Machine Translation 2

COMP-550

Nov 23, 2017

Outline

IBM Model 1

IBM Model 2

Phrase-based MT

MT Decoding

Recent Developments

Statistical Machine Translation

Let's look at a popular direct-transfer approach to statistical machine translation: the **noisy channel model**.

$$\underset{P(E)}{\text{English}} \xrightarrow[P(F|E)]{} \text{Russian}$$

When I look at an article in Russian, I say:
'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.'

Warren Weaver, 1955

IBM Model 1

IBM developed a series of five influential models that make increasingly powerful assumptions.

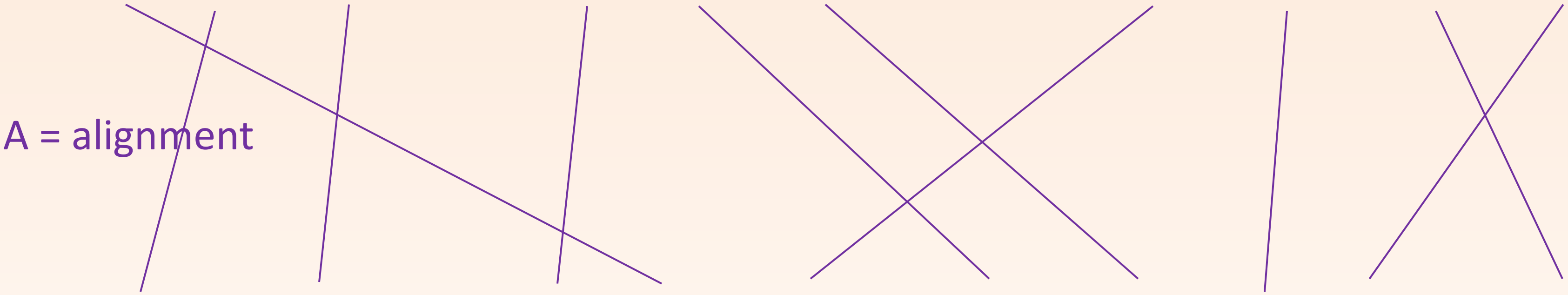
Model 1 is the most basic:

- Each source word is aligned to **zero or one** target word
- Don’t try to model different **distortions** of word order (e.g., completely flipping word order vs. just swapping the orders of one or two words)
- Don’t try to model likelihood of **fertility** (some phrases, e.g., *take a walk*, might be translated as one unit)

Word Alignment

E = target sentence

NULL The petitioners are calling for containers to be standardized to the metric system



Les pétitionnaires demandent l’ uniformisation des contenants au système métrique

F = source sentence

- NULL node allows words in F to align to nothing in E.
- Since each source word is aligned to **zero or one** target word, |A| = |F|.
- Can represent A as indices: {1, 2, 4, 0, 9, 5, 6, 10, 13, 12}

Word Alignment Probabilities

$$P(F|E) = \sum_{A} P(F, A|E) = \sum_{A} P(F|E, A) \times P(A|E)$$

Probability of source sentence, given the target sentence, and knowing which words are aligned with which.

Probability of the alignment, given the target sentence.

$P(A|E)$

IBM Model 1 makes a very strong simplifying assumption:

- Uniform probability of translation length (i.e., length of A)
- Uniform probability for each possible alignment

$$P(A|E) \propto C$$

or

$$P(A|E) = \frac{\epsilon}{(I+1)^J}$$

, where I is the number of target words, J is the number of source words, ϵ is there to make sure things normalize across different possible values of J.

Why the + 1?

$P(F|E,A)$

Decompose this into individual word alignments

$$P(F|E,A) = \prod_{j=1}^{J} t(f_j|e_{a_j})$$

How do we learn $t(f_j|e_{a_j})$?

- If we had observed word alignments in the training corpus, we could simply do MLE:

$$t(f|e) = \frac{\text{Count}(f,e)}{\text{Count}(e)}$$

- We don't, so it's time for …?

Expectation-Maximization

1. Initialize the parameters $t(f|e)$ randomly
2. Iterate for a while:
   - **E-step**: Given the current parameters, compute the expected value of $\mathrm{Count}(f, e)$ over the training data
   - **M-step**: Given the current $\mathrm{Count}(f, e)$, compute the new MLE $\theta_k = t(f|e)$

Probability of Alignments

To get the expected counts, what we really need is the probability of an alignment: $P(A|E,F)$

$$P(A|E,F) = \frac{P(A,E,F)}{P(E)P(F|E)} = \frac{P(F,A|E)}{P(F|E)} = \frac{P(F,A|E)}{\sum_A P(F,A|E)}$$

Since $P(F,A|E) = P(F|E,A) \times P(A|E)$, and $P(A|E)$ is the same for all alignments, we get:

$$P(A|E,F) = \frac{P(F|E,A)}{\sum_A P(F|E,A)}$$

Recall that $P(F|E,A) = \prod_{j=1}^{J} t(f_j|e_{a_j})$.

Thus, we're set, given some initial model of $t(f|e)$.

Example

Let's do one round of EM training for the following mini-corpus:

red house *the house*

maison rouge *la maison*

Initialize the model $t(f|e)$ uniformly:

$t(maison \mid red) = \frac{1}{3}$	$t(rouge \mid red) = \frac{1}{3}$	$t(la \mid red) = \frac{1}{3}$
$t(maison \mid house) = \frac{1}{3}$	$t(rouge \mid house) = \frac{1}{3}$	$t(la \mid house) = \frac{1}{3}$
$t(maison \mid the) = \frac{1}{3}$	$t(rouge \mid the) = \frac{1}{3}$	$t(la \mid the) = \frac{1}{3}$

Exercise

Do the second round of EM training.

Details, Details

In practice, don't initialize $t(f|e)$ uniformly:

- Given reasonable sizes of lexicon, too many parameters = too much memory and computation!
- Rather, restrict it to word pairs e', f', where e' and f' occur is some aligned sentence pair in the training set.

When sentence lengths are high, need to efficiently compute probabilities of all possible alignments.

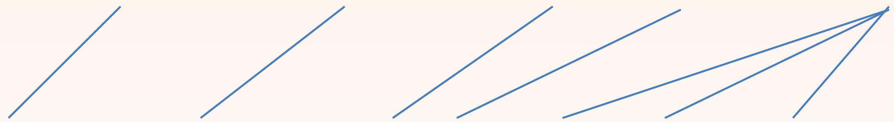
- Can adapt our algorithm to implicitly sum over all alignments

IBM Model 2

Does not assume that all possible alignment structures are equiprobable.

- For many language pairs, alignment should proceed without much crossing:

And the programme has been implemented.



Le programme a été mis en application.

Can also draw alignment as a table.

IBM Model 2

$t(f|e)$ as before; the probability of source word f given target word e

$q(j|i,l,m)$ **distortion** probability that $a_i = j$, given length of $F = m$ and length of $E = l$.

Recall that in Model 1, $P(A|E) = \frac{\epsilon}{(I+1)^J}$

Now:

$$P(A|E) = \epsilon \prod_{i=1}^{m} q(a_i|i,l,m)$$

$$P(A|E,m) = \prod_{i=1}^{m} q(a_i|i,l,m)$$, for a given m

Exercise

Given the following sentence pair:

And the programme has been implemented.

Le programme a été mis en application.

Write down A, then the expression for $P(F, A|E, m)$ in terms of factors $t(\ldots)$ and $q(\ldots)$.

$$P(F|E, A) = \prod_{j=1}^{J} t(f_j|e_{a_j})$$

$$P(A|E, m) = \prod_{i=1}^{m} q(a_i|i, l, m)$$

Parameter Estimation in IBM Model 2

In MLE:

$$t(f|e) = \frac{\text{Count}(f,e)}{\text{Count}(e)}$$

$$q(j|i,l,m) = \frac{\text{Count}(j,i,l,m)}{\text{Count}(i,l,m)}$$

For EM, need probability of a specific edge in the alignment $\delta_k(i,j)$ of aligning the ith word of F to the jth word of E in sample k:

$$\delta_k(i,j) = \frac{q(j|i,l_k,m_k)t(f_i^k|e_j^k)}{\sum_{j'=0}^{l_k} q(j'|i,l_k,m_k)t(f_i^k|e_{j'}^k)}$$

Further Notes

Each iteration of EM increases training corpus likelihood.

EM on IBM Model 2 may converge on local optima; *different initializations lead to different solutions*.

- So, need a good initialization
- Trick: initialize with the result of running IBM Model 1

Extensions

Higher IBM models

- Model 3: model **fertility**—how many words are used to translate a word

HMM alignment

- Cast computation of $P(F, A|E)$ as an HMM sequence labelling problem
- Use this to prefer alignments that are close to diagonal (works for some language pairs like English-French, English-Spanish)

Phrase-Based SMT

What about dealing with phrases that are better translated as a unit?

coup	*blow*
foudre	*lightning*
coup de foudre	*love at first sight*

Non-constituents also benefit:

Spass am	*fun with the*

Phrase-based, rather than word-based SMT can solve this problem by adding a little more context.

Need to learn **phrase table**

A Model of Phrase-based MT

1. Split sentence into phrases

$$E = e_1 e_2 \ldots e_I = ep_1 ep_2 \ldots ep_N$$

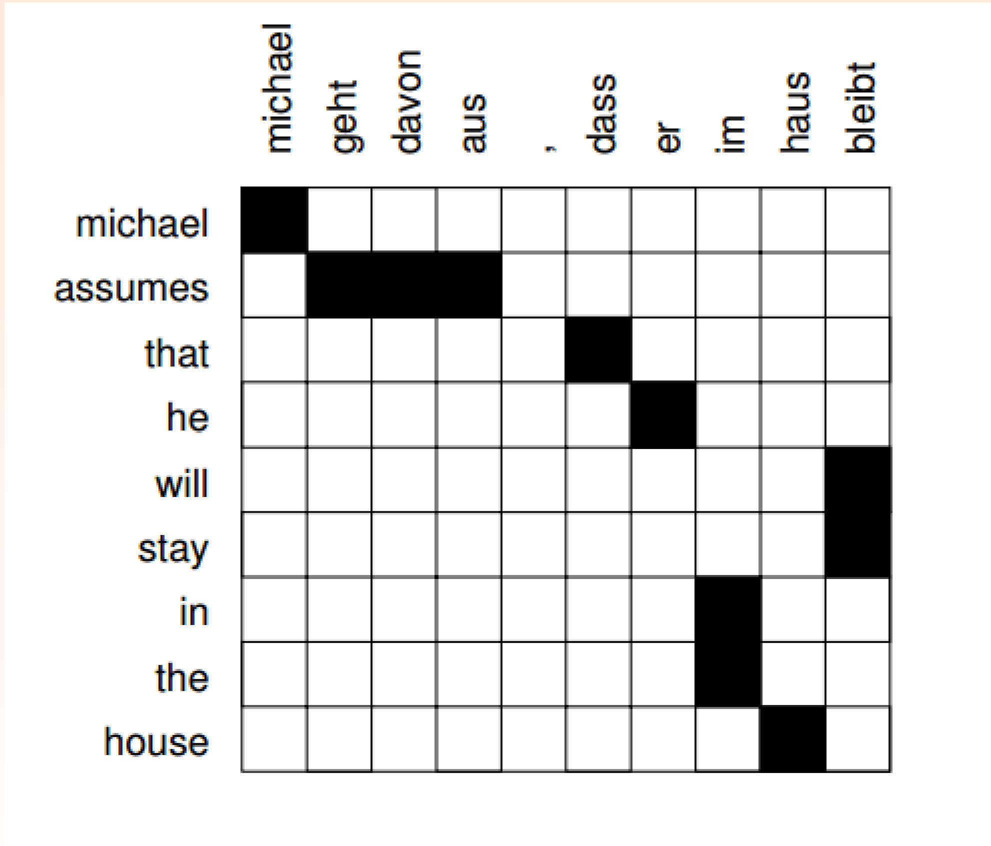
2. Translate each phrase with **phrase translation probability** $P(fp|ep)$
3. Rearrange phrases with some **reordering probability** $d(dist)$
   - e.g., penalty for changing position

$$P(F|E) = \prod_n P(fp_n|ep_n) d(dist_n)$$

Learning a Phrase Table

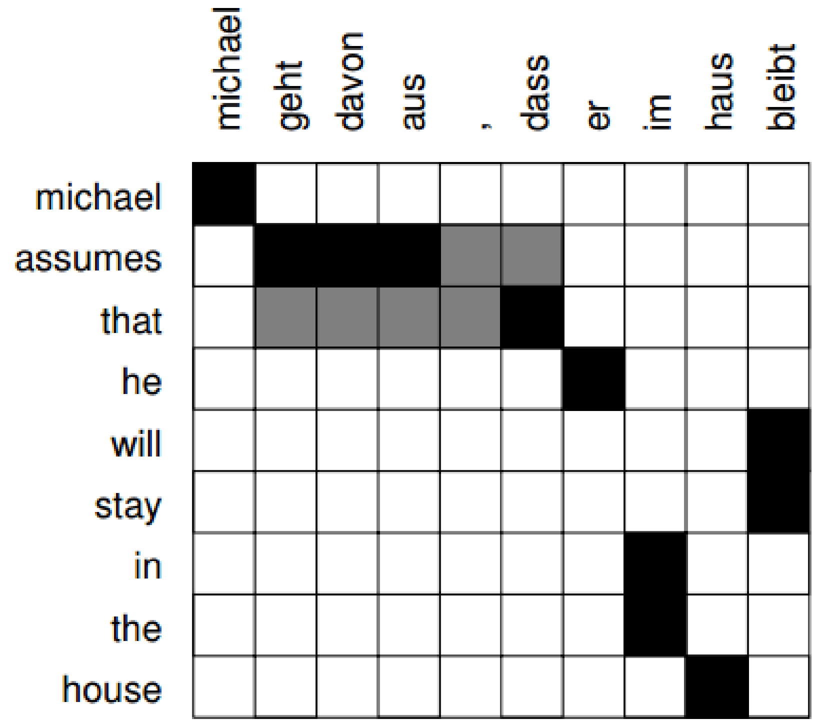
1. Start with word alignment
   - e.g., use an IBM model
2. Extract phrase pairs
3. Score phrase pairs

Word Alignment



Example drawn from Koehn, (2009), Ch. 5

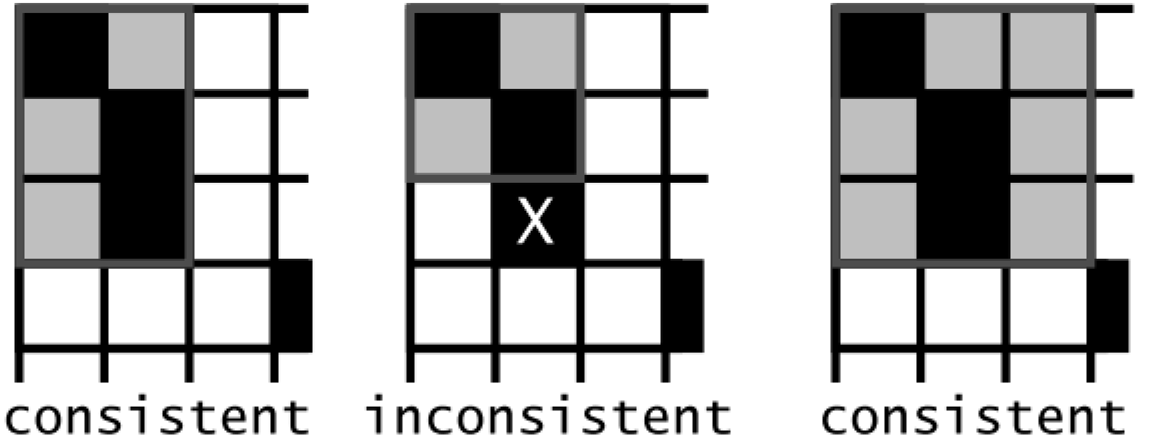
Extracting Phrase Pairs



extract phrase pair consistent with word alignment:

assumes that / geht davon aus , dass

Note Consistency Constraints



ok

violated
one alignment
point outside

ok
unaligned
word is fine

All words of the phrase pair have to align to each other.

Scoring Phrase Translations

Relatively simple affair:

$$P(fp|ep) = \frac{\text{Count}(fp, ep)}{\sum_{fp'} \text{Count}(fp', ep)}$$

MT Decoding

We still need a **decoding algorithm** to *search* for the best possible translation predicted by a given model.

Many search algorithms can be used:

- A* search
- Greedy hill-climbing
- Beam search
- …

Let's briefly describe a greedy hill-climbing method (Germann et al., 2001)

Greedy Hill-Climbing

Start by creating one complete candidate translation

- e.g., translate each word separately

$$e^* = \mathrm{argmax}_{\mathrm{e}}\, P(f|e)$$

This gives an initial translation:

Diese Woche ist die gruene Hexe zuhause.

$$\downarrow$$

This week is the green witch at home.

Hill Climbing

Then, apply change operators:

- Change the translation of a word or phrase
- Combine the translation of two words into a phrase
- Split up the translation of a phrase into two subphrases
- Rearrange parts of the translation

At each point, we evaluate all of the transformations by computing $P(E)P(F, A|E)$, and select the change the maximizes this.

We iteratively run this process until reaching a local optimum.

Recent Developments in MT

Neural network methods have become very popular in MT over the past two years.

e.g., the following paper at ACL 2014:

> Devlin et al. Fast and Robust Neural Network Joint Models for Statistical Machine Translation.
>
> http://acl2014.org/acl2014/P14-1/pdf/P14-1129.pdf

Neural Network Joint Model

The model directly predicts the output translation given the input translation, and previous translation decisions:

$$P(T|S) \approx \prod_{i=1}^{|T|} P(t_i | t_{i-1} \dots, t_{i-n+1}, \Sigma_i)$$

$\Sigma_i = \sigma_1 \sigma_2 \dots \sigma_m$ is a subsequence within S that is predicted to be important for translating t_i.

This is done by an initial word alignment step.

Neural Network Model Structure

t_i

Output prediction (softmax)

$\uparrow$

512-dimensional hidden layer

$\uparrow$

512-dimensional hidden layer

Hidden layers (tanh activation functions)

$\uparrow$

192-dimensional vector representation of each word

$\uparrow$

$t_{i-3} \quad t_{i-2} \quad t_{i-1} \quad \sigma_1 \quad \sigma_2 \quad \sigma_3 \ldots \quad \sigma_{12}$

Input words

BLEU Results

Combined with an existing MT decoder, this model achieves very good BLEU results:

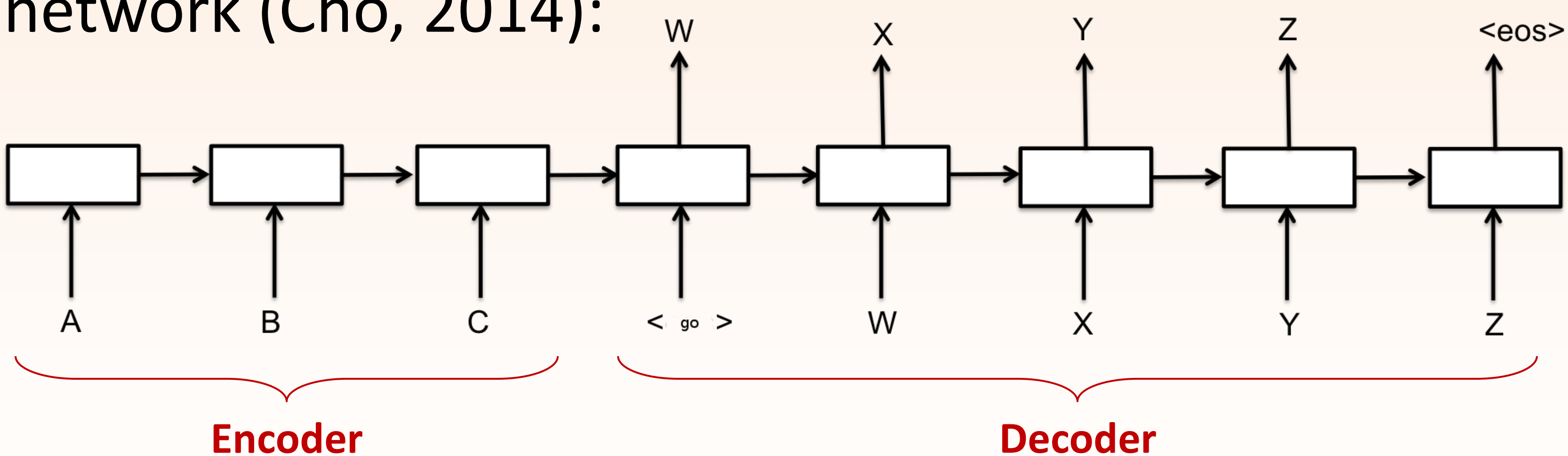
NIST MT12 Test		
	Ar-En	Ch-En
	BLEU	BLEU
OpenMT12 - 1st Place	49.5	32.6
OpenMT12 - 2nd Place	47.5	32.2
OpenMT12 - 3rd Place	47.4	30.8
...	...	...
OpenMT12 - 9th Place	44.0	27.0
OpenMT12 - 10th Place	41.2	25.7
Baseline (w/o RNNLM)	48.9	33.0
Baseline (w/ RNNLM)	49.8	33.4
+ S2T/L2R NNJM (Dec)	51.2	34.2
+ S2T NNLTM (Dec)	52.0	34.2
+ T2S NNLTM (Resc)	51.9	34.2
+ S2T/R2L NNJM (Resc)	52.2	34.3
+ T2S/L2R NNJM (Resc)	52.3	34.5
+ T2S/R2L NNJM (Resc)	52.8	34.7
"Simple Hier." Baseline	43.4	30.1
+ S2T/L2R NNJM (Dec)	47.2	31.5
+ S2T NNLTM (Dec)	48.5	31.8
+ Other NNJMs (Resc)	49.7	32.2

Table 3: Primary results on Arabic-English and Chinese-English NIST MT12 Test Set. The first

Joint Alignment and Translation

Another method is to jointly train a model to align and translate **at the same time**.

Consider a **sequence-to-sequence** recurrent neural network (Cho, 2014):



- Each block above is an RNN cell, such as a **LSTM** block

Attention Mechanism

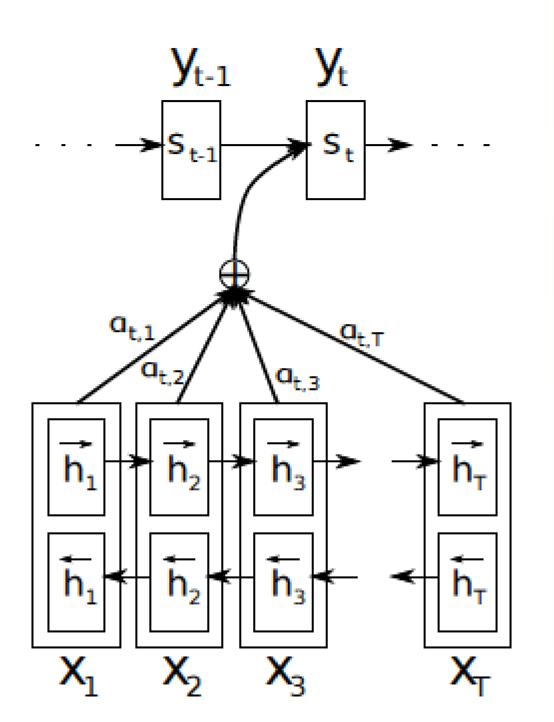
At decoder step, take a weighted combination of the hidden representations in the encoder for use in predicting next word (Bahdanau et al., 2015):

$$c_i = \sum_j \alpha_{ij} h_j$$ Used in decoding at time i

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k e_{ik}}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

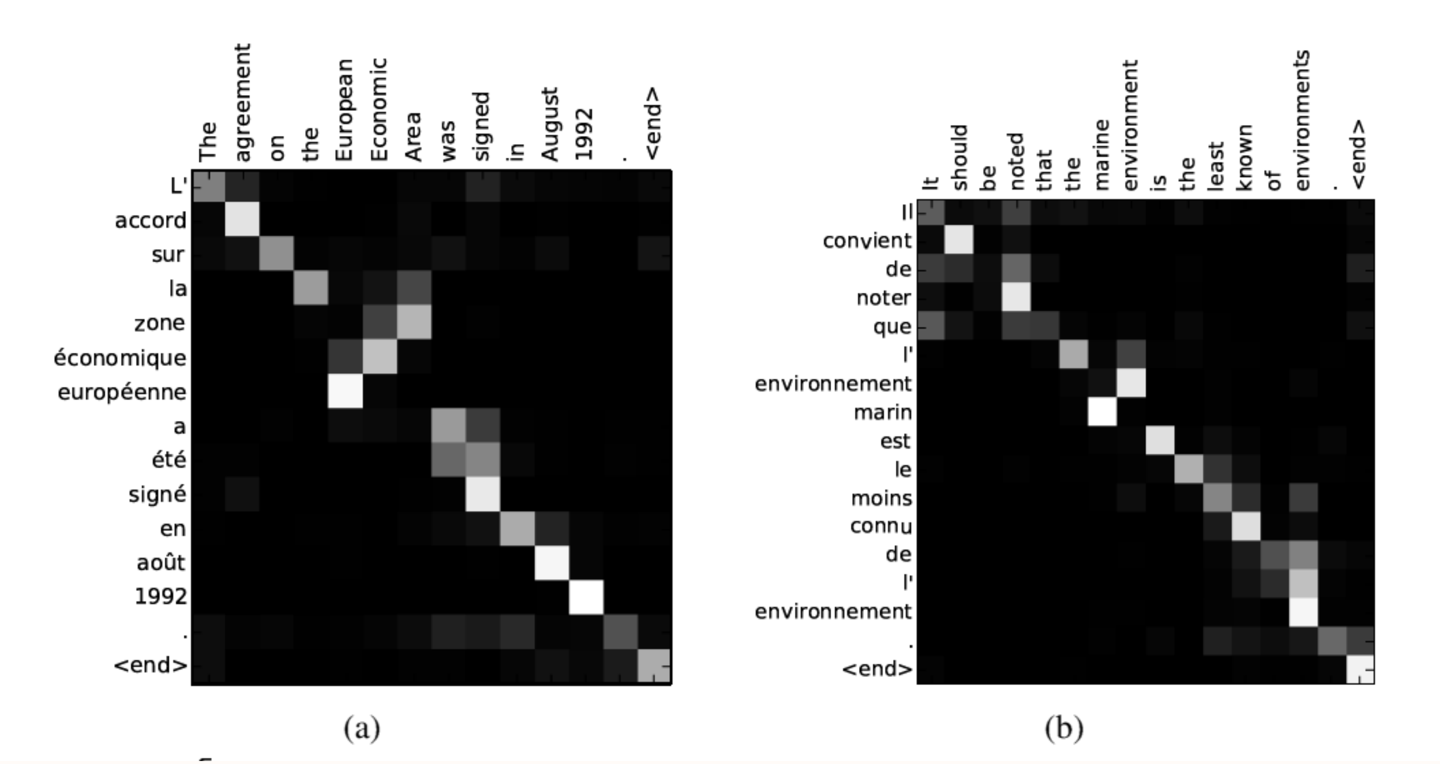
where a is a feed-forward NN



Decoder

Encoder

Visualization of Attention Weights



from (Bahdanau et al., 2015)

Use of attention now widespread in NLP!

Reference

- Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015.
- Devlin et al. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. ACL 2014.
- Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast Decoding and Optimal Decoding for Machine Translation. ACL 2001.