Paper Report

NGO HO Anh Khoa

December 5, 2017

Contents

Part I
Paper report

1 Unsupervised Neural HMMs [8]

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 ? [3]

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 [6]

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 judgment (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 [2]

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

5 Alignment by Agreement [4]

5.1 Main idea

This approach is similar to ensemble method. In this unsupervised approach, the two simple asymmetric models are trained jointly to maximize a combination of data likelihood and agreement between the models. The new point is encouraging the agreement between these models during training.
In fact, intersecting the alignment prediction of two asymmetric models reduces AER. An assumption is that this intersection (Agreement) could reduce the loss of each asymmetric model.
Moreover, we could see the transformation from asymmetric models to symmetric model.

5.2 How to train these two model ?

- Two models are:

$$\begin{aligned} p_1(x, z; \theta_1) &= p(e)p(a, f|e; \theta_1) \\ p_2(x, z; \theta_2) &= p(f)p(a, e|f; \theta_2) \end{aligned} \tag{16}$$

where
$x = (e, f)$ is an input sentence pair,
$z = z_{i,j} \in \{0, 1\} : 1 \leq i \leq I, 1 \leq j \leq J$ is a set of indicator variables for each potential alignment edge.

- The data likelihood:

$$\prod_x p_k(x;\theta_k) = \prod_x \sum_z p_k(x,z,\theta_k) \tag{17}$$

  where $k \in 1,2$ is models.

- **Joint objective function** is maximizing the data likelihood:

$$\max_{\theta_1,\theta_2} \sum_x [logp_1(x;\theta_1) + logp_2(x;\theta_2)] \tag{18}$$

  Because of the probability that two models agree the alignments on an example x being

$$\sum_z p_1(z|x;\theta_1)p_2(z|x;\theta_2) \tag{19}$$

- Therefore,

$$\max_{\theta_1,\theta_2} \sum_x [logp_1(x;\theta_1) + logp_2(x;\theta_2) + log \sum_z p_1(z|x;\theta_1)p_2(z|x;\theta_2)] \tag{20}$$

5.2.1 Optimization via EM

From the standard EM,

$$\begin{aligned} &E : q(z;x) := p(z|x;\theta) \\ &M : \theta^{'} := argmax_\theta \sum_{x,z} q(z;x) \quad log\, p(x,z;\theta) \end{aligned} \tag{21}$$

becomes

$$\begin{aligned} &E : q(z;x) := \frac{1}{Z_x} p_1(z|x;\theta_1)p_2(z|x;\theta_2) \\ &M : \theta^{'} := argmax_\theta \sum_{x,z} q(z;x)\, log\, p_1(x,z;\theta_1) + \sum_{x,z} q(z;x)\, log\, p_2(x,z;\theta_2) \end{aligned} \tag{22}$$

where Z_x is a normalization constant. It is sum of the product of two model posteriors over the set of possible z with nonzero probability under both models.

In E-step,

$$E : q(z;x) := \frac{1}{Z_x} p_1(z|x;\theta_1)p_2(z|x;\theta_2) \tag{23}$$

becomes

$$q(z;x) := \prod_{i,j} p_1(z_{ij}|x;\theta_1)\, p_2(z_{ij}|x;\theta_2) \tag{24}$$

where each $p_k(z_{i,j}|x;\theta_k)$ is the posterior marginal probability of the (i,j) edge.

This posterior $q(z;x)$ could show the disagreement because this product $p_1(z|x;\theta_1)p_2(z|x;\theta_2)$ is small.

5.3 Analysis

This agreement approach does not guarantee to increase the joint objective. The experiment shows that it could provide an effective method of achieving model agreement and a significant accuracy gains over independent training.

6 Word Alignment Modeling with Context Dependent Deep Neural Network [10]

6.1 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.

6.2 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 [9].

$$p(a, e|f) = \prod_{i=1}^{|e|} P_{lex}(e_i|f_{a_i}) P_d(a_i|a_{i-1}) \tag{25}$$

- Given a sentence pair (e,f)
- P_{lex}: Lexical translation probability, emission probability
- P_d: HMM alignment probability [9], 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 $(a_i - a_{i-1})$ [9]).

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:

$$s_{NN}(a|e, f) = \prod_{i=1}^{|e|} t_{lex}(e_i, f_{a_i}|e, f) t_d(a_i, a_{i-1}|e, f) \tag{26}$$

- $s_{NN}(a|e, f)$ is the score of an alignment based on the source and target sentence.

- 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.

$$t_{lex}(e_i, f_j|e, f) = functions_{NN}(window(e_i), window(f_j)) \tag{27}$$

- t_d: Distortion score.

$$t_d(a_i, a_{i-1}|e, f) = t_d(a_i - a_{i-1}|window(f_{a_{i-1}})) = functions_{NN}(window(f_{a_{i-1}})) \tag{28}$$

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

$$t_d(a_i, a_{i-1}|e, f) = t_d(a_i - a_{i-1}) \tag{29}$$

6.3 Loss in supervised learning

$$loss(\theta) = \sum_{every(\mathbf{e},\mathbf{f})} \max(0, 1 - s_\theta(a^+|\mathbf{e}, \mathbf{f}) + s_\theta(a^-|\mathbf{e}, \mathbf{f})) \tag{30}$$

where

- a^+: Gold alignment path
- a^-: Highest scoring incorrect alignment path under θ
- s_θ: Score for alignment path defined in (26)

6.4 Analysis

6.4.1 Two loss functions during training

They pre-train word embedding with monolingual data. They train firstly NN with the loss from lexical translation score and then the loss (30) because they recognize that using only the loss (30) is not efficient.
The loss of lexical translation score:

$$loss(\theta) = \sum_{every(\mathbf{e},\mathbf{f})} \max(0, 1 - t_{lex,\theta}((e, f)^+|\mathbf{e}, \mathbf{f}) + t_{lex,\theta}((e, f)^-|\mathbf{e}, \mathbf{f})) \tag{31}$$

where

- $(e, f)^+$: Correct word pair
- $(e, f)^-$: Incorrect word pair
- t_{lex}: Score of (27)

6.4.2 Result

Baselines are HMM and IBM4. IBM4+NN takes the first place, followed by IBM4 and then HMM+NN.

6.4.3 Notes

They conclude also that the size of window influence on the accuracy of translation score. The large number of hidden layer does not return any improvement.

7 Recurrent Neural Networks for Word Alignment Model [7]

7.1 Main idea

This research is mainly based on [10]. The difference is that the score of alignment is calculated in recurrent approach RNNs, which means that a_j depends on all previous position $a_{0...j-1}$. The score of an alignment in [10] has two components: One is for lexical translation and the other is for alignment. This research proposes a single score.

7.2 How does RNNs work in calculating alignment score ?

The equation of [10] (Reverse the source and the target sentence)

$$s_{NN}(a_1^J|e_1^I, f_1^J) = \prod_{j=1}^{J} t_d(a_j, a_{j-1}|window(e_{a_{j-1}})) * t_{lex}(f_j, e_{a_j}|window(e_{a_j}), window(f_j)) \tag{32}$$

It is modified for RNNs:

$$s_{NN}(a_1^J|e_1^I, f_1^J) = \prod_{j=1}^{J} t_{RNN}(a_j|a_1^{j-1}, e_{a_j}, f_j) \tag{33}$$

7.3 Loss in unsupervised learning

$$loss(\theta) = max(0, 1 - \sum_{(\mathbf{f}^+, \mathbf{e}^+) \in T} E_\Phi[s_\theta(\mathbf{a}|\mathbf{f}^+, \mathbf{e}^+)] + \sum_{(\mathbf{f}^+, \mathbf{e}^-) \in \Omega} E_\Phi[s_\theta(\mathbf{a}|\mathbf{f}^+, \mathbf{e}^-)]) \tag{34}$$

where

- T: Training data (Bilingual sentences) as Observed data
- Ω : Full translation search space as neighborhood of observe data, .
- Φ : Set of all possible alignments given ($\mathbf{e}$,$\mathbf{f}$)

- $E_\Phi[s_\theta]$: Expected value of the scores s_θ on Φ
- e^+ : A target language sentence in T
- e^- : A pseudo-target language sentence
- $\sum_{(\mathbf{f}^+,\mathbf{e}^+)\in T} E_\Phi[s_\theta(\mathbf{a}|\mathbf{f}^+,\mathbf{e}^+)]$: Expectation term for observed data
- $\sum_{(\mathbf{f}^+,\mathbf{e}^-)\in \Omega} E_\Phi[s_\theta(\mathbf{a}|\mathbf{f}^+,\mathbf{e}^-)])$: Expectation term for neighborhood

They would like to reduce the computation, they randomly select N pseudo-target language sentences for each f^+. (34) becomes:

$$loss(\theta) = \sum_{\mathbf{f}^+\in T} max(0, 1 - E_{GEN}[s_\theta(\mathbf{a}|\mathbf{f}^+,\mathbf{e}^+)] + \frac{1}{N}\sum_{\mathbf{e}^-} E_{GEN}[s_\theta(\mathbf{a}|\mathbf{f}^+,\mathbf{e}^-)]) \tag{35}$$

where

- $\mathbf{e}^-$: a pseudo-target language sentence with the same length $|\mathbf{e}^-| = |\mathbf{e}^+|$
- GEN: A subset of all possible alignment generated by beam search

7.4 Agreement constraints

See [4]

Why: HMM-based model is asymmetric. It is demonstrated that encouraging directional models to agree improves alignment performance.
How: They propose training two directional models including alignment F→E and E→F. Each model has a different modified loss function

$$loss_{\theta_{F\to E}} = loss(\theta_{F\to E}) + \alpha \parallel \theta_{E\to F} - \theta_{F\to E} \parallel \tag{36}$$

$$loss_{\theta_{E\to F}} = loss(\theta_{E\to F}) + \alpha \parallel \theta_{F\to E} - \theta_{E\to F} \parallel \tag{37}$$

where

- $\theta_{E\to F}$: Weight of layers in model $E \to F$
- α: Weight parameter of the agreement constraint $\parallel \theta_{E\to F} - \theta_{F\to E} \parallel$
- $\parallel \theta \parallel$: Norm of θ
- $loss(\theta_{F\to E})$: Loss from (34)

7.5 Analysis

Baselines are IBM4, FFNN [10]. They do the experiments of word alignment and machine translation in both unsupervised/supervised learning.

7.5.1 Result

RNNs captures alignment paths based on long alignment history. This can be viewed as phrase-level alignment, which is more effective in non-similar-order-language (Japanese-English) than French-English.

RNNs could help to reduce the size of training data. They show the similar results between RNNs-Small data and IBM4-Large data.

They highlight the role of agreement constraints in alignment performance improvement.

8 Hierarchical Multiscale Recurrent Neural Networks [1]

8.1 Main idea

Hierarchical Multi-scale RNNs learns the latent hierarchical structure of a sequence and its temporal representation with non-fixed timescales (Without explicit boundary information).
For example, this model consider a sentence as a list of characters, not only words. It is trained to know autonomously the beginning and the end of a word. For each layer, it could group the units.
The key idea is an update mechanism with a binary boundary detector (Gate determining the boundary of an object).

The advantages of "Multi-scale" over standard RNNs:

- Updating the high-level layers is less frequently → Computational efficiency and Vanishing gradient problem. Solving vanishing gradient problem means that this model could delivering efficiently long-term dependencies.
- Flexible resource allocation. The model could modify the number of hidden units for an information. For example, the model could use more units to the higher layers (Long-term dependency) and less units to the lower layers (Short-term dependency). This means that it could decide to keep the long-term or short-term information.
- A good generalization performance (Dropout approach)
- The internal process of RNN becomes more interpretable. We could count the number of operation execution time (Update, Flush and Copy operation)

The problem of the previous kind of RNNs:

- LSTMs: LSTMs has also an update mechanism with forget and update gates, which means that it could operate with different timescales. However, these timescales are not organized hierarchically.
  Problem:
  - In practice, after a few hundred time steps, long-term information is gradually diluted at every time step.
  - LSTMs is computationally expensive (Update at every time step for each unit)
- Clockwork RNNs and its soft timescale approach: The hidden units are grouped into several modules. Different timescales are assigned to these modules (i.e: A module i is updated at every 2^{i-1} -th time step).
  Problem:
  - It is not easy to find a good timescales because timescale depends on data.
  - It is unclear how to deploy more layers than the number of boundary levels that is explicitly observed in the data.
- Hierarchical RNNs with Known explicit hierarchical boundary structure (Not multi-scale).
  Problem: It is the same with Clockwork RNNs that boundary information is not explicit and expensive to be discovered.
- CNNs with 1-D kernels: It could provide also high-level representation of sequence.
  Problem: We need to observe a full sequence with a reduced length.
- Zoneout, a regularization technique similar to dropout.
  Problem: Its strength is still an hyper-parameter.

8.2 Update mechanism

They describe Hierarchical Multi-scale RNN (HM-RNN) based on LSTM update rule. It is better to see the graph in paper.

Why it is not based on RNN ?

At each time step t, for each layer l, there are:

$$h_t^l,\, c_t^l,\, z_t^l = f^l(c_{t-1}^l, h_{t-1}^l, h_t^{l-1}, h_{t-1}^{l+1}, z_{t-1}^l, z_t^{l-1}) \tag{38}$$

The function includes Update, Copy and Flush operation depending on binary detector - boundary states z_{t-1}^l (Previous time step) and z_t^{l-1} (Below layer). The difference between three operations are the use of context c which keeps the summary information of the previous units.

- **Flush**: It shows the beginning of a segment or "word", the boundary is found previously.

The unit h_t^l does not get information from same-level context c_{t-1}^l (Eject) but higher-level unit h_{t-1}^{l+1} (This is the connection between two words, it is re-installed with more long-term information from previous words).

- **Copy**: It shows not-end state of a "word", none of boundary is found. the unit h_t^l only copies from previous unit h_{t-1}^l, its context copies from c_{t-1}^l (It reduces the number of operation execution, leading to computational efficiency).

- **Update**: It is similar to update operation of LSTM (Forget, input and output gate) but it is executed sparsely. It is the end of a "word", the boundary is found in the unit below.
  The unit h_t^l gets also information from the same-level context c_{t-1}^l and information from the lower-level unit h_t^{l-1}, but not from the higher-level unit h_{t-1}^{l+1} (It does not need information from previous words).

8.3 Analysis

The result for character-level language model does not show clearly a good performance.

9 Hybrid Neural Network alignment and Lexicon Model in direct HMM for Statistical Machine Translation

Part II
Report

10 Overview about statistical alignment

10.1 Word-based alignment ? [5]

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: $a_1^J = a_1, ..., a_j, ...a_J$

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.

$$e_1^I = argmax_{e_1^I} p(e_1^I|f_1^J) = argmax_{e_1^I} \frac{p(f_1^J|e_1^I)p(e_1^I)}{p(f_1^J)} \tag{39}$$

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 modeling $p(f_1^J|e_1^I)$. 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).

(40)

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

$$p_\theta(f_1^J|e_1^I) = \sum_{a_1^J} p_\theta(f_1^J, a_1^J|e_1^I) \tag{41}$$

10.2 Hidden Markov Alignment Model [5]

The alignment model is re-structured:

$$p(f_1^J, a_1^J|e_1^I) = p(J|e_1^I) \prod_{j=1}^{J} p(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \tag{42}$$

$$= p(J|e_1^I) \prod_{j=1}^{J} p(f_j|f_1^{j-1}, a_j, e_1^I) * p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \tag{43}$$

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:

$$p(f_1^J, a_1^J|e_1^I) = p(J|I) \prod_{j=1}^{J} p(f_j|e_{a_j}) * p(a_j|a_{j-1}, I) \tag{44}$$

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

10.2.1 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 [9].

$$p(i|i', I) = \frac{c(i - i')}{\sum_{i''=1}^{I} c(i'' - i')} \tag{45}$$

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

10.2.2 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:

$$p(i+I|i',I) = p_0\delta(i,i') \tag{46}$$

$$p(i+I|i'+I,I) = p_0\delta(i,i') \tag{47}$$

$$p(i|i'+I,I) = p(i|i',I) \tag{48}$$

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

10.2.3 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.

11 Word alignment model in Neural Network

We start from this formula (41):

$$p_\theta(f_1^J|e_1^I) = \sum_{a_1^J} p_\theta(f_1^J, a_1^J|e_1^I) \tag{49}$$

11.1 Probability approach

11.1.1 [8]

Unsupervised neural HMM of [8] shows how to apply NN in HMM. In general, we try to calculate the probability by using two different neural networks.The significant point of NN is embedding context.

1. They use Generalized EM to estimate θ, (49) becomes:

   $p_\theta(f_1^J|e_1^I) = \sum_{a_1^J} p_\theta(f_1^J, a_1^J|e_1^I)$
   $= E_{q(a_1^J)}[lnp_\theta(f_1^J, a_1^J|e_1^I)] + H[q(a_1^J)] + KL(q(a_1^J)\|p_\theta(a_1^J|f_1^J, e_1^I))$

2. Updating θ requires only maximizing $E_{q(a_1^J)}[lnp_\theta(f_1^J, a_1^J|e_1^I)]$.
   The gradient is:

   Why?: See [8]

   $J(\theta) = \sum_{a_1^J} \quad q(a_1^J) \quad \bigtriangledown log\, p_\theta(f_1^J, a_1^J|e_1^I)$

3. In the step E, we model $q(a_1^J) = p_{\theta'}(a_1^J|f_1^J, e_1^I)$. Therefore,

$$J(\theta) = \sum_{a_1^J} \quad p_{\theta'}(a_1^J|f_1^J, e_1^I) \quad \bigtriangledown \, log\, p_\theta(f_1^J, a_1^J|e_1^I)$$

4. From (44), we have
$p(f_1^J, a_1^J|e_1^I) = \prod_{j=1}^J p(f_j|e_{a_j}) * p(a_j|a_{j-1})$ with the assumptions about I, J. Therefore,

$$\begin{aligned}
&J(\theta) = \sum_{a_1^J} \quad p_{\theta'}(a_1^J|f_1^J, e_1^I) \quad \bigtriangledown \, log\, p_\theta(f_1^J, a_1^J|e_1^I) \\
&= \sum_{a_1^J} \quad p_{\theta'}(a_1^J|f_1^J, e_1^I) \quad \bigtriangledown \, log\, [\prod_{j=1}^J p_\theta(f_j|e_{a_j}) * p_\theta(a_j|a_{j-1})] \\
&= \sum_{a_1^J} \quad p_{\theta'}(a_1^J|f_1^J, e_1^I) \quad \bigtriangledown \, [\sum_{j=1}^J log\, p_\theta(f_j|e_{a_j}) + log\, p_\theta(a_j|a_{j-1})] \\
&= \sum_{j=1}^J \sum_{a_j} p_{\theta'}(a_j|f_j, e_{a_j}) \bigtriangledown log\, p_\theta(f_j|e_{a_j}) \quad + p_{\theta'}(a_j, a_{j-1}|f_j, e_{a_j}) \bigtriangledown log\, p_\theta(a_j|a_{j-1})]
\end{aligned}$$

5. We choose a is state, f is observation. We remove e. Therefore,

$$J(\theta) = \sum_{j=1}^J \sum_{a_j} p_{\theta'}(a_j|f_j) \bigtriangledown log\, p_\theta(f_j|e_{a_j}) \quad + p_{\theta'}(a_j, a_{j-1}|f_j) \bigtriangledown log\, p_\theta(a_j|a_{j-1})]$$

How to apply NN ?

- Emission matrix $p_\theta(f_j|e_{a_j})$: It is not the normal way of translation. We receive the target-language sentence and return the probabilities for all source-language vocabulary.
- Transition matrix $p_\theta(a_j|a_{j-1})$: We receive the target-language sentence and return this matrix.
- The posteriors $p_{\theta'}(a_j|f_j)$ and $p_{\theta'}(a_j, a_{j-1}|f_j)$: We set source-language sentence as a set of observation, Baum-Welch algorithm returns forward and backward messages (How is the size of these messages)

What is the problem of transition matrix in programming ? The input of transition model is target sentence. The model returns the matrix with its size being $(I * I)$.
However, this matrix is hard to return. This is why I return matrix $(I * V_e)$ with V_e is the size of target language vocabulary. From this matrix, I could select the probabilities of the words existed in target sentence. The final result is a matrix with its size being $(I * I)$.
This method breaks the structure of the tensor tree. The cost calculated can not be back-propagated because there is no relationship between this cost and this matrix $(I * V_e)$.

11.2 Non-probability - Score approach

11.2.1 [10], [7]

12 Symmetrical alignment

13 Techniques in Alignment

13.1 Agreement between models

14 Evaluation of a translation/alignment

1. Alignment error rate
2. Attention concentration: Entropy [3], Absentmindedness penalty [6]
3. Attention loss: Compared with known soft alignment
4. Confidence metric - Auto-evaluation: Coverage deviation penalty and Absentmindedness penalty [6]

References

[1] J. Chung, S. Ahn, and Y. Bengio. Hierarchical Multiscale Recurrent Neural Networks. *ArXiv e-prints*, September 2016.

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

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

[4] Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.