KB Context Switching Algorithm for NLP

A.M.S.S.A.U.B. Attanayake
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka

Abstract—**Those who develop Natural Language Processing (NLP) systems sometimes find it convenient to develop several representations of knowledge. Especially in question and answer generating machines it is more logical to have several knowledge bases (KB) to answer each specific types of questions. This paper discusses a statistical learning based algorithm to port a specific question to a desired KB. The algorithm allows selection of KBs based on previously learnt patterns. Due to probabilistic parsing and POS (Part of Speech) tagging makes this algorithm much suitable for short questions or short input sentences.**

Index Terms—**Answer Generation, Chat Bot, Knowledge Representation, NLP.**

I. Introduction

NATURAL Language Processing (NLP) deals with inherently ambiguous human language. In most cases the knowledge bases (KB) or ontology used to represent the system's knowledge differs significantly from system to system. The differences can exist in the structure or the way the knowledge is traversed or how the knowledge is organized [1], [2]. Specific representation of knowledge allows optimization in terms of quality of the answer, convergence or in terms of computing resource usages like memory or CPU power. Concept net[1] is a classic example of how different knowledge elements can be bound together for easy traversal and reasoning.

In question and answer generating systems it is sometimes important to select a particular KB depending on the question being asked. For instance if the user asks "what is the meaning of fortification" the system can answer the question by easily referring to a dictionary. On the other hand if the question is "Who is the father of American president Bill Clinton" the question can be answered by searching the internet, encyclopedia or some other type of KB.

The format of a generated answer also depends on type of question being asked. For example the answer for the question "Where is Melbourne Australia" will look nice if you can show a map with the search results. Although it is possible to look for keywords like "where" and then show a map, current existing systems sometimes make mistakes; for example if you ask a question like "Where is beautiful sunny Melbourne".

Current systems sometimes ignore the word ordering and word context. For instance the question "Sunny Melboune, Where is it?" is much similar to the question "Where is Melbourne" and should return similar answers. Identifying similar sentences using context can be a very useful tool in NLP. Specially in learning based machines finding similarity between learned set and real input set is vital to find matching pairs.

II. Existing Methods For KB Switching

1) Popular method of selecting the end format or KB is looking for keywords and then doing the selection. This can have major drawbacks as some times "Where is Melbourne" can mean a title of a song instead of a place. The system can get even more confused if the raised question is "What is, where is Melbourne".

2) Usual POS tagged query context analysis in answer generating machines happens from start to end considering leaf match. This method prevents analyzing complex questions. For instance "where is the god's love" is same as "the god's love, where is it". Consider word break down tree in Fig. 1 & Fig. 2. (The word tags are in accordance with University of Pennsylvania (Penn) Treebank Tag Set[2]).

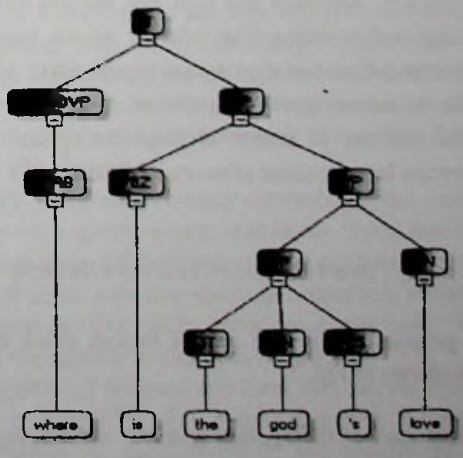


Fig. 1. Where is the god's love.

[1] http://conceptnet.media.mit.edu/

[2] http://www.cis.upenn.edu/~treebank/

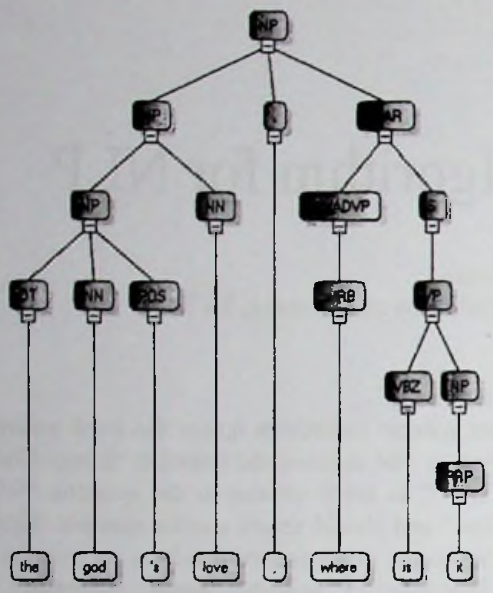


Fig. 2. the god's love , where is it.

Both trees are produced using OpenNLP[3] tool kit and statistical model is based on Wall Street Journal and Browns Corpus.

The POS tag sequence when represented as a string gives

a. where/WRB is/VBZ the/DT god/NN 's/POS love/NN
b. the/DT god/NN 's/POS love/NN ,/, where/WRB is/VBZ it/PRP

In both cases start to end leaves matching gives little similarity, but in reality they mean exactly the same.

3) Stop word filtering is another common method where noise words like "the", "a", "an" are first removed to extract more important words like proper nouns. The sentence "where is the god's love" will be converted to something like "where god love" before selecting the KB. Although this is a fast method suitable for high end systems like internet search engines, the method does not give a convergent result when used in an answer generating system. In most cases when the content of search is large the system tends to return large amount of unwanted data.

III. Sub tree based context Switch

The proposed algorithm can be broken down to several steps as shown in Fig. 3.

At step one POS tagging involves identifying part of speech for the given query. POS tagging is a well established research area and Bayesian statistics based POS taggers [3], [4] and maximum entropy models [5] are commonly used due to their relative speed and accuracy. Modern POS taggers have accuracy well over 80%.

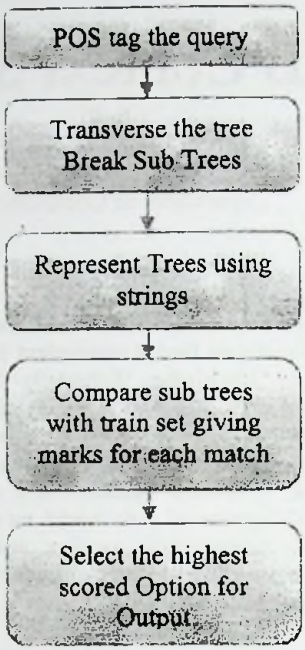


Fig. 3.

Step two involves breaking sub trees and a novel method of sub tree representation and storing. Instead of using predefined structures like linked list or some advance trees; KB switching simply uses strings Fig. 4. Modern programming languages have good impedance match with strings and fast look up using String.Contains("value") method. In most occasions the underlying platform assures only one memory location is allocated for a same type of string and internal indexing is done for quick retrieval. String interning [6] is a common method in almost all modern languages. (E.g. Java, .NET)

The super sub tree (sub tree containing all the sub trees) of earlier given sentence "Where is the god's love" will be converted into

(S (WHADVP (WRB where)) (VP (VBZ is) (NP (NP (DT the) (NN god) (POS 's)) (NN love))))

The bracketed method assures sub trees are preserved. Like the super sub tree each sub tree will be represented with a separate string and will be stored in an index form Fig. 4.

If the exact word is not important and only the word type is the concern you can replace the exact word with a tag like TOK (token). The new tag will turn the above string into

(S (WHADVP (WRB TOK)) (VP (VBZ TOK) (NP (NP (DT TOK) (NN TOK) (POS TOK)) (NN TOK))))

Saving only TOK reduces space required to store sub trees but gives poor separation between sentences. During tree traversal an algorithm like Depth First Search [7] can be used.

[3] http://opennlp.sourceforge.net/

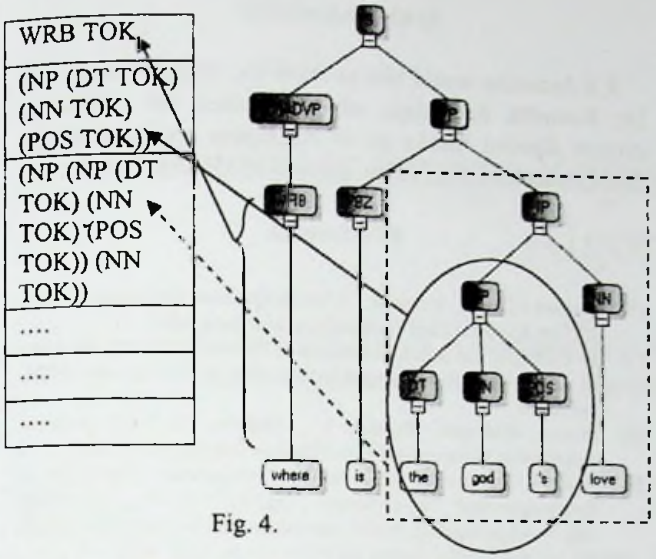


Fig. 4.

For example if there are three "choices" to be made depending on the context; the representation will be done as on Fig. 5. The choices can be either type of knowledge base, or final output format or some set of decision points. To preserve the generality the choices will be named A, B, C.

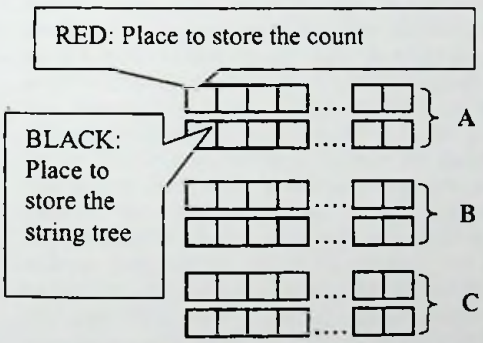
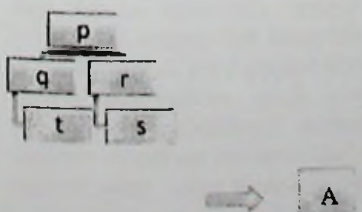


Fig. 5.

Note that although there are three black slots; in reality the internal design will assure only one merged set of black slots will be available to store the sub trees.

During training sessions sub trees will be stored in black slots along with their counts in red slots. When a previously known sub tree is encountered its count is incremented by one.

Consider Fig. 6 two train sets for selection of A.



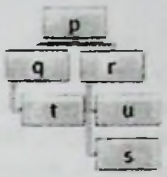


Fig. 6.

At the end of training following will be stored in A's slot TABLE 1.To reduce unwanted noise leaf nodes can be eliminated.

TABLE 1

Count	1	1	2	2	1	1
Tree	p((q(t)) (r(s)))	p((q(t)) (r((s)(u))))	q(t)	r(s)	r(u)	r((s)(u))

During real matching phase each input sub tree will be compared to a tree slot and the count is added to score the final result. For example if the input has only one q(t) and 4 r(s) and nothing else to match A's train set; A's score will be 1*2+4*2=10. Finally the option with the highest score is selected as the matching output. For instance if A's score is 10, B's score is 6 and C's score is 26; C will be selected as the preferred choice.

IV. Complexity Analysis

Due to statistical construction of the POS tag tree, the parse tree is not a well balanced one. This makes complexity analysis bit more had hoc. Assuming that the constructed tree is similar to a balanced binary tree the complexity analysis can be easily carried out. Note that major part of memory and CPU consumption happen in sub tree construction. Complexity of the POS tagging is not discussed as it depends on the method of POS tagging being used.

For a tree of V average vertexes and E average edges the running time of Depth First Search is $\Theta(V+E)$ [7] . The insertion of a constructed sub tree and retrieval depends on string saving and string.contains() functions in the underlying platform. String saving can be done in a constant time and usually the length of input string can be taken as a constant as this can be fixed, looking at max length (for say at length 200 characters).

Memory complexity mainly depends on the number of sub trees in a given query. Assuming there are no over lapping sub trees (if there are over lapping sub trees *intern string* will make sure less space is consumed, making such an assumption fine with worst case analysis) we can calculate the number of sub trees as follows.

The number of different sub trees that can be constructed at depth r is $2^r+2^{(r+1)}$. For a sentence with K number of average words (K divisible by 2, and can be taken as a constant if fixed at max length), the total number of sub trees including the root is $\sum_{r=1}^{r=log_2(K)} (2^r + 2^{r+1}) + 3$. The value $log_2(K)$ is the depth of the tree and the final 3 is the number of sub trees you get from the root.

After summation the number of sub trees is $3 * (2^{\log_2 K} - 1)$=3(K-1) giving a memory complexity for a single sentence as O(K). If there are N number of sentences in the train set total memory requirement is O(KN). Note that this gives a unique feature of having a memory requirement proportional to train set size. Taking off overlapping sub trees make it even less memory intensive.

Taking a real world data set in an extreme case; assuming 1byte is required to store a letter, average 8 letters per word, 16 words per sentence, size of the sentence is 1*8*16=128 bytes. The memory consumption for tree can be reduced if each POS tag is represented with a unique number 1,2,3....rather than NNP, VP etc. As the number of POS tags is less than 99 let's assume we represent each tag with a number 1,2,3... , so that a POS tag at a tree node requires only 2 bytes each. When the 16 word sentence forms a balance binary tree; bottom sub trees (at depth 3) require 16*(2+2+8) + 8*(1+8+8+6) =376 bytes (note that you have to set provision for brackets used to mark sub trees). The depth 2 sub trees require 4*(2+6+24*2)=224 bytes. At depth 1 the size of sub trees is 2*(2+6+56*2)=240 bytes. Similarly at depth 0 the size of the tree is 248 bytes. All together the string table requires 376+224+240+248=1088bytes. If the train set has 100,000 unique trees (which is very unlikely) total memory requirement is around 104MBs. This amount of memory can easily be managed with modern hardware.

The algorithm automatically counts for tree rotation changes and gives close marks for sentences where only difference is some sub tree rotation or parent mismatch.

V. Conclusion

Options (KBs, final output format etc) switching using string based sub tree method provides an easy to implement yet fast method of option selection. Memory requirement to store the train set is a linear function of the input (Forgetting over lapping sub trees) giving a good memory usage feature. The algorithm can be used not only in NLP but also in other situations where trees have to be compared for rotations and sub tree similarities. Due to use of statistic based POS tagging, the method is fast only for small sentences. The algorithm can be extended to dynamically select the path based on the path it has already traversed and previously learn path patterns.

Acknowledgment

S.S Aravinda would like to thank Dr. Shehan Perera and Dr. Ravindra Koggalage who supervised the AIEPmora project. Special thanks go to AIEPmora project team A. Maha Aracchchi A.I, Vithanagama S and Philips G.L.L.

References

[1] Richard J Cooper, Stefan M, "A Simple Question Answering System" *9th Text REtrieval Conf,* Gaithersburg, Maryland, 2000.

[2] H. Helbig, C. Gnorlich, "Multilayered Extended Semantic Networks as a Language for Meaning Representation in NLP Systems"ACM, published.

[3] Sharon Goldwater, Thomas. L . Griffiths, "A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging" unpublished.

[4] Ankit K Srivastava, (2008, March) "Unsupervised Approaches to ParUnsupervised Part-of-Speech Tagging" [Online]. Available: http://www.computing.dcu.ie/~asrivastava/docs/UnsupervisedPOS.pdf

[5] Joshua Goodman, Microsoft Research, "Sequential Conditional Generalized Iterative Scaling" Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, (2002, July) pp. 9-16.

[6] String Interning [Online]. Available: http://en.wikipedia.org/wiki/String_intern_pool

[7] T. H. Cormen, C. E. Leiserson, R. L Rivest, C. Stein, "Introduction to Algorithms" 2nd ed. Prentice Hall, India, New Delhi, 2003, pp. 540-547