ISSN: 2229-6956(ONLINE)
DOI: 10.21917/ijsc.2015.0143

TRAINING TREE ADJOINING GRAMMARS WITH HUGE TEXT CORPUS USING SPARK MAP REDUCE

Vijay Krishna Menon[1], S. Rajendran[2] and K.P. Soman[3]
Centre for Excellence in Computational Engineering and Networking, Amrita Vishwa Vidyapeetham, India
E-mail: [1]m_vijaykrishna@cb.amrita.edu, [2]rajushush@gmail.com, [3]kp_soman@amrita.edu

Abstract

Tree adjoining grammars (TAGs) are mildly context sensitive formalisms used mainly in modelling natural languages. Usage and research on these psycho linguistic formalisms have been erratic in the past decade, due to its demanding construction and difficulty to parse. However, they represent promising future for formalism based NLP in multilingual scenarios. In this paper we demonstrate basic synchronous Tree adjoining grammar for English-Tamil language pair that can be used readily for machine translation. We have also developed a multithreaded chart parser that gives ambiguous deep structures and a par dependency structure known as TAG derivation. Furthermore we then focus on a model for training this TAG for each language using a large corpus of text through a map reduce frequency count model in spark and estimation of various probabilistic parameters for the grammar trees thereafter; these parameters can be used to perform statistical parsing on the trained grammar.

Keywords:

TAGs, Spark, Probabilistic Grammar, RDDs, Parsing

1. INTRODUCTION

TAGs were proposed for language models earlier by Vijay-Shankar and Joshi in [4]. Unlike the Chomskian formalisms, the elementary objects manipulated by TAG are trees; structured objects and not strings. Such structured formalisms have properties that relate directly to strong generative capacity (structure descriptions), which is linguistically more relevant than string sets (weak generative capacity). So we call TAGs as a tree generating system rather than a string generating system. The set of all trees derived in a TAG constitute the object language. Hence, in order to describe the derivation of a tree in the object language, we will need to know about 'derivation trees'. The derivation trees are important in both syntactic and semantic senses. TAGs also have some interesting linguistic properties. Lexicalization is one of the key motivations for the study of TAGs, both linguistic and formal. The lexical phenomena now explain many linguistic theories previously thought to be purely syntactic. So the information in lexicons, have increased both in amount and complexity. From the formal perspective, lexicalization allows us to associate every elementary structure (trees) with a lexicon (any word). The famous Greibach Normal Form (also Chomsky Normal Form or CNF) for CFGs is a kind of lexicalization. However it is a weak lexicalization, as the structure of the original grammar is not preserved and all rules cannot be lexicalised. Thus TAGs provide an edge to this errand over conventional CFGs.

TAGs were introduced by Joshi [1] and later studied by Vijay-Shankar [2]. It is known that tree adjoining languages (TALs) generate some context sensitive languages and fall in the class of the so called 'mildly context sensitive' languages [2]. TALs properly contain context-free languages and are properly contained by indexed languages.

Formally, as mentioned in [2], A tree adjoining grammar (TAG), G consists of a quintuple (N, T, S, I, A) where,

1. T is a finite set of terminal symbols. N is a finite set of non-terminal symbols that $T \cup N = \phi$
2. S is a Sentential symbol such that such that.
3. I is a finite set of trees called initial trees, with the following properties
   a. Interior nodes are labelled by non-terminal symbols.
   b. The leaf nodes of all initial trees are labelled by terminals or non-terminals; non-terminals symbols on the frontier of any tree in I-set are marked for substitution conventionally using a down arrow (↓).
4. A is a finite set of trees called auxiliary trees, with the following properties:
   a. Non-leaf nodes are labelled by non-terminal symbols.
   b. The leaf nodes of auxiliary trees are labelled by terminal symbols or non-terminal symbols. Non-terminal symbol on the leaves in A-set are marked for substitution except for one node, called the foot node conventionally marked with an asterisk (*); foot node must be exactly identical to the root node.

The grammar generates parse trees which in turn yield sentences of the language, through an iterative and cross recursive parse process, involving multiple combinations of elementary trees through substitution and/or adjunction. The parsing and the nature of TAGs are discussed in a separate section in this paper.

When TAG grammar yields (generates) derived trees by derivation, the information to trace the history of such combination is not given. Unlike CFGs, the derived tree does not contain information as to which basic rules (in our case, elementary trees) were used to construct it. Hence we require a new object that gives us information regarding all operations and elementary trees used to build a derived tree. This structured object is called a derivation tree. It uniquely specifies what operation was used to combine which particular trees. Both adjunctions and substitutions are considered for derivation.

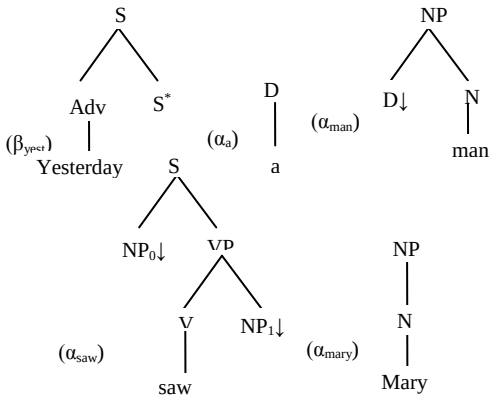

Fig.1. Lexicalised elementary tree for '*yesterday a man saw Mary*' adopted from [2].

2. DERIVATION STRUCTURES IN TAGS

Consider the example sentence "Yesterday a man saw Mary". This example has been adopted from [2]. One can imagine how the derived tree for the above English sentence might be. But this tree does not give any relevant information regarding how it can be constructed. For this we define the derivation tree for the same sentence. Figure 1illustrates the necessary elementary trees required to derive α_S. Note that α trees are initial trees and the β ones are auxiliary. This convention will be prevailing throughout this paper whenever referring to TAG trees.

Now the derivation tree for this example is shown in Fig.2. Along with exemplifying the process of building a derivation we also show how a proper lexicalization of TAG is achieved. All the elementary trees in the Fig.1 are properly and completely lexicalised with every elementary tree mapped to at least one lexicon. So every tree will have at least one anchor node.

The roots of all derivation trees are labelled by the name of an S-type initial tree. All child nodes are labelled by auxiliary trees which adjoined or initial trees which are substituted. The notion of tree address is used here to indicate where the composition happened. This will uniquely identify a node in a given tree. This address is referred to as the Gorn index; used for multiple array of purposes and is specifically important from an implementation point of view.

The Gorn index system starts with index 0 for the root node. For the 1st level children the numbering starts with 0.1 (or just 1) for the leftmost and increasing towards the right. For the 2nd level children say the child of the second leftmost child will be given 0.2.1 (or just 2.1) and so on. The system is simple and intuitive. Now if an adjunction takes place at this node of the tree, the derivation tree node labelled with the adjoining auxiliary tree will also carry the Gorn index 0.2.1, so we know exactly where the adjunction or substitution has occurred.

The case with α_{Mary} is no different, except that it is substituted at for node 0.2.2. But $\beta_{\text{yesterday}}$ is an auxiliary tree and is adjoined at the root node of α_{saw} as it contains the Gorn index pointing to the root. The main idea here is the Gorn indices given in a derivation tree's node, points to an address in its parent node's tree where the substitution or adjunction has been done. Further it also demonstrates how lower composition happens, like α_{a} substituted on α_{man}. Unlike as represented, substitutions need not be discriminated with dotted lines alone. The target node tree can solve the conflict by its type as in initial or auxiliary. Another counter intuitive fact is that adjoining happens even at the root node. But controlling adjunctions will help us control the grammars generative ability and restrict the constructs it creates. So every node in the derivation tree will have distinct indices for a given parent node. This way of representing derivation not only captures the syntactic structure of the target tree but also contains semantic dependencies. This has been demonstrated in Rambow and Joshi [6]; they were the first to investigate this property for TAG derivations. Later, in [7] they gave a dependency grammar based on TAG formalism. However we shall give a different picture of the same idea here. To illustrate this let us isolate the basic words of the above given example itself. Before we go into detail of this we will need to define dependency functions of each word with respect to the parts of speech (POS) of each word. Consider initially the verb saw. Now 'saw' is a transitive verb, so it will have dependencies in 2 ways, one with its subject and the other with the object. Hence the dependency function will look like this (basic argument structure):

$$f[d](\text{saw}) = \text{verb}[t](\text{sub}, \text{obj}) = \text{verb}[t](\text{man}, \text{Mary}) \qquad (1)$$

This show the dependencies of the transitive verb saw to depend on the subject as to who or what saw to the object as to saw whom or what.

This is exactly what we get in the derivation; "man saw Mary" giving us the dependency function for saw to be saw (Man, Mary). All the other words will have dependencies too as well. As for the Noun man the function is different and addresses the number or specificity. That means that nouns have articles or adjectives that describe them. This is their dependency. The above derivation also gives man(a) which is the dependency function for the word. The dependencies of a word can be easily found from the children of the given node in a derivation tree.

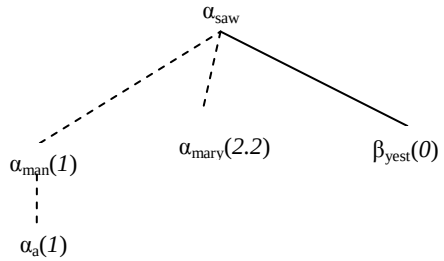

Fig.2. The Derivation tree for the example sentence in [2]

From the above insight, we must gather that saw in this example is not just transitive. That is to say it has a subject, an objects and an adverb. Thus the definition of the function should be having an extra parameter, one that specifies time in this case hence we have saw (Man, Mary, Yesterday). This property of TAG derivation greatly helps for representation of agglutinative languages, where the verbal inflection will depend on its subject or object or both. Subject verb agreements are crucial especially in Indian languages. Some tree maps are given below from our synchronous grammar.

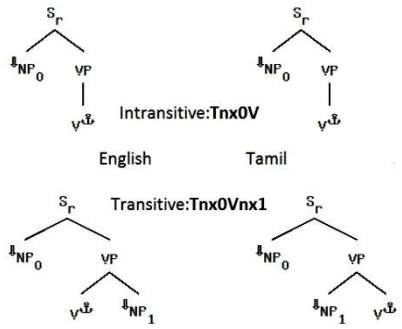

Fig.3. Synchronised TAG trees for English and Tamil [5]

3. TAMIL TAG AND PARSER IMPLEMENTATION

Tamil is a morph rich language, so to do pure syntax based dependency mining from it we will need to set aside the morphological considerations for the while and focus on the syntactic and psycho syntactic models. We have hand developed a Tamil TAG. Though its scope is quite restricted and tested mainly on tourism and health based corpora, it is effective enough for text book class sentences. Since such sentences only have limited or light dependencies, it might just prove to be insufficient for detailed analysis, however our attempt can be considered a step one into TAG based semantic analysis for Tamil language.

The Tamil TAGs were mainly created as part of a Machine Translation project, using synchronous TAGs. So these grammar trees are synchronised over a subset of XTAG English grammar. Efforts are being made for this to be expanded to a comprehensive grammar not just limited to Tamil. Unlike general XTAG trees we have designed single anchor trees; a grammar tree can be lexicalised only with one lexicon. This way we maintain a one to one relation between lexicons and derivation nodes so that the node represents only the dependency relation of that particular lexicon. Also note that we do not do any kind of statistical parsing or context based ranking of parses over the sentence. To fully observe the dependencies, all syntactically ambiguous parses are needed, so as to obtain different points of views and preserve the natural ambiguity. Before we observe the parses we need to describe the main aspects of the grammar. We have captured the following few main constructs of Tamil Language in this TAG.

1. Noun, Postpositions (morphemes), Conjunctions, Adverbs.
2. Recursive adjectives.
3. Basic Clefts (If clefts in a limited way).
4. Transitive Verb.
5. Intransitive Verb.
6. Ergative Verb.
7. PP complement.
8. PP small Clause.
9. Sentential Complement.
10. Sentential Subject.

These constructs are represented using elementary trees and auxiliary trees as was seen fit linguistically and by ease of parsing. The Parser is a multithreaded java implementation of the 'Earley Type TAG parsing' algorithm in [17] and [1]. It generates both parse trees and derivation trees over each and every ambiguous parse it can find from the grammar provided. Fig.3 and Fig.4 demonstrate the Tamil TAG trees as rendered by our viewer. The annotations for the nodes are consistent with the XTAG conventions except that all trees have single anchor node that houses the POS category the tree belongs to. The figures also show mappings between the English and Tamil trees as earlier mentioned. This however does not diminish the generative capacity of the Tamil grammar to independently parse and generate derivation trees on its own accord. Positively it helps to align the dependency with English like dependency and base it on the Stanford dependency set of 40 major semantic relations.

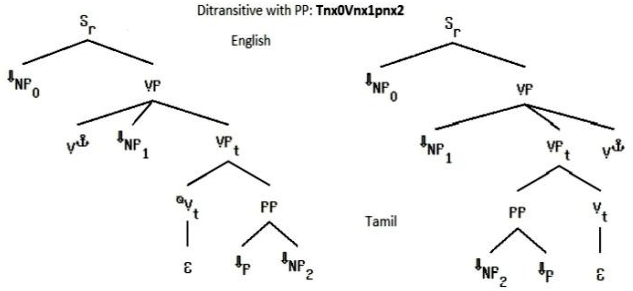

Fig.4. Further examples synchronous TAGs for Tamil

4. NATURE OF TAG PARSING

TAG parsing is considered by many computational linguists as a grey area in NLP. The main issue is the hardness of the problem itself. The parse tree for TAGs are not constructed in a predictable fashion like it is in CFGs. CFGs rely on a particular property called the valid prefix for its bottom up guided parsing. This means that given a half parse prefix of a sentence then any extension to that string can be parsed by extending the current reduction chart. This is an unlikely feature considering the complexity of TAG parsing as already shown by Vijay-Shankar in [3]. We still use it due to the interesting nature of its derivation construct as opposed to a complicated and counter intuitive parse algorithm. This also rules out the possibility of having an online parser for normal TAGs. However, there is an alternative formalism derived from TAGs called the Spinal-LTAG, which supports an incremental parsing algorithm overcoming the former claim, in [8]. But conversion to spinal formalism is again a formidable process and might prove detrimental to translation mechanism that we employ for normal TAGs and LTAGs.

There are three main algorithms that have been successfully used for parsing TAGs. The oldest is the one proposed by Vijay-Shankar [3]; a CYK type algorithm with obvious implementation difficulties. The second is the more practical and easily implementable Earley type parser, proposed by Shabes and Joshi [17].This is the one that we have adopted and modified as a multi-threaded prediction chart parser. The next one is proposed and implemented by Sarkar in [9], and is used in the XTAG system. This is a statistical parser and works with trained

LTAG trees only, using a head-corner algorithm for TAGs originally proposed in [10].

Statistical parsing for TAGs obviously has its advantages as it reduces the amount of symbolic computations done by the parser. Our grammar consists only of single anchor trees with only one site for lexicalisation per tree. The grammar was designed to facilitate machine translation and hence had to be a minimalistic and domain based one rather than a full coverage comprehensive grammar such as in the XTAG project. Our grammar generation process which is mostly linguistic in nature and the working of the parser has been already detailed in our previous works this year in [11]. The English grammar is evidently a pruned and minimised version of the XTAG English grammar, while the Tamil grammar we have developed is moderately synchronous with English. We will also use the statistical model in [9] for training this minimal grammar. However the estimation of the probabilistic parameters in our case is a bit different. This is the next main focus in this paper.

5. MODEL FOR PROBABILISTIC TAGs

TAGs are defined as a quintuple formalism, GTAG (*N, T, S, I, A*) where Nis the set of all non-terminals, *T* is the set of all terminal symbols S being a start symbol and $I \cup A$ is the set of all elementary trees such that they have nodes, n_i;

$$n_i = N \cup T, \forall i \neq 0.$$

We shall call this union as *V* the set of all nodes (vertices) in this grammar, called GTAG. Further the language accepted by GTAG will be $L_{TAG}(G)$. Let us consider a string *w* being parsed by this grammar so it is accepted.

i.e., $$w \in L_{TAG}(G).$$

Now GTAG is not a statistical grammar. So GTAG will give many ambiguous derivations of *w* which cannot be ordered in any way. What we are looking for is a quantisation of these sets of derivations to have a partial order. Probabilistic grammars are able to do the same by assigning probability parameters to grammar rules and associated operations. We shall do the same here in the most intuitive way. We can assign various parameters to this grammar by using just one function. Before we tweak this model, it is necessary to see the traditional model proposed by Shabes in [12] and refined by Sarkar in [9].

For any initial tree, $\tau \in I$ to be the root of a derivation we estimate the initial probability for it. $P_{INIT}(\tau)$. Consider a derivation $d_i \in D_w$ for *w*, where D_w is the set of all ambiguous derivations for *w*. For all the initial tress that can start any d_i, then total probability for initiation is exhaustive. i.e.:

$$\forall \tau \in I_w^{start} \subseteq I; \sum_{\tau} P_{INIT}(\tau) = 1 \quad (2)$$

In the course of derivation, combination of trees happens either by substitution or by adjunction. At this point we shall follow Sarkar's suggestion that these need not be distinguished and may be treated as a singular grammatical operation; hence we call them as attachment. Attachment is a ternary operation with 3 parameters, the parent tree, the node where the attachment is done and the child tree that has been attached. Attachment happens independently. An attachment of tree *t* onto node $n_j \in N$ is given as $n_j \mapsto t$. So we have:

$$\forall \tau' \in I_w \cup A_w \subseteq I \cup A; \sum_{\tau'} P_{ATTACH}(\tau, n \mapsto \tau') = 1 \quad (3)$$

For the completeness of the model we shall also have a complementing parameter for attachments where no attachment happens on a given node. This is called the null adjunction probability. Even though there is little training to be done on this parameter, without it the model would be incomplete.

$$P_{ATTACH}(\tau, n \mapsto \phi) \quad (4)$$

So combining this with the above generic attachment parameter, we shall rewrite it as a single relation.

$$\forall \tau' \in I_w \cup A_w \cup \phi \subseteq I \cup A; \quad \sum_{\tau'} P_{ATTACH}(\tau, n \mapsto \tau') = 1 \quad (5)$$

Derivation $d_i \in D_w$ of any string '*w*' can now be assigned a probability computation, and hence so doing can rank the best derivation for *w* in D_w.

$$\Pr(D, w_0, \ldots w_n) = P_{INIT}(\tau_\alpha, w_i) \times \prod_{k \leq n, k \neq i}^{0} P_{ATTACH}(\tau, \eta \mapsto \tau_k, w_k) \times \prod_{m-n} P_{ATTACH}(\tau, n \mapsto \phi) \quad (6)$$

We shall not deliberate on the various consistency aspects of this model as that deviates from the main focus of this paper; more details on the consistency conditions of probabilistic TAGs are available in [9], [12] and [13]. However we promised to redefine TAGs with a simple tweak of this above model maintaining its completeness; we define probabilistic TAG, GPTAG as a sextuple (*N, T, S, I, A,* ψ) where ψ is the probability of attachment, given as a triple relation

$$R_\varphi = (I \cup A) \times N \times (I \cup A). \quad (7)$$

This model caters to all probability parameters discussed before. Now for any Initial tree that starts derivation on a string w can now be stated to have:

$$\forall \tau \in I_w^{start} \subseteq I; P_{INIT}(\tau) = \varphi(\phi, \phi, \tau) \quad (8)$$

And for any adjunction or substitution of trees it will capture the attachment parameter likewise as follows:

$$\forall \tau' \in I_w \cup A_w \cup \phi \subseteq I \cup A; \quad P_{ATTACH}(\tau, n \mapsto \tau'_k, w_k) = \varphi(\tau, n, \tau'_k) \quad (9)$$

We will use simple bi-gram like relations to estimate all parameter. Like a bigram model any attachment and initiation probability comes from counting usage frequencies. We shall design an experiment that uses textual big data processing to incrementally estimate such statistical parameters for a single TAG grammar. We have also considered the destructive probability assignments and dirty grammar problems mentioned by Booch and Thomson via [9]. The solutions to these lie in continuous refinement of the model estimates and retuning of the grammar.

6. EXPERIMENT DESIGN AND PARAMETER ESTIMATION

In order to have a statistical grammar, it is pivotal to have syntax annotated corpus of the language samples. Since we have fewer or no such resource especially for most Indian languages, we need to do this innovatively. Building the Tamil TAG and synchronising it with a minimalistic English TAG was the initial step towards this task. The estimation itself is a continuous and self-refining process and need to be performed incrementally.

We devise the use of our existing non-statistical parser to yield complete derivations of the huge batch of sentences and use them as a corpus for estimation of parameters. We perceive this task as a Big Data problem that can be resolved using huge monolingual unsupervised corpus of text. Since we would like to keep the experiment simple we have chosen sentences from textbook domain and Wikipedia, to be our initial corpus. We will use spark core engine to distribute all map-reduce processes that we will be deploying. The benefit is obvious as we have mentioned that our training requires to be done incrementally; resilient distributed data sets (RDDs) in spark are persistent data holders that can be reused and piped to batches of map-reduce processes and need to be built only once. This is precisely our requirement so that we can improve the model without repeating the entire training. This can also be done on streaming basis. This justifies our perspective that it is a contemporary Big Data problem.

We shall start by proposing RDDs holding tree frequencies from a collection of derivations. The text is held in either a Kafka queue or in any distributed file system within the cluster. A phrase processor splits sentences, POS tags them and pre-processes them such as morph analysis and / or stemming as it is required by the language. The processed and tagged sentence is fed to the parser. The parser yields ambiguous derivations and forwards them as an RDD to a module that counts the grammatical operations and lexicalizations, called the derivation mapper; this is the first map process that is spawned within the cluster. This map generates lot of key value pairs where the values are frequency counts and keys are tree pairs. For the time being we can ignore the node relations in the attachment operations. In fact this process creates multiple frequency counts. Attachment and usage counts are kept as separate key value pairs or Mapped RDDs. A collection of such RDDs can be reduced to form a flattened RDD with cumulative frequency counts. This process can be repeated innumerate number of times so as to incrementally modify the frequency counts as more and more data arrives. This is an embedded advantage of using spark RDDs instead of normal Hadoop map-reduce. The Fig.5 illustrates various RDDs and transformation.

The final RDDs in the lineage are subscripted as RDD1 and RDD2. When these RDDs are partially reduced (Shuffled, Reduce by Key Transform) they yield 2 sets of important parameters; the counts of usage of individual trees and the counts of attachments made as parent-node–child tree pair. For maintaining simplicity in the initial estimation, we drop the node relation and simply estimate the parent-child tree frequency irrespective of the node. However when considering the attachment probability, the node matching heuristic will see to it that destructive probabilities do not interfere in this case.

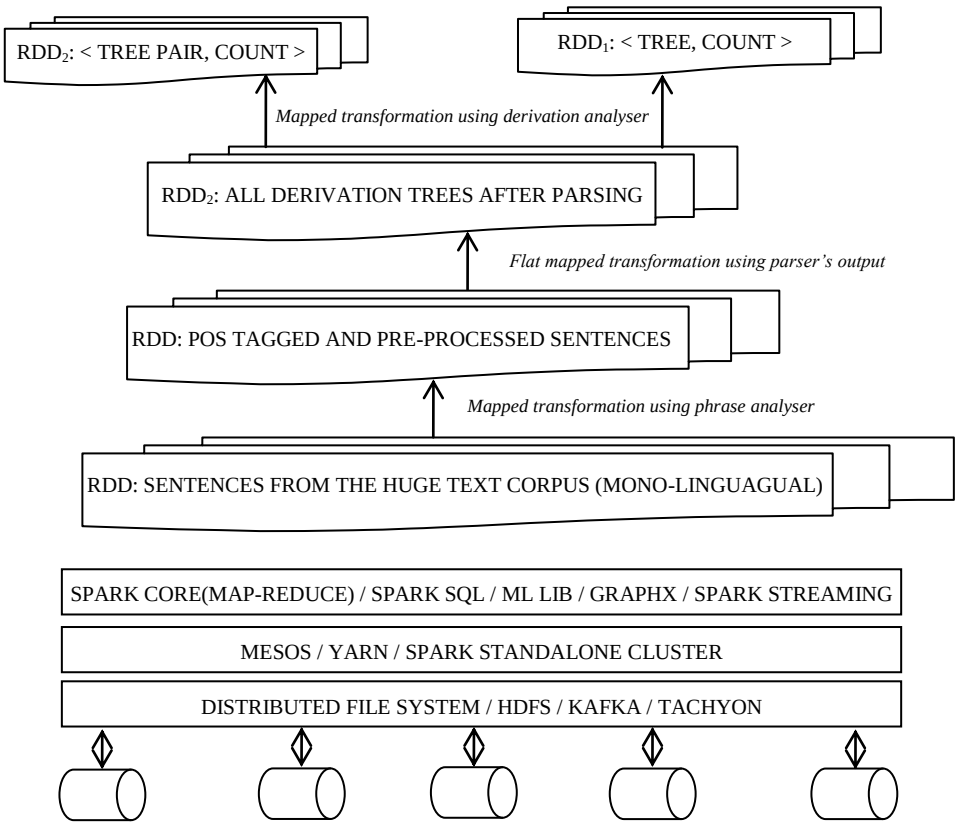

Fig.5. RDDs counting out tree frequencies for estimation of parameters

In case of initial probability of a tree to start derivations the reduce on the key value 'null-x' for RDD2 is used as the numerator for a tree 'x' over the total count of 'null-?' for all derivations. Similarly for each attachment pair too we can find the frequency and divide by the total occurrence to yield a basic probability. The advantage must be clear that these assignments of basic 'init' and 'attach' can be calculated from massive data in a distributed mode. Although the measure is complete, it need not always be consistent. This requires many runs to refine the estimate over multiple training epochs and some convenient heuristics

7. CONCLUSION AND FUTURE WORK

This model of PTAGs is previously verified for consistency and completeness in many previous works. We have simply devised a Big Data approach to parameter estimation for the same model. We have found this to be sufficient enough for the single anchor TAGs that we use mainly for machine translation of language. Currently the estimate is based on unlabelled mono lingual data. Ours' is a synchronous grammar. This approach however does not help to model the transfer statistics. That will be the scope of another publication as there are many parameters that need consideration. In order to maintain the focus on the basic estimation we have omitted all discussion on how such synchronous grammar can be trained.

Our primary objective for achieving a probabilistic grammar is to impart a way to quantise derivations that are too many and too ambiguous. The advantage of statistical parsing is secondary in our list of concerns. We must be able to not only quantise but qualify the estimates as being non-destructive in nature. This is a diligent process and might take time beyond our estimation. Similarly the problem of unreachable tree can also be avoided and grammar can be refined using these measures. We have also omitted lexicalisation parameters for the time being as we are not dealing with full LTAG.

REFERENCES

[1] Aravind K Joshi, Leon S Levy and Masako Takahashi, "Tree Adjunct Grammars", *Journal of Computer and System Science*, Vol. 10, No. 1, pp. 136-163, 1975.

[2] Aravind K. Joshi and Yves Schabes, "Tree-adjoining grammars", *Handbook of formal languages*, Vol. 3, pp. 69-123, 1997.

[3] K. Vijay-Shanker, "A Study of Tree Adjoining Grammars", Ph.D. diss., Department of Computer of Science, University of Pennsylvania, 1987.

[4] K. Vijay-Shankar and Aravind K. Joshi, "Some Computational Properties of Tree Adjoining Grammars", *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pp. 82-93, 1985.

[5] Vijay Krishna Menon, Rajendran S, Anandkumar M, and Soman K P, "Dependency resolution and semantic mining using Tree Adjoining Grammars for Tamil Language", *International Conference on Tamil language computing and Tamil Internet*, pp. 337-344, 2015.

[6] Owen Rambow and Aravind Joshi, "A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena", In: L. Wanner, (ed.) "Recent Trends in Meaning-Text Theory. Studies in Language", *Companion Series*, Vol. 39, pp. 167–190, 1997.

[7] Aravind Joshi and Owen Rambow, "A formalism for dependency grammar based on tree adjoining grammar", *Proceedings of the Conference on Meaning-Text Theory*, pp. 207-216, 2003.

[8] Libin Shen, Lucas Champollion and Aravind K. Joshi, "LTAG-spinal and the Treebank", *Language Resources and Evaluation*, Vol. 42, No. 1, pp. 1-19, 2008.

[9] Anoop Sarkar, "Statistical Parsing Algorithms for Lexicalized Tree Adjoining Grammars", Ph.D Thesis, University of Pennsylvania, 2002.

[10] Gertjan Van Noord, "Head Corner Parsing for TAG", *Computational Intelligence*, Vol. 10, No. 4, pp. 525-534, 1994.

[11] Vijay Krishna Menon, S. Rajendran and Soman K P, "A Synchronised Tree Adjoining Grammar for English to Tamil Machine Translation," 4th International Symposium on Natural Language Processing (NLP'15) co affiliated with International Conference on Advances in Computing, Communications and Informatics, 2015 (in press).

[12] Yves Schabes, "Stochastic tree-adjoining grammars", *Proceedings of the workshop on Speech and Natural Language*, pp. 140-145, 1992.

[13] Rebecca Nesson and Stuart M. Shieber, "Extraction phenomena in synchronous TAG syntax and semantics", *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pp. 9-16, 2007.

[14] Vijay Krishna Menon, "English to Indian Languages Machine Translation using LTAG", Master's Thesis, Amrita Vishwa Vidyapeetham University, 2008.

[15] XTAG Group, "A lexicalized tree adjoining grammar for English", Technical Report, IRCS-01-03, Institute for Research in Cognitive Science, University of Pennsylvania 2001.

[16] William Schuler, "Preserving semantic dependencies in synchronous Tree Adjoining Grammar", *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 88-95, 1999.

[17] Yves Schabes and Aravind K. Joshi, "An Earley-type parsing algorithm for Tree Adjoining Grammars", *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pp. 258-269, 1988.