Final project report for CMPT826: Automated Learning and Reasoning (Summer 2011)

Discovering Lexical Hierarchy from Word Sequences

Honghua Li , Robert Hadley
School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6

Abstract

In his article, *Finding Structure in Time*, Jeffrey L. Elman (1990) proposed that networks with memory (e.g. Simple Recurrent Network - SRN) were able to find interesting structures from temporal sequences. In the simulation *Discovering lexical classes from word order*, however, the training and testing corpus contains only 2- and 3-word sentences, which made his experiment somewhat unrealistic. In this project, I used Emergent software (previously called PDP++) to simulate an SRN, and trained it with more complicated corpora, which contained that- and preposition-clauses and had longer sentence lengths. Besides discovering lexical classes and testing the trained network with a novel untrained noun, I also examined the distribution of hidden activations of each word in different contexts, and found that the network was able to learn to distinguish the syntax roles of each word.

1. Introduction

In fixed word-order languages (e.g. English), the order of words is highly restricted. It is well known that there is an underlying abstract structure, which provides a more insightful understanding of the generation of word orders. However, without the knowledge about the underlying abstract structure the surface strings of a sentence also provide some clues for it. The task of this project is to try to learn some aspects of the underlying structure (e.g. lexical classes) from the surface strings using a Simple Recurrent Network (SRN).

The SRN has 4 layers, as showed in Fig. 1. The hidden layer of an SRN encodes each input as an internal representation which in turn enable the network to produce the output. The context layer stores a copy of the activations in the previous hidden layer, which gives the network memory.

In this project, the network was trained with two large corpora, which contained 2,227 and 3,000 sentences respectively. Because sentences with clauses were included, the tasks in this project were more difficult than that in Elman's paper. After the network was trained, my analysis of the activation patterns in the hidden layer was as follows: (1) examined the distribution of all hidden activations of a given word under different contexts to clarify Tony Plate's concern about averaging technique; (2) clustered the internal representations of each word to discover the lexical categories; (3) for a noun that appears in both subject and object positions, compared the average activation pattern produced from subject position with that from object position; (4) replaced one of the nouns with a novel, non-trained noun, then tested the entire corpus and compared the activation patterns produced with that produced by the original noun.

2. Train the SRN

The sentence generator in Elman's paper can only create two- and three-word short sentences. To make the task more challenging, my sentence generator started from a basic template NOUN-VERB-NOUN, and inserted a that-

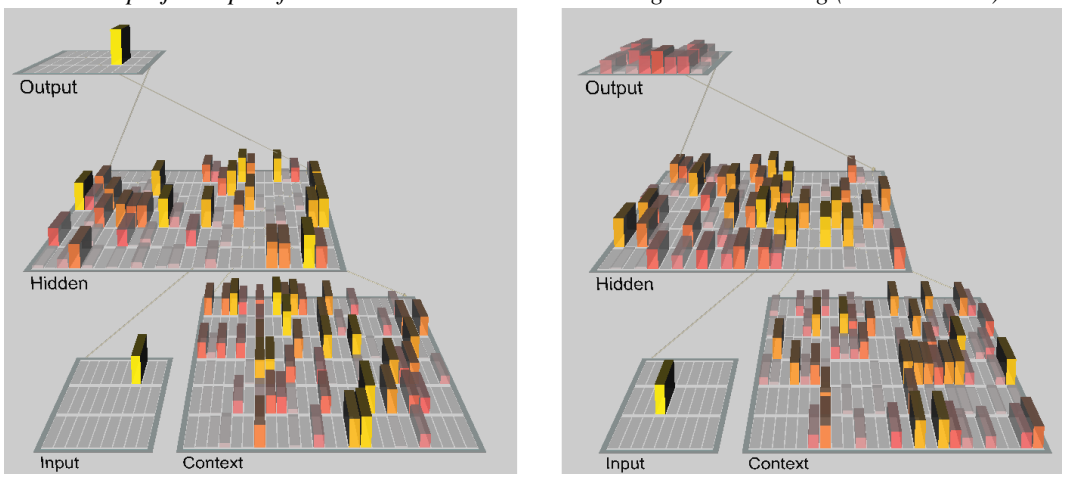

Figure 1: The SRN created by Emergent: (left) the output in the training phase was forced to be correct, (right) the output in the testing phase is just the likelihood of each potential successor.

Table 1: Templates of basic structure and that-/preposition-clause

template	Example
NOUN-VERB-NOUN (basic structure)	*girls hate cats*
that-NOUN-VERB	birds *that girls love* hit dogs
that-VERB-NOUN	boys like cats *that chase mice*
from-NOUN	birds *from girls* hit balls
with-NOUN	girls *with balls* love dogs

or preposition-clause (showed in Table 1) after any noun in the sentence. The insertion was recursively applied. All the 26 words used by the sentence generator included 12 nouns, 11 verbs, two prepositions and 1 pronoun.

Two corpora were used in this project: corpus-A and corpus-B. Sentences in corpus-A were created by either basic template (NOUN-VERB-NOUN) or inserting prepositional clause once to the basic template, thus with a length less or equal than 5 words. Besides what corpus-A had, corpus-B also contained sentences created by recursively inserting all types of clauses to basic template up to 4 times. There were 2,227 sentences in corpus-A, and 3,000 sentences in corpus-B. While corpus-A was more complicated than that in Elman's experiment, corpus-B was harder than corpus-A. Both corpora were used to train and test the SRN, with the hope to see the performance of SRN under tasks with different complexity.

Each word was represented by a 27-bit vector, in which only one bit was set to one while all the others were zero. One extra bit was reserved for further analysis. Because all words were orthogonal to each other in the 27 dimensional space, the encoding method reflected neither grammar role nor meaning of each word.

I used Emergent software to simulate an SRN (Fig. 1), in which the input and output layers contained 27 unites each and the hidden and context layers contained 150 units each. The task for the network was to learn to predict the successor words in the sequence. The training strategy was as follows. All the sentences in corpus-A were concatenated to form a single input stream of 8,105 27-bit vectors. In each *trial*, the network was trained to predict the next word right after the input word in the sequence. After all the 8,105 trials were completed, the other *epoch* began without any break. My training phase included 10 complete epoches.

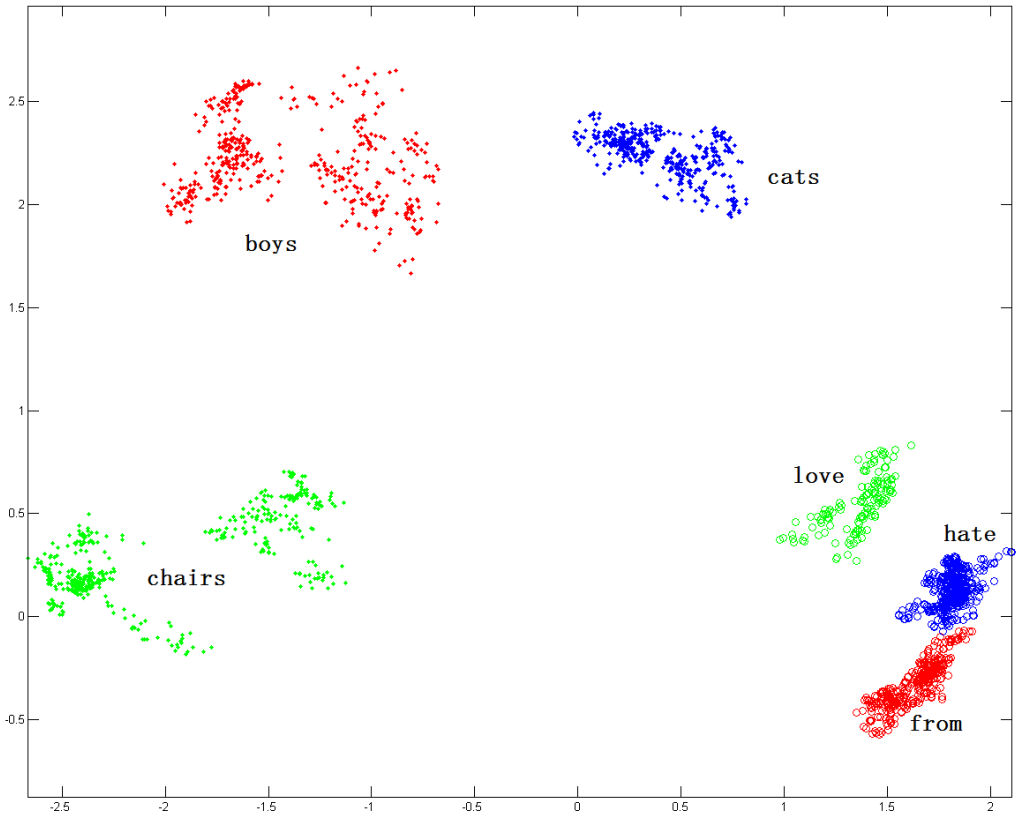

Figure 2: Distribution of internal representations in the 2D embedding space. The points in the same group (indicated by dots with the same color and mark) are internal representations of the same word in different contexts.

3. Internal Representation

After training the network with ten epoches, I froze all the connection weights that had been trained. The input stream was processed by the network for one final pass to test what the network had learned.

Elman has showed the network had learned to approximate the likelihood ratios of potential successors by the activation of its output units. On the right of Fig. 1, the activations of output units were the likelihood of all possible successors instead of one single successor.

Besides the activations in the output layer, the internal representations captured by the hidden layer should also reveal some patterns. During the testing phase, the activations of hidden units were saved to yield 8,105 150-bit vectors. Each word occurred many times in the corpus. By averaging all the hidden activations produced by a given word, a single 150-bit vector was generated for each of the 26 words.

Elman mentioned in the footnote that Tony Plate had pointed out this averaging technique was dangerous. To clarify their concerns, I examined the distributions of all hidden activations under different contexts in the 150-dimension space. It was impossible to visualize these internal representations in the 150-dimension space directly, so I reduced the dimensions and analyzed the distribution in a 2D embedding space.

Firstly, I used the Principle Component Analysis (PCA) tool in the Emergent software to discover the two principle components. To do so, one needs to create a new data table including all the hidden layer activations, and find the *data_proc* $\rightarrow$ *data_anal* object in the left browser. At the bottom, select *HighDim/PCA2d Prjn* and specify the source data in the pop-up dialog. Hit OK, and the embedded data will be saved at *data* $\rightarrow$ *AnalysisData*. Then I plotted these embedded internal representations using Matlab, as shown in Fig. 2. The points produced by the

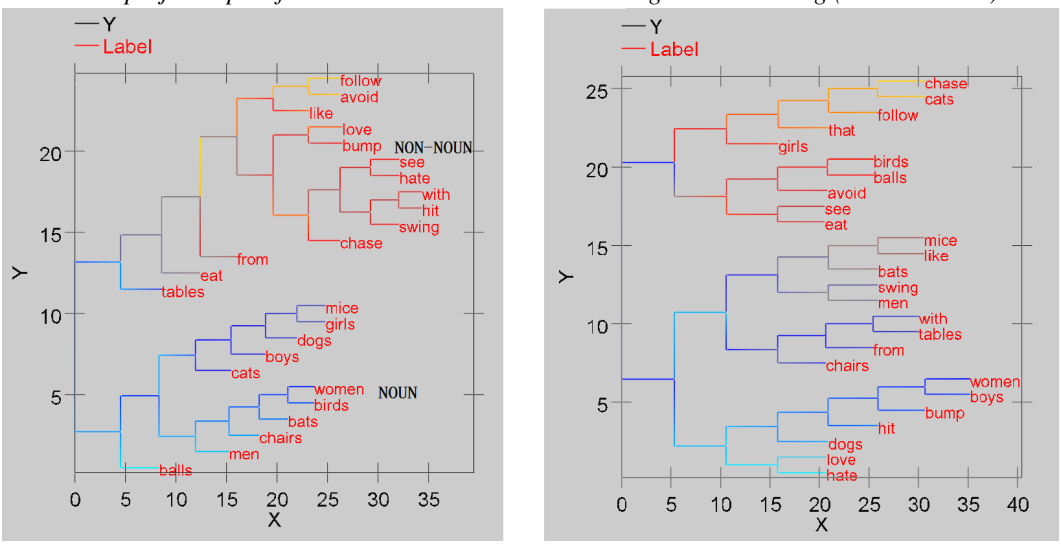

Figure 3: Hierarchical cluster trees of hidden activation patterns of corpus-A (left) and corpus-B (right). The labels of the two axes do NOT correspond to the real distance among words in the representation space, but are default scales from the clustering software.

same word were plotted with the same color and mark. It was very obvious that each word corresponds to a tight cluster in the internal representation space without overlapping. Thus, it was safe to average all the hidden layer activations of each word in different contexts to generate a single internal representation for each word.

4. Lexical categories

Each word had many instances in the corpus, which generated different activation patterns in the hidden layer according to the different contexts. As a simple approximation, all these activation patterns of a given word were averaged, resulting in a single 150-bit vector as the internal representation for each of the 26 words in corpus-A. I clustered the 26 activation vectors using the single-hierarchical method, and the resulting tree is shown on the left in Fig 3.

The hierarchical cluster tree of corpus-A reflected two major categories of the vocabulary, NOUN and NON-NOUN. There was one exception, *tables*, which was located between these two major categories and more closer to the NON-NOUN category. One highly possible reason was that *table* only occurred at object position in all sentences, while all the other nouns occurred on both subject and object position.

The same clustering was applied on corpus-B, and the hierarchical cluster result was showed on the right of Fig 3. In this case, the network failed to discover the major categories. The main reason might be corpus-B contained very complicated sentences, e.g. *boys that like girls that eat cats that birds avoid swing women,* which was also hard for our human to capture its meaning. Due to the fact that the SRN only has one context layer which records the previous hidden activations, its learning ability is limited. Some complicated tasks, like corpus-B, are beyond the learning ability of the simple recurrent networks.

5. Subject v.s. Object

A given noun can serve as either the subject or the object in a sentence. In our training sequence, a noun at the subject position was most likely followed by a non-noun (verb, pronoun or preposition), while the noun at the object position mostly had a noun successor (e.g. the first word of the next sentence, which was always the

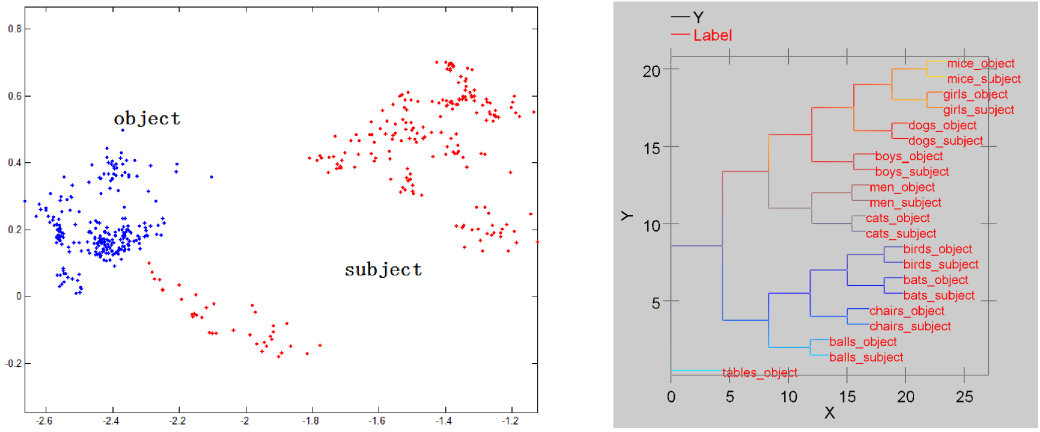

Figure 4: Left: All the hidden activations of *chairs* in different contexts were plotted in the 2D embedding space, among which the red dots were produced at subject position and the blue dots were at object position. Right: The hierarchical cluster tree of all the internal representations produced at either subject or object position for each word.

subject). If indeed the network had learned the syntax structures of the training corpus, we should expect to see the difference between activations of subject and that of object.

At the first step, the grammar role of each noun was identified in its context. In the sentence that contained clauses, identifying a noun's grammar role was not trivial. The clause could be inserted after any noun in the sentence, so it was not easy to *see* the syntax structure of a sentence. My strategy was analyzing the syntax structure and decomposing a complicated sentence into one basic sentence with several clauses. One import observation for syntax structure analysis was that the clause was never inserted at the middle of other clauses, but at the end of other clauses. In other words, if we come across *that* in a sentence, the two successors must belong to this that-clause. For example, *"boys that like girls that eat cats that birds avoid swing women"* was decomposed into one basic structure *"boys swing women"* and three that-clauses *"that like girls"*, *"that eat cats"* and *"that birds avoid"*. After the decomposition, the grammar role of each word was identified in the two- or three-word syntax components. According to the templates in Table 1, this task was easily accomplished by considering a word's position in these syntax components.

In Fig 2, the activation group of each noun had two obvious subgroups, which inspired me to guess that one subgroup revealed the subject and the other the object. To examine my guess, hidden activations produced by *chairs* at subject and object positions were plotted by different colors on the left in Fig. 4. There were two obvious subgroups, without overlapping. **It seemed the network also learned to distinguish the syntax roles of a given word!**

All the hidden activations of each noun at either subject or object position were averaged to yield two internal representations for each noun. All these internal representations were also clustered, and the hierarchical tree was showed on the right in Fig 4. The syntax structures the network had learned were substructures of each word, which could also be proved by the distributions of groups and subgroups in Fig 2.

6. Introducing a novel word

I replaced one of the nouns (*birds*) in the training corpus with a novel, untrained noun (*NOVEL-NOUN*). This replacement was accomplished by replacing the 27-bit vector of *birds* with a new 27-bit vector, in which only the reserved bit was turned on. The trained network was tested with the entire modified training corpus.

The same analysis as that in section 4 was applied, and the hierarchical cluster tree is shown in Fig. 5. By

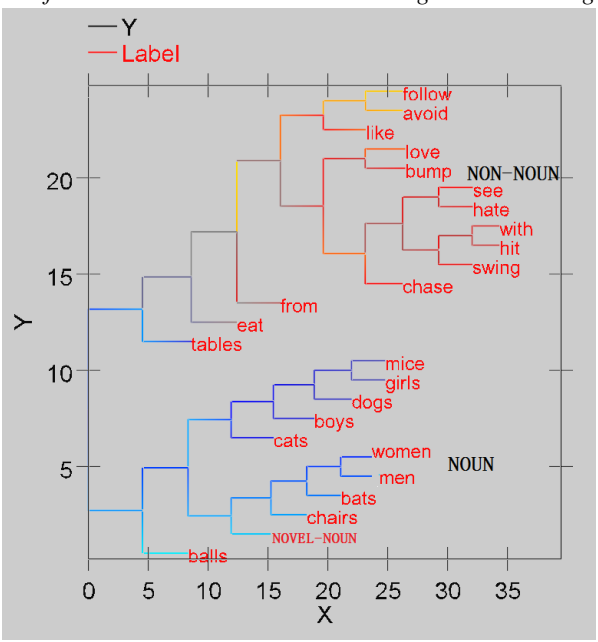

Figure 5: The hierarchical cluster tree of the modified corpus-A, in which *birds* was replaced by a novel noun *NOVEL-NOUN*.

comparing it with the left tree in Fig. 3, it seemed the network had generated an internal representation for NOVEL-NOUN that was very similar to the internal representation of the word *bird* in the original training corpus.

7. Conclusions

In this project, I reproduced Elman's simulations with more complicated corpora. Besides what Elman did, I also made some novel contributions:

- Through examining the distribution of hidden activations in an embedding 2D space, I clarified Tony Plate's worry on the technique of generating internal representation of each word by averaging all the hidden activations of the given word in different contexts.
- Developed an algorithm to identify the grammar role (subject, object) of a given noun in a sentence that contains recursive clauses.
- Analyzed the distribution of hidden activations produced by the same word at different grammar positions. The SRN was able to discover the syntax structures of each noun, which were substructures of the hierarchical tree in the internal representation space. This is an important observation, and is an extension to Elman's discoveries.

It also showed that the SRN failed to discovering lexical categories on very complicated corpus-B, which contained recursive clauses.

References

[1] Jeffrey L. Elman, Finding structure in time, *Cognitive Science*, Vol. 14, pp. 179-211, 1990.

[2] Emergent documentation, http://grey.colorado.edu/emergent/index.php/User_hub.

[3] LaTeX documentation on wikibooks, http://en.wikibooks.org/wiki/LaTeX.