 2002, 2)

This means that the BLEU score is related to count the number of matches, not the fluency of the MT output. The metric that determines the BLEU score changes depending on corpus which is used to create SMT engine. In the MT engine I created, the BLEU score is obtained by using unigram (1-gram) model. Additionally, the other n-gram models can be used. The words can be chunked together to form single entities, for example, fuel tank can be chunked together as one word. Normally it is 2-gram model but it can be used as single entity. This may help to predict what the next word will be (cf. Brownlee, 2017). To conclude, the different BLEU scores which can be seen in the examples are related to the corpus which I used. When we add new parallel data to the corpus (related to domain), we could get higher quality results which are closer to the reference text and this means we could get higher BLEU scores. To be able to

measure the BLEU score we can check the words in the MT output and assign 1 for the ones that are seen in the reference translation, and 0 for the ones that are not seen in the reference translation. Then the number of words that are seen in the reference translation can be divided by the total number of words in the output sentence. This is how unigram precision works. But this changes depending on different parameters such as the length of the sentences, the number of the characters, and so on. And also, BLEU does not consider the meaning or sentence structure (cf. Tatman, 2017). For example:

Araç gazı ve güvenlik Yangın durumunda yapılacaklar UYARI! (HT)

vehicle gas and safety yangın yapılacaklar uyari! 1 (MT) (23.761019)

Gerekliliğin amacı güvenliği tehlikeye düşürecek durumları mümkün olduğunca önlemektir. (HT)

gerekliliğin defect güvenliği tehlikeye düşürecek durumları possible önlemektir. (MT)
(1.304298)

In the first example, there are 3 words in the MT output that are seen in reference translation (HT). The total number of words in the MT output sentence is 8. When the number of words that are seen in the reference translation is divided by the total number of words in the output sentence, the score is not 23.761019. The same is applicable for the second example. It should be noted that BLEU is just precision not a result showing the quality or usefulness of the translation results. It just compares the reference text and MT output and gives an average score to help MT engine producers to check the progress of training the MT engine.

6 CONCLUSION

The purpose of this thesis was to show the basic technical steps of creating an SMT engine, and creating corpus and making this corpus ready to be used for statistical machine translation engine. The reader of this thesis can find useful information to create her/his own SMT engine, and may make profit for the company s/he works for or herself/himself. For example s/he can use his own engine and do translation in a shorter time. People who spend time on preparing corpus in big volumes in a specific domain may sell their engines or they can take part in an engine creation process. And these people can benefit from this study. In order to create SMT engine, parallel corpus needs to be used. In order to create parallel corpus, big volumes of data needs to be aligned. Of course there are some ethical considerations on data usage. Translators, editors, project managers or anyone who take part in a translation process and work for a language service provider (LSP), multilingual language vendor (MLV) or an individual client need permission of the owner of data before using it for their engines, although translation is done by themselves.

People who are not familiar with Linux operating system, and Moses, can benefit from this thesis to create their own statistical based machine translation engines. There are other systems much easier to use but the basis of most of these systems is Moses and it provides users to make more individual parameter calibrations. In this thesis, it is not claimed to show all kinds of parameter. Most of the resources on Moses suppose that the user has basic information on coding and using Linux. For this reason, these resources give the codes and expect the user to adapt them accordingly. In this thesis, it is aimed to explain the codes in a simple way.

In order to create the SMT engine I got training on basic coding and using Linux operating system. Then I adapted the codes accordingly and made the MT engine ready for translation. As a result, I have an engine that can translate automotive texts but due to the volume of the data used, the results are not high-quality. In this study, it is tried to explain what a code means and how it can be adapted by an individual user. In the first chapter I outlined the CAT tools and machine translation. In the following chapter, I summarized the effects of technological developments on translators and their transformation. In chapter three, I mentioned the different machine translation systems,

and corpus. The fourth chapter includes the technical steps of creating SMT engine. In the fifth chapter, I briefly gave information on BLEU and showed the examples obtained from the engine which I have created. I hope this basic study will function as an initiative for translation students.



REFERENCES

- Bahdanau, Dzmitry, KyungHyun Cho and Yoshua Bengio. 2015. "Neural Machine Translation by Jointly Learning to Align and Translate." In *ICLR Conference*, San Diego, the USA, 07-09. arXiv:1409.0473.
- Blommaert, Eef. 2012. "YouTube." *YouTube* (blog), Accessed January 8, 2019. <https://www.youtube.com/watch?v=aIxI8mGbNuY>.
- "Boost C Libraries." (n.d.). "Welcome to Boost.org." Accessed March 1, 2019. <https://www.boost.org/>.
- Brownlee, Jason. 2017. "A Gentle Introduction to Calculating the BLEU Score for Text in Python." Section on Deep Language Processing. Accessed August 20, 2019. <https://machinelearningmastery.com/calculate-bleu-score-for-text-python/>.
- Clark, Stephen. (n.d.). "ACS Statistical Machine Translation Lecture 2: Introduction to SMT Models." Lecture, University of Cambridge, the UK.
- Crandall, Richard E. 2017. "Industry 1.0 to 4.0: the Evolution of Smart Factories." ASCM. Accessed April 13, 2019. <http://www.apics.org/apics-for-individuals/apics-magazine-home/magazine-detail-page/2017/09/20/upgrading-smart-manufacturing-with-industry-4>.
- Esselink, Bert. 2000. *A Practical Guide to Localization*. Amsterdam/Philadelphia: John Benjamins.
- Federico, Marcello, Nicola Bertoldi and Mauro Cettolo. 2008. *Language Modeling Toolkit Version 5.20.00 User Manual*. Ebook. Trento: FBK-irst. Accessed April 19, 2019. http://hermes.fbk.eu/people/bertoldi/teaching/lab_2010-2011/img/irstlm-manual.pdf.
- "GIZA Installation and Running Tutorial." 2016. Okapi Framework. Accessed July 12, 2019. https://okapiframework.org/wiki/index.php/GIZA++_Installation_and_Running_Tutorial.
- Koehn, Phillipp. 2019. *Statistical Machine Translation System User Manual and Code Guide*. Edinburgh: Edinburgh University Press.
- Kumar, Prachi. 2017. *Crossroads: The ACM Magazine for Students* (blog), Accessed June 21, 2019. <https://blog.xrds.acm.org/2017/10/introduction-n-grams-need/>.

- O'Dowd, Tony. 2013. "The History of Machine Translation Pt. 1." *KantanMT* (blog), Accessed May 12, 2019. <https://kantanmtblog.com/2013/07/12/the-history-of-mt-pt-1/>.
- O'Dowd, Tony. 2014. "RBMT vs SMT." *KantanMT* (blog), Accessed March 16, 2019. <https://kantanmtblog.com/2014/02/13/rbmt-vs-smt/>.
- Othman, Achraf. 2017. "How to Install Moses (Statistical Machine Translation) on Ubuntu?" Dr. Achraf Othman, Accessed June 27, 2019. <http://achrafothman.net/site/how-to-install-moses-statistical-machine-translation-in-ubuntu/>.
- Öner, Işın. 2006. "Yerelleştirme'nin Tanımı." *Varlık* 1185: 33-35.
- Öner Bulut, Senem. 2019. "Future Professional Profile and Agency of the Human Translator: A Survey on Human-Machine Tension in the Context of Technologization of Translation." In *Research in Translation Studies* (ed. Seda Taş), 93-122. İstanbul: Hiperyayın.
- Papineni, Kishore, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. "BLEU: a Method for Automatic Evaluation of Machine Translation." *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 311-318. <https://www.aclweb.org/anthology/P02-1040.pdf>.
- Pym, Anthony. 2003. "Redefining Translation Competence in an Electronic Age. In Defence of a Minimalist Approach." *Meta* 48 (4): 481- 497. <https://doi.org/10.7202/008533ar>.
- Rathod, Sarita G. and Shanta Sondur. 2012. "Machine Translation of Natural Language Using Different Approaches: ETSTS, English to Sanskrit Translator and Synthesizer." *International Journal of Emerging Technology and Advanced Engineering* 12 (2): 379-383.
- Schumacher, Phyllis and Janet Morahan-Martin. 2001. "Gender, Internet and Computer Attitudes and Experiences." *Computers in Human Behavior* 17 (1): 95-110. [http://dx.doi.org/10.1016/S0747-5632\(00\)00032-7](http://dx.doi.org/10.1016/S0747-5632(00)00032-7).
- Tatman, Rachael. 2019. "Evaluating Text Output in NLP: BLEU at Your Own Risk." *Towards Data Science*. Accessed May 27, 2019. <https://towardsdatascience.com/evaluating-text-output-in-nlp-bleu-at-your-own-risk-e8609665a213>.
- "The Complete Open-Source and Business Software Platform." *SourceForge*. Accessed April 22, 2019. <https://sourceforge.net/>.

Tian, Liang, Fai Wong and Sam Chao. 2011. “Word Alignment Using GIZA on Windows.” *Machine Translation Summit 13*: 369-372. Accessed May 14, 2019. <http://www.mt-archive.info/MTS-2011-Tian.pdf>.

Van der Meer, Jaap. 2015. TAUS. Accessed February 22, 2019. <https://blog.taus.net/the-game-changers-of-2016>.

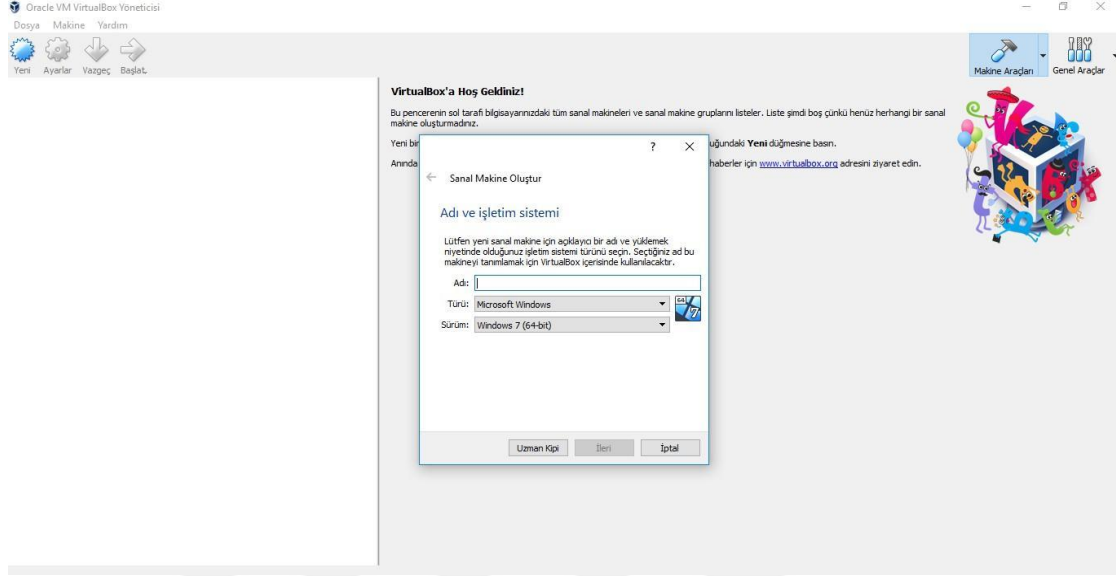
“What Is Technical Translation?” Technitrad, 2016. Accessed July 14, 2019. <https://www.technitrad.com/what-is-technical-translation/>.

Wiggins, Dion. “Omniscien.” *Omniscien* (blog). Accessed May 12, 2019. <https://omniscien.com/migrate-from-smt-to-nmt/>.



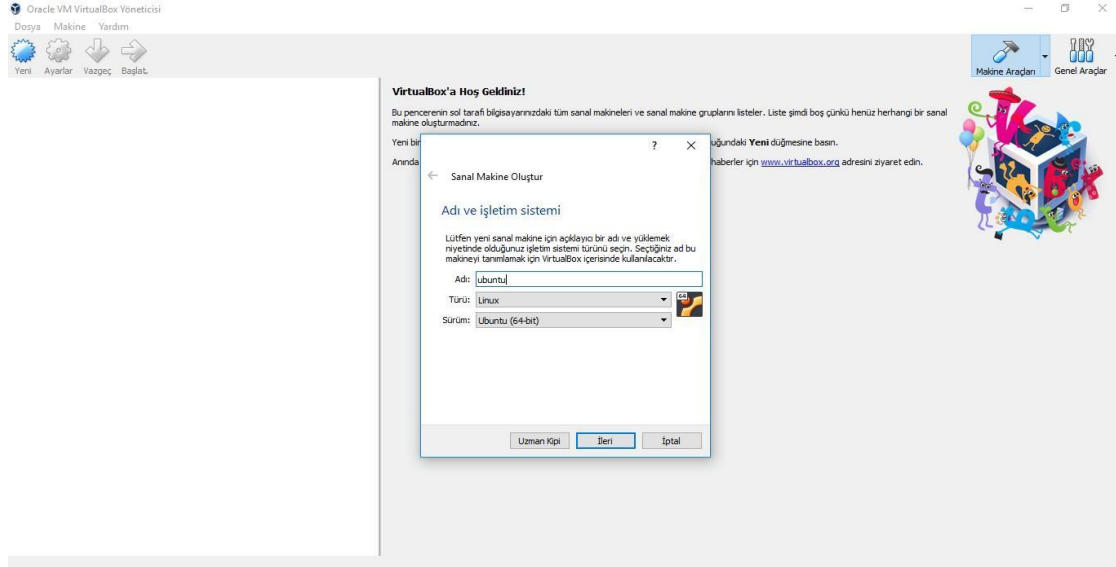
APPENDICES

APPENDIX I



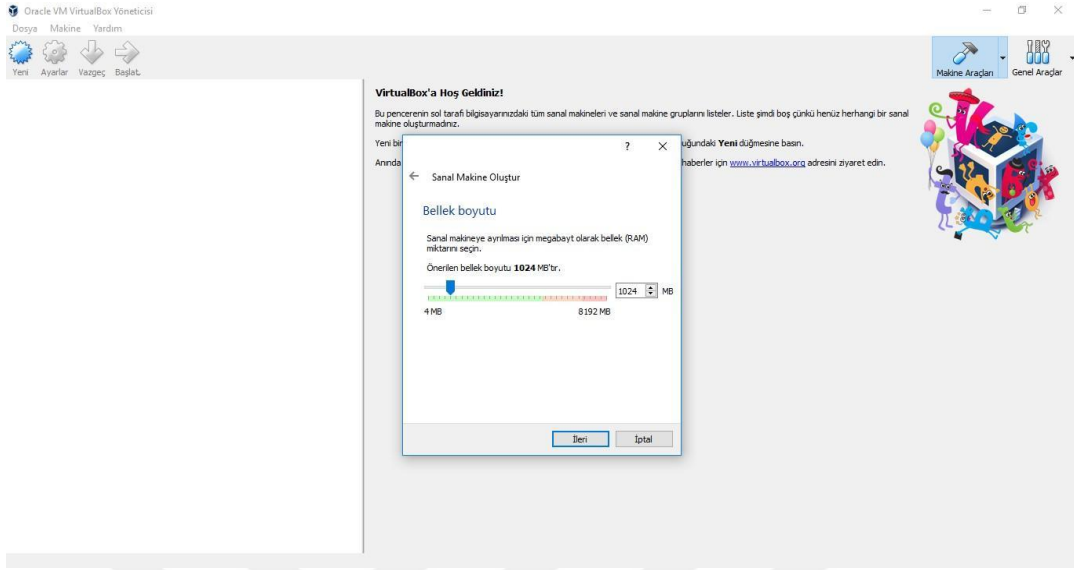
Screenshot of Selecting the Operating System

APPENDIX II



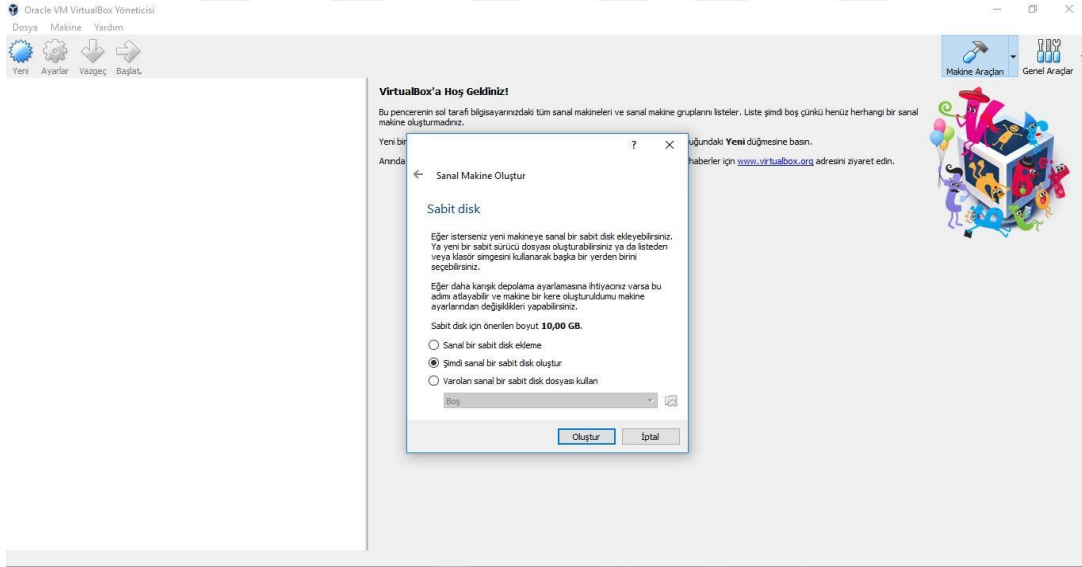
Screenshot of Selecting the Ubuntu Operating System

APPENDIX III



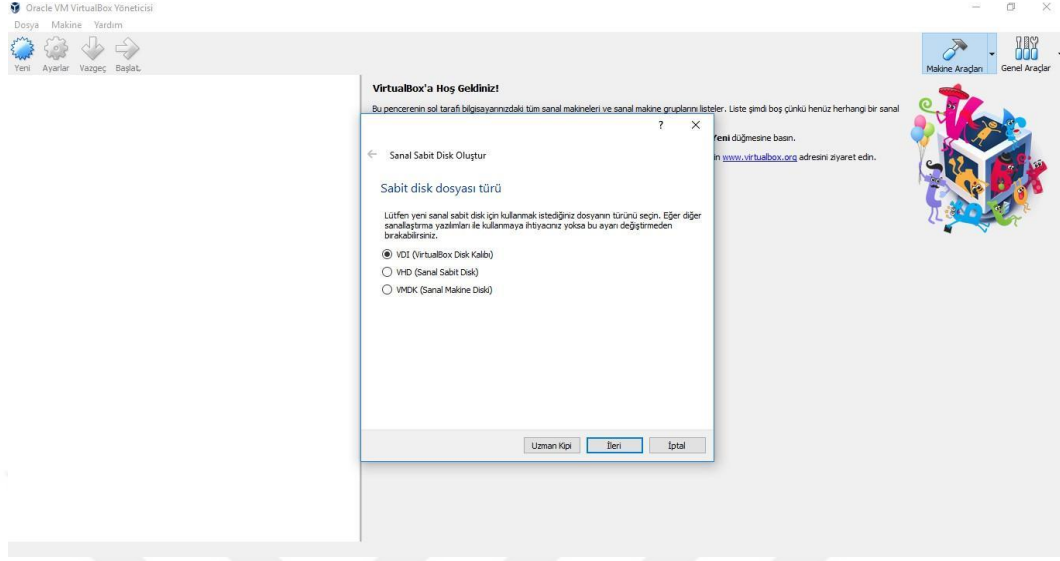
Screenshot of Selecting The Memory Size of VM

APPENDIX IV



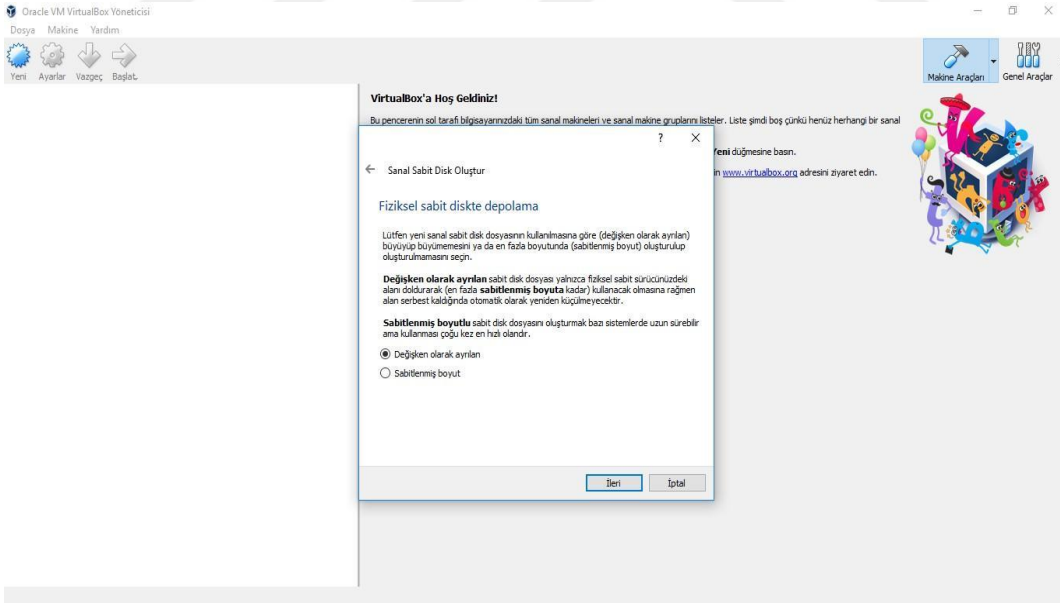
Screenshot of Creating a Virtual Disk

APPENDIX V



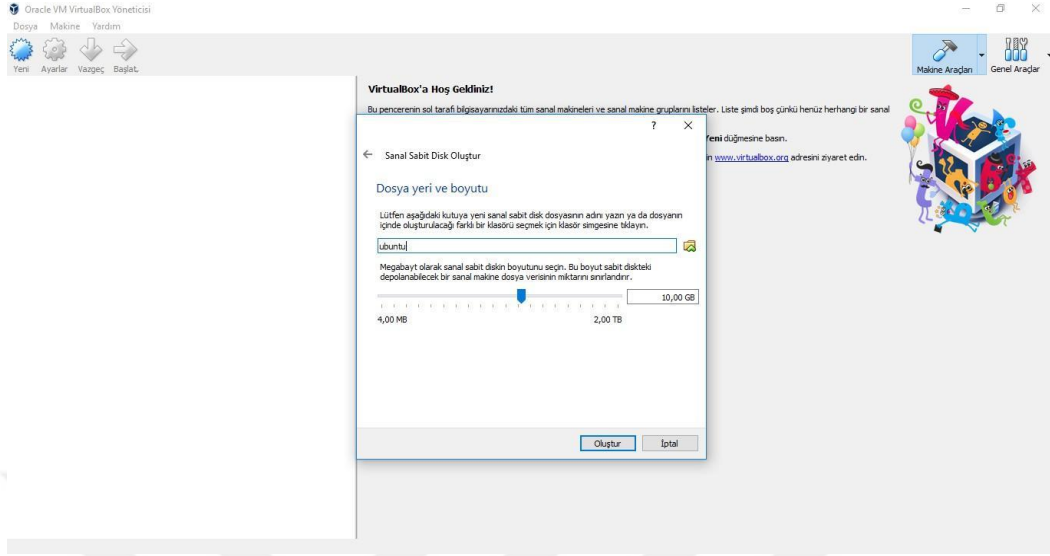
Screenshot of Selecting Virtual Disk Image

APPENDIX VI



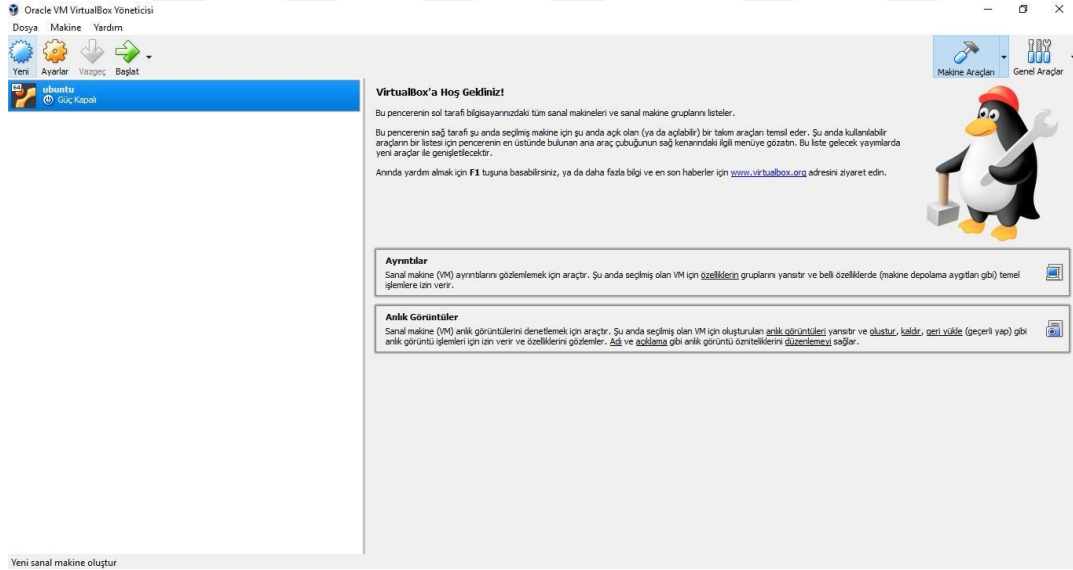
Selecting the Dynamically Allocated Storage Space

APPENDIX VII



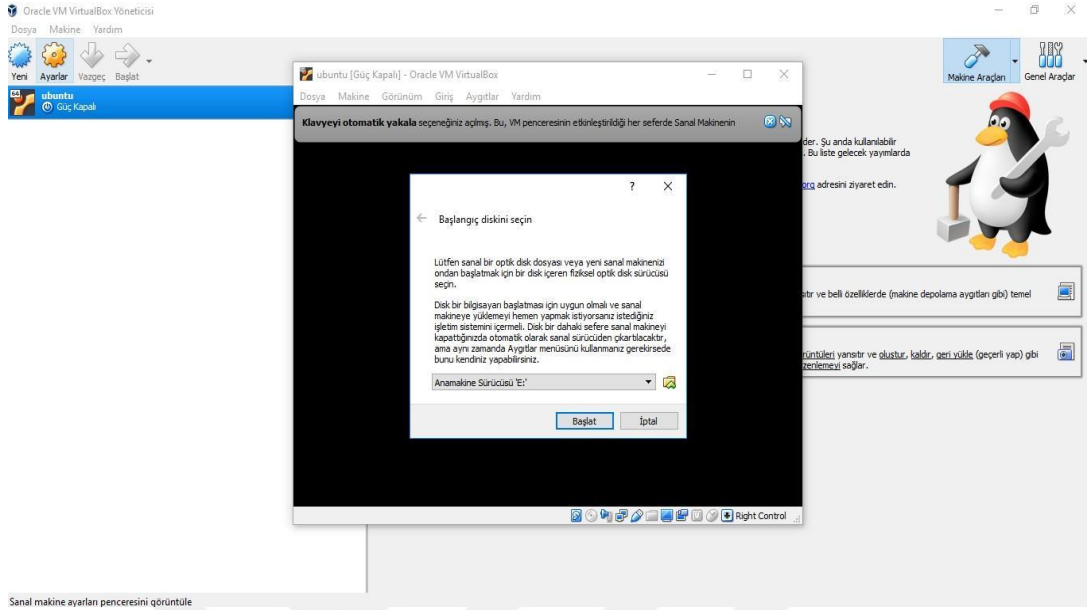
Screenshot of Selecting the Disk Size of VM

APPENDIX VIII



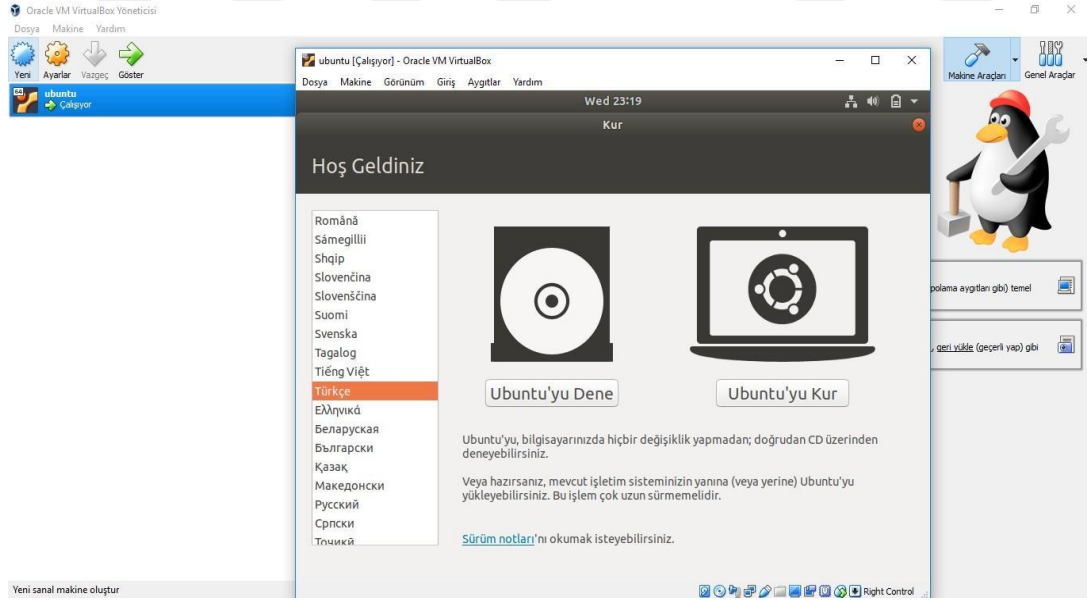
Screenshot of Initializing the Installation

APPENDIX IX



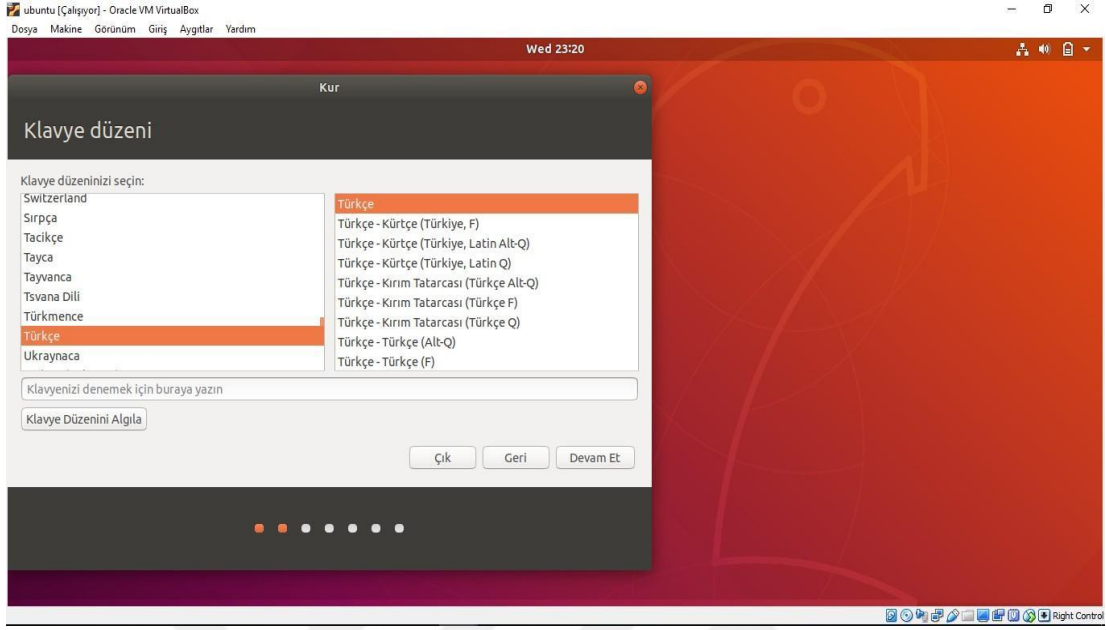
Screenshot of Selecting the ISO file

APPENDIX X



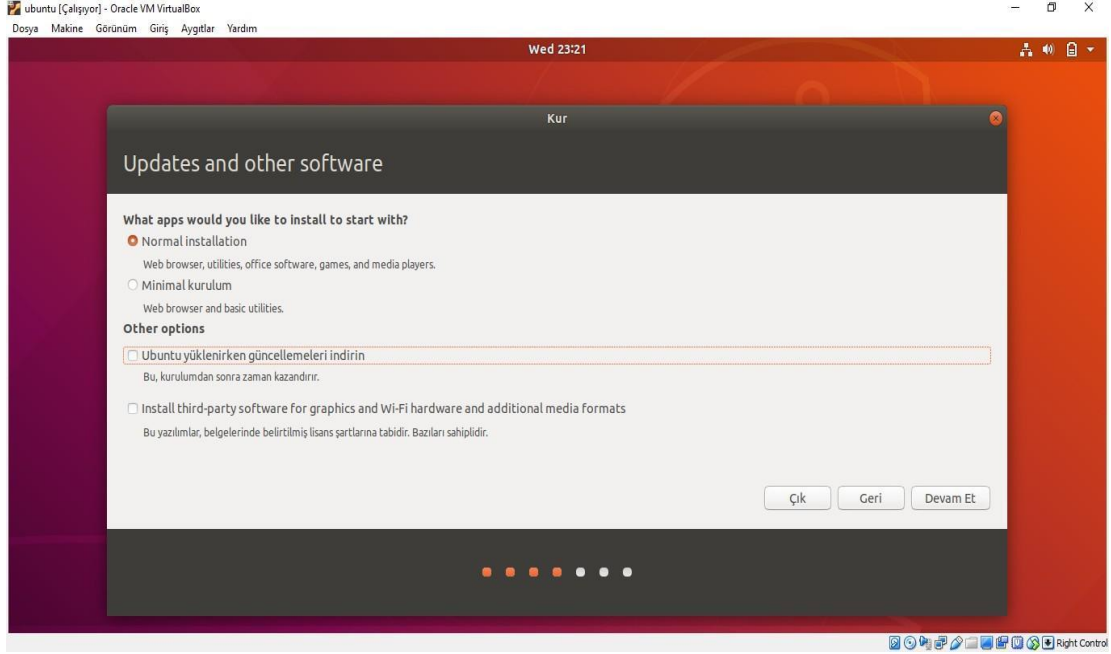
Screenshot of Selecting the Language

APPENDIX XI



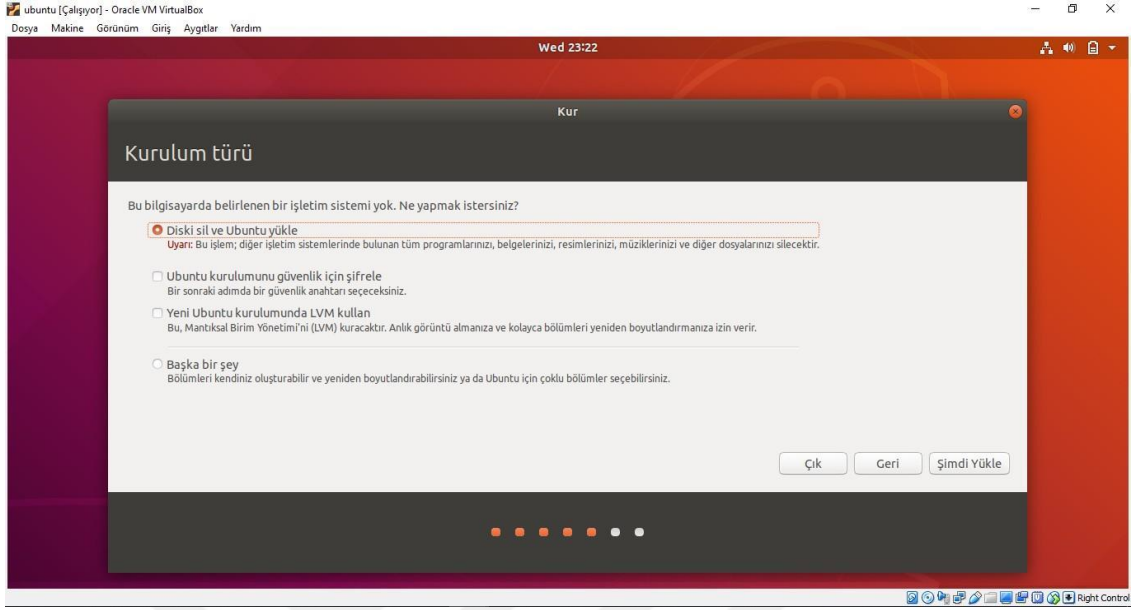
Screenshot of Keyboard Selection

APPENDIX XII



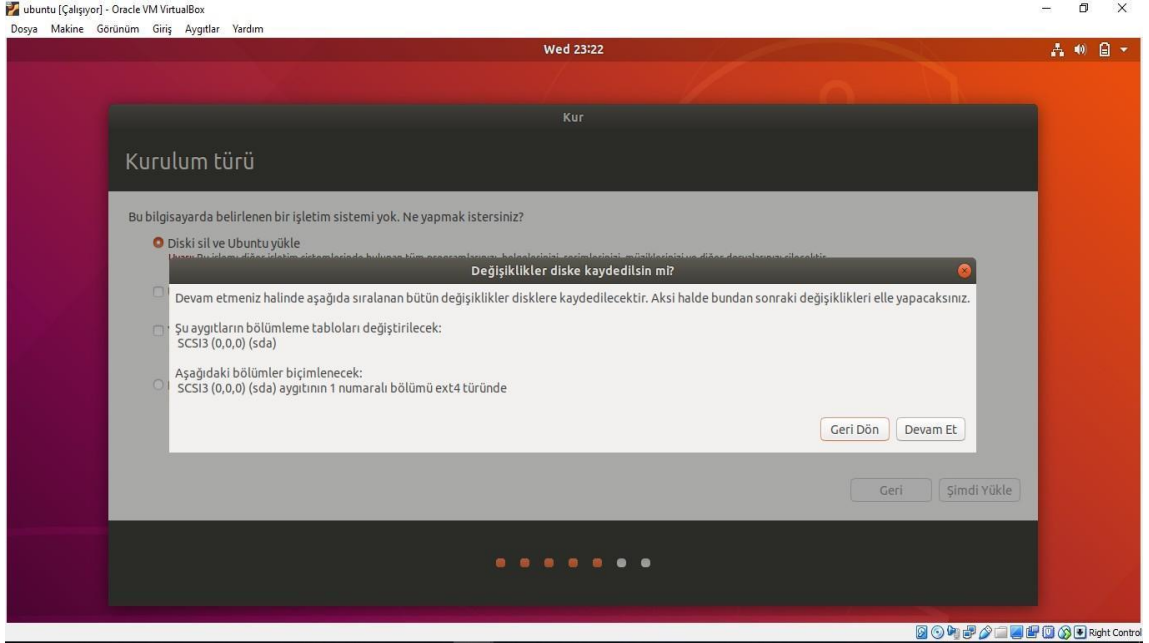
Screenshot of Starting the Complete Installation Process

APPENDIX XIII



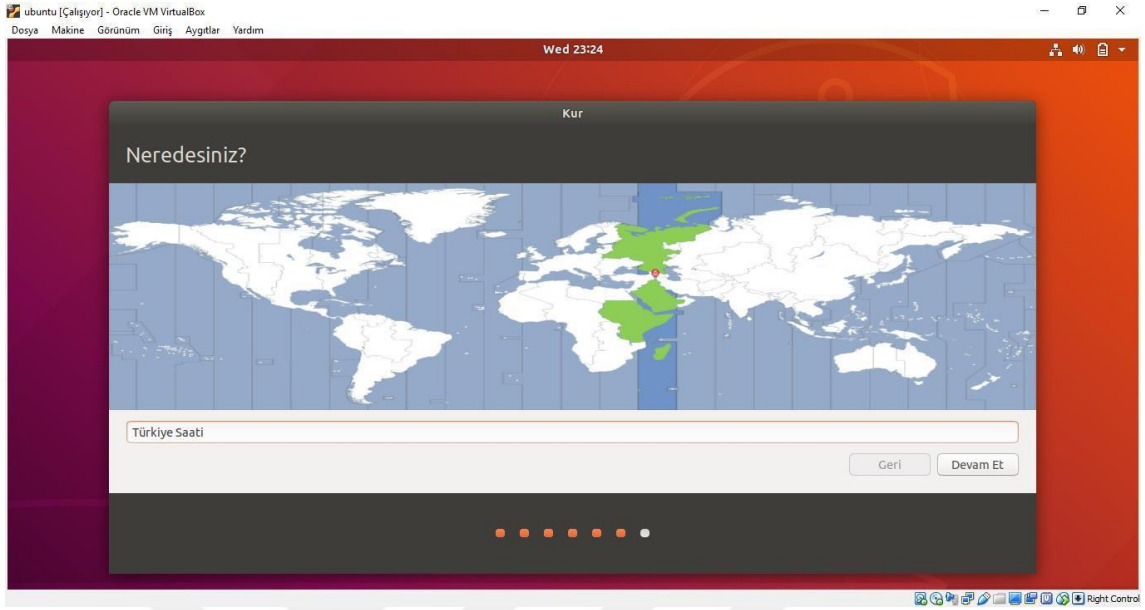
Screenshot of Erasing Disk and Installing Ubuntu

APPENDIX XIV



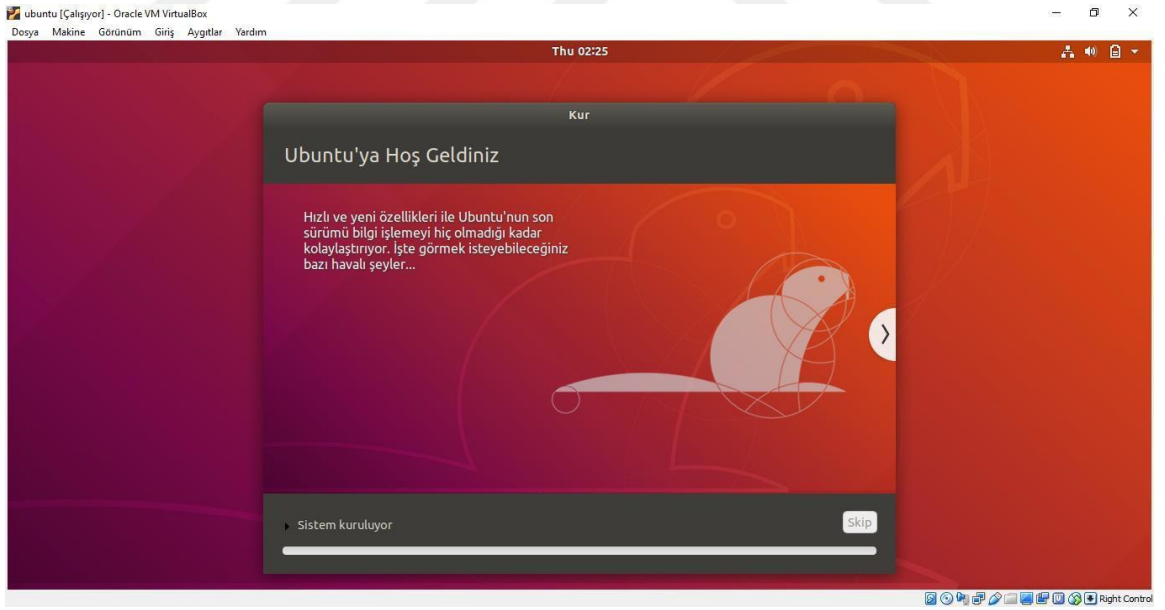
Screenshot of Ongoing Installation Process

APPENDIX XV



Screenshot of Selecting the Region

APPENDIX XVI



Screenshot, Showing the System Installation

ÖZGEÇMİŞ

ÖZGEÇMİŞ			
Adı, Soyadı	Alper	ÇALIK	
Doğum Yeri ve Yılı	Kocasinan	1993	
Bildiği Yabancı Diller	İngilizce	Almanca	
ve Düzeyi	(İleri)	(İleri)	
Eğitim Durumu	Başlama - Bitirme Yılı	Kurum Adı	
Lise	2007	2011	24 Kasım Anadolu Lisesi
Lisans	2011	2016	Trakya Üniversitesi
Yüksek Lisans	2016	2019	İstanbul 29 Mayıs Üniversitesi
Doktora			
Çalıştığı Kurum/lar	Başlama - Ayrılma Yılı	Çalışılan Kurumun Adı	
1.	2018	2018	Localex Language Service Provider
2.	2018	-	Bartın Üniversitesi
Üye Olduğu Bilimsel ve Mesleki Kuruluşlar	-		
Katıldığı Proje ve Toplantılar	-		
Yayımlar:	1) Çalık, A. Creating an Automotive Oriented Statistical Based Machine Translation Engine - Enriching Translation Studies through Re-Readings [28.03.2018] 2) Çalık, A. Çeviri Teknolojileri Bağlamında Makine Çevirisinin Çevirmen Kimliğine Olumlu/Olumsuz Etkileri: “Post-Editör” Çevirmen Kimliğine Genel Bir Bakış - AsosCongress IV [26.10.2018] 3) Akcan, G., Çalık, A. Teknolojinin Zaman Yönetimi Konusunda Çevirmenler Üzerinde Yarattığı Baskının İncelenmesi – <i>Çevirmen Psikolojisi</i> [Aralık 2018]		
Diğer:			
İletişim (e-posta):	acalik@bartin.edu.tr		
	Tarih İmza Adı Soyadı	13/09/2019 Alper ÇALIK	

