The Multi-Liaison Algorithm

Ms. Anjali Ganesh Jivani
Dept. of CSE
The M. S. University of Baroda
Vadodara, India

Ms.Amisha Hetal Shingala
Dept. of MCA,
SVIT, Gujarat Technological University,
Vasad, India

Dr. Paresh. V. Virparia
Dept. of CS, SP University
V.V.Nagar, India

***Abstract*— In this paper we present an approach for extracting multiple connections or links between subject and object from natural language input (English), which can have one or more than one subject, predicate and object. The parse tree visualization and the dependencies generated from the Stanford Parser are used to extract this information from the given sentence. Using the dependencies we have generated an output which displays which subject is related to which object and the connecting predicate. Finding the subjects and objects helps in determining the entities involved and the predicates determine the relationship that exists between the subject and the object. An algorithm has been developed to do so and this algorithm is elucidated in detail step-wise. We have named our algorithm 'The Multi-Liaison Algorithm' since the liaison between the subjects and objects would be displayed. The word 'liaison' has been used since we are displaying the relationship and association between the subjects and predicates. This output would be useful for natural language processing (NLP), information retrieval, information extraction and also text mining applications.**

Keywords- parse; liaison; relationship; extraction; information retrieval; text mining.

I. INTRODUCTION

An English sentence can have multiple subjects and objects and the Multi-Liaison Algorithm as proposed by us would display them with the connecting verbs/predicates. According to the approach presented in [3], a triplet in a sentence is defined as a relation between subject and object, the relation being the predicate. We have made an enhancement to it by including multiple connections or liaisons depending on the type of sentence that is given as input. The sentence is parsed with the help of the Stanford parser and then using the output of the parser as input to our program, all the subjects, objects and the predicates i.e. the multiple liaisons are displayed. The subjects can either be nouns or even pronouns. Moreover, one subject can be related to multiple objects and vice-versa.

The algorithm was developed in JAVA using Net Beans IDE 6.5 RC2. The performance of the application was measured using a system with CPU 1.99 GHz and 2.86 GB RAM. The algorithm was tested with a variety of input and some sample inputs and outputs are shown further. This algorithm is expected to be useful to researchers involved in NLP and Text Mining.

II. THE STANFORD PARSER

The Stanford Parser is a probabilistic parser which uses the knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences. This package is a Java implementation of probabilistic natural language parsers.

The Stanford dependencies provide a representation of grammatical relations between words in a sentence for any user who wants to extract textual relationships. The dependency obtained from Stanford parser can be mapped directly to graphical representation in which words in a sentence are nodes in graph and grammatical relationships are edge labels [10].

We have used them to extract the relation between multiple subjects and objects when the sentence to be parsed is a little complicated. Stanford dependencies (SD) are triplets: name of the relation, governor and dependent.

A. The Parse Tree and Dependencies

The parse tree generated by the Stanford Parser is represented by three divisions: A sentence (S) having a noun phrase (NP), a verbal phrase (VP) and the full stop (.). The root of the tree is S.

The Stanford typed dependencies representation was designed to provide a simple description of the grammatical relationships in a sentence that can easily be understood. The current representation contains approximately 52 grammatical relations [4]. The dependencies are all binary relations. The definitions make use of the Penn Treebank part-of-speech (POS) tags and phrasal labels.

To find the multiple subjects in a sentence our algorithm searches the NP sub tree. The predicate is found in the VP sub tree and the objects are found in three different sub trees, all siblings of the VP sub tree containing the predicate. The sub trees are: PP (prepositional phrase), NP (noun phrase) and ADJP (adjective phrase).

III. THE MULTI-LIAISON ALGORITHM

We explain our algorithm in detail followed by the output. In the first example we have displayed the Stanford output of the sentence followed by the output of our algorithm.

```
Function: CONVERT_ SENTENCE (Input_Str)
Returns: POS tagging, Parse tree, Typed Dependencies
Input_Str: Sentence to be parsed

[Run the Stanford parser with Input_Str as input]

Output_Str ← i) POS of each word
             ii) The parse tree generated
             iii) The typed dependencies
Return Output_Str

Function: MULTI_LIAISON (Output_Str)
Returns: Multiple liaisons or error message
        Function GET_TRIPLETS (Output_Str)
        Function GET_RELATIONSHIP (Output_Str)
Display the multiple liaisons
```

Figure 1. The Multi-Liaison Algorithm.

A. The Detailed Algorithm

We start with parsing a sentence by the Stanford parser and storing the result in some intermediate file so that it can be taken as input for our algorithm. The triplet extraction algorithm of [3] has also been considered before finding the liaisons.

As shown in Fig. 1, the Multi-Liaison Algorithm takes as input the POS of each word, the parse tree and the typed dependencies [9]. Two functions are then called, the first is the GET_TRIPLETS and the second is the GET_RELATIONSHIP.

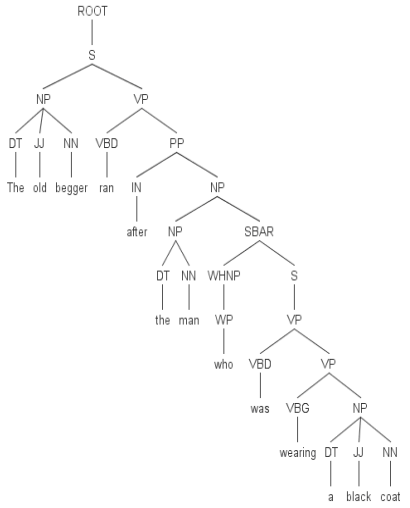

Figure 2. The Stanford Parse Tree.

```
Function: GET_TRIPLET (Output_Str)
Returns: Multiple subjects, objects and predicates
[Read level 1 of Parse Tree – refer Fig. 2]
If tree contains 'NP' or 'NNP' then
        Function GET_SUBJECT (NP sub tree)
Else
        Return error message
If tree contains 'VP' then
       Function GET_PREDICATE (VP sub tree)
       Function GET_OBJECT (VP sub tree)
Else
       Return error message

Function: GET_SUBJECT (NP sub tree)
Returns: Subject(s) and adjective(s)
For (all nodes of NP sub tree) do
    If NP sub tree contains 'NN' or 'NNP' or 'NNS'
then
          Store POS as a subject
    If NP sub tree contains 'JJ' then
          Store POS as an adjective
Return the subject(s) and adjective(s)

Function: GET_PREDICATE (VP sub tree)
Returns: Predicate(s)
For (all nodes of VP sub tree) do
    If VP sub tree contains 'VB?' then
          Store POS as a predicate
    Else
          Return error message
Return the predicate(s)

Function: GET_OBJECT (VP sub tree)
Returns: Object(s)
For (all nodes of VP sub tree) do
    If VP sub tree contains 'NP' then
          For (all nodes of VP_NP sub tree) do
              If VP_NP sub tree contains 'NP' or 'NN'
then
                    Store POS as an object
              Else
                    Return error message
    Else
          Return error message
Return the object(s)
```

Figure 3. The GET_TRIPLETS Function.

As shown in Fig. 3, the GET_TRIPLETS function takes as input the Stanford Parse Tree and by considering the nodes under the NP sub tree and the VP sub tree, finds all the subjects, objects and predicates.

The GET_RELATIONSHIP finds and displays the relationships between the subjects and objects. The algorithm is displayed in Fig. 4.

```
Function: GET_RELATIONSHIP (Output_Str)
Returns: Multiple liaisons / relations
[Read the Stanford typed dependencies from Output_Str]
For (all terms in typed dependencies) do
    If typed dependencies contain 'NSUBJ' then
      Store both words of NSUBJ as S1 and S2
      For each value of subject from GET_SUBJECT do
          If subject matches S2 then
            [Check for predicates]
            For each value of predicate from
            GET_PREDICATE do
                If predicate matches S1 then
                    [Concatenate subject and predicate as
                     R1]
                    Store R1 in the relation
   If typed dependencies contain 'DOBJ' or 'PREP' then
      Store both the words as D1 and D2
      For each value of object in GET_OBJECT do
          If object matches D2 then
            Store value of object in the relation as R2
Return R1+R2
```

Figure 4. The GET_RELATIONSHIP Function.

B. The Ouput of the Multi-Liaison Algorithm

As per the algorithm discussed above, the output is shown below. In the first example, the outputs of the Stanford parse as well as the output of the Multi-Liaison both are displayed including the parse tree. In subsequent examples the parse tree is not displayed but the tagging, dependencies and the Multi-Liaison output is displayed. Fig. 2 displays the parse tree.

Example 1: The old beggar ran after the rich man who was wearing a black coat

The Stanford Parser output:

Tagging:

The/DT old/JJ beggar/NN ran/VBD after/IN the/DT rich/JJ man/NN who/WP was/VBD wearing/VBG a/DT black/JJ coat/NN

Parse Tree:

```
(ROOT
  (S
    (NP (DT The) (JJ old) (NN beggar))
    (VP (VBD ran)
      (PP (IN after)
        (NP
          (NP (DT the) (JJ rich) (NN man))
          (SBAR
            (WHNP (WP who))
            (S
              (VP (VBD was)
                (VP (VBG wearing)
                  (NP (DT a) (JJ black) (NN coat)))))))))))
```

Typed Dependencies:

det(beggar-3, The-1)
amod(beggar-3, old-2)
nsubj(ran-4, beggar-3)
det(man-8, the-6)
amod(man-8, rich-7)
prep_after(ran-4, man-8)
nsubj(wearing-11, man-8)
aux(wearing-11, was-10)
rcmod(man-8, wearing-11)
det(coat-14, a-12)
amod(coat-14, black-13)
dobj(wearing-11, coat-14)

The Multi-Liaison Output:

Subject: 1
NN beggar
Predicate: 3
VBD ran
VBD was
VBG wearing
Object: 2
NN man JJ rich
NN coat JJ black

Relationship:

beggar - ran - man
man - wearing - coat

Figure 5. Example 1.

As shown above, the Multi-Liaison Algorithm displays the relationship between the subject and object (beggar and man) as well as the relationship between the two objects (man and coat).

Example 2: The dog and the cat ran after the mouse and the mongoose

Tagging:

The/DT dog/NN and/CC the/DT cat/NN ran/VBD after/IN the/DT mouse/NN and/CC the/DT mongoose/NN

Typed Dependencies:

det(dog-2, The-1)
nsubj(ran-6, dog-2)
det(cat-5, the-4)
conj_and(dog-2, cat-5)
nsubj(ran-6, cat-5)
det(mouse-9, the-8)
prep_after(ran-6, mouse-9)
det(mongoose-12, the-11)
prep_after(ran-6, mongoose-12)
conj_and(mouse-9, mongoose-12)

The Multi-Liaison Output:
Subject: 2
NN dog
NN cat
Predicate: 1
VBD ran
Object: 2
NN mouse
NN mongoose

Relationship:
dog - ran – mouse - mongoose
cat - ran - mouse - mongoose

Figure 6. Example 2.

Example 3: Jack and I visited the zoo with our children

We have also considered pronoun as a subject and therefore we have got the relationship with 2 subjects in terms of noun and pronoun.

Tagging:
Jack/NNP and/CC I/PRP visited/VBD the/DT zoo/NN with/IN our/PRP$ children/NNS

Typed Dependencies:
nsubj(visited-4, Jack-1)
conj_and(Jack-1, I-3)
nsubj(visited-4, I-3)
det(zoo-6, the-5)
dobj(visited-4, zoo-6)
poss(children-9, our-8)
prep_with(visited-4, children-9)

The Multi-Liaison Output:
Subject: 2
NNP Jack
PRP I
Predicate: 1
VBD visited
Object: 2
NN zoo
NNS children
PRP$ our

Relationship:
Jack - visited - zoo - children
I - visited - zoo - children

Figure 7. Example 3.

All the three examples shown in Fig. 5, Fig. 6 and Fig. 7 have different number of subjects and objects and the relationship between them is also not similar. The Multi-Liaison Algorithm output in this way can be very useful for text mining applications where a variety of sentences are to be mined.

IV. Performance of Multi-Liaison Algorithm

The application was written in JAVA using Net Beans IDE 6.5 RC2. It parsed a single sentence of 12 words in 8.35 seconds and displayed the output as shown in the examples above.

This algorithm works equally well with simple as well as complex sentences and the output is very clear and precise.

V. Conclusion

In this paper we have presented an algorithm which displays the relationships between subjects and objects in sentences where there are multiple subjects and objects. The Stanford parser output was used to generate this result.

This algorithm would be usable not only by text mining experts and computational linguists but also by the computer science community more generally and by all sorts of professionals like biologists, medical researchers, political scientists, business and market analysts, etc. [10]. In fact it would be easy for users not necessarily versed in linguistics to see how to use and to get value from the simple relationship that is displayed so effectively.

This work can be carried forward for different text mining and natural language processing applications. In continuation to this we aim to use the Multi-Liaison Algorithm in text summarization and Information Retrieval.

Acknowledgment

We are grateful to our Departmental Heads and Research Guides for their timely suggestions and technical support. Moreover we are extremely thankful to all the authors mentioned below whose research papers we have referred. Their work has formed the foundation and backbone for us in developing this algorithm.

References

[1] D. Klein, C. D. Manning, “Fast exact inference with a factored model for natural language parsing” in Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10, 2003.
[2] D. Klein, C. D. Manning, “Accurate unlexicalized parsing” in Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430, 2003.
[3] Delia Rusu, Lorand Dali, Blaz Fortuna, Marko Grobelnik, Dunja Mladenic, “Triplet extraction from sentences” in Artificial Intelligence Laboratory, Jožef Stefan Institute, Slovenia, Nov. 7, 2008.
[4] D. Lin, P. Pantel, “DIRT - Discovery of inference rules from text” in Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001. pp. 323-328, 2001.
[5] J. Leskovec, M. Grobelnik, N. Milic-Frayling, “Learning sub-structures of document semantic graphs for document summarization” in Proceedings of the 7th International Multi-Conference Information Society IS 2004, Volume B. pp. 18-25, 2004.
[6] J. Leskovec, N. Milic-Frayling, M. Grobelnik, “Impact of linguistic analysis on the semantic graph coverage and learning of document extracts” in National Conference on Artificial Intelligence, 2005.
[7] O. Etzioni, M. Cafarella, D. Downey, A. M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A.Yates. Unsupervised named-entity extraction

from the Web: An experimental study. Artificial Intelligence, Volume 165, Issue 1, June 2005, Pages 91-134.

[8] Marie-Catherine de Marneffe, Bill MacCartney, Christopher D. Manning, "Generating typed dependency parses from phrase structure parses" in LREC 2006.

[9] Marie-Catherine de Marneffe, Christopher D. Manning, "The Stanford typed dependencies representation" in COLING Workshop on Cross-framework and Cross-domain Parser Evaluation, 2008.

[10] Marie-Catherine de Marneffe, Christopher D. Manning, "The Stanford typed dependencies manual" in Revised for Stanford Parser v1.6.2, February, 2010.

[11] Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, Christopher D. Manning, "Parsing to Stanford dependencies: Trade-offs between speed and accuracy" in 7th International Conference on Language Resources and Evaluation (LREC 2010).

[12] Dick Grune and Ceriel Jacobs, "Parsing Techniques – A Practical Guide," in Proceedings of the 8th International Conference, CICLing 2007, Mexico City, A. Gelbukh (Ed), pp. 311-324, Springer, Germany, 2007.

[13] Takale, S. A. (2010). Measuring Semantic Similarity between Words Using Web Documents. International Journal of Advanced Computer Science and Applications - IJACSA, 1(4), 78-85.

[14] Firdhous, M. F. M. (2010). Automating Legal Research through Data Mining. International Journal of Advanced Computer Science and Applications - IJACSA, 1(6), 9-16.

[15] Satapathy, S. K., & Mishra, S. (2010). Search Technique Using Wildcards or Truncation : A Tolerance Rough Set Clustering Approach. International Journal of Advanced Computer Science and Applications - IJACSA, 1(4), 73-77.

AUTHORS PROFILE

Ms. Anjali Ganesh Jivani is an Associate Professor in the Department of Computer Science and Engineering, The Maharaja Sayajirao University of Baroda. She is pursuing her doctorate in Computer Science and her research area is Text Mining. She has published a number of research papers related to Text Mining. Her paper titled "'The Shared Nearest Neighbor Algorithm with Enclosures (SNNAE)" has been published by IEEE Computer Society and World Research Organization, ISBN 978-0-7695-3507-4, DOI 10.1109/CSIE2009.997, pg. 436. The paper is available on the ACM DL portal. She has co-authored a book titled 'SQL & PL/SQL Practice Book', ISBN 978-81-910796-0-9.

Ms. Amisha Hetal Shingala is an Assitant Professor in the Department of MCA, Gujarat Technological University. She is pursuing her doctorate in Information Retrievel. She has published a number of papers. She has co-authored a book titled 'SQL & PL/SQL Practice Book', ISBN 978-81-910796-0-9.

Dr. Paresh V. Virparia is an Associate Professor in the Dept. of Computer Science, SP Universiy. He has done doctorater in Simulation and Modeling. He has guided 9 Ph.D. students.