Paper Report

NGO HO Anh Khoa

November 13, 2017

Contents

Part I
Paper report

1 Unsupervised Neural HMMs [6]

1.1 Main idea

This research show how to apply unsupervised hidden Markov model in neural network approach. In fact, they would like to prove that a simple nn models trained to maximize the marginal likelihood could outperform more complicated models in unsupervised learning.

1.2 What is their concentration ?

- There are three components:
  - Set of latent variables Z (Tags)
  - Set of observed variables X (Words)
  - Model parameters θ (Emission and transitions probability)
- Purpose: Find θ which maximize p(X $|\theta$)
- How: Use Generalized EM to estimate θ
  1. : Maximizing $p(X)$ means

$$p(x) = \sum_z p(X, Z) = E_{q(Z)}[\ln p(X, Z|\theta)] + H[q(Z)] + KL(q(Z)||p(Z|X, \theta)) \tag{1}$$

  2. E-step: Estimate $p(Z|X)$ based on current θ
  3. M-step: Consider $q(Z) = p(Z|X)$
     $KL(q(Z)||p(Z|X, \theta)) = 0$
     $H[q(Z)]$ constant
  4. Maximizing $p(X)$ becomes maximizing $E_{q(Z)}[\ln p(X, Z|\theta)]$
- Result: Gradient of the joint probability scaled by the posteriors

$$J(\theta) = \sum_Z p(Z|X) \frac{\partial}{\partial \theta} \ln p(X, Z|\theta) \tag{2}$$

- Problem: How to calculate $p(X, Z)$?

1.3 What is the role of HMM ?

- Assumption:
  - Every word token is generated by a latent class (Tag)
  - The current class at time t is conditioned on the previous class at time (t - 1)
- Therefore, the probability of a given sequence of observation X and latent variables Z (Factorization of the joint probability):

$$p(X, Z) = \prod_{t=1}^{n+1} p(z_t|z_{t-1}) \prod_{t=1}^{n} p(x_t|z_t) \tag{3}$$

- Result: Combine $p(X, Z)$ (3) and gradient $J(\theta)$ (2)

$J(\theta) = \sum_Z p(Z|X) \frac{\partial}{\partial \theta} \ln p(X, Z|\theta)$

$= \sum_Z p(Z|X) \frac{\partial}{\partial \theta} \ln[\prod_{t=1}^{n+1} p(z_t|z_{t-1}, \theta) \prod_{t=1}^{n} p(x_t|z_t, \theta)]$

$= \sum_Z p(Z|X) \frac{\partial}{\partial \theta} [\sum_{t=1}^{n+1} \ln p(z_t|z_{t-1}, \theta) + \sum_{t=1}^{n} \ln p(x_t|z_t, \theta)]$

$= \sum_t \sum_{z_t} p(z_t, z_{t-1}|X) \frac{\partial}{\partial \theta} \ln p(z_t|z_{t-1}, \theta) + p(z_t|X) \frac{\partial}{\partial \theta} \ln p(x_t|z_t, \theta)$

$$J(\theta) = \sum_t \sum_{z_t} p(z_t, z_{t-1}|X) \frac{\partial}{\partial \theta} \ln p(z_t|z_{t-1}, \theta) + p(z_t|X) \frac{\partial}{\partial \theta} \ln p(x_t|z_t, \theta) \tag{4}$$

- Problem:
  - How to calculate $p(z_t, z_{t-1}|X)$ and $p(z_t|X)$? They propose Baum-Welch
  - How to calculate $p(z_t|z_{t-1}, \theta)$ and $p(x_t|z_t, \theta)$? They propose Neural Networks

1.4 Where is neural network ?

- Input: A sentence $X = x_1, ..., x_t, ...x_{L_x}$, a set of vocabulary $W = w_1, ..., w_i, ...w_{L_W}$ a set of tags $Z = z_1, ..., z_j, ...z_{L_z}$
- Output: $p(z_t|z_{t-1}, \theta)$ and $p(x_t|z_t, \theta)$ at each time t.
- How:

1. Embedding X and Z by θ: Vector embedding of W v_W (Using CNN - Convolution for Morphology) and vector embedding of Z v_Z (Simple feed-forward nn having a lookup table following by a non-linear activation ReLU). v_W and v_Z have the same dimension.
2. Calculate $p(x_t|z_t, \theta)$ (Emission matrix): Probability of a word w_i in Vocabulary is generated by a tag z_j (Do not care about time t).

$$p(w_i|z_j) = \frac{exp(v_{z_j}^T * v_{w_i} + b_i)}{\sum_{w \in W} exp(v_{z_j}^T * v_w + b)} \tag{5}$$

3. Calculate $p(z_t|z_{t-1}, \theta)$ (Transition matrix): Probability of a tag z_j at time t is generated by a tag $z_{j'}$ at time (t - 1).
   Input: Vector of word w at x_t noted v_{x_t}. It is query embedding (Using LSTMs)
   Result: Matrix of $L_z * L_z$ noted T. It means all transition probabilities of each tag at (t - 1) to all tags at time t.

$$T = U^T * v_{x_t} + b \tag{6}$$

2 What does Attention in NMT pay Attention to ? [2]

2.1 Main idea

This research compares between Attention Models (Non-recurrent attention model/ Global attention and Recurrent attention/ Input-feeding model) and known Word Alignment. The result is that their differences depends on the word type being generated.

2.2 How to compare ?

Higher consistency between Attention and Alignment leading to better translation.

- **Spearman's rank correlation between attention quality** (Attention loss compared known human alignment) **and translation quality** (Word prediction loss). Higher correlation means a closer relationship between translation quality and consistency of attention versus alignment.
  - Attention loss

$$L_{At}(outToken) = -\sum_{inToken} Al(in, out) * log(At(in, out)) \tag{7}$$

    * Al(in,out): Weight of alignment link between input token and output token

        * At(in,out): Weight of attention between input token and output token
    - Word prediction loss: Softmax()

- Attention concentration: Entropy of attention distribution (Soft-hard attention problem)

$$E_{At}(outToken) = -\sum_{in} At(in, out) * log(At(in, out)) \tag{8}$$

    - At(in,out): Weight of attention between input token and output token

2.3 Analysis

The analysis is based on POS tags experiments.

- Impact of Attention between Non-recurrent (NR) and Recurrent attention (IF): IF has lower AER (Hard attention) and Attention loss (Soft attention).
- Translation quality: Consistency between attention and word alignment depends on POS tags. For example,
    - There is a higher consistency in the case of nouns. However, attention captures other information in the case of verbs.
    - Translation quality of Verbs is better than Nouns. It means attention does not follow alignment for translating Verbs.
- Attention concentration: Review the case of Verbs and Nouns.
    - Nouns have a lower attention entropy (Higher concentration), lower attention loss (Closer to Alignment), which is that attention entropy can be used as a measure of closeness of attention to alignment in the case of nouns
    - Verbs have a lower correlation between attention entropy and word prediction loss, which means that attention concentration is not necessary for translating verbs.
- Attention distribution: It shows how a POS tag of target sentence depends on other POS tags of source sentence.

3 Confidence through Attention [4]

3.1 Main idea

Auto-evaluation metric without reference. This research is that attention distribution becomes a confidence metric (Translation quality and Decoder confidence)

- Filtering out bad translation from a large back-translated corpus (Provide a better parallel corpus)
- Selecting the best translation in a hybrid setup of 2 translation systems

The result is that this metric could be consider as an human judgement (Not so true!, just about 50%) and leads to BLEU score improvement (in some cases).

3.2 How to calculate Confidence metric ?

Penalty measures: Coverage deviation and Absentmindedness

- Coverage Deviation Penalty: Lacking attention and Too much attention per input token, which mean penalizing the sum of attention per input token for going to far from 1.0 (Why 1.0: Replaced by token's expected fertility)

$$CDP = -\frac{1}{inSentLen} \sum_{inToken} log(1 + (1 - \sum_{outToken} \alpha_{out-inToken})^2) \quad (9)$$

- Absentmindedness Penalty (Entropy): The attention of confident output tokens should concentrate on a small number of input tokens and vice versa (Assumption).

$$AP_{out} = -\frac{1}{inSentLen} \sum_{inToken} \sum_{outToken} \alpha_{out-inToken} * log(\alpha_{out-inToken}) \quad (10)$$

- Combination:

$$Confidence = CDP + AP_{out} + AP_{in} \quad (11)$$

3.3 Analysis

3.3.1 Comparison with human evaluation

They use Kendall rank correlation coefficient for looking at the pairs where human scores differ. They recognize that their metric over-penalizes the translations which do not follow the source word-by-word.

3.3.2 Exp: Filtering Back-translated Data

They compare their confidence metric with language model method in filtering the best translated sentences. Both methods have the similar levels of overlapping the human evaluation. One point should be considered is that their metric does not require any additional model (LM).
For BLEU score, their method show a better performance on some cases, which is in general insignificant.

3.3.3 Exp: Hybrid Decisions

The difference between two baseline systems influences on the final BLEU score. A small difference leads to the small improvements, a large difference causes a score drop. It is well-reported that hybrid selection overlaps about 50% human selection.

4 Word Translation without Parallel data [1]

4.1 Main idea

The research is about **building a bilingual dictionary without parallel data** by aligning monolingual word embedding spaces in a unsupervised way (GAN) and proposing a similarity metric CSLS.
Its results show a strong performance of using Procrustes-CSLS. GAN in this case is a step necessary to overcome unsupervised learning.

4.2 How does it works ?

Input: Two large monolingual corpora.
Output: Linear mapping W between the source and target space. Two steps of training:

1. Training GAN:

   - A discriminator distinguishes between n mapped source embeddings and m mapped target embeddings.
     Objective function:

     $$Loss_D(\theta_D|W) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}(source = true|Wx_i) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}(source = false|y_i) \tag{12}$$

   - A generator creates these embeddings to fool discriminator.
     Objective function:

     $$Loss_W(W|\theta_D) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}(source = false|Wx_i) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}(source = true|y_i) \tag{13}$$

   - Update parameter:
     Orthogonality advantages: Reservation of monolingual quality of the embeddings (Dot product of vectors or distances, rotation $\rightarrow$ An isometry of the Euclidean space); Stable training)

     $$W \leftarrow (1+\beta)W - \beta(WW^T)W \tag{14}$$

2. Refinement procedure (Solution for rare words that GAN does not well solve) repeats until reaching stopping condition.

(a) Extracting a synthetic high-quality dictionary (Most frequent words) evaluated by CSLS

(b) Applying Procrustes solution for generating more accurate dictionary.

3. Stopping condition/Best hyper-parameters selection - Validation step: Similarity measure CSLS between mapped source and target words.
Why: It is based on K-NN and overcomes a problem of two spaces and "Hubs and Anti-hubs" (Some points are highly near many other points while there are some points are not nearest any point in high-dimensional spaces).
They proposed a bipartite neighbourhood graph. Each word of a language is connected to K words of an other language.

$$CSLS(Wx_s, y_t) = 2\cos(Wx_s, y_t) - r_T(Wx_s) - r_S(y_t) \tag{15}$$

- $r_T(Wx_s) = \frac{1}{K}\sum_{y_t \in N_T(Wx_s)} \cos(Wx_s, y_t)$
Mean similarity of a source embedding x_s to its target neighbours.
- $r_S(y_t) = \frac{1}{K}\sum_{Wx_s \in N_S(y_t)} \cos(y_t, Wx_s)$
Mean similarity of a source embedding x_s to its target neighbours.
- $N_T(Wx_s)$: Neighbour of a source word Wx_s in target word space.

5 Word Alignment Modeling with Context Dependent Deep Neural Network [8]

5.1 Main idea

The research describes how to use Context Dependent Deep NN in HMM-based word alignment model. This means that bilingual word embedding could capture not only lexical translation and context from surrounding words. An example is translating a rare word by using its surrounding words.
It is noted that this version uses a smaller number of parameters than the classic HMM model.

5.2 How does DNN work in word alignment ?

The role of DNN in this paper is learning automatically feature from raw text. They start from the factorization of the joint probability [7].

$$p(a, e|f) = \prod_{i=1}^{|e|} P_{lex}(e_i|f_{a_i}) P_d(a_i|a_{i-1}) \tag{16}$$

- Given a sentence pair (e,f)
- P_{lex}: Lexical translation probability, emission probability

- P_d: HMM alignment probability [7], transition probability. The paper has a different notation: $P_d(a_i - a_{i-1})$ and it is called Jump distance distortion probability (Why: The alignment probability depends only on the jump width $(a_i - a_{i-1})$ [7]).

They propose using a score instead of these probabilities because they would like to avoid the softmax normalization step of a large vocabulary. The formula above becomes:

$$s_{NN}(a|e,f) = \prod_{i=1}^{|e|} t_{lex}(e_i, f_{a_i}|e,f) t_d(a_i, a_{i-1}|e,f) \tag{17}$$

- f_{a_i} represents not only this word but the context around this word. Surrounding words of both source and target word are input of DNN, which is handling the context of both sides. This could reduce the explosion of parameter number.
  In this case, they use fixed length windows surrounding both e_i and f_j.

- t_{lex}: Lexical translation score.

$$t_{lex}(e_i, f_j|e,f) = functions_{NN}(window(e_i), window(f_j)) \tag{18}$$

- t_d: Distortion score.

$$t_d(a_i, a_{i-1}|e,f) = t_d(a_i - a_{i-1}|window(f_{a_{i-1}})) = functions_{NN}(window(f_{a_{i-1}})) \tag{19}$$

  They recognize that this lexicalized distortion does not produce a better alignment. They reverse to the simple version.

$$t_d(a_i, a_{i-1}|e,f) = t_d(a_i - a_{i-1}) \tag{20}$$

5.3 Result

6 Recurrent Neural Networks for Word Alignment Model [5]

6.1 Main idea

This research is mainly based on [8]. The difference is that the alignment score is calculated in recurrent approach RNNs, which means that a_j depends on all previous position $a_{0...j-1}$.

6.2 How does RNNs work in calculating alignment score ?

Part II
Report

7 Overview about statistical alignment [3]

7.1 General problem

Input:

- A source sentence: $f_1^J = f_1, ..., f_j, ...f_J$
- A target sentence: $e_1^I = f_1, ..., f_i, ...f_I$

Output: Alignment map $j \rightarrow i = a_j$

- An alignment: $a_1^J = a_1, ..., a_j, ...a_J$

7.1.1 How to model a word alignment ?

Our work is modelling the relationship between a source sentence and a target sentence.
We start from the view of translation model, which is finding this best translation e_1^I for a source sentence f_1^J.

$$e_1^I = argmax_{e_1^I} p(e_1^I|f_1^J) = argmax_{e_1^I} \frac{p(f_1^J|e_1^I)p(e_1^I)}{p(f_1^J)} \tag{21}$$

In this case, the translation direction is changed from $p(e_1^I|f_1^J)$ to $p(f_1^J|e_1^I)$. This should be noted while reading alignment papers.

From now, the work of translation is modelling $p(f_1^J|e_1^I)$. Our work is more complicated by adding an alignment component a_1^J which maps from a source position j to a target position a_j (It could be i, but there is also empty words).

$$p(f_1^J|e_1^I) = \sum_{a_1^J} p(f_1^J, a_1^J|e_1^I) \tag{22}$$

With the model parameters θ, our main problem becomes $p_\theta(f_1^J, a_1^J|e_1^I)$

7.2 Hidden Markov Alignment Model

The alignment model is re-structured:

$$p(f_1^J, a_1^J|e_1^I) = p(J|e_1^I) \prod_{j=1}^{J} p(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \tag{23}$$

$$= p(J|e_1^I) \prod_{j=1}^{J} p(f_j|f_1^{j-1}, a_j, e_1^I) * p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \tag{24}$$

For this new structure, there are three different probabilities:

- $p(J|e_1^I)$: Length probability
- $p(f_j|f_1^{j-1}, a_j, e_1^I)$: Lexicon probability
- $p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I)$: Alignment probability

We need some assumptions to put this model into HMM:

- $p(f_j|f_1^{j-1}, a_j, e_1^I) \rightarrow p(f_j|e_{a_j})$: The lexicon probability depends only on the word at position a_j
- $p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \rightarrow p(a_j|a_{j-1}, I)$: The alignment a_j depends on a_{j-1} (First-order dependence)
- $p(J|e_1^I) \rightarrow p(J|I)$: Simplify this probability (Not because of HMM, it's just simplification)

Therefore, $p(f_1^J, a_1^J|e_1^I)$ is decomposed under these assumptions as follows:

$$p(f_1^J, a_1^J|e_1^I) = p(J|I) \prod_{j=1}^{J} p(f_j|e_{a_j}) * p(a_j|a_{j-1}, I) \tag{25}$$

From this formula, we need to calculate these two components:

- $p(f|e)$: Translation probability
- $p(a_j|a_{j-1}, I)$ or $p(i|i', I)$: HMM alignment probability

7.2.1 Assumption about HMM alignment probability $p(a_j|a_{j-1}, I)$ or $p(i|i', I)$

Alignment probability depends on the difference in the alignment positions rather than on the absolute position [7].

$$p(i|i', I) = \frac{c(i - i')}{\sum_{i''=1}^{I} c(i'' - i')} \tag{26}$$

where $c(i - i')$ is non-negative.

7.2.2 Extension: Empty word in target sentence

An source word could give an empty word, which is that each target word has an extra empty word. This leads to the length of target sentence being e_1^{2I} and the empty word zone being e_{I+1}^{2I}. The word e_i has an empty word e_{i+I}.
They enforce the constraints:

$$p(i + I|i', I) = p_0\delta(i, i') \tag{27}$$

$$p(i + I|i' + I, I) = p_0\delta(i, i') \tag{28}$$

$$p(i|i' + I, I) = p(i|i', I) \tag{29}$$

where p_0 is the probability of a transition to the empty word.

7.2.3 Extension: Refinement of alignment

- Purpose: Add more assumptions that $p(a_j|a_{j-1}, I)$ depends on $e_{a_{j-1}}$ or f_j.
- Problem: In the case of large corpora or large size of vocabulary, it leads to a large alignment parameters.
- Solution: They use classes G which are the mappings of words to classes.

8 Word alignment model in Neural Network

We start from this formula 25:

$$p(f_1^J, a_1^J|e_1^I) = p(J|I) \prod_{j=1}^{J} p(f_j|e_{a_j}) * p(a_j|a_{j-1}, I) \tag{30}$$

In general, we try to calculate these probability by using two different neural networks. The significant point of NN is Context.

8.1 Non-probability approach, we use score

The papers include [8], [5].

9 Evaluation of a translation/alignment

1. Alignment error rate
2. Attention concentration: Entropy [2], Absentmindedness penalty [4]
3. Attention loss: Compared with known soft alignment
4. Confidence metric - Auto-evaluation: Coverage deviation penalty and Absentmindedness penalty [4]

References

[1] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. Word Translation Without Parallel Data. *ArXiv e-prints*, October 2017.

[2] H. Ghader and C. Monz. What does Attention in Neural Machine Translation Pay Attention to? *ArXiv e-prints*, October 2017.

[3] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March 2003.

[4] M. Rikters and M. Fishel. Confidence through Attention. *ArXiv e-prints*, October 2017.

[5] Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. Recurrent neural networks for word alignment model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1470–1480, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[6] M. Ke Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. Proceedings of the workshop on structured prediction for nlp. pages 63–71. Association for Computational Linguistics, 2016.

[7] Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[8] Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. Word alignment modeling with context dependent deep neural network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–175. Association for Computational Linguistics, 2013.