Optimal Code Length Based Cost for Unsupervised Grammar Induction

Rohit J. Kate
Department of Health Informatics and Administration
Department of Computer Science
University of Wisconsin-Milwaukee, USA
katerj@uwm.edu

Abstract

An effective grammar can be induced from natural language sentences by simultaneously minimizing the cost of encoding the grammar and the cost of encoding the corresponding derivations of the sentences. In previous work, the cost of encoding a derivation was computed in terms of the number of bits it requires to encode which of the possible productions is used to expand each of its non-terminals. However, this ignored the fact that if some productions are used more often than others in the derivations, then they could be encoded with fewer bits using optimal code length based encoding. This paper presents a new derivation cost that uses such an encoding and applies it for inducing grammars. Minimizing this new derivation cost also corresponds to maximizing the probability of the derivations. Thus besides being theoretically more appealing, experimental results on sentences from clinical reports show that this new derivation cost also leads to induction of grammars that have better parsing performance.

Keywords: Grammar induction, unsupervised learning, optimal code length, clinical reports.

1 INTRODUCTION

In several natural language processing tasks, it is useful to first obtain syntactic parses of sentences based on the grammar of the language. The syntactic parsers and the grammar are typically learned automatically from supervised data of several thousand sentences manually annotated with syntactic parses by linguists [1]. However, creating such a supervised data requires a lot of manual effort by trained linguists and this effort is required every time the parser needs to be ported to any other domain which has different types of sentences. An alternate to learning from supervised data is to learn syntactic parser and the underlying grammar from unsupervised data of only sentences with no annotations, which is very easy to obtain for any domain. Unsupervised grammar induction has an additional advantage that it may induce an appropriate grammar best suited for the particular natural language processing application at hand as opposed to using the grammar that was pre-determined by the linguists for annotations.

Several methods for unsupervised syntactic parsing have been proposed, however, most of them only give bracketing information [2, 3] for the parses and do not actually induce the grammar and hence do not show the parses according to the induced grammar. Some other methods force the user to specify the number of non-terminals in the grammar [4] or restrict the induced grammar to binary form [4, 5]. The simplicity bias or cost reduction method for unsupervised grammar induction [6, 7, 8] does not have these drawbacks and is capable of inducing grammars without any restriction. This method starts with a trivial grammar in which there is a separate production for every sentence in the data. Then it iteratively searches for a better grammar by applying grammar transformation operators that modify the grammar. The search is guided by the cost reduction principle, according to which the best grammar will be the one which requires least number of bits to encode it as well as to encode the corresponding sentence derivations. Previous work has shown that this method is capable of inducing good grammars [6, 7], including for the clinical report sublanguage [8].

To apply the cost reduction method for inducing grammar, the cost of a grammar was computed in the search procedure by counting the number of bits it requires to encode all the symbols in the grammar (terminals and non-terminals) and then correspondingly to encode all the productions in the grammar. The cost of each derivation was computed by counting the number of bits it requires to encode the information regarding which of the possible productions is used to expand each of the non-terminals in the derivation. In this latter cost, no distinction was made between different productions that expand the same non-terminal and hence equal number of bits were counted no matter which production is used to expand a non-terminal in the derivation. However, the cost of this encoding can be reduced if fewer bits are used for a production which is used more often to expand a non-terminal and more bits are used if it is used less often, in accordance with the optimal code length principle [9, 10]. This paper presents a new derivation cost that makes use of optimal code lengths for encoding the derivations and uses it for unsupervised grammar induction. The paper also points out that minimizing this encoding cost of derivations corresponds to maximizing the probability of the derivations. Experimental results show improvement in the accuracy of parsing sentences from clinical reports [8] when the new derivation cost is used for inducing grammar.

2 BACKGROUND

This section provides some background on unsupervised grammar induction, the cost reduction method for it and optimal code lengths. The next section will describe the optimal code length based new derivation cost for doing unsupervised grammar induction.

2.1 Unsupervised Grammar Induction

Given several natural language sentences, say in English or in any other natural language, the task of unsupervised grammar induction is to automatically induce a grammar that can generate the given sentences as well as other well-formed sentences from the same language, but however, does not generate any ill-formed sentences. As a very small example, given the sentences “the cat saw a mouse”, “the mouse heard a cat” and “the dog chased the cat”, the induced grammar should be able to generate these as well as other sentences like “the cat heard a dog” and “the cat chased the mouse” etc., however, it should not generate sentences like “heard the dog cat” or “mouse the the cat saw” etc. The grammar induction process is called *unsupervised* if no other information about the sentences is given. If the sentences are already manually annotated with grammatical parses or some related information, then the grammar induction process is called *supervised.*

For the small example of sentences given before, an example of a good grammar that an unsupervised grammar induction process could induce will have terminals “cat”, “mouse”, “dog”, “saw”, “heard”, “chased”, “the” and “a”. The non-terminals could be “N”, “V”, “DET”, “NP”, “VP” and “S”, where N represents nouns (“cat”, “mouse” and “dog”), V represents verbs (“saw”, “heard” and “chased”), DET represents determiners (“a” and “the”), NP represents noun phrases (“a mouse”, “the cat” etc.) and VP represents verb phrases (“saw a mouse”, “chased the cat”, etc.). Finally S is the start symbol of the grammar representing entire sentences. The productions of a good example grammar could be “S→ NP VP”, “NP → DET N” and “VP → V NP”. Using this grammar, the *derivation* or *syntactic parse* of the sentence “the cat saw a mouse” could thus be represented in the following tree or labeled-bracketing structure: (S (NP (DET the) (N cat)) (VP (V saw) (NP (DET a) (N mouse)))). Given an induced grammar, it is straightforward to obtain a derivation like this for a sentence if the grammar generates it. Such syntactic parses are useful in analyzing sentences automatically for several natural language processing tasks. Note that this grammar will also generate other unseen sentences like “the cat heard a dog”, but will not generate an ill-formed sentence like “heard the dog cat”.

The above example of grammar induction was very small, but in general, a system may have to automatically induce grammar from thousands of sentences as input where the sentences could be of much longer lengths. Instead of unsupervised grammar induction, most of the work in this area

has been for directly doing *unsupervised syntactic parsing* in which the system directly generates syntactic parses for sentences without inducing any grammar [2, 3]. For example, these systems may generate the parse "((the cat) (saw (a mouse)))" for the sentence "the cat saw a mouse", without labeling the phrases. While this information may be good for some applications, for many other applications it is good to also know the labels as well as the underlying grammar. A few other approaches induce grammar but force it to be of a particular form, like maximum two non-terminals on the RHS of productions [4, 5] or some maximum number of non-terminals in the entire grammar [4]. The next subsection describes a method for unsupervised grammar induction that induces grammar without any such restrictions.

2.2 Cost Reduction Method for Unsupervised Grammar Induction

Several grammars can generate the same set of given sentences. For example, for the three sentences shown at beginning of the previous subsection, a trivial grammar with three productions "S → the cat saw a mouse", "S → the mouse heard a cat" and "S→ the dog chased the cat" will also generate those three sentences. This grammar, however, is overly-specific and will not be able to generate any other sentence, like "the cat heard a dog". Another trivial grammar can have the productions "S → S S", "S→ the", "S → cat", "S→ saw" etc., i.e. S expanding to two Ss as well as to every terminal. While this grammar will generate those three sentences as well as other unseen sentences, it is overly-general and will also generate almost any sentence with those words, including ill-formed sentences like "heard the dog cat" etc. Hence the best grammar is in-between these two extremes.

The cost reduction method [6], also known as simplicity bias method or minimum description length method, is based on the idea of language and data compression [11]. The basic assumption is that the smallest grammar that also leads to smallest derivations of the sentences is likely to be the best grammar in terms of neither being too general nor too specific. This method starts with a trivial grammar in which there is a production corresponding to every sentence and then heuristically searches for a smaller grammar by repeatedly applying grammar transformation operators of combining and merging non-terminals. The size of the grammar and derivations is measured in term of the encoding cost using ideas from information theory. The grammar is essentially viewed as a means to compress description of the given set of sentences, the more the compression the better the grammar. The encoding cost is measured in bits and consists of two components: the encoding cost of the grammar itself and the encoding cost of the derivations given the grammar. These are briefly described next using the notation from [7, 8].

Cost of Grammar: The cost C_P of encoding a production P of the form A → β, where A is a non-terminal and β is a non-empty sequence of terminals and non-terminals, will be: $C_P = (1+|\beta|)\log(|\Sigma|)$, where $|\beta|$ is the length of the right-hand-side (RHS) of the production and $|\Sigma|$ is the total number of terminals and non-terminals in the symbol set Σ of the grammar. This is because it takes $\log(|\Sigma|)$ bits to encode each symbol in the grammar and the production has $(1 + |\beta|)$ symbols, $|\beta|$ symbols on the RHS and one symbol on the left-hand-side (LHS). Hence the cost C_G of encoding an entire grammar G will be:

$$C_G = \sum_{i=1}^{p} (1+ |\beta_i|) \log(|\Sigma|)$$

where *p* is the number of productions in the grammar and β_i is the RHS of the *i*th production.

Cost of Derivations: Given the productions of the grammar, the derivation or generation of a sentence proceeds by expanding the available non-terminals in the current sequence, starting with the sequence of only the start symbol and ending when there is no non-terminal left in the sequence. Hence at each step in the derivation process, a decision needs to be made about which production is be used to expand a non-terminal. Out of all the productions which have that non-terminal on its LHS, one production is chosen. Hence this is the only information what needs to be

encoded at every step in the derivation process to encode the derivation of the entire sentence. If s_k is the non-terminal that is expanded at the *k*th step in the derivation process, and $P(s_k)$ is the set of productions that have s_k on their LHS, then $\log(|P(s_k)|)$ bits will be needed to encode which of those $|P(s_k)|$ number of productions is to be used to expand s_k. As an example, if there is only one production in the grammar that expands a non-terminal, then it is obvious which production will be used to expand it, and hence it will require log(1) or zero bits to encode that information. Equal number of $\log(|P(s_k)|)$ bits are used to encode this information no matter which production is used. In the next section, we present a better encoding cost for this, but this is what was used in the previous work [6, 7, 8]. Using this encoding cost for every derivation step, the cost to encode an entire derivation D_j of the *j*th sentence will be:

$$C_{D_j} = \sum_{k=1}^{m_j} \log(|P(s_k)|)$$

where m_j is the number of steps needed in the derivation of the *j*th sentence. Thus the cost C_D of encoding all *q* given sentences will be:

$$C_D = \sum_{j=1}^{q} \sum_{k=1}^{m_j} \log(|P(s_k)|)$$

Total Cost: In [6, 7], the total encoding cost C of both the grammar and the derivations was taken to be simply the sum of the two separate encoding costs C_G and C_D. Instead of weighing them equally, in [8], a parameter *f*, that takes value between 0 and 1, was introduced to relatively weigh the two components to determine the total cost. The best performance was achieved when the two components were weighed unequally using this parameter. Hence in this paper also the total cost C is defined using this parameter as:

$$C = f * C_G + (1 - f) C_D$$

where C_G and C_D are the grammar and derivation encoding costs respectively as defined earlier.

Grammar Search for Minimum Cost: It should be pointed out that there is a trade-off between the encoding cost of the grammar and the encoding cost of the derivations. For example, the overly-specific trivial grammar, which has a separate production for every given sentence, will have very little derivation cost because of the small derivations, but will have a large grammar cost because each production will be long in length. In contrast, the overly-general trivial grammar mentioned at the beginning of Subsection 2.2, which has a separate production for every word, will have very little grammar cost because of the small productions, but will lead to large derivations and hence a large derivation cost. Thus the best grammar will be in between the two extremes and will have the least total grammar and derivation cost.

Using the total cost as defined in the previous equation, the unsupervised grammar induction process [6, 8] starts with overly-specific trivial grammar in which there is a separate production for every sentence and then greedily applies one of the possible instances of grammar transformation operators that reduces the total cost by the largest amount. An operator transforms the grammar as well as appropriately changes the derivations. This process continues till no reduction in cost can be obtained. Two types of grammar transformation operators were used: *combine* and *merge*. The *combine* operator combines two or more non-terminals to form a new non-terminal. For example it may combine "V NP" to form "VP", by introducing the production "VP → V NP". Of course, the name of the new terminal will be a system-generated name instead of "VP". Although it introduces a new production, this operator may actually reduce the cost of the grammar if "V NP" appears too often in the grammar. It has no effect on the cost of the derivations because there will be only one way to expand the new non-terminal. The *merge* operator merges two non-terminals into one non-terminal, for example, it may replace all instances of NP and N with the new non-terminal M. This operator leads to merging of productions which are same except for the non-terminals being merged. Merging of productions reduces the size of the grammar as well as reduces how many ways a non-terminals can be expanded thus also reducing the derivation cost. However, if there are

productions that have either of the two non-terminals on their LHS, then merging them, in fact, increases the number of ways the merged non-terminal could be expanded and could thus increase the derivation cost. The reader may refer to [8] for more details about these two operators and also for the implementation details.

2.3 Optimal Code Length

According to information theory [10], given *n* possible symbols to send over an information channel, it would require log(*n*) bits to encode each of those symbols. However, if some of the symbols are more frequent than others, then the total number of bits sent over the channel could be reduced using optimal length encodings. According to Shannon's source coding theorem [9, 10], the optimal code length for a symbol *s*, which will lead to the least number of bits sent over the channel, is –log(P(*s*)), where P(*s*) is the probability of the symbol *s*. This encoding will assign fewer bits (shorter codes) for more frequent symbols and consequently more bits (longer codes) for less frequent ones. Overall, this reduces the number of bits sent over the channel, or if viewed differently, this leads to a better data compression. The next section describes use of this encoding method to improve the derivation cost.

3 OPTIMAL CODE-LENGTH-BASED DERIVATION COST

Subsection 2.2 described a cost of derivations that was used in the previous work [6, 7, 8]. At *k*th step in the derivation process, a decision needs to be made about which production will be used to expand the non-terminal s_k. If there are $P(s_k)$ productions in the grammar that expand s_k, then this information was encoded using $\log(|P(s_k)|)$ bits equally irrespective of which of the $P(s_k)$ productions was actually used. As outlined in Subsection 2.3, if fewer bits are used to encode a production that is used more often, then it will lead to an overall saving in the number of bits needed to encode each derivation step and hence to encode an entire derivation. This is a better estimate of the real cost of encoding a derivation because it uses an optimal encoding.

Suppose ${}^{sk}p_1$, ${}^{sk}p_2$, … , ${}^{sk}p_n$ are the *n* productions in the grammar that expand the non-terminal s_k, i.e. they have s_k on their LHS, and if N(p) denotes the number of times a production p is used in derivations of all the sentences, then the maximum likelihood probability $P({}^{sk}p_{rk})$, that a production ${}^{sk}p_{rk}$ will be used to expand s_k in a derivation, will be:

$$P({}^{s_k}p_{r_k}) = \frac{N({}^{s_k}p_{r_k})}{\sum_{i=i}^{n} N({}^{s_k}p_i)}$$

Hence the number of bits that should be used to encode the information that the production ${}^{r}p_{sk}$ was used to expand non-terminal s_k, i.e. the optimal encoding cost $C({}^{sk}p_{rk})$ will be:

$$C({}^{s_k}p_{r_k}) = -\log(P({}^{s_k}p_{r_k}))$$

Accordingly, analogous to the costs of derivations shown in Subsection 2.2, the new cost C_{Dj} of encoding the *j*th derivation and the new cost C_D of encoding all the derivations will be:

$$C_{D_j} = \sum_{k=1}^{m_j} -\log(P({}^{s_k}p_{r_k}))$$

$$C_D = \sum_{j=1}^{q}\sum_{k=1}^{m_j} -\log(P({}^{s_k}p_{r_k}))$$

where m_j is the number of steps needed in the derivation of the *j*th sentence, and there are total *q* sentences. The total cost C of grammar and derivations can be computed by substituting this new derivation cost C_D in the equation of the total cost in Subsection 2.2.

It is important to point out that because summing logs of probabilities is equivalent to taking products of probabilities, hence the cost of derivations in the above equations is, in fact, related to the combined probability of all the derivations, where probability of a derivation is product of the probabilities of its productions. Note that the way probability of a production has been defined here is exactly same as it is defined in a standard probabilistic context-free grammar [1]. Thus searching a grammar and correspondingly the derivations while minimizing the new derivation cost as defined above, is also equivalent to finding derivations of maximum probability. Hence the search for the minimum encoding cost of grammar and derivations is also a search for a grammar that would lead to maximum probability derivations. In effect, the new derivation cost when used in the grammar induction process will prefer grammars in which productions are used more often in the derivations, i.e. productions have high probabilities, as opposed to grammars in which productions are used less often, i.e. they have low probabilities.

4 EXPERIMENTS

This section presents experimental results of unsupervised grammar induction comparing the new derivation cost to the earlier derivation cost.

4.1 Methodology

The same dataset and the same experimental methodology was used in the experiments as was used in [8]. To create the dataset, first 5000 sentences were taken from the clinical discharge summaries section of the Pittsburgh corpus [12] using Stanford CoreNLP's[1] sentence segmentation utility. Clinical reports were chosen as the domain because natural language processing in the clinical reports have several important applications, and to our best knowledge, unsupervised grammar induction has not been applied to this domain with the exception of [8].

For the experiments, unsupervised grammar induction was not done directly on the raw sentences because due to the large vocabulary size it is difficult to directly induce grammars using the words themselves. Instead, it was done on sentences in which the words were replaced by their part-of-speech tags (for example, noun, verb, det, aux, prep, conj etc.) and UMLS [13] semantic types (for example, medical_device, biologicall_active_substance etc.) using MetaMap [14]. All the 5000 sentences were chosen to have maximum length of 20 words because MetaMap appeared to run endlessly on longer sentences. Since many UMLS semantic types are very fine-grained, only 27 of them were chosen which seemed relevant for clinical reports. Using MetaMap, all the occurrences of these semantic types were substituted for the actual words and phrases in the sentences, the words which were not part of any semantic type were substituted by their part-of-speech tags. The experiments were run using the dataset of sentences transformed into this form. It may be noted that the accuracy of this dataset is limited to the accuracy of MetaMap. As in [8], hard rules were introduced to never allow verb, det, prep, aux and conj part-of-speech tags to be the second or latter RHS terms in any production in the grammar induction process. These rules were introduced to prevent some obvious errors in the induction process. These biases could be learned in future from a small amount of supervised data in a semi-supervised grammar induction setting.

Out of the 5000 sentences, 4000 were separated to be used as training sentences to induce grammar. The remaining 1000 were used to test how well the induced grammar works on novel sentences, i.e. whether the induced grammar is able to parse them or not. Ealrey's context-free grammar parsing algorithm [15] was used to obtain parses of the sentences using the induced grammar. However, without knowing the correct parses, it is not possible to know the accuracy of the parses obtained. Due to the unavailability of any corpus of clinical report sentences annotated with syntactic parses, 100 sentences out of the 1000 test sentences were manually annotated [8] with correct parse bracketing information. These sentences were then used to measure the accuracy of the parses obtained. Since the non-terminals created by the grammar induction process have

[1] http://nlp.stanford.edu/software/corenlp.shtml

system-generated names, it is not possible to measure the parsing accuracy based on labeled constituents, hence only the bracketing accuracy was measured. It was measured in terms of *precision* (how many brackets obtained using the induced grammar matched with the correct brackets), *recall* (how many brackets in the correct parses were present in the parses obtained using the induced grammar) and *F-measure* (harmonic mean of precision and recall). In the results, only the F-measures are reported.

4.2 Results

Figure 1 shows the F-measures of parsing brackets obtained with different *f* parameter values of the grammar induction process. This parameter relatively weighs the encoding cost of the grammar and the derivations (see Total Cost in Subsection 2.2). For inducing the grammar, 2000 training sentences were used (more training sentences did not significantly improve the performance as the next graph shows). The graph compares the F-measures obtained using the new derivation cost based on optimal code lengths and the derivation cost based on equal code lengths. It can be seen that the new derivation cost obtains better performance overall. Although the actual numbers are not high, they are comparable to the performance of unsupervised parsing of sentences of similar maximum lengths using other approaches in other domains.

In the Figure 2, learning curves are plotted to show how the performance changes as more training sentences are used to induce the grammar. For each point, only the best F-measure obtained by varying the *f* parameter is shown. It is clear from the graph that better performance is obtained using the optimal code lengths at all amounts of training data. However, when sentences of smaller maximum lengths (10 and 15) were used, there was not a significant difference in performances between optimal code length based cost and equal code length based cost. This may be because it is easier to parse smaller sentences and the new derivation cost did not offer any advantage.

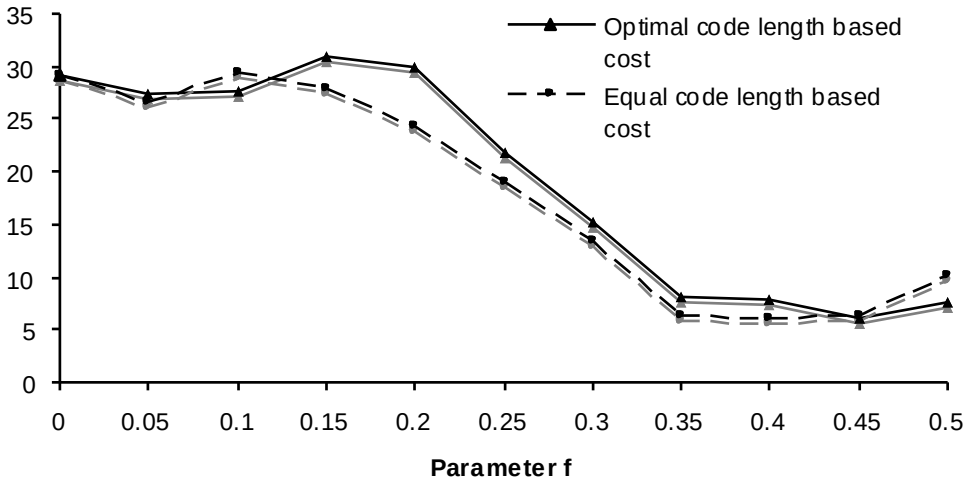

FIGURE 1. F-measure of parsing brackets using the new optimal code length based derivation cost and the equal code length based derivation cost while varying the value of parameter *f* which relatively weighs the cost of the grammar and cost of the derivations during the grammar induction process.

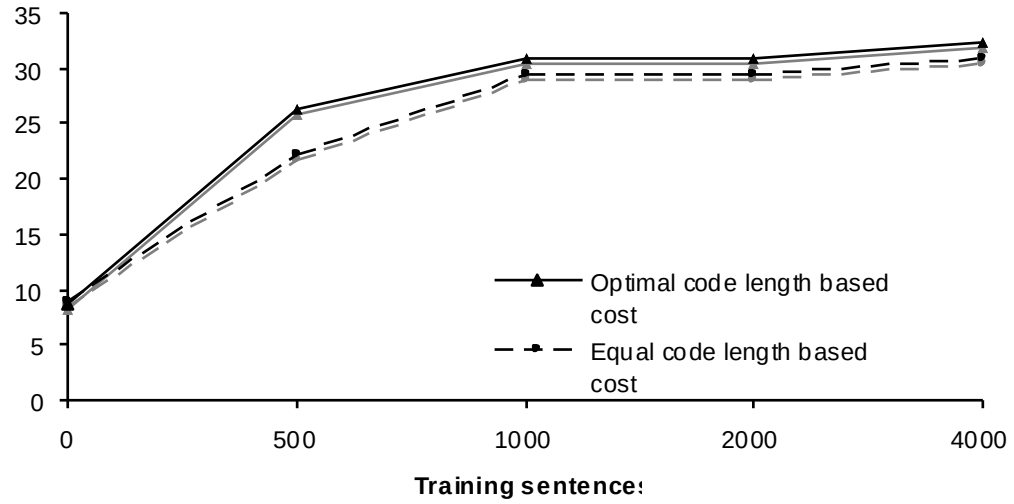

FIGURE 2. Learning curves for the maximum F-measure of parsing brackets obtained using the new optimal code length based derivation cost and the equal code length based derivation cost.

5 FUTURE WORK

The experiments presented in the last section used only hundred sentences to measure parsing performance. An obvious future work will be to use a bigger corpus of annotated sentences for testing. Also, besides discharge summaries, it should be tested on other genres of clinical reports present in the Pittsburgh corpus. Although optimal length codes were used in computing the new derivation cost, equal bits were used to encode each terminal and non-terminal in the grammar. In future, optimal length codes could be used to encode grammar as well, for example using fewer bits to encode a non-terminal that is used often in the grammar. But given that a symbol is not used in a grammar as many times a production is used in derivations, optimal length encoding of grammar symbols may not have the same impact.

Although this work was about unsupervised grammar induction, but as there is no direct way to measure accuracy of a grammar, the accuracy was actually measured for unsupervised parsing. As the method was not set to be tested for unsupervised parsing, a single default parse was obtained during testing for every sentence using the induced grammar ignoring any potential ambiguity in the parsing process. In future work, it will be more appropriate to obtain the best parse whenever there are multiple parses obtainable for any sentence. The new derivation cost could also be used to obtain the best parse by defining the best parse to be the one which has the lowest encoding cost. As was pointed out in Section 3, this is equivalent of obtaining maximum probability parse. Another avenue for future work is to incorporate grammar ambiguity in the grammar induction process as well. The grammar induction process continues in iterations, applying a grammar transformation operator in each iteration and appropriately changing the grammar and the derivations. Although more computationally expensive, a better way will be to obtain the best derivation for each sentence by parsing the sentences using the new grammar in each iteration. This will also give better counts of how many times a production is used in the derivations and thus better optimal lengths for encoding the derivations.

6 CONCLUSIONS

The paper presented a new encoding cost for sentence derivations to be used in the cost reduction method for unsupervised grammar induction. The cost uses optimal code lengths for encoding the information regarding which production is used in expanding a non-terminal in a derivation. The induced grammar using the new cost was shown to obtain a better parsing performance on a dataset of clinical report sentences.

7 REFERENCES

[1] D. Jurafsky and J. H. Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics", 2nd edition, Prentice-Hall, 2009.

[2] E. Ponvert, J. Baldridge, and K. Erk. "Simple unsupervised grammar induction from raw text with cascaded finite state models," in *Proc. of ACL-HLT*, 2011, pp. 1077–1086.

[3] Y. Seginer. "Fast unsupervised incremental parsing," in *Proc. of ACL*, 2007, pp. 384–391.

[4] D. Klein and C. Manning. "A generative constituent-context model for improved grammar induction," in *Proc. of ACL*, 2002, pp. 128–135.

[5] D. Klein and C. D. Manning. "Corpus-based induction of syntactic structure: Models of dependency and constituency," in *Proc. of ACL*, 2004, pp. 479–486.

[6] P. Langley and S. Stromsten. "Learning context-free grammar with a simplicity bias," in *Proc, of ECML*, 2000, pp. 220–228.

[7] T. Chen, C. Tseng, and C. Chen. "Automatic learning of context-free grammar," in *Proc. of Conference on Computational Linguistics and Speech Processing*, 2006.

[8] R. J. Kate. "Unsupervised Grammar Induction of Clinical Report Sublanguage", to appear in the special session on *Machine Learning for Biomedical Literature Analysis and Text Retrieval* in *Proc. of ICMLA*, 2011.

[9] C. E. Shannon. "A Mathematical Theory of Communication", *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[10] T. M. Cover. "Elements of Information Theory", John Wiley & Sons, 2006.

[11] J. G. Wolff. "Language acquisition, data compression, and generalization," *Language and Communication*, vol. 2, pp. 57–89, 1982.

[12] W. W. Chapman, M. Saul, J. Houston, J. Irwin, D. Mowery, H. Karkeme, and M. Becich, "Creation of a repository of automatically de-identified clinical reports: Processes, people, and permission," in *AMIA Summit on Clinical Research Informatics*, 2011.

[13] O. Bodenreider. "Unified Medical Language System (UMLS): integrating biomedical terminology". *Nucleic Acids Research*, Vol. 32, No. suppl 1, 2004.

[14] A. R. Aronson and F. Lang, "An overview of MetaMap: Historical perspectives and recent advances," *Journal of the American Medical Informatics Association*, vol. 17, pp. 229–236, 2010.

[15] J. Earley, "An efficient context-free parsing algorithm," *Communications of the Association for Computing Machinery*, vol. 6, no. 8, pp. 451–455, 1970.