Papaer Synopsis

Jie He

March 1, 2017

Abstract

The three sections in this report respectively summarize the main ideas as well as some of my own ideas on the three papers: End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures (Miwa and Bansal, 2016), Coherent Dialogue with Attention-based Language Models (Mei, Bansal, and Walter, 2016), A Joint Speaker-Listener-Reinforcer Model for Referring Expressions (Yu, Tan, Bansal, and Berg, 2016).

1 End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures (Miwa and Bansal, 2016)

1.1 Problem Statement

Extraction of semantic relations between two entities plays an important role in information extraction and natural language processing. In a sentence where two entities are presented, the relation between the two entities can be inferred by the context words. One prevailing approach to automatically extract this relation is using neural network model which builds on recurrent/recursive neural networks (RNNs) or convolutional neural networks (CNNs).

But the previous RNN-based models which use long short term memory (LSTM) don't take full use of the linguistic information in the sentence, such as word sequence and dependency tree structure, and don't perform entity detection and relation extraction jointly. Based on these weaknesses, this paper presents a new model which applies both word sequence and dependency tree structure, and models entities and relations jointly.

1.2 Model

The model proposed by this paper mainly consists of three layers: embedding layer, sequence layer and dependency layer.

Embedding layer: This layer handles different type of embedding representations for each word, such as word embedding, part-of-speech(POS) embedding, dependency type embedding and entity label embedding. We can get these embeddings from this layer when we need them in the subsequent layers.

Sequence layer: This layer includes a bidirectional (forward and backward) LSTM-RNNs. It takes the concatenation of word and POS embeddings as input, and outputs the concatenation of the hidden state vectors in both directions to represent the word sequence.

Entity detection: Over the sequence layer, entity detection is realized by a two-layered neural network (NN) which consists of a hidden layer and a softmax output layer. For a given word in the sentence, this NN receives the concatenation of its output vector from the sequence layer and the output vector of its previous word on this NN. The output vector is the entity tag (using the BILOU encoding schema) for the current word.

Dependence layer: This layer, which is based on the bidirectional (top-down and bottom-up) tree-structured LSTM-RNNs, is used to generate the relation vector for a given pair. The existing tree-structured LSTMs do not divide the children into different types, so the authors propose a new variant of tree-structured LSTMs which allows variable number of children types, and where children of the same type share the weight matrices. To find the better structure to model the relation in a pair, the authors consider three structure options: 1) the shortest path tree (SP-Tree), which takes only the nodes on the shortest path between the two entities in a pair and is widely used in relation classification models. 2) SubTree, which takes all the nodes below the lowest common ancestor node. 3) FullTree, which takes the full dependency tree. A type mapping function is defined to map the two types of nodes (on shortest

paths or not) to their weight matrices for SubTree and FullTree. For a given node (word), the input of this layer is the concatenation of its output from sequence layer, dependency type embedding from embedding layer and its output from the entity detection layer. The output is the relation vector which takes the concatenation of the hidden state vector of the lowest common ancestor node and the hidden state vectors of the last words of the two entities.

The LSTM unit for t-th node use following equations:

$$i_t = \sigma\left(W^{(i)}x_t + \sum_{l\in C(t)} U^{(i)}_{m(l)}h_{tl} + b^{(i)}\right), \tag{1a}$$

$$f_{tk} = \sigma\left(W^{(f)}x_t + \sum_{l\in C(t)} U^{(f)}_{m(k)m(l)}h_{tl} + b^{(f)}\right), \tag{1b}$$

$$o_t = \sigma\left(W^{(o)}x_t + \sum_{l\in C(t)} U^{(o)}_{m(l)}h_{tl} + b^{(o)}\right), \tag{1c}$$

$$u_t = \tanh\left(W^{(u)}x_t + \sum_{l\in C(t)} U^{(u)}_{m(l)}h_{tl} + b^{(u)}\right), \tag{1d}$$

$$c_t = i_t \odot u_t + \sum_{l\in C(t)} f_{tl}\odot c_{tl}, \tag{1e}$$

$$h_t = o_t \odot \tanh(c_t), \tag{1f}$$

where $C(t)$ is the children and $m(\cdot)$ is the type mapping function.

Relation classification: To extract and classify the relation between the detected entities, the model firstly builds all possible combinations among the detected entities and then for each of these combination (pair) puts the corresponding path as input on the dependence layer and gets the relation candidate vector. To take into account more information from the sequence layer and the entity, this relation candidate vector is also concatenated with the mean vectors of all words in each entity from the sequence layer. The new relation candidate vector is then passed to a two-layered NN (with a hidden layer and a softmax output layer) to predict the relation.

Training: During the training, the model applies back-propagation through time (BPTT), Adam, gradient clipping, L2-regularization and dropout to update weights and biases, adapt learning rate, prevent exploding gradients, regularize weights and reduce overfitting. Besides, scheduled sampling and entity pretraining are applied as enhancements.

1.3 Results and Analysis

The model is evaluated on three datasets: ACE05 and ACE04 for relation extraction, and SemEval-2010 Task 8 for relation classification. ACE05 defines 7 coarse-grained entity types ad 6 coarse-grained relation types. ACE04 defines 7 coarse-grained entity types and 7 coarse-grained relation types. SemEval2010 Task 8 defines 9 relation types and a tenth type *other*, and consists of 8,000 training and 2,717 test sentences. During the evaluation, several ablation tests are performed to show the influence of different structure options and the effect of sharing parameters. The evaluation metrics used in the tests are micro-F1, micro-precision and micro-recall, and macro-F1. The results show gains over the state-of-art performance on ACE04 and ACE05 (entity detection and relation extraction) with appropriate tree structure (SP-Tree) and enhancements, as well as comparable scores on SemEval-2010 Task 8 (relation classification) to previous models.

1.4 My Questions and Ideas

Questions:

1) Are the embeddings of POS tags, dependency types and entity labels one-hot vectors?

2) What is "parameter averaging" during the training?

3) Does the $\uparrow h_{p_A}$ in $d_p = [\uparrow h_{p_A}; \downarrow h_{p_1}; \downarrow h_{p_2}]$ is the concatenation of the hidden state vectors of the last nodes on the left and right bottom-up paths?

4) Have the authors tried a max pooling which gathers the information from all the hidden state vectors (Xu et al., 2015) to generate the relation candidate vectors, instead of taking the hidden state vectors of the last nodes on the paths?

Ideas:

In some of the cases, there can be more than one relation between two entities, i.e., *Paris is located in France* and *Paris is part of France*, where the relation between Paris and France can be both *PHYS* and *PART-WHOLE*. Inspired from the model in the paper, is it possible to add certain algorithm to deal with mulit-relation entities? For example, if the output of a relation candidate is like following: $R1 - 0.15, R2 - 0.1, R3 - 0.3, R4 - 0.32, R5 - 0.08, R6 - 0.05$ where R_i is the relation class and the decimal is the probability. If the entity pair actually has two relations ($R3$ and $R4$) but the annotated relation class is $R3$ instead of $R4$, this correct predication will be missed although there is just a tiny difference (0.02) in the output distribution. Is it worthwhile to investigate this problem? I was thinking that it would be useful if there is a threshold δ and the output takes all the relation classes whose probability is bigger than $P_{max} - \delta$, where P_{max} is the biggest probability in the output.

2 Coherent Dialogue with Attention-based Language Models (Mei, Bansal, and Walter, 2016)

2.1 Problem statement

Dialog systems are important to many applications such as chatbots, technical assistants, and customer service agents. Recently, a lot of work focusing on building a more natural and coherent dialog model has been proposed, including using the recurrent neural newwork (RNN) sequence-to-sequence framework to train the dialog models and regarding the whole conversation as a single sequence.

The authors in this paper think that it's more appropriate to use a language model to train dialog model instead of a sequence-to-sequence model, since a language model can learn how the conversation continues when new responses generate, while a sequence-to-sequence model only learns how the most recent response is generated and is more suitable to transform information from one form to another such as translation. To train a dialog language model with higher coherence in the continued conversations, this paper proposes an attention-based neural language model which applies a dynamic attention mechanism to capture the most related words in the whole conversation history (including the currently generated sentences).

2.2 Model

This section introduces the basic language model (RNN-LM), the sequence-to-sequence model (RNN-Seq2Seq) and the application of attention mechanism.

RNN-LM deals a sentence as a set of tokens. The next token in a sentence is predicted by the softmax output function which takes the hidden state vector of the current token and the vectors of all the tokens in the vocabulary to compute distribution. RNN-LM is trained by maximizing the log-likelihood function which indicates the average sum of the log probability of each token in each trained sentence.

RNN-Seq2Seq consists of encoder and decoder. The encoder takes a sequence (all the tokens in a source sentence) as input and encodes them into a set of hidden states which is then used to predict the target sequence (target sentence) token-by-token in the decoder.

Attention in RNN-Seq2Seq Models. These models include an attention module, which takes the sequence of hidden states from the encoder and the hidden state of previous step from the decoder. The calculation of attention is based on the previous hidden state from decoder and the whole set of hidden states from encoder. The attention is represented by a context vector which is then passed (together with the vectors of already predicted tokens) to decoder to predict the next token.

Attention in RNN-LM. The attention-RNN language model (A-RNN-LM) proposed in this paper firstly encodes the entire existing context into a sequence. Each token is represented by the concatenation of its word embedding and hidden state. Then the attention module takes as input the hidden state of the current token and the sequence of all token representations and yields the context vector. The context vector and the hidden state of the next token are then passed to the output function to predict the next token. This model not only learns how the conversation evolves but also relates each predicted token to the entire history. besides, the attention information can be visualized since it's a weighted sum over the hidden states of the history.

The equations in the attention module and the output function in this model are as following:

$$\beta_{ti} = b^\top \tanh(W h_{t-1} + U r_i), \tag{2a}$$

$$\alpha_{ti} = \exp(\beta_{ti}) / \sum_{i=0}^{t-1} \exp(\beta_{ti}), \tag{2b}$$

$$z_t = \sum_{i=0}^{t-1} \alpha_{ti} r_i, \tag{2c}$$

where r_i is token representation. b, W, and U are model parameters.

$$g(h_t, z_t, v_j) = O_{v_j}^\top (O_h h_t + O_z z_t), \tag{3a}$$

$$P(w_t = v_j | w_{0:t-1}) = \frac{\exp g(h_t, z_t, v_j)}{\sum_i \exp g(h_t, z_t, v_i)}, \tag{3b}$$

where O_{v_j} is a output vector corresponding to v_j.

2.3 Experiments and Results

The tests are performed on *MovieTriples* and *Ubuntu Troubleshoot* using different kinds of evaluation metrics such as perplexity (PPL), word error rate (WER), recall@N, Distinct-1, BLUE and etc. Besides, the authors choose Latent Dirichlet Allocation (LDA) based re-ranker to evaluate the similarity between the original conversation and its continuations in topic coherence test. Based on the two datasets, the results show that the model has the best performance on the evaluations of primary dialog modeling, generation diversity and topic coherence, as well as preliminary human evaluation, compared with other models such as the plain vanilla RNN language model, the RNN-Seq2Seq model and etc.

2.4 My Ideas

Considering that the existing dialog systems are pure data-driven models and have evolved quickly in different tasks, also inspired by the attention-RNN language model which shows good performance in long-distance memory and information association, I was thinking about building a ***task-driven*** dialog system which also incorporates data-driven model.

The reasons for *task-driven* are: 1) To guide the model on what to say, what seems interesting for it and when to end. 2) To explore the application of dialog modeling in real-world. In my opinion, this task-driven mechanism can be realized by constructing an ordered check list for the model to inform it what to say next. For example, if we build an Ubuntu technical support agent, we might construct a check list which needs the agent to finish for current customer. This check list can be a set of key phrases, e.g., *check question type, check system information, check solutions, check results, and end service*. (This list might also be trained from expert documents in this domain.) Once the dialog system has this check list, it will follow the key phrases one-by-one to gather and process the information provided by customer and search possible solutions until the problem is solved or the customer gives up.

One of the main problems in this task-driven agent model is that how the model can verify all the information it needs to solve the problem and how it processes this information gathered from customer.

One possible solution is using topic-matching to make sure the information provided by customer is relevant, and using attention-RNN language model to make the conversation coherent. Another difficult is that how the model finds precise solutions for given problem description and additional information (Ubuntu version, kernel version, and etc.). Maybe the attention mechanism can also be applied to get inspiration for this problem. Other problems include the lack of prepared datasets.

Is it worthwhile to investigate the feasibility of this idea?

3 A Joint Speaker-Listener-Reinforcer Model for Referring Expressions (Yu, Tan, Bansal, and Berg, 2016)

3.1 Introduction

Referring expressions are very commonly used in daily life, which includes two parts, a speaker who gives referring expression about certain object and a listener who interprets it and points out the corresponding object. This can also be computationally modeled by a Referring Expression Generation (REG) which models the listener and a Referring Expression Comprehension (REC) which models listener. Recent approaches can be classified into two types. One uses a CNN-LSTM encoder-decoder model to generate expressions for a given object while the other uses a joint-embedding model which transforms both the visual representations of an object and its referring expressions into an embedding space and then calculates their distance.

This paper proposes an unified model to perform both the comprehension and expression tasks. This model consists of a CNN-LSTM speaker, a embedding-based listener and a reward-based reinforcer.

3.2 Model

The model consists of three modules.

Specker: This module uses a CNN-LSTM framework where CNN is a pre-trained model which gives a visual representation of the given object. The visual representation is a concatenation of the 4 features, i.e., the target object, context, location/size and two visual comparison features, and is then passed to a multi-layer perception (MLP) in the listener. The module then concatenates the output from the MLP with the original vector (from the CNN) as feature vector to feed a LSTM and generates most likely referring expression. This module is trained by minimizing the negative log-likelihood function. The max-margin Maximum Mutual Information (MMI) is also introduced and incorporated with two hinge losses over a triplet (one positive match and two negative matches) as a training option:

$$\begin{aligned} L_2^s(\theta) = -\sum_i [&\lambda_1^s \max(0, M + \log P(r_i, o_k) - \log P(r_i, o_i)) \\ &+\lambda_2^s \max(0, M + \log P(r_j, o_i) - \log P(r_i, o_i))], \end{aligned} \tag{4}$$

where the two losses mean that, for each training pair, they tend to minimize both the probability of current expression for a random wrong object, and the probability of a random wrong expression for current object, while maximize the probability of current expression for current object.

Listener: The aim of this module is to convert the pair of visual representation and referring expression into a joint embedding space and force them to be similar during the training, and find the most similar one which comes from visual representation during the comprehension task. This module receives referring expression as input and encodes it by a LSTM. Then a MLP (two fully connected layers with ReLU between them) and a L2 normalization is added to the encoded expression vector and the visual representation, respectively. Thus the pair is projected to a common embedding space. A hinge loss over triplet is defined to train the parameters in the MLPs and the LSTM:

$$\begin{aligned} L^l(\theta) = -\sum_i [&\lambda_1^l \max(0, M + \log P(r_i, o_k) - \log P(r_i, o_i)) \\ &+\lambda_2^l \max(0, M + \log P(r_j, o_i) - \log P(r_i, o_i))], \end{aligned} \tag{5}$$

Reinforcer: This module is used to make the speaker generate less ambiguous expressions by a learned reward function. During the training, the reinforcer samples words (including <EOS> token) step-by-step from the softmax output of the speaker's LSTM to construct the expression. Since the ambiguity of the sampled expression (non-differentiable operation) remains unknown until testing it with the reward function, this module performs a policy gradient to update the parameters (of the speaker):

$$\nabla_\theta J = E_{p(w_{1:T})}[F(w_{1:T})\nabla_\theta \log p(w_{1:T};\theta)], \tag{6}$$

where $\log p(w_{1:T})$ is from the softmax output of the speaker's LSTM and $F(w_{1:T})$ is the score of the sampled expression from the reward function.

To train the reward function, the authors take the object and expression pairs as input and encode the expression by another LSTM. The output of this LSTM is then concatenated with the visual representation of the object and a MLP (logistic regression) adds to this concatenation to get a match score (between 0 and 1). The LSTM and MLP here are trained with cross-entropy loss.

Joint model: The whole model is a joint framework for the reasons: 1) In the listener module, the output from the MLPs for the visual representation embedding is reused in the speaker module (concatenated with the original visual representation) as an additional input to the LSTM to generate expressions, which means the speaker has information from the listener. 2) The sampled triplets in both the hinge loss of the speaker and the listener are the same. 3) The word embeddings are shared between the speaker and the listener. 4) The sampled expressions in the reinforcer come from the speaker's LSTM.

The whole model is regarded as a multi-task model where the loss function:

$$\theta = \arg\min L_1^s(\theta) + L_2^s(\theta) + L^l(\theta) - J(\theta), \tag{7}$$

where $L_1^s(\theta)$ is the negative log-likelihood function in speaker, $L_2^s(\theta)$ is the MMI (generalized by the authors), $L^l(\theta)$ is the triplet loss function in listener and $J(\theta)$ is the reward function in reinforcer.

3.3 Experiments and Discussion

Three referring expression datasets, RefCOCO, RefCOCO+ and RefCOCOg, are used during the experiments.

Comprehension task: The model can use either the speaker module or the listener module to perform this task. (The listener predicts the target object by choosing the closest object embedding in its learned embedding space. The speaker can generate expressions for all the objects in the image and predict the object by finding the best match with the target expression.) During the task, the intersection-over-union (IoU) with threshold of 0.5 is applied to decide whether a prediction is correct. Besides, the authors run different ablation tests to study the effect of each module. The results show that all speaker modules trained with MMI has better performance than without MMI, and speaker module can be improved with joint training and incorporating with reinforcer module. Besides, listener generally works better than speaker in comprehension task and reinforcer module has a limited performance.

Generation task: To perform generation, the speaker firstly generates a set of expression by beam search, then the listener reranks them and chooses the less ambiguous one. The authors also introduce cross compression and expression diversity to fully use the listener. During the generation task, METEOR and CIDEr are used as metrics. The results show that the speakers in jointly trained model get higher scores and the full model with "rerank" gets the highest scores.

3.4 My Questions and Ideas

Questions:

1) Is it a typo for the unnecessary minus sign in the loss functions $L_2^s(\theta)$ in speaker module and $L^l(\theta)$ in listener module? It seems no sense with minus and there is no minus in the definition of max-margin MMI (Mao et al., 2015).

2) During the training of the reward function in the reinforcer, are the input pairs ground-truth pairs of object and expression or the combinations of all possible objects and expressions?

3) Have the authors tried to perform a generation and comprehension task? This task means the model firstly generates the referring expressions on the test dataset and then uses these genereated expressions to feed the listener to point out the original objects.

4) The best score of TestA in RefCOCO(detected) for ablation study using the speaker module for comprehension task (Yu et al., 2016, Table 1) should be 72.95% from ***speaker***+*listener*+*MMI* instead of 72.88% from ***speaker***+*listener*+*reinforcer*+*MMI*.

Ideas:

1) About the reinforcer

From the results, I noticed the performance of the reinforcer is limited in most of the ablation tests, e.g., six out of seven of scores from *speaker*+***listener***+*reinforcer*+*MMI* even drop a little when comparing with *speaker*+***listener***+*MMI* in the ablation study using listener or ensembled listener+speaker modules for the comprehension task (Yu et al., 2016, Table 2). And the evaluation is not performed for the ablation ***speaker***+***listener***+*MMI* (ensembled listener+speaker without reinforcer).

Since the listener has a good performance on comparing the two embeddings (one from the visual representation and the other from the expression), is it possible to use another individual listener as the reward function instead of the existing one? Because an individual listener can be trained independently with ground-truth object and expression pairs. Besides, it might be reasonable to assume that this individual listener can also perform well on computing the similarity (reward score) between the object embedding and the expression embedding.

2) A framework only focusing on semantic translation

Inspire from the unified framework in this paper which associates referring expressions with objects in image, can we extend this framework to one only focusing on the semantic part of the referring expression? Specifically, we simply substitute all the entities with different entity identifications in the framework proposed in this paper.

In my opinion, this new framework will only focus on learning the association of semantic reference with its representation in the image, e.g., the expression *Object A is in front of object B* will only teach the model how the reference "in front of" means visually and the expression *the back of woman with a green coat and black purse* only needs to teach the model about "of", "with" and "and" in an image.

I intuitively think this framework will be more efficient since it only focuses on the visual representation of the reference part in the expressions. And as the image object recognition/classification has already reached a high level. It is possible to integrate these models directly into the framework to perform referring expression generation and comprehension tasks. To integrate existing object detection models, maybe the multi-model can give some inspiration (Hendricks et al., 2016). Is it worthwhile to further develop this idea?

References

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. Deep compositional captioning: Describing novel object categories without paired training data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. *CoRR*, abs/1511.02283, 2015. URL `http://arxiv.org/abs/1511.02283`.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Coherent dialogue with attention-based language models. *arXiv preprint arXiv:1611.06997*, 2016.

Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL 2016*. arXiv preprint arXiv:1601.00770, 2016.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying relations via long short term memory networks along shortest dependency path. *CoRR*, abs/1508.03720, 2015. URL http://arxiv.org/abs/1508.03720.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. A joint speaker-listener-reinforcer model for referring expressions. *arXiv preprint arXiv:1612.09542*, 2016.