Assessing LSTMs' Ability to Predict Chinese Nominal Classifiers Based on Syntactic Cues

Chenyu Zhang

Abstract

The LSTM (long-short term memory) model is a popular recurrent neural network (RNN) used in deep learning and natural language processing, that serves as the foundation for many advanced models. Vanilla RNNs are known to suffer from difficulties of detecting long-range dependencies (such as English number-word agreement), and the design of LSTM aims to solve this problem with special mechanics such as the input, output, and forget gates.

In this work, we investigate whether LSTMs are able to predict a special form of long-range dependency, namely, the agreement between Chinese nouns and the numeral classifiers, specifically the sortal classifiers, with which each head noun is uniquely associated with. The model is able to make reasonable predictions that beat two of the baselines, but is unable to beat the strong baseline where the model doesn't need to handle the dependency as the most likely head noun is given directly instead of need to be inferred from the text sequence. We also manipulate the amount of intervention between the head noun and classifier position (gap for prediction), and discovered that more intervention makes it harder for the model to make the correct prediction. The results show the difficulty of the task. While the LSTM model does have its advantage, it is by no means perfect.

Introduction

Classifiers are function words that express quantity of nouns often used in Asian languages such as Chinese, Japanese, and Korean. Classifiers appear in quantifier phrases; one example is 'piece' in 'a piece of paper.' Most English nouns are not modified by classifiers (such as in 'a book'), but in Chinese, it is obligatory for nouns to be preceded by classifiers ('a book' is translated as 'yi ben shu', where the second word is the classifier).

Bond and Paik (2000) identified five major types of classifiers that have different properties—*sortal* (which classify the kind of noun they quantify, e.x. liàng, classifier for "vehicle"); *event* (used to quantify events, e.x. cì as "time"); *mensural* (which measure the amount of some property, e.x. mǐ as "meter"); *group* (which refer to a collection of members, e.x. zǔ as "group"); *taxonomic* (which force the noun to be interpreted as a generic kind, e.x. zhǒng as "kind"). It is common to simplify the classification to sortal and mensural classifiers. Each sortal classifier has a fixed association with a particular group of nouns, where the association is usually based on physical properties of the nouns (Tai 1994). For instance, the classifier 'pian' is associated with flat or sliced objects such as leaves. Mensural classifiers, on the other hand, hold a more contingent or temporary relationship with the nouns (Tai 1994). They are units of quantification that do not associate with particular nouns. It is easy to find English counterparts of mensural classifiers (e.x. 'a bowl of rice'), but not sortal classifiers. For the purpose of this study, we focus on sortal classifiers to make the prediction task easier.

The main task of this project is to test whether LSTM can learn from natural language to predict the correct Chinese classifier given an input sentence with a classifier gap. We mainly pose two questions here. First, LSTMs are known to be sensitive to syntactic structures. In the classifier prediction case, how well does the model capture syntactic dependencies? Does incorporating syntactic information help or hurt performance? If the model is able to successfully identify a head noun for the prediction task, the complete model would be expected to do better than a baseline model where the input is simply the closest noun (not necessarily the head noun) to the classifier. Another target of investigation is how much syntactic attraction affects the performance. Will the model do better when there is no intervening noun between the classifier gap and the head noun? How will the choice of the correct classifier be affected by the intervention? Those questions will provide some insights into LSTM's syntactic abilities.

While the experiment here in section 3 focuses on computer simulated prediction, it is also interesting to see how humans do when using classifiers. There have been human experiments focusing on children's acquisition of classifiers. According to Li, Huang, and Hsiao (2010), children start to learn classifier-noun associations at age three and can even generalize to novel objects. However, it is not easy to use classifiers perfectly. Learning the use of classifiers is a part of Chinese elementary school education—intuition is sometimes not sufficient even for native speakers. For Chinese language learners, learning classifiers can be very difficult. In this project, we will see how the neural network model face the same level of difficulty.

Background

Long short-term memory (LSTM) neural network is a language processing technique characterized by its ability to capture long-distance structural dependencies. This project is inspired by previous work (Linzen, Dupoux, & Goldberg, 2016) on the evaluation of the ability of LSTMs to learn syntax-sensitive dependencies, specifically measured by its performance on "number prediction tasks" in English subject-verb agreement. Languages that make use of numeral classifiers, such as Chinese, typically lack plural markings (Mok, Wen, Eshley, & Bond, 2012). Some Chinese classifiers, instead, function like English plural suffixes. Therefore, to extend the study of LSTMs into Chinese, we can use classifiers as the indicator of the incorporation of syntactic structure. Classifiers, however, are more complicated than verb number, since the former has many variations (196 identified in this project, including mensural and sortal) whereas the latter only has two (singular or plural). This higher number of classes plausibly makes classifier prediction harder than number prediction.

Even without syntactic structure, classifier-noun association on its own is a difficult task. There have been previous studies focusing on constructing models that assign a correct classifier to a given head noun. The most common approach was to predict classifiers based on ontologies and lexicons. Mok et al. (2012) and Bond and Paik (2000) associated classifiers with semantic classes and generated classifiers based on semantic hierarchies. They obtained relatively high generation score (around 80% in Mok et al. (2012)). Guo and Zhong (2005) considered SVMs with syntactic and ontological features; the best feature set results in a 58.71% accuracy for automatic evaluation and 94.2% for human evaluation where multiple classifiers can be correct.

But their task focused on the head noun as the predictor and concluded that contextual features do not have an effect.

Peinelt, Liakata, and Hsieh (2017) used distributed representations rather than ontological resources to train models for classifier prediction. They experimented with adding contextual features for head words and classifier gaps and achieved substantial improvements. They argued that context is important for classifier selection, because a head word may have multiple associated classifiers and the final selection can be restricted by the context. Their LSTM model excluded head noun annotation and reached micro f1 71.51 and macro f1 30.56, which could be compared to results obtained in this project.

Experiment – Classifier Prediction

The classifier prediction task is a multi-class classification task where each classifier is a class. In order to do this task, the model needs to do well in two parts: first identify the head noun associated with the classifier gap, and then match the correct classifier to the head noun.

Data is obtained from “The Lancaster Corpus of Mandarin Chinese” (McEnery & Xiao, 2004), an openly available part-of-speech tagged Chinese corpus. The character-based corpus contains over 45,000 sentences. 422 distinct classifiers are identified from the corpus, and after filtering out those with less than five appearances and those with more than one character (usually reduplicated words or mensural units), 196 remain. 12,944 sentences (sents_all) that contain at least one of those 196 classifiers are identified and split into train, dev, and test sets (4:1:1). In addition, 67 common sortal classifiers are manually identified and they correspond to 8044 sentences (sents_sortal), which are split in the same way. Each sentence then has its first classifier substituted by a classifier gap <CL> and the classifier will be the class label.

Example training sentence:
第四 天 的 早晨 ， 白素 醒来 后 ， 见 天 还 没有 亮 ， 便 在 床 上 躺 了 一会 ， 想 着 多多 的 许多 奇异 之 处 ， 忽然 之间 ， 心中 就 冒 出 了 那 <CL> 话 出来 。 句

The model consists of an encoder LSTM that encodes the gapped sentence and a linear classifier that predicts the class label based on the output of the encoder. The embedding size of the LSTM is 50 and the hidden state size is 256.

The strong baseline model is a ‘closest noun’ model, where the input is, instead of a whole sentence, simply the first noun that appears after the classifier. This noun is usually the head noun. By adopting this baseline, we can see how incorporating syntactic information help or hurt the task performance. Another baseline is the ‘ge’ baseline, where every sentence gets assigned the universal classifier ‘ge’. In the case of the 67 sortal classifiers, the accuracy obtained by always predicting ‘ge’ is 25.2%. We also consider the ‘random guessing’ baseline, where the probability of guessing each classifier is 1/67 = 1.5%, thus the accuracy for this baseline is 1.5%.

To evaluate the model, an overall accuracy is obtained using the test data set. Then, in order to test the effects of the attractor nouns (nouns that occur between the classifier and the head noun), an attraction test is designed where three types of handcrafted test sentences are used. The three types are: 1, sentences where the head noun directly follows the classifier (no intervention at all), 2, sentences with no intervening noun but may have intervening adjectives, 3, sentences with one intervening noun. Example test sentences are shown in table 1. For each type, the probability of getting the correct class label is compared with the probability of getting the classifier for the attractor. We expect the probability of getting the correct classifier to be higher in type 1 and type 2 sentences than in type 3 sentences. An ideal model that can learn syntactic dependencies would have high accuracy even for type 3 sentences.

Type 1	我 有 一 <CL> 石头 。 I have a <CL> stone.
Type 2	我 有 一 <CL> 巨大 的 石头 。 I have a <CL> huge stone.
Type 3	我 有 一 <CL> 海滩 上 捡来 的 石头 。 I have a <CL> beach got from stone. (I have a stone that I got from the beach.)
Correct / attractor classifier	块 / 片

Table 1: example test sentences
'块' is the classifier for '石头' (stone) whereas '片' is the classifier for '海滩' (beach)

Results

The test set accuracies of the models and the baselines are shown in table 2. The overall model accuracy is relatively low. One reason is that many classifier gaps can be filled in with multiple classifiers, but there is only one correct answer in the test set. The test set accuracy for sents_all is lower than that of sents_sortal because 'all classifiers' include mensural words, and mensural classifiers are not associated with specific nouns, meaning that one noun can predict multiple mensural classifiers that all make sense.

Because the accuracy for sents_all is quite low, most of the rest of the experiment is conducted with sents_sortal and the 67 selected sortal classifiers. Thus, the closest noun baseline is only trained on classifiers and nouns from sents_sortal; the attraction test is only done with the model trained on sents_sortal.

The test set accuracies of both sents_all and sents_sortal are higher than the respective 'ge' baselines and much higher than the 'random guessing' baselines, but is lower than the accuracies of the 'closest noun' baseline, suggesting that while the model is able to learn to some extent to identify the head noun and associate the classifier with that noun, such identification of head noun is not very successful.

A closer look at the predicted classifiers reveals that the model is biased to predict ‘ge’ most of the time, which is why the accuracy is close to the ‘ge’ baseline. The tendency to predicting ‘ge’ may be because ‘ge’ is much more common than any other classifiers in the limited training data and thus is learned the most.

Table 3 shows the comparison between the probabilities of the correct classifier and the classifier of the attractor for the three types of sentences. (See the code for a complete list of test sentences.) I trained the model twice and obtained similar results. The differences between the probabilities across the three types of sentences are small, but we do see that type 1 sentences have the highest probabilities and type 3 have the lowest. This fits with the expectation that type 1 sentences are the easiest and type 3 sentences are the hardest. Model predictions are distracted by the attractor between the classifier and the head noun in type 3 sentences.

It is interesting to notice from table 3 that the first trained instance of the model always has higher probability for the correct label than the attractor label, but the opposite is true for the second trained instance. It seems that second model instance is more fooled by the attractors at least for this particular set of test sentences.

	all	sortal
Complete model	15.35%	31.87%
Ge	14.97%	26.49%
Random guess	0.51%	1.49%
Closest noun		38.34%

Table 2: test set accuracies
‘all’ means using all 196 classifiers and ‘sortal’ means using 67 selected sortal classifiers. ‘Complete model’ is the model trained on classifier gapped sentences; the other three are baseline models. The closest noun baseline for ‘all’ was not computed.

	Correct label	Attractor label
Type 1	0.0717	0.0705
Type 2	0.0714	0.0706
Type 3	0.0708	0.0707
average	0.0713	0.0706

	Correct label	Attractor label
Type 1	0.0915	0.0994
Type 2	0.0909	0.1022
Type 3	0.0899	0.1031
average	0.0908	0.1016

Table 3: probabilities of the correct and attractor label for the type 1 (no intervention), 2 (adjective intervention), 3 (noun intervention) sentences using two model instances trained with sents_sortal

Discussion

From results shown in table 2, we can see that the model does better than ‘ge’ and ‘random guessing’ baselines, suggesting that it is able to learn something from the syntactic information to predict the correct classifier. But it does worse than the ‘closest noun’ baseline, showing that providing full-sentence context does not improve performance over providing the closest noun alone. The complete model does not do well in identifying the head noun from the given sentence. This seems to contradict results from previous studies, especially Peinelt et al. (2017), where the LSTM model without marked head noun performed the best. One explanation is the difference in the model used for the baseline; they used SVM and Logistic Regression for their head noun model, whereas here for the ‘closest noun’ baseline I used the same LSTM encoder plus linear classifier set up for consistency. This may explain why the ‘closest noun’ baseline here does better than the complete model, but the Peinelt et al. (2017) head noun models does worse. Note that although this project used the closest noun instead of the head noun, it is reasonable to guess that a model trained with the head noun and the classifier will get even higher accuracy.

The test set accuracy of the model is lower than the accuracy obtained by the LSTM model in Peinelt et al. (2017) for several reasons. First, the data size for this project is much smaller; while only 8,044 sentences are used in this project, Peinelt et al. (2017) has 858,472 sentences for training. Second, this project does not use any pre-trained word embedding as in Peinelt et al. (2017). Third, the scale of the LSTM is smaller (256 hidden units in this project compared to 480 hidden units in Peinelt et al. (2017)). With larger data size, better word embeddings, and a larger-scale model, we should be able to replicate the results in Peinelt et al. (2017).

Furthermore, the designed test shows the effects of intervening words between the classifier gap and the head noun. When there is one intervening word, the prediction task becomes harder than without; intervening nouns are harder than adjectives. In order for the model to do better in capturing classifier-noun dependency relationship and to better predict the correct classifier, the problem of intervening words need to be solved.

Conclusion & Future Work

This project investigates whether LSTMs can learn structural dependencies that have many variations, specifically Chinese nominal classifiers. Although LSTM models have been successful in a variety of tasks, we have seen here that the model experiences difficulty in the classifier prediction task, where training data is limited. The result shows that it is difficult to predict the correct classifiers based solely on sequential information without annotated head nouns. The model clearly experiences difficulty in identifying the head noun, because it performs worse than the ‘closest noun’ baseline where a ‘fake’ head noun is given. On the bright side, it is not much worse than the baseline, meaning that it is still able to capture some syntactic information. The model experiences more difficulty in making correct associations between nouns and classifiers, which we can see from the overall low accuracy including the ‘closest noun’ baseline. This is due to the variability of classifiers and the lack of training data.

At the beginning of the experiment, I considered limiting the training and testing data to nouns that have only one choice of classifier. This would significantly improve the classification result. Doing so would require parsing the sentences to get the head nouns. Moreover, most Chinese nouns can be associated with multiple classifiers, thus restricting the data would further decrease the training size and limit the model’s ability to generalize.

In natural language, most of the head nouns immediately follow the classifier. This may explain the relatively high accuracy of the ‘closest noun’ baseline. To see how reliable the closest noun is as the head noun, we can parse the corpus to identify the head noun and match them with the closest noun for each sentence to see the percentage of matches. (In fact, I have tried parsing the corpus with the Stanford Constituency Parser, but it takes too long and I was short of time.) We can also adopt a ‘head noun’ baseline to see how much it can do better than the ‘closest noun’ baseline. This can be compared with the complete model to see the model’s ability to capture syntactic dependencies.

Besides attractors between the classifier and the head noun, nouns after the head noun may interfere as well. In fact, in the current setup, any noun in the sentence may interfere with the prediction of the classifier. This is what makes this task really hard. Some of the classifiers have only five training samples possibly with different head nouns, making it impossible for the model to capture the head noun and the classifier-noun association. Thus, a larger dataset is necessary. It would also be interesting to do a more detailed analysis on which noun attractors (the ones before the classifier, between the classifier and the head noun, or after the head noun) have the greatest effect on prediction.

Whereas the training data size is small as in a machine learning experiment setting, it is definitely large enough for human learners to acquire the use of the classifiers. Young children learn to use sortal classifiers with limited linguistic input and can generalize the classifier-noun associations to novel nouns based on physical properties with high accuracy (Li et al., 2010). On the other hand, our language model cannot capture the classifier-noun associations based on limited data, let alone generalizing to novel nouns. Thus, for neural network models to catch up with human intelligence, there is still a long way to go.

Appendix

The Colab Notebook with the experiment code is available at:
https://drive.google.com/open?id=1raoexAjUmj_A_8JIp4FQ14xyoHo1wAd3

References

Bond, F., & Paik, K. (2000). Re-using an ontology to generate numeral classifiers. In 18th international conference on computational linguistics (pp. 90–96). Saarbrucken, Germany.
Guo, H., & Zhong, H. (2005). Chinese classifier assignment using svms. In Proceedings of the fourth sighan workshop on chinese language processing.
Li, P., Huang, B.,&Hsiao, Y. (2010). Learning that classifiers count: Mandarin-speaking children's acquisition of sortal and mensural classifiers. Journal of East Asian linguistics, 19, 207–230.
Linzen, T., Dupoux, E., & Goldberg, Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. Transactions of the Association for Computational Linguistics, 4, 521–535.
McEnery, A., & Xiao, Z. (2004). The lancaster corpus of mandarin chinese: A corpus for monolingual and contrastive language study.
Mok, H., Wen, S., Eshley, G. H., & Bond, F. (2012). Using wordnet to predict numeral classifiers in chinese and
japanese. In Gwc 2012 6th international global wordnet conference (p. 264–271).
Peinelt, N., Liakata, M., & Hsieh, S.-K. (2017). Classifierguesser: A context-based classifier prediction system for
chinese language learners. In Companion volume of the ijcnlp 2017 proceedings: System demonstrations (p. 41-44).
Tai, J. H.-Y. (1994). Chinese classifier systems and human categorization. In honor of William S.-Y. Wang: Interdisciplinary studies on language and language change, 479—494.