CS584: Natural Language Processing **Fall 2019**

Lecture 9: Machine Translation, Seq2seq models, Attention models

Lecturer: Yue Ning *Scribes: Junzhe Wang*

9.1 Machine Translation

Machine Translation (MT) is the task of translating a sentence x from one language (the source language) to a sentence y in another language(the target language).

9.1.1 History of Machine Translation

- 1950s: Early Machine Translation(Rule-based)

  Russian → English (motivated by the Cold War)

  Systems were mostly rule-based, using a bilingual dictionary to map Russian words to their English counterparts

- 1990s - 2010s: Statistical Machine Translation (Statistics-based)

  Core idea: Learn a probabilistic model from data

  The best systems were extremely complex

- 2014 - : Neural Machine Translation

9.1.2 Statistical Machine Translation

The core idea is to learn a probabilistic model from data. The probabilistic model is constructed from a language model which tells us how likely a given sentence/phrase is overall, and a translation model which tells us what a sentence/phrase in a source language most likely translates into. Given French sentence f, we want to find best English Sentence e, mathematically:

$$P(e|f) = \frac{P(e)P(f|e)}{P(f)}$$

$$T(f) = \hat{e} = argmax_e P(e|f) = argmax_e P(e)P(f|e)$$

These components were used to build translation systems based on words or phrases. There is additional complexity due to different sentence lengths and word orders in the languages. The best systems were extremely complex in several aspects. The Systems have many separately-designed subcomponents. They need large amount of parallel data, which needs lots of human effort to maintain.

9.1.3 Alignment for Statistical Machine Translation

To learn translation model $P(x|y)$ from the parallel corpus, we actually want to consider

$$P(x, a|e)$$

where a is the alignment, i.e. word-level correspondence between French sentence x and English sentence y. Alignment is the correspondence between particular words in the translated sentence pair. We have to note that some words have no counterpart [1]. Alignment is complex, because it can be many-to-one, one-to-many, or many-to-many.

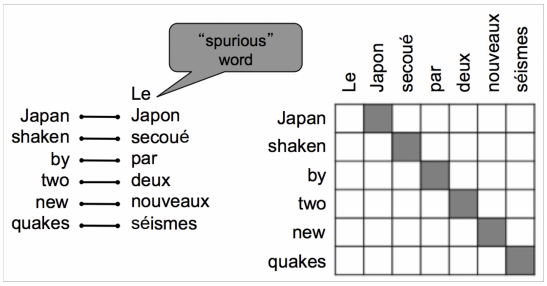

Figure 9.1: Words with no counterpart

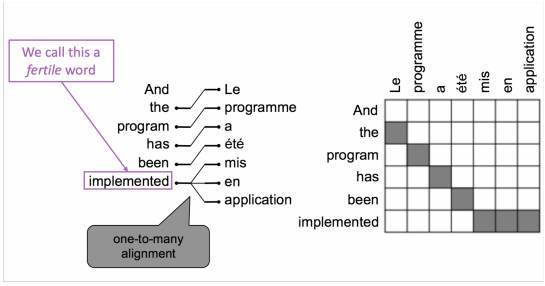

Figure 9.2: One-to-many

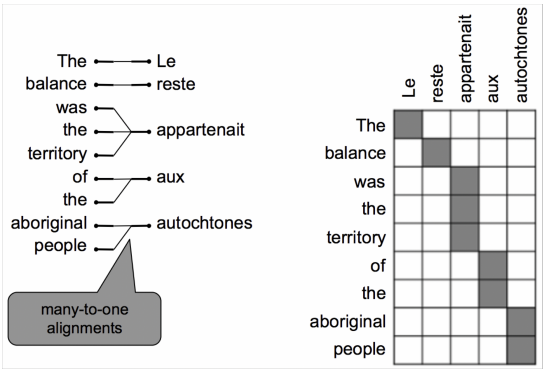

Figure 9.3: Many-to-one

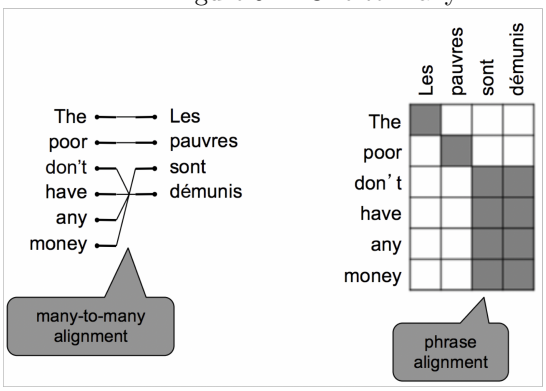

Figure 9.4: Many-to-many

We learn $P(f, a|e)$ as a combination of many factors, including

- Probability of particular words aligning (also depends on position in sent)
- Probability of particular words having particular fertility (number of corresponding words)
- etc.

9.1.4 Word Alignment - Vector Representation

We assume each French word f is aligned to exactly one English word e, including NULL. The alignment vector is represented

$$a = [2, 3, 4, 5, 6, 6, 6]$$

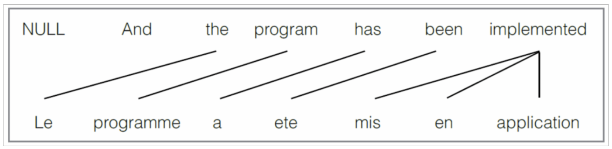

Figure 9.5: word alignment

The length of a equals the length of sentence f, and a_i = j if French position i is aligned to English position j. There are $(l+1)^m$ possible alignments in A. We formalize the connection between word alignments and the translation model

$$p(f_1, f_2, ..., f_m|e_1, e_2, ..., e_l, m) = \sum_{a \in A} p(f_1, ..., f_m, a_1, ..., a_m|e_1, ..., e_l, m)$$

We define a conditional model by projecting word translations through alignment links.

9.1.5 Decoding for SMT

We search for the best translation in the space of possible translations

$$T(f) = \hat{e} = argmax_e P(e)P(f|e)$$

where $P(e)$ is the language model and $P(f|e)$ is the translation model. To compute this argmax, we could enumerate every possible e and calculate the probability. However, this approach is too expensive. We use a heuristic search algorithm to search for the best translation, discarding hypotheses that are too low-probability.

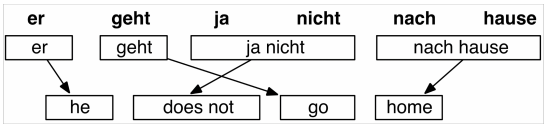

Figure 9.6: Phrase to phrase alignment

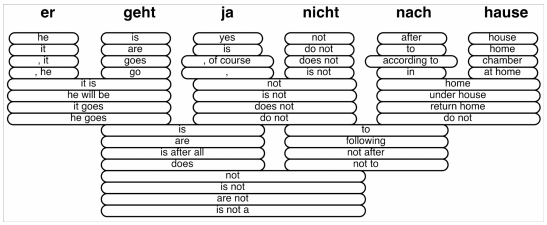

Figure 9.7: Decoding for SMT

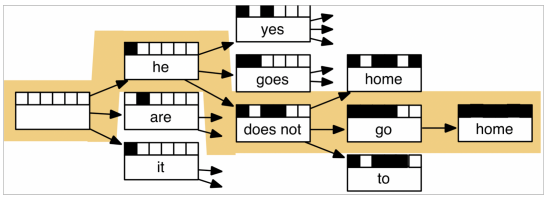

Figure 9.8: High-probability hypothesis

9.2 Neural Machine Translation with Seq2Seq

Sequence-to-sequence, or "Seq2Seq", is a relatively new paradigm, with its first published usage in 2014 for English-French translation[5]. At a high level, a sequence-to-sequence model is an end-to-end model made up of two recurrent neural networks:

- an encoder, which takes the model's input sequence as input and encodes it into a fixed-size context vector
- an decoder, which uses the context vector from above as a "seed" from which to generate an output sequence.

For this reason, Sequence-to-sequence models are often referred to as "encoder-decoder models".

Compared to Statistical Machine Translation(SMT), Neural Machine Translation(NMT) has better performance. NMT is more fluent, and it has a better use of context and phrase similarities. NMT only has a single neural network to be optimized end-to-end, which means no subcomponents are needed to be individually optimized. It also requires much less human engineering effort, because it uses the same method for all language pairs without feature engineering.

However, compared to SMT, NMT is less interpretable because it is hard to debug, and it is difficult to control since we cannot easily specify rules or guidelines for translation.

9.2.1 Seq2Seq - encoder

The goal of the encoder network is to read the input sequence to our Seq2Seq model and generate a fixed-dimensional context vector C for the sequence. To do so, the encoder will use a recurrent neural network cell(usually an LSTM or GRU) to read the input tokens one at a time. The final hidden state of the cell will then become context vector C. However, it is so difficult to compress an arbitrary-length sequence into a single fixed-size vector(especially for difficult tasks like translation), the encoder will usually consist of stacked RNNs. The final layer's RNN hidden state will be used as C.

9.2.2 Seq2Seq - decoder

The decoder is also an LSTM or GRU network. It is used as a language model which is "aware" of the words that it has generated so far and of the input. We initialize the hidden state of the decoder with the context vector generated by the encoder. We use a stack of several recurrent units(one or more layers) to predict an output y_t at a time step t. The decoder will literally use the context of the input to generate an output. Once the decoder is set up with its context, we will pass in a special token to signify the start of output generation; in literature, this is usually an $< EOS >$ token appended to the end of the input and the end of the output. Then we will run all RNN layers, followed by a softmax on the final layer's output to generate the first output word. Then, we pass that word into the first layer as the input, and repeat the generation.

9.2.3 Seq2Seq - Greedy decoding

Greedy decoding takes most probable word on each step. In other words,

$$x_t = argmax_{\tilde{x_t}} \mathbb{P}(\tilde{x_t}|x_1, ..., x_n)$$

We keep decoding until the model produces a <END>token. This technique is efficient and natural. However, it explores a small part of the search space, and it has no way to undo decisions, which means if we make a mistake at one time step, the rest of the sentence could be heavily impacted.

9.2.4 Seq2Seq - Exhaustive Search Decoding

We compute the probability of every possible sequence, and we chose the sequence with the highest probability. Ideally we want to find translation y that maximizes:

$$P(y|x) = P(y_1|x)P(y_2|y_1,x)P(y_3|y_1,y_2,x)...,P(y_T|y_1,...,y_{T-1},x) = \sum_{t=1}^{T} P(y_t|y_1,...,y_{t-1},x)$$

However, this technique does not scale at all to large outputs as the search space is exponential in the size of the input. On each step t of the decoder, we track V^t possible partial translations, where V is the vocabulary size. The complexity is $\mathcal{O}(V^T)$ which is far too expensive

9.2.5 Seq2Seq - Beam Search Decoding

The idea is to maintain K candidates at each time step. On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses). A hypothesis y_1, ..., y_t has a score which is its log probability:

$$score(y_1,...,y_t) = logP_{LM}(y_1,...,y_t|x) = \sum_{i=1}^{t} logP_{LM}(y_i|y_1,...,y_{i-1},x)$$

Scores are all negative, and higher score means better translation. We keep decoding until the model produces a <END>token. Different hypotheses may produce <END>tokens on different time steps. When a hypothesis produces the <END>token, that hypothesis is complete. We place it aside and continue exploring other hypotheses via beam search. We usually continue beam search until we reach time step T (where T is some pre-defined cutoff), or we have at least n completed hypotheses (where n is pre-defined cutoff).

When we have our list of completed hypotheses, we select top one with highest score. However, longer hypotheses have lower scores. To fix this problem, we normalize the score by the length of the hypothesis:

$$\frac{1}{t}\sum_{i=1}^{t} logP_LM(y_i|y_1,...,y_{t-1},x)$$

Beam search is not guaranteed to find optimal solution, but it is much more efficient than exhaustive search.

9.2.6 Evaluation of Machine Translation Systems

In 2002, IBM researchers developed the Bilingual Evaluation Understudy(BLEU)[4] that remains, with its many variants to this day, one of the most respected and reliable methods for machine translation. Although developed for translation, it can be used to evaluate text generated for a suite of natural language processing tasks, such as language generation,image caption generation, text summarization, and speech recognition.[2]

The BLEU algorithm compares the precision score of a candidate machine-written translation to one or several human-written translations. The BLEU metric ranges from 0 to 1. Few translations will attain a score of 1 unless they are identical to a reference translation. For this reason, even a human translator will not necessarily score 1. The reference human translation is assumed to be a model example of a translation, and we use n-gram matches as our metric for how similar a candidate translation is to it. We also add a penalty for too-short system translations.

9.3 Attention Model

9.3.1 Motivation

Sequence-to-sequence model has the bottleneck problem. All the relevant information to decode is fed through a fixed size vector. When the encoding sequence is long, it often fails to capture the complex semantic relations between words entirely. Words occurring at the beginning of the encoding sentence may contain information for predicting the first decoding outputs. It is often complex for the model to capture long term dependencies.

To overcome the information bottleneck and long-term dependencies limitations, attentions model have been introduced. On each step of the decoder, we use direct connection to the encoder to focus on a particular part of the source sequence. There are many types of attention, but we will examine the one introduced by Luong et al.[3]

9.3.2 Luong et al. NMT model

We have the hidden states given by the encoder h_1, ..., h_N, and the hidden states of the decoder s_t, ..., s_n. On each timestep t, we have decoder hidden state $s_t \in \mathbb{R}^N$. we compute an attention vector over the encoder hidden. On timestep t, we get the attention scores e^t at for this time step

$$e^t = [s_t^T h_1, ..., s_t^T h_N] \in \mathbb{R}^N$$

We can take softmax to get the attention distribution a^t for this step

$$a^t = softmax(e^t) \in \mathbb{R}^N$$

We use a^t to take a weighted sum of the encoder hidden states to get the attention output a^t

$$a_t = \sum_{i=1}^{N} a_i^t h_i \in \mathbb{R}^h$$

Finally we concatenate the attention output a_t with the decoder hidden state s_T and proceed as in the non-attention sequence-to-sequence model

$$[a_t; s_t] \in \mathbb{R}^{2h}$$

9.3.3 Benefits and Applications

Attention significantly improves NMT performance. It is very useful to allow decoder to focus on a certain part of the source. It solves the bottleneck problem by allowing decoder to look directly at source. It also helps with vanishing gradient problem by providing shortcut to far away states. Finally, attention provides some interpretability. By inspecting attention distribution, we can see what the decoder was focusing on. The network just learns alignment by itself, so we get (soft)alignment for free without training an alignment system explicitly.

Attention is a general deep learning technique. We can use attention in many architectures other than seq2seq model and many tasks(not just machine translation). We can define attention in a more general way

- Given a set of vector values, and a vector query, attention is a technique to compute a weighted sum of the values, dependent on the query.

We sometimes say that the query attends to the values.

For example, in the seq2seq + attention model, each decoder hidden state (query) attends to all the encoder hidden states (values).

The weighted sum is a selective summary of the information contained in the values, where the query determines which values to focus on. Attention is a way to obtain a fixed-size representation of an arbitrary set of representations (the values), dependent on some other representation (the query).

We have some values $h_1, ..., h_N \in \mathbb{R}^{d_1}$ We can compute the attention scores $e \in \mathbb{R}^N$, and take softmax to get attention distribution α

$$\alpha = softmax(e) \in \mathbb{R}^N$$

We use attention distribution to take weighted sum of values:

$$a = \sum_{i=1}^{N} \alpha_i h_i \in \mathbb{R}^{d1}$$

thus we obtain the attention output a(sometimes called the context vector).

There are several ways to compute $e \in \mathbb{R}^N$ from $h_1, ..., h_N \in \mathbb{R}^{d_1}$ and $s \in \mathbb{R}^{d_2}$

- Basic dot-product attention

  $$e_i = s^T h_i \in \mathbb{R}$$

  This assumes $d_1 = d_2$.

- Multiplicative attention

  $$e_i = s^T W h_i \in \mathbb{R}$$

  Where $W \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix.

- Attentive attention

  $$e_i = v^T tanh(W_1 h_i + W_2 s) \in \mathbb{R}$$

  Where $W_1 \in \mathbb{R}^{d_3 \times d_1}$, $W_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $v \in \mathbb{R}^{d_3}$ is a weight vector.

References

[1] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[2] Estimation lemma. Estimation lemma — Wikipedia, the free encyclopedia, 2010.

[3] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[4] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[5] I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. *Advances in NIPS*, 2014.