Guiding Symbolic Natural Language Grammar Induction via Transformer-Based Sequence Probabilities

Ben Goertzel[1], Andres Suarez-Madrigal[1,2], Gino Yu[2]

[1]SingularityNET Foundation [2]The Hong Kong Polytechnic University

Abstract. A novel approach to automated learning of syntactic rules governing natural languages is proposed, based on using probabilities assigned to sentences (and potentially longer sequences) by transformer neural network language models to guide symbolic learning processes like clustering and rule induction. This method exploits the learned linguistic knowledge in the transformers, without any reference to their inner representations; hence, the technique is readily adaptable to the continuous appearance of more powerful language models. We show a proof-of-concept example of our proposed technique, using the method to guide unsupervised symbolic link-grammar induction methods drawn from our prior research.

1 Introduction

Unsupervised grammar induction – learning the grammar rules of a language from a corpus of text or speech without any labeled examples (e.g. of sentences labeled with their human-created syntax parses) – remains in essence an unsolved problem. State of the art performance is improving but still fairly mediocre [6].

Recent transformer neural network models have shown powerful abilities at language prediction and generation, indicating that at some level they do internally "understand" the rules of grammar. However, the rules of grammar don't seem to be found in the neural connections in these networks in any straightforward manner [1][5], and are not easily extractable in an unsupervised manner. Supervised extraction of grammatical knowledge from the BERT network reveals that, to map the state of a transformer network when parsing a sentence into the sentence's parse, complex and tangled matrix transformations are needed [4].

Here we explore an alternate approach: Don't try to milk the grammar out of the transformer network directly, rather use the transformer's language model as a *sequence probability oracle*, a tool for estimating the probabilities of word sequences; then use these sequence probability estimates to guide the behavior of symbolic learning algorithms performing grammar induction. This is work in progress, but preliminary results have been obtained and look quite promising.

Full human-level AI language processing will clearly involve additional aspects not considered here, most critically the grounding of linguistic constructs in non-linguistic data [8]. However, the synergy between symbolic and sub-symbolic

aspects of language modeling is a key aspect of generally intelligent language understanding and generation which has not been adequately captured so far, and we feel the work presented here makes significant progress in this direction.

2 Methodology

Transformer network models like BERT and GPT2 and their relatives provide probabilistic language models which can be used to assess the probability of any given sentence. The probability of sentence S according to such a language model tells you the odds that, if you sampled a random sentence output by the model (used in a generative way), it would be S. If S is not grammatical according to the grammar rules of the language modelled by the network, its probability will be very low. If S is grammatical but senseless, its probability should also be quite low.

Having a sentence (or more generally word sequence) probability oracle of this nature for a language provides a way to assess the degree to which a given grammar G models that language. What one wants is that: The high-probability sentences according to the oracle tend to be grammatical according to G, the low-probability sentences according to the oracle are less likely to be grammatical according to G, and G is as concise as possible. The grammars that are best according to these conjuncted factors are the best grammatical models of the language in question.

This concept could be used to cast grammar induction as a probabilistic programming problem, in a relatively straightforward but computationally exorbitant way. Just sample random grammars from some reasonable distribution on grammar space, and evaluate their quality by the above factors.

What we propose here is conceptually similar but more feasible: Begin with a symbolic grammar learning algorithm which is capable of incrementally building up a complex grammar, and use sentence probability estimates from a neural language model to guide the grammar learning. One could view this as an instance of the probabilistic programming approach with a linguistic-theory-based heuristic method of sampling grammar space.

Guided by our prior work on symbolic unsupervised grammar induction [3], we describe an algorithm for inducing a dependency grammar from an unlabeled corpus, involving two major steps:

- Separate the vocabulary of interest into word categories (functionally equivalent to parts of speech, with a certain level of granularity). An implicit sub-step here is the disambiguation of polysemous words in the vocabulary, so that a single word could be assigned to more than one category.
- Do rule induction to find rules that specify how words in the said categories can be connected to form grammatical sentences.

In more detail, our approach involves:

1. Infer word-senses and parts of speech from vectors built using a neural language model as sentence probability oracle

2. Infer grammatical rules from symbolic pattern-analysis of the corpus tagged with these senses and parts of speech
3. Assemble a grammar incrementally from inferred rules
4. Evaluate the addition of an inferred rule to one's grammar via using a tree transformer network to generate sentences consistent with one's rule, and alternately with mutations of one's rule
5. Use a neural model as a sentence probability oracle to estimate whether the inferred rule leads to better generated sentences than its mutation

The overall grammar learning architecture encompassing this process is depicted in Figure 1

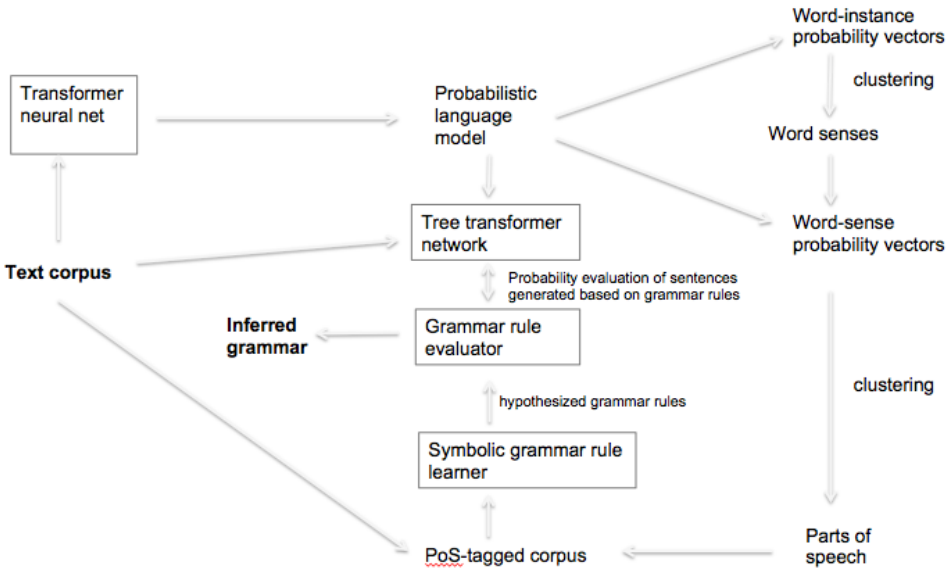

Fig. 1. High-level grammar learning architecture involving symbolic learning guided by estimated word sequence probabilities from a transformer network.

For our early experiments, we have chosen BERT [2] as the transformer to use, but it could easily be extended to any other similar pre-trained network.

2.1 Assessing sentence probability

We now explain some of the formal details of our approach, beginning with the computation of sentence probability according to a neural language model.

Given a sentence $S = [w_0, w_1, ..., w_N]$, composed of N words $w_i, i \in [0, 1, ..., N]$, we want to calculate its probability P(S). A way to decompose that probability

into conditional probabilities is:

$$P_f(S) = P(w_0, w_1, ..., w_N) = P(w_0) \cdot P(w_1|w_0) \cdot P(w_2|w_0, w_1) \cdot ... \cdot P(w_N|w_0, w_1, ..., w_{N-1}),$$

which we call *forward sentence probability*.

Now, each $P(w_i|w_{i-1}, ..., w_0)$ can be obtained from BERT's masked word prediction model by taking the whole sentence, masking all the words which are not conditioned in the term (including w_i), and obtaining BERT's estimation for the probability of w_i.

To exemplify the idea, we summarize how to calculate the forward probability of the sentence "She answered quickly". The probability is given by

$$P_f(\text{She answered quickly}) = P(\text{She}) \cdot P(\text{She answered}|\text{She}) \cdot P(\text{She answered quickly}|\text{She answered}).$$

Each of these terms translates to a BERT Masked Language Model prediction for a sentence with masked tokens. For example,

$$P(\text{She answered}|\text{She}) = P(\text{MASK2=answered}|\text{She MASK2 MASK3}),$$

and we get probability values such that "answered" is predicted as the second token in the BERT Masked Language Model.

Now, to take advantage of BERT's bi-directional capabilities, we can estimate the sentence's *backwards probability* in a similar fashion:

$$P_b(S) = P(w_0, w_1, ..., w_N) = P(w_N) \cdot P(w_{N-1}|w_N) \cdot P(w_{N-2}|w_{N-1}, w_N) \cdot ... \cdot P(w_0|w_1, w_2, ..., w_N)$$

We finally approximate the sentence probability as the geometric-mean of the two directional ones:

$$P(S) = \sqrt{P_f(S) \cdot P_b(S)}$$

2.2 Word Category Formation

Following our prior work on symbolic grammar induction [3] and a number of previous works, we propose to generate embeddings for the words in the vocabulary, and cluster them using a proximity metric in the embedding space. Each final cluster can be considered a different word category, whose connection rules to other clusters will be later defined in the induced grammar. Unlike prior work, we use sentence probabilities for the embedding features.

We expand each sentence in the corpus into N sentences with a "blank" token in a different position, where N is that sentence's length. Each of those sentences with a blank is a feature for the word-vectors we will build. Hence, we can think of a word-sentence matrix M, where the rows are the words in the vocabulary and the columns are unique sentences with blanks in them. Figure 2 shows such a matrix.

We fill each cell in the matrix with the probability of the corresponding sentence-with-a-blank (column), when the blank is substituted by the corresponding word (row). That is, if S'_j is the sentence-with-a-blank in column j and w_i is the word in row i, then the cell $M_{i,j} = P(S'_j|\text{blank filled with } w_i)$.

Once the matrix is filled, word categories are obtained by clustering the obtained word vectors. Or, if one has performed word sense disambiguation (which can be done based on different computations from this same matrix, as will be described below), by clustering similar vectors corresponding to word senses.

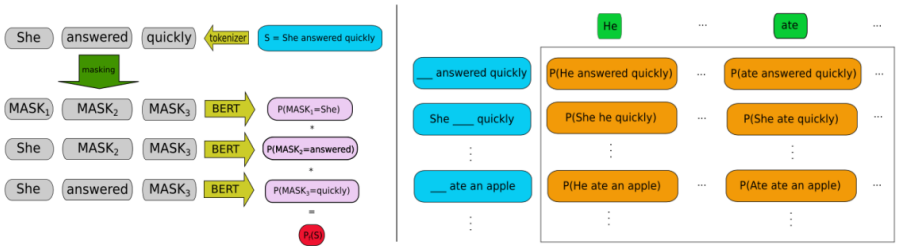

Fig. 2. *Left*: An example of forward sentence probability calculation. *Right*:The matrix of words versus sentences-with-one-blank, each entry of which gives the probability of the sentence with the given word filling in the blank. We use these for word sense disambiguation and word category learning.

2.3 Word Sense Disambiguation

Embeddings obtained from transformer networks by supervised learning have been used to derive word senses [10]; here we pursue a related goal but in an unsupervised approach. From an unlabeled training corpus, we obtain a transformer embedding for each instance of each word in its given context. Then, for each word in the vocabulary, we gather all of its embeddings and cluster them. The resulting clusters can be used as the different word senses of a word.

Specifically, in our approach, a word-instance can be represented by a vector whose i'th entry is the probability the neural language model assigns to the sentence obtained by: Replacing the word with word i, in the sentence (and discourse context) where it appears.

Consider the word-instance "test" in "The test was a success." We can make a vector for the sentence-with-one-blank-word "The ___ was a success" via evaluating the probabilities of sentences like:

- $W[1]$ = "frog", $S[1]$ = "The frog was a success"
- $W[2]$ = "which", $S[2]$ = "The which was a success"

Based on these we can create a vector for the intension (contextual properties) of the word-instance, via

- V(test, The ___ was a success)[1] = P(The frog was a success)
- V(test, The ___ was a success)[2] = P(The which was a success)
- ...
- V(test, The ___ was a success)[i] = P(S[i]).

Word Category Formation in Depth Based on the above, a word-sense can be represented by a vector whose i'th entry is the probability the neural language model assigns to the sentence obtained by: Inserting the word into the blank in the i'th sentence on a list of sentences-with-one-blank-word (and checking to be sure the resulting sentence gives the appropriate sense of the word). For instance, if we have

- Sentence 1 = The ___ was a success.
- Sentence 2 = Dating her was a constant ___ of my sanity.
- Sentence 3 = I would rather ___ him first.

then building the vector for "test" (in the sense of exam) looks like

- S(1, test) = The test was a success.
- S(2, test) = Dating her was a constant test of my sanity.
- S(3, test) = I would rather test him first.
- ...
- $V(test)[i] = P(S(test, i))$

Clustering these vectors creates Part of Speech categories and finer-grained syntactico-semantic categories.

2.4 Grammar Induction

After word categories are formed, grammar induction can take place by figuring out which groups of words are allowed to link with others in grammatical parses. A grammar can be accumulated by starting with one rule and adding more incrementally, using the neural language model to evaluate the desirability of each proposed addition. The choice of which additions to the grammar to propose at each stage is made by a symbolic rule induction algorithm; so far we have used the Grammar Learner process described in [3].

For a grammar rule proposed as an addition to the partial grammar already learned, we generate sentences that use that rule within the given grammar and obtain their sentence probabilities $P(S)$. Then we corrupt the rule in some manner, adjust the grammar accordingly, generate sentences from this modified grammar starting with the mutated rule, and evaluate their $P(S)$. If the sentences from the modified grammar decrease significantly in quality (where the threshold is a parameter), then the original rule is taken as valid. The rationale here is that correct grammar rules will produce better sentences than their distortions.

In the case of the link grammar formalism [7], which we have used in our work so far, a grammar rule consists of a set of disjuncts of conjunctions of typed "connectors" pointing forward or backward in a sentence. A mutation of this type of rule can be the swapping of each connector in the rule, which also implies a word-order change.

For example, if we have a rule R that connects the word "kids" with the word "the" on the left and the word "small" also on the left, in that order:

`kids: small- & the-`,
which allows the string "the small kids", then the mutated rule R^* would be
`kids: small+ & the+`,
which accepts the string "kids small the"[1].

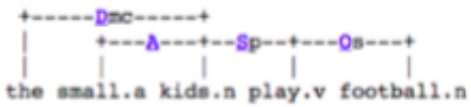

Fig. 3. Link parse of "The small kids play football" according to the standard English link grammar dictionary.

This methodology requires some way to generate series of sentences from proposed grammars. One approach is to use a given grammar to guide the attention within a Tree Transformer [9]. The standard Tree Transformer approach guides attention based on word-sequence segmentation that is driven by mutual information values between pairs of adjacent words. One can replace these probabilities with mutual information values between pairs of words that are linked in partial parses done according to a provided grammar.

Currently we are using a simpler stochastic sentence generation model in our proof-of-concept experiments, and planning to shift to a Tree Transformer approach for the next phase of work.

So, the rule R guides the generation of sentences like S = "The small kids play football". The rule R^* guides the generation of sentences like $S*$ ="The kids small play football". The language model says $P(S) > P(S^*)$ – arguing in favor of adding R to one's grammar (and then continuing the incremental learning process).

Alternatively, instead of producing mutated rules, one could also compare the probabilities of sentences generated with the rule under evaluation against those of a set of reference sentences of the same length, like those in the corpus used to derive the grammar, or the word categories obtained previously.

3 Proof of concept

Scalable implementation and testing of the ideas described above is work in progress; here we describe some basic examples we have explored so far, which validate the basic concepts (but do not yet provide a thorough demonstration). We chose to perform our initial experiments using BERT, due to its popularity in several downstream tasks (e.g. word sense disambiguation by [10]). [2]

[1] Notice that connectors in the rules for small and kids also have to be modified to accommodate this mutation, i.e. they need to swap `kids+ to kids-`

[2] In particular, we use Huggingface's implementation of BERT, contained in their "transformers" package [11] https://huggingface.co/transformers

Following the workflow of the grammar induction process, we first show an example of word sense disambiguation, then one for word category formation, and finally grammar rule evaluation.

3.1 Word sense disambiguation

For an initial simple experiment, we created a small corpus of 16 sentences containing 146 words, out of which 8 are clearly ambiguous (for a human). Both syntactic and semantic ambiguities were included.

Some practical considerations for our proof of concept WSD experiment are as follows:

- We use the average of the last 4 attention layers of BERT as word instance embeddings, following the second best result from the original paper [2] in a Named Entity Recognition task[3].
- When the BERT tokenizer splits an out-of-vocabulary word into subwords, we use the arithmetic mean of the embeddings of each sub-token, following [10].
- For clustering, we tried the out-of-the-box KMEANS, DBSCAN and OPTICS models in scikit's sklearn library[4].

We ran the corpus through our disambiguator, and found that KMeans clustering did the best job at separating word senses in our test. Using 2 clusters, the algorithm achieved an F1-score of 0.91. As examples, the disambiguation for the word "fat", which was perfect, looks as follows:

```
Cluster #0 samples:
santiago became FAT after he got married .
there are many health risks associated with FAT .
the negative health effects of FAT last a long time .
Cluster #1 samples:
the FAT cat ate the last mouse quickly .
there is a FAT fly in the car with us .
```

The clustering for "time", on the other hand, had one mistake, and looks like this:

```
Cluster #0 samples:
i was born and raised in santiago de cuba , a long TIME ago .
my mouse stopped responding at the same TIME as the keyboard .
the negative health effects of fat last a long TIME .
Cluster #1 samples:
you will TIME the duration of the dress fitting session .
TIME will fly away quickly .
```

The disadvantage of using this straightforward implementation of KMeans is that one has to specify the number of clusters. When requesting more clusters

[3] The best result uses the concatenation of the same 4 last layers, which creates embeddings that are 4 times longer, but it's only marginally better.

[4] https://scikit-learn.org/stable/index.html

than there are senses for a word, the algorithm spreads instances with similar meanings to different clusters. This is especially the case with words that we wouldn't consider ambiguous, like function words (we have sought to filter these by explicitly not disambiguating the top 10% most frequent words in the corpus). However, this may not be a terrible problem in our use case, as the word category formation algorithm will simply create more word-sense vectors per word, which then it could cluster together in the same word category. Future WSD experiments will involve alternative like automatically learning K, or using EM as an alternative.

3.2 Word category formation

Here, working with the same corpus as for WSD, we used the disambiguation results described above to build word vectors, thus allowing for words to be catalogued in more than one group. Rather than KMeans, we found that OPTICS, a method that doesn't require a parameter for the number of clusters and can leave vectors uncategorized (shown below as Cluster #-1), offers a very good cluster quality, with a good level of granularity.

The quality of the formed clusters (#0-14) is remarkable:

```
Cluster #-1: [fat, fat, ate, last, mouse, mouse, quickly, quickly,
 ., there, there, many, many, health, health, associated, with,
 with, stopped, responding, same, time, as, will, fly, fly, negative,
 of, a, a, long, in, us, tomorrow, she, she, was, was, wearing,
 lovely, brown, brown, dress, attendees, did, not, properly, for,
 occasion, became, after, got, married, ', ', s, deteriorated,
and, de, ,, ago, fitting, wasn, t, year, smith, protagonize, ]
Cluster #0: [the, my, his, ]
Cluster #1: [born, able, ]
Cluster #2: [raised, growing, bought, ]
Cluster #3: [cat, keyboard, car, session, feed, family, microsoft, ]
Cluster #4: [duration, episode, series, ]
Cluster #5: [are, is, ]
Cluster #6: [morning, night, ]
Cluster #7: [away, out, ]
Cluster #8: [they, he, i, you, ]
Cluster #9: [risks, effects, ]
Cluster #10: [at, to, ]
Cluster #11: [santiago, cuba, ]
Cluster #12: [time, will, long, ]
Cluster #13: [dress, and, ]
Cluster #14: [of, in, ]
```

An evident problem with this result is that most of the words remain in Cluster #-1, which are the uncategorized words. Although we would expect the full iterative grammar learning algorithm we propose to be able to live with that and cluster some of the remaining words in the next pass, we will first try to fine-tune the procedure to alleviate this situation, as well as explore some other clustering algorithms. At the same time, we predict that the results will improve

when we use a larger number of features (instead of only 16 sentences for a total of 146 different features). A very simple expansion of the vocabulary to cluster (not shown) already showed a similar number of more populated clusters.

3.3 Grammar Rule Evaluation

We show a simple use case for grammar rule evaluation, using the simple rule modification strategy proposed in the methodology: swapping the direction of the connectors that make up a rule, and comparing the sentences it generates with and without the mutation.

For this experiment, we created a proof of concept grammar with 6 words divided in 6 word categories: determiner, subject, verb, direct object, adjective, adverb. Then, we assigned relationships among the classes. Using the semi-random sentence generator, this grammar produces sentences like "the small kids eat the small candy quickly." (that being the longest possible sentence derived from this grammar).

We then introduced some extra spurious rules to the grammar by hand. From a total of 21 rules (15 correct ones vs. 6 spurious ones), the grammar can generate sentences like "kids eat the the small candy kids eat candy the small quickly quickly.", which clearly shows that the grammar is not correct anymore (this grammar has loops, so this is not even the longest sentence permitted by these simple modification).

Finally, we ran our first version of the grammar rule evaluation algorithm, to find out that all of the spurious rules were detected and rejected. Three of the "correct" rules suffered the same fate.

It is interesting to notice that among the "correct" rules that were discarded, at least one:

```
eat: kids-,
```

generates sentences with no direct object, like "the kids eat." This sentence, although valid, might not be very common for the BERT model, and thus obtain a low probability. Similarly, the reverse of this rule, as modified by the rule evaluation algorithm:

```
eat: kids+,
```

generates sentences like "eat the kids.", which is also grammatically valid, and maybe as common as the previous case. This would be one sensible explanation for the rule's rejection.

4 Conclusion and Future Work

Our proof-of-concept experiments give intuitively strong indication of the viability of the methodology proposed for synergizing symbolic and sub-symbolic language modeling to achieve unsupervised grammar induction. The next step is to create a scalable implementation of the approach and apply it to a large corpus, and assess the quality of the results. If successful this will constitute

significant progress both toward unsupervised grammar induction, and toward understanding how different types of intelligent subsystems can come together to more closely achieve human-like language understanding and generation.

References

1. Clark, K., Khandelwal, U., Levy, O., Manning, C.D.: What Does BERT Look At? An Analysis of BERT's Attention. arXiv:1906.04341 [cs] (Jun 2019), http://arxiv.org/abs/1906.04341, arXiv: 1906.04341
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs] (May 2019), http://arxiv.org/abs/1810.04805, arXiv: 1810.04805
3. Glushchenko, A., Suarez, A., Kolonin, A., Goertzel, B., Baskov, O.: Programmatic Link Grammar Induction for Unsupervised Language Learning. In: Hammer, P., Agrawal, P., Goertzel, B., Ikle', M. (eds.) Artificial General Intelligence, vol. 11654, pp. 111–120. Springer International Publishing (2019)
4. Hewitt, J., Manning, C.D.: A Structural Probe for Finding Syntax in Word Representations p. 10 (2019)
5. Htut, P.M., Phang, J., Bordia, S., Bowman, S.R.: Do Attention Heads in BERT Track Syntactic Dependencies? arXiv:1911.12246 [cs] (Nov 2019), http://arxiv.org/abs/1911.12246, arXiv: 1911.12246
6. Li, B., Cheng, J., Liu, Y., Keller, F.: Dependency Grammar Induction with a Neural Variational Transition-based Parser. arXiv:1811.05889 [cs] (Nov 2018), http://arxiv.org/abs/1811.05889, arXiv: 1811.05889
7. Sleator, D.D.K., Temperley, D.: Parsing english with a link grammar (1995)
8. Tomasello, M.: Constructing a Language: A Usage-Based Theory of Language Acquisition. Harvard University Press (2003), http://groups.lis.illinois.edu/amag/langev/paper/tomasello03book.html
9. Wang, Y.S., yi Lee, H., Chen, Y.N.: Tree transformer: Integrating tree structures into self-attention. In: EMNLP/IJCNLP (2019)
10. Wiedemann, G., Remus, S., Chawla, A., Biemann, C.: Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. arXiv:1909.10430 [cs] (Oct 2019), http://arxiv.org/abs/1909.10430, arXiv: 1909.10430
11. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. ArXiv **abs/1910.03771** (2019)