ISSN 1512-1232

PRE-PROCESSING PHASE IS MUST FOR MACHINE TRANSLATION SYSTEM

Vivek Dubey

Associate Professor in Dept. of Computer Sc. & Engg in Shri Shankaracharya College of Engg. & Tech.,
Bhilai, CG, India.

Abstract

A Machine Translation (MT) is the process of translating a source language sentence to target language sentence with same meaning and similar construction. However fully automatic MT system (FAMTS) has been demand. It includes preprocessing phase, core MT phase and post-processing phase. In this paper, preprocessing phase has been explored and explained using examples and also illustrated algorithm to construct model. Already, FAMTS has been designed and developed in C/C++ using example based approach. Experimental results of system and evaluation of system with other MT systems have been proven its quality and demand.

Keywords: *Segmentation, Conjunction Words, Parsing Sentence, Multiword Expressions, Chunking Sentence.*

1 Introduction

Machine Translation (MT) is the automated process of translating text units of the one natural language called as source language (SL) into a second natural language called as target language (TL) equally by using computer [8,12]. Machine Translation (MT) applications have been proved its importance in Internet world. Consequently, the demand of online MT system has been increased in all application [1]. In fact, there are many cases in which the natural translation results in diverse form than that of the original [4].

Online MT system includes pre-processing phase, core machine translation phase and post-processing phase. Although, online MT system accepts inputs in various forms like a file, a paragraph, a sentence as well as a word but it always translates inputs word-by-word. The core objective of pre-processing is to provide reliable data i.e. standard sentences, standard words. The pre-processed inputs have been used for parsing sentence, chunking sentence and core MT phase, so that translation has been performed well and post processing phase generates ‘faithful’ image of the source language (SL) into target language (TL).

A fully automatic machine translation system (FAMTS) using transfer approach has been designed and developed in C/C++ for English-Hindi language pair including pre-processing phase, core MT phase and post processing phase and 1000 English sentences have been translated and evaluated. Obtained results feels importance of post processing phase and with only its effectiveness the results of proposed system shows better than other available MT system.

In this paper, mainly pre-processing phase of FAMTS has been focused. Section-2 explains organization of preprocessing phase of MT, section-3 briefs text normalization, section-4 describes sentence segmentation process, section-5 illustrates handling conjunction words of compound/complex sentences, section-6 explains parsing process of sentences, and section-7 illustrates handling multi words expressions, section-8 presents chunking process and section-9 concludes paper.

Vivek Dubey is with the Shri Shankaracharya College of Engineering and Technology, Bhilai, Chhattishgarh, India; (e-mail: vivekdubey22@ gmail.com).

ISSN 1512-1232

2 Organization of Preprocessing Phase

Preprocessing phase is prior to linguistic analysis [15]. Making source language (SL) reliable, this phase has been responsible formed sequence of events like normalization of input text, sentence segmentation, handling conjunction word, parsing sentence, handling multi-word expressions and chunking sentence.

Usually text preprocessing services are included in NLP application. However, there are many constraints in using these services such as interfacing, operating, etc. Also they are not reliable since many results are found to be incorrect. However, it is strange that existing MTS are unable to translate long sentences, compound-complex sentences and even some simple sentences. Therefore, it proves to be a good motivation to design and develop an advanced preprocessor. The architecture of preprocessor has been illustrated in Fig. 2.1 with input as real-world text and output as preprocessed text.

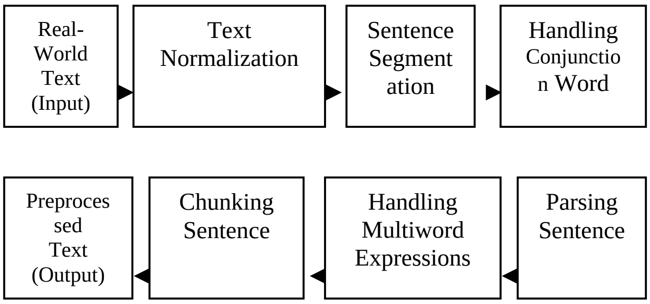

Fig.2.1 Architecture of Language Preprocessor

3 Text Normalization

The role of this phase is identified valid tokens and separate out invalid tokens from input text. Valid tokens are constructed by alphabets (A-Z, a-z), numeric (0-9) and special characters (. , ; " ' [ ] { } ( ) < > - _ + * % / & $ = ? ~ ^ # @ !). Invalid tokens are constructed by control character and non-printable characters [2, 13]. Some examples of invalid character are “, ”, ‘,’,|, etc. White characters like tab and new line are also invalid characters.

Two token are separated by a blank space. Consider a sentence “I am a computer engineer ”. In this sentence, since there are five black spaces, therefore there are five tokens. However, more than one blank space is also called as invalid characters. Consider a sentence “I am <BlackSpace> a doctor ”. In this sentence, even though there are 12 black spaces and two tabs, but there are four tokens. It means 8 blank spaces are invalid characters. Consider a set of tokens “How are you? I am happy. Oh! ”. In this set, there are seven tokens. However, tokens ‘you?’, ‘happy.’ and ‘Oh!’ have been separated from end of sentence marks i.e. ‘?’, ‘.’ and ‘!’, since tokens ‘you’, ‘happy’ and ‘Oh’ are not abbreviated words. Consider a sentence “Mr. and Mrs. Sharma are going to S.S.C.E.T. ”, in this sentence since tokens ‘Mr.’, ‘Mrs.’ and ‘S.S.C.E.T’ are abbreviated, there fore no need of separate out character ‘.’. Similarly in tokens ‘100%’, ‘$1000’, ‘12.50’, ‘10/06/1967’, ‘four-story’, no need of separate out character ‘%’, ‘$’, ‘/’, ‘-'.

Some time short words have been used instead of long words. Consider a sentence, “I don’t know my college.”. In this sentence, since token “don’t” is in short form, it has been normalized in long form as “do not”. To normalize sentences, an algorithm TextNormalization has been designed and developed which is explained in Fig. 3.1. It uses mapping Short-Form-Word i.e. don’t to Long-Form-Word i.e do not through algorithm ConvertLongForm (see Fig. 3.2). Also, it uses algorithm JoinHyphenatedWord as shown in Fig. 3.3. The input and output of TextNormalization are RealWorldFile and NormalizedText respectively.

ISSN 1512-1232

```
NormalizedText TextNormalization(RealWorldFile)
Do while End-of-file
        Read word
        If word = {<Enter>, <Tab>, <Black Space>, <Non-ASCII>} ; Continue
        If word = {short form like don't, didn't,etc}/*Rule Based Approach*/
                LongFormWord = ConvertLongForm(word)
                Write LongFormWord;
                Continue
        If Hyphenated Word
                Word = JoinTwoHyphenatedWord(Hyphenated Word)
                Write Word;
                Continue
        Write Word
End Do
```

Fig. 3.1 Algorithm of TextNormalization

```
SubFunction LongFormWord ConvertLongWord ( ShortFormWord)
Do while End-of-ShortWord
        Read ShortWord
        If ShortWord = ShortFormWord
                ShortFormWord = LongFormWord
                Break
        End if
End Do
```

Fig.3.2 Algorithm ConvertLongWord

```
SubFunction Word JoinTwoHyphenatedWord ( HyphenatedWord)
Do while End-of-HyphenatedWord
        Read a Character
        If Character = '-' ; Continue
        If Character = BlankSpace ; Continue
        Join Character in Word
End Do
```

Fig.3.3 Algorithm JoinTwoHyphenatedWord

4 Sentence Segmentation

Parsers and MT have been worked on sentence properly. [7, 11, 15] has been reported that end-of-sentence punctuation marks are ambiguous. Input text, whenever, are in form of paragraph, then paragraph must be segmented into sentences. Mostly sentences are used punctuation marks as period (.), question mark (?) and exclaimnation marks (!). Consider a paragraph P1 and its equivalent sentences S1-S3.

P1: Hai ! My name Ramgopal . What is your name ?
S1: Hai !
S2: My name is Ramgopal .
S3: What is your name ?

In such paragraph, there is no confusion to system in understanding or detection end of sentence. However, if sentence is with more than one punctuation mark, system has not been detection end-of-mark and consequence has been segmented paragraph in wrong way. Consider a paragraph P2 and its equivalent sentences S4-S7.

P2: He is Prof. Ramgopal. His brother's name is Mr.Rajgopal.
S4: He is Prof.
S5: Ramgopal.
S6: His brother's name is Mr.
S7: Rajgopal.

In paragraph P2, since words 'Prof.' and 'Mr.' are abbreviated words means they are used as it is. Therefore, after word 'Mr', punctuation period has been used as a part of abbreviated word and not used as end-of-mark. Hence instead of four sentences i.e. S4-S7, they has been segmented as two sentences i.e. S8-S9.

ISSN 1512-1232

S9: He is Prof. Ramgopal.

S9: His brother’s name is Mr. Rajgopal.

Sentence segmentation initially has been isolated words. Secondly it has been identified words as abbreviated words or not and if it is found as abbreviated word, end-of-mark has been ignored as end-of-sentence detection. An algorithm has been design and developed for sentence segmentation. Input to algorithm is paragraph and list of abbreviated word and output of algorithm is segmented sentences. The complete algorithm has been shown in Fig. 4.1.

```
SegmentedSentence SentenceSegmentation (Paragraph, ListOfAbbreviatedWord)
        While not end of Paragraph
                Read a Word
                If character in Word = PunctuationMark
                        If word = AbbreviatedWord
                                Write Word;
                                Write BlankSpace
                        Else Write Word
                                Write BlankSpace;
                                Generate NewLine
                        Endif
                Endif
        EndDo
```

Fig.4.1 Algorithm of Sentence Segmentation

5 Handling Conjunction Word

Usually, input sentences are in complex and compound nature. Always, they are after MT unable to maintain source language sense in target language. Hence therefore they have been used break before conjunction words by adding suffix comma [6]. Consider a compound sentence English sentence SentEeng5.1 and its Hindi sentence SentHin5.2 as shown in Fig. 5.1.

SentEng5.1: Ram’s father is an engineer and Ram’s mother is a doctor.

SentHin5.1: राम के पिताजी एक इंजीनियर हैं और राम की माताजी एक डाक्टर हैं ।

Fig.5.1 Complex/Compound Sentence

In SentEng 5.1, since there are two simple sentences, after formatting input sentence, the updated input sentence has been become as “<Ram’s father is an engineer> **,** and <Ram’s mother is a doctor>. Hence therefore at the post processing during target language generation what ever before comma has been generated first and then rest has been generated.

In Table 5.1, conjunction words have been listed and in Fig 5.2, ConjHandle algorithm has been shown. Its input is FormattedSentence and output is Update Sentence.

Table 5.1. Conjunction Word List

Conjunction Type	Example
Coordinating Conjunction	For, and, or, nor, but, yet, so
Subordinating Conjunction	After, although, as, as if, as long as, as much as, as though, because, before, by the time, even if, even though, if, I order that, in case, least, once, only if, providing that, since, so that, than, that, though, till, etc
Correlative Conjunctions	Both… and, either… or, neither… nor, not only… but also, whether… or, etc

ISSN 1512-1232

```
UpdatSentence ConjHandle(FormattedSentence)
Do while End-of-file
        Read Word
        If word = {<and>, <or>, <for>, etc} ;Write ‘, ’ ;Endif
        Write Word
End Do
```

Fig 5.2 Algorithm ConjHandle

6 Parsing Sentence

Words are divided into different parts-of-speech (POS) [19, 22] like Noun (i.e. Ram, Dog, etc), Pronoun (i.e. You, We, etc), Adjective (i.e. beautiful, much,etc), Verb (i.e. am, go, etc), Adverb (i.e. yesterday, much), Preposition (i.e. at, for, etc), Conjunction (i.e. and, else etc), Interjection.

POS tagging (POST or just tagging for short) is the process of assigning a POS or other lexical class marker or tag to each word or token using a penn Treebank tagset like Noun (i.e. Ram/NNP), Pronoun (i.e. You/PRP), Adjective (i.e. Bigger/JJR), Verb (i.e. go/VB),Adverb (i.e. yesterday/RB), Preposition (i.e. at/IN), Conjunction (i.e. and/CC), Determiner (i.e. a/DT).

The input to a tagging algorithm is a string of words and a specified tagset. The output is a single best tag for each word. The output of POST of input words like “going” and “book” are “going/VBG” and “book/NN” respectively. However, POST is done not only word to word but also sentence to sentence, for example, tagging of English sentence “Book that flight” is “Book/VB that/DT flight/NN”.

GENIA tagger is freeing available to download from internet; it has been used in proposed preprocessing phase for tagging input sentences. Fig 6.1 illustrates its working where option “–i” is for input file and option “–o” is for output file and Fig 6.2 explains its input and output.

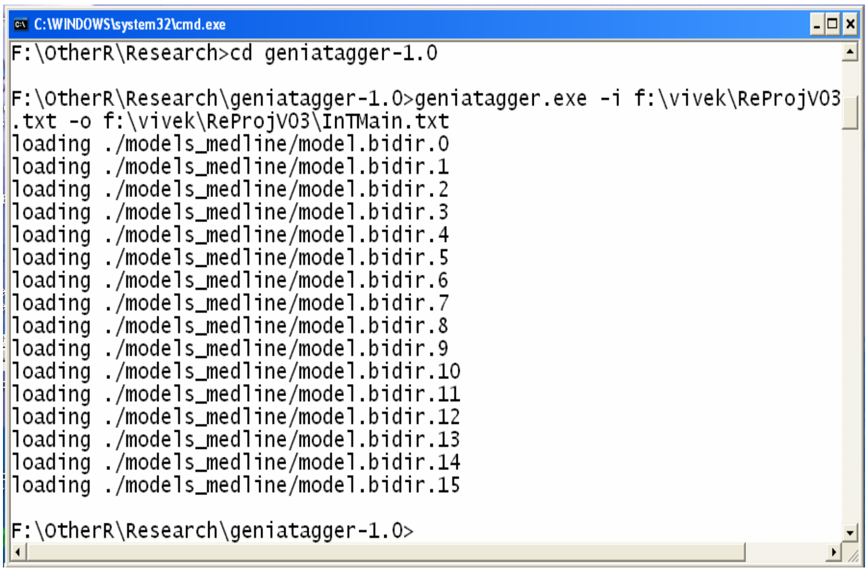

Fig 6.1: Working Sample of GENIA Tagger

Input Sentence	**Output Tagged Sentence**
The grand jury commented on a number of other topics	The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS

Fig 6.2: Input-Output Tagged Sentence of GENIA Tagger

ISSN 1512-1232

Handling Multiword Expressions

Quality Machine Translation (QMT) is not achievable without considering Multiword Expressions (MWEs) [3, 5, 10, 14, 21] and normally meaning cannot be derived from their head word. MWEs are one of the stumbling blocks for more precise MT systems and other NLP applications.

MWEs include a large range of linguistic phenomena, such as nominal compounds (police car, coffee machine), phrase verbs (break down, rely on), idiomatic expressions (rock the boat, let the cat out of the bag), terminology (Computer Network) and institutionalized phrases (salt and pepper). Table 7.1 enlists various MWEs such as Proverbs, Phrases in pairs, Phrases, Idioms and two word verbs etc.

Table 7.1. Multiword Expression List

Kind of Multiword Expression	**Examples**
Two Word Verb	Come to, used to, make of, put on, set on
Idioms	A bed of roses, to bring to light
Phrase	Cold war, out of date, all in all
Proverbs	Care kills the cat, save life, might is right

Frequency has been counted of two words verb in brown corpus [18] and has been analyzed and shown statistic of higher first ten in Fig 7.1.

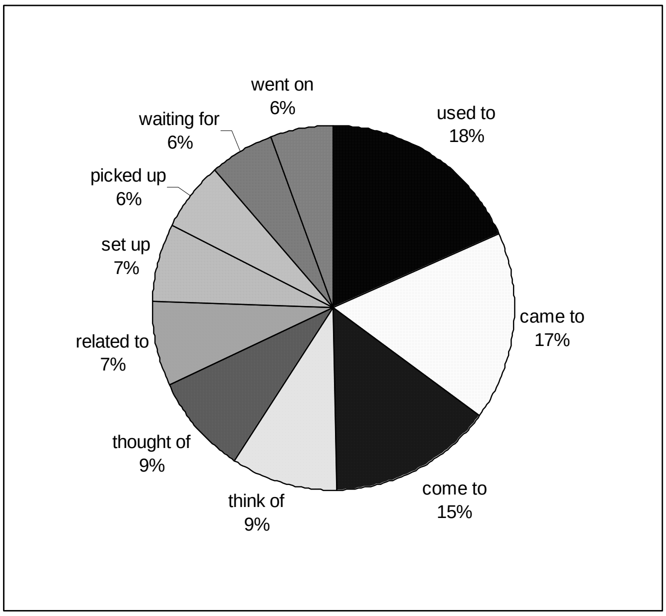

Fig 7.1: Statistic of Frequency count of Two Word Verbs

Consider an English sentence “You can ask for anything”. If verb ‘ask’ has been assumed alone, the MT has been translated it in Hindi as shown in Hin7.1. However, if ‘ask for’ has been assumed as two words verb, MT has been translated correctly it in Hindi as shown in Hin7.2 as in dictionary there is an entry “ask_for =मांगना”. Such MWEs have been handled as illustrated in Table 7.2 and its algorithm has been shown in Fig 7.2.

Hin7.2 : आप कुछ भी के लिये कह सकते हैं

Hin7.2: आप कुछ भी मांग सकते हैं

ISSN 1512-1232

Table 7.2. Handling Two-Words Verb

Steps	English Sentence
Step-1 (Input Sentence)	You can ask for anything
Step-2 (Parsing Sentence)	You/PRP can/MD ask/VB for/IN anything/NN
Step-3 (Handle MWE)	You/PRP can/MD ask_for/VB anything/NN

```
HandledMWE HandlingMWE( ParsedEngSent)
Begin
        Do while not end of ParsedEngSent
        Read Word
        If WordPOS = Verb(VB, VBN, VBD, VBG, VBP)
                Read NextWord
                IF NextWordPOS = IN
                        Concatenat (Word, '_')
                        Concatenat (Word, NextWord)
                        Concatenat (Word, "/IN")
                End if
        End if
        Write Word
End do
```

Fig 7.2. Algorithm Handling MWE

8 Chunking Sentence

The most basic unit of linguistic structure is the word. Words are always found in the chunks / groups / phrases. Mainly there are four chunks [20] – Noun Chunk (NX), Verb Chunk (VX), Adjective Chunk (AdjX) and Adverbial Chunk (AdvX). The examples of NX are "A boy", "A good boy"; VX are "go", "am going"; AdjX are "easy to understand", "very honest"; AdvX are "inside the room", "rapidly like a bat".

The process of segmenting tagged sentences into meaningful structure is known as chunking [9]. It is also referred to as chunk parsing or shallow parsing. In the last few years in MT, it has been a relatively active field. Mainly, there are three advantages of using chunks: 1. Sentence having infinite word length sentence having can be computed. 2. Small structure can be translated accurately. 3. Reordering can be implemented.

In NC rule, there is at most one determiner (Det), one cardinal (Card), one ordinal (Ord), Quant and AdjC as shown in RNP1.

NP $\rightarrow$ (Det) (Card) (Ord) (Quant) (AP) Nominal ...(RNP1)

The VX may be single auxiliaries or main verb or auxiliaries + main verb. Auxiliaries include the modal verbs can, could, may, might, must, will, would, shall, and should, the perfect auxiliary have, the progressive auxiliary be, and the **passive auxiliary** be. A sentence can have multiple auxiliary verbs, but they must occur in a particular order:

Modal < Perfect < Progressive < Passive ...(RVP1)

Consider an affirmative sentence "He is a student". Its tagged sentence has been computed as "He/PRP is/VBZ reading/VBG a/DT book/NN". During chunking process in step-1, NX and VX

ISSN 1512-1232

have been computed. In step-2, Subject has been attached with VX, so that during language modeling, mapping SL→TL would be computed as “He is reading →पढ़ रहा है ”. The complete process has been explained in Table 8.1.

Table 8.1. Chunking Process of Affirmative Sentence

Step-1	**Step-2**
NP→He/PRP VX→is/VBZ reading/VBG NP→a/DT book/NN	NP→He/PRP VX→He/PRP is/VBZ reading/VBG NP→a/DT book/NN

In proposed chunking, there are two steps: first it has been chunked and second it has been managed subject with verb agreement. Fig 8.1 has been shown its algorithm.

```
ChunkedTaggedSentence Chunking (TaggedSentence)
Begin
        /*************************Case 1**************************/
        Do      Read TaggedToken
                If TaggedToken is from Noun # (Det) (Card) (Ord)  # (Quant) (AX) Nominal POS
                        Concanate (NP, Noun)
                Endif
        While TaggedToken <> Noun
        If NP <> Null
                Write NP ; NP ← Null; If TaggedToken = Null ; Stop End if
        End if
        /*************************Case 2**************************/
        Do      Read TaggedToken
                If TaggedToken is from Pronoun1 # I he she we you they
                        Concanate (NP, Noun)
                Endif
        While TaggedToken <> Pronoun1
        If NP <> Null
                Write NP ; NP ← Null; If TaggedToken = Null ; Stop End if
        End if
        /*************************Case 3**************************/
        Do      Read TaggedToken
                If TaggedToken is from NounPronoun2 # my our brother
                        Concanate (PAdjX, Noun)
                Endif
        While TaggedToken <> NounPronoun2
        If PAdjX <> Null
                Write PAdjX ; PAdjX ← Null; If TaggedToken = Null ; Stop End if
        End if
        /*************************Case 4**************************/
        Do      Read TaggedToken
                If TaggedToken is from NounPronoun3 # Adjective Noun
                        Concanate (AdjQuX, Noun)
                Endif
        While TaggedToken <> NounPronoun2
        If AdjQuX <> Null
                Write AdjQuX ;AdjQuX ← Null; If TaggedToken = Null; Stop End if
        End if
        /*************************Case 5**************************/
        Do      Read TaggedToken
                If TaggedToken is from Verb # VB VBP VBZ VBG VBD VBN MD
                        Concanate (VX, Verb)
                Endif
        While TaggedToken <> Verb
        If VX <> Null
                Write VX ;VX ← Null; If TaggedToken = Null; Stop End if
        End if
        /*************************Case 5**************************/
        Write TaggedToken #Other
End
```

Fig 8.1. Chunking Algorithm

ISSN 1512-1232

9 Conclusion

In this paper, step-by-step preprocessing phase of FAMTS has been explained. Examples in each phase have been explained. All phases - text normalization, sentence segmentation, handling conjunction words, parsing process of sentences, handling multi words expressions and chunking process of a sentence have been explored experimentally. Algorithms – TextNormalization, ConvertLongWord, JoinTwoHyphenatedWord, Sentence Segmentation, ConjHandle, Handling MWE and Chunking have been described.

References

1. F. Acikgoz and O. Sert, "*Interlingual Machine Translation: Prospects and Setbacks*", Translation Journal 10 (3), July 2006, pp. 10.
2. Clark, "*Pre-processing Very Noisy Text*", Proc. of Workshop on Shallow Processing of Large Corpora, 2003.
3. Colin Bannard, Timothy Baldwin, Alex Lascarides, "A statistical approach to the semantics of Verb-Particles", Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment, pp.65-72, July 12, 2003.
4. B. J. Dorr, "*Machine Translation Divergences: A Formal Description and Proposed Solution*", In Proceedings of Computational Linguistics, pp. 597-633, 1994.
5. V. Dubey and H.R. Sharma, "*Generic Implementation of Multiword Verb Phrases and Idioms Expression In English-Hindi Machine Translation*", International Journal of Engg. Research & Indu. Apllis (IJERIA), ISSN 0974-1518, Vol.3, No. II, May 2010, pp. 285-293, http://www.ascentjournals.com.
6. Eriksson and T. Myhrman , Department of Linguistics and Philology, Språkteknologiprogrammet, (Language Technology Programme), Master's Tthesis in Computational Linguistics, June 7, 2010, Supervisors: Tiedemann, Uppsala University Eva Pettersson, Convertus AB.
7. G. Grefenstette and P. Tapanainen,"*What is a word, what is a sentence? Problems of tokenization*", In Proceedings of the 3rd Conference on Computational Lexicography and Text Research (COMPLEX'94), 1994.
8. D. Gupta, "*Contributions to English to Hindi Machine Translation using Example-Based Approach*", Ph.D. Thesis, Dept. of Mathematics, IIT Delhi, 2005.
9. Igor Boehm,"Rule Based vs. Statistical Chunking of CoNLL Data Sets", 2005.
10. H. J. Kaalep, K. Muischnek , "Inconsistent Selectional Criteria in Semi-automatic Multi-word unit extraction", 2003, 7th Conference on Computational Lexicography and Corpus Research, Ed. By F. Kiefer, J.Pajzs, Research Institute for Linguistics, Hungarian Academy of Sciences, Budapest 2003, pp. 27-36.
11. P. Potamites , "Modeling Abbreviations and Punctuation for Sentence Segmentation" , P. :December 16, 2007.
12. Y. Salem and B. Nolan, "Designing an XML lexicon architecture for Arabic machine translation based on role and reference grammar", MEDAR 2009: 2nd International Conference on Arabic Language Resources & Tools, 22-23 April 2009, Cairo, Egypt, pp.221-229.
13. Sproat R., Black A., Chen S., Kumar S., Ostendorf M. and Richards C., "*Normalization of non-standard words*", presented at Computer Speech & Language, 2001, pp.287-333.
14. S. Ganesh, S. Harsha, P. Pingali, and V. Varma, "Statistical Transliteration for Cross Language Information Retrieval using HMM alignment and CRF", IJCNLP 2008: 2nd International Workshop on Cross-Lingual Information Access (CLIA) Proceedings of the workshop, 11 January 2008, Hyderabad, India; pp.42-47.
15. K. Tomanek, J. Wermter, and U. Hahn, "*Sentence and token splitting based on conditional random fields*", PACLING 2007 - Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pp. 49-57. Melbourne, Australia, September 19-21, 2007.

ISSN 1512-1232

16. J. Xu, R. Zens, and H. Ney, “*Sentence Segmentation Using IBM Word Alignment Model 1*”, In Proceedings of the 10th Annual Conference of the European Association for Machine Translation (EAMT 2005), pp. 280-287, Budapest, Hungary, May 2005.
17. W. Wang, S. DeNeefe, K. Knight, and D. Marcu, “*What can syntax-based MT learn from phrase-based MT?*”, EMNLP-CoNLL-2007: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic; pp. 755-763, 2007.
18. WWW1 http://en.wikipedia.org/wiki/Brown_Corpus (viewed on 25/04/2011)
19. Dorr B.J. and Monz C., “Intro to Computational Linguistics”, CMSC 723 / LING 645, November 3, 2004.
20. Jurafsky D. and Martin J., “Speech and Language Processing: An Introduction to Speech Recognition, Computational Linguistics and Natural Language Processing”, Second Edition, 2006.
21. Anna Mundelein, “Master’s Thesis: Identification of Idiomatic Expressions Using Parallel Corpora”, October 29, 2008.
22. Cassel S., “Thesis: MaltParser and LIBLINEAR Transition-based dependency parsing with linear classification for feature model optimization”, December 7, 2009.

Article received: 2012-02-08