Machine Translation

From Rules to Transformers

Ramaseshan Ramachandran

CONTENTS

A Short History
of Machine Translation

SEVENTY-FIVE YEARS, FOUR IDEAS

- **Rules** encode the grammar of both languages
- **Examples** translate by analogy with past translations
- **Statistics** learn from aligned parallel text
- **Neural** methods learn one continuous model end to end

Each replaced the one before it. Each took roughly two decades.

1947–1949: TRANSLATION AS DECODING

- In 1947 Warren Weaver writes to Norbert Wiener: could translation be treated as a **cryptography** problem?
- In July 1949 Weaver circulates his memorandum *Translation* to some 200 researchers
- The framing that launched the field:

  "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." *(Weaver, 1947)*

The noisy-channel model, which we meet later, is this idea made precise.

1954: THE GEORGETOWN–IBM EXPERIMENT

- In January 1954 in New York, an IBM 701 translates about **60 Russian sentences** into English
- A vocabulary of **250 words** and **6 grammar rules**
- Carefully chosen sentences; chemistry and general subjects
- The press reported translation as essentially solved
- Researchers predicted the problem would be finished in three to five years

It funded a decade of work.

1966: THE ALPAC REPORT

- The Automatic Language Processing Advisory Committee reviews a decade of US funding
- Its finding: machine translation was **slower**, **less accurate** and **roughly twice as expensive** as human translation
- It saw no immediate prospect of useful machine translation
- Recommendation: fund basic computational linguistics instead

US funding collapsed. The first winter lasted more than ten years.

WHAT ALPAC GOT RIGHT AND WRONG

Right

- The systems really were poor
- Word-by-word substitution cannot resolve ambiguity
- Claims had outrun results

Wrong

- Assumed the ceiling was permanent
- Missed that data, not rules, was the bottleneck
- Post-editing was dismissed. It is standard practice today

A lesson worth carrying into any claim about today's models.

1968–1980S: THE RULE-BASED ERA

- In 1968 SYSTRAN is founded, used by the US Air Force and later the European Commission
- In 1968 Vauquois publishes his survey. The **pyramid** becomes the map of the field
- Direct, transfer and interlingua systems
- METEO translates Canadian weather bulletins from 1977. The domain is narrow and the system is genuinely useful

Lesson of the era: restrict the domain and rules work.

1984: TRANSLATION BY ANALOGY

- Makoto Nagao proposes **example-based** machine translation
- People do not translate by deep grammatical analysis; they translate by analogy with things they have seen translated
- Store a bilingual corpus; find the closest matching fragments; recombine them
- The first argument that **data** should carry the translation, not rules

1988–1993: IBM AND THE STATISTICAL TURN

- The Candide project at IBM applies speech-recognition methods to translation
- Brown, Della Pietra, Mercer and colleagues publish **IBM Models 1–5**
- Trained on the **Hansards**, the proceedings of the Canadian parliament, in English and French
- Introduces **word alignment** as the central problem

No linguist wrote a rule. The model learned the correspondence.

1999–2007: PHRASE-BASED SMT

- In 1999 the Johns Hopkins summer workshop reimplements the IBM models; GIZA and later GIZA++ make alignment available to everyone
- In 2003 Koehn, Och and Marcu introduce **statistical phrase-based translation**
- Translate multi-word phrases, not single words. Local reordering and idiom come almost free
- In 2007 **Moses** is released. Phrase-based SMT becomes the standard baseline for a decade
- In 2006 Google Translate launches, statistical from the start

2014–2017: THE NEURAL TURN

- In 2014 Sutskever and colleagues introduce **sequence to sequence** learning, and Cho and colleagues the encoder–decoder
- In 2014–15 Bahdanau, Cho and Bengio add **attention**, removing the fixed-length bottleneck
- In 2016 Google replaces ten years of phrase-based infrastructure with a neural system
- In 2017 comes **the transformer**: attention alone, no recurrence, and it parallelises

Machine translation is where the transformer was invented.

WHERE IT STANDS

- Multilingual models translate hundreds of language pairs in one set of weights
- Large language models translate **without being trained to**. Translation is a by-product of prediction
- Still open: low-resource languages, domain terminology, gender and other bias, evaluation itself

The problem Weaver posed in 1949 is not closed. It moved.

THE ARC IN ONE TABLE

Period	Approach	Carries the knowledge
1949–1966	Direct, word-for-word	Dictionaries
1968–1990	Rule-based transfer	Hand-written grammars
1984–1990s	Example-based	A bilingual corpus
1990–2014	Statistical (SMT)	Alignment counts
2014–2017	Neural seq2seq	Learned weights
2017–	Transformer, LLM	Attention over everything

Read the last column downwards: less hand-built each time.

LET US TRANSLATE - SWAHILI TO ENGLISH

My dog

mbwa	wangu
My	My
Dog	dog

My house

Nyumba	Yangu
My	My
House	House

My cycle

mzunguko	wangu
My	My
Cycle	Cycle

Your house

Nyumba	yako
My	My
House	House

Your cycle

mzunguko	Wako
Your	Your
Cycle	Cycle

This is a house

hii	ni	nyumba
This	This	This
is	is	is
a	a	a
house	house	house

This is your cat

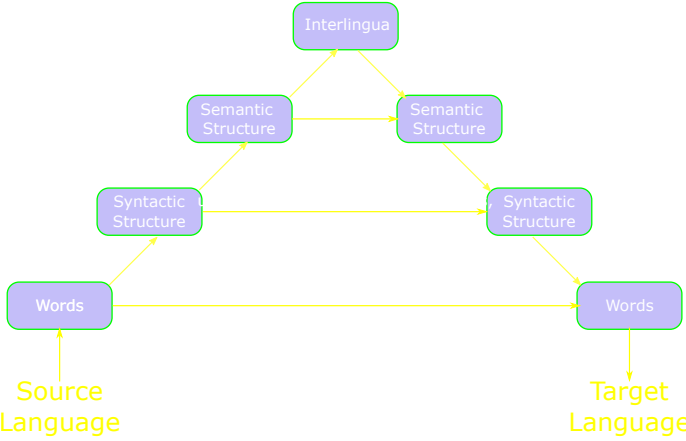
huyu	ni	paka	wako
This	This	This	This
is	is	is	is
your	your	your	your
cat	cat	cat	cat

nyumba	yangu	ni	nyumba	yako

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, 1947)

VAUQUOIS PYRAMID - VARIOUS APPROACHES TO MT

Vauquois pyramid [1]



WORD2WORD OR LITERAL TRANSLATION

Every word from the source language is converted into the target language, one word at a time with out considering the whole sentence as context

SYNTACTIC TRANSLATION

1. The source sentence is parsed to create a syntax tree
2. The nodes of the source tree is mapped to the nodes the similar syntax tree created for the target language -

$$(\mathrm{subject})_s \rightarrow (\mathrm{subject})_t$$
$$(\mathrm{noun})_s \rightarrow (\mathrm{noun})_t$$
$$(\mathrm{det})_s \rightarrow (\mathrm{det})_t$$
$$(\mathrm{adj})_s \rightarrow (\mathrm{adj})_t$$

3. Generate the sentence in the target language sentence from the parse tree

SEMANTICS-BASED TRANSLATION

- The meaning of the source sentence is obtained
- Using the semantics derived from the source sentence, the target sentence is generated

INTERLINGUA TRANSLATION

- A meta-language format for representing knowledge independent of any language
- Instead of Translation systems for all possible pairs of languages, one representation would be used to generate translations
- $O(n^2) \rightarrow O(n)$
- Difficult to design efficient and comprehensive knowledge representation formalisms and due to the large amount of ambiguity

AUTOMATIC MACHINE TRANSLATION

- The idea of *the ability to make anyone speak to anyone without the boundary of languages* is the most appealing idea
- The goal of the automatic translation is to produce error-free translation
  - Preserve the meaning of the source language
- AMT is a hard problem
- Parallel corpora aid in the development of AMT

- Translation by analogy: Example based machine translation (EBMT) (lazy learning)

  Hii ni nyumba yangu - This is my house

  Mbwa wangu anapenda kukimbia - My dog loves to run

  Mimi kukimbia na mbwa wangu - I run with my dog

  This is my dog -

- Translation by analogy: Example based machine translation (EBMT) (lazy learning)

  Hii ni nyumba yangu - This is my house
  Mbwa wangu anapenda kukimbia - My dog loves to run
  Mimi kukimbia na mbwa wangu - I run with my dog
  This is my dog - Huyu ni mbwa wangu

- Learn MT models from data: Statistical Machine Learning
  - Translation models with language-specific parameters
  - Train model parameters & apply to unseen data

Translations are generated using parameters and models which are derived from the analysis of bilingual text corpora.

- Every French string, f, is a possible translation of e. We assign to every pair of strings $\{e, f\}$ a number $P(f|e)$, which we interpret as the probability that a translator, when presented with e, will produce f as his translation
- Given a French string f, the job of our translation system [2] is to find the string e that the native speaker had in mind when he produced f

F	E
comment allez-vous?	How are you?
	How do you do?
	How are you doing?
Comment ça va ?	
Vous allez bien?	
Ça va ?	

DEFINITIONS

Let us assume that the task is to translate a French sentence f with a sequence $(f_1, f_2, f_3, ...f_m)$ of length m and f_j for $j \in (1,2,3,...m)$ is the j^{th} word.
The translated English sentence will be assumed to have the sequence $(e_1, e_2, e_3, ...e_n)$ and n is the length of the English sentence.
Let us assume that the corpus consists of the pair of source and translated sentences, $(f^k \text{ and } e^k)$.
$f^k = (f_1^k, f_2^k, \ldots, f_m^k)$ where f_j^k is the j^{th} word in the k^{th} French sentence of length m
$e^k = (e_1^k, e_2^k, \ldots, e_n^k)$ where e_j^k is the j^{th} word in the k^{th} English sentence of length n

The parallel copora are available from Canadian parliamentary proceedings (the *Hansards*) and from Europarl data. Europarl data consists of proceedings from the European parliament, and consists of translations between several European languages

PARALLEL CORPORA

A parallel corpora is a collection of corpus that contains a collection of original text and its translation in various languages. In most cases, parallel corpora contain data from two languages.

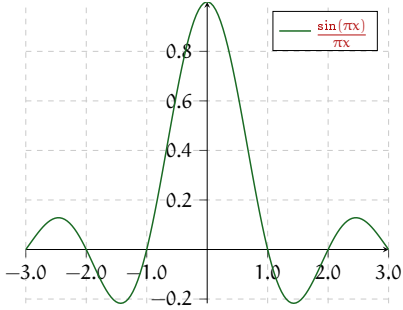
English	French
Resumption of the session	Reprise de la session
I declare resumed the session of the European Parliament adjourned on Friday 17 December 1999, and I would like once again to wish you a happy new year in the hope that you enjoyed a pleasant festive period.	Je déclare reprise la session du Parlement européen qui avait été interrompue le vendredi 17 décembre dernier et je vous renouvelle tous mes vux en espérant que vous avez passé de bonnes vacances.
You have requested a debate on this subject in the course of the next few days, during this part-session	Vous avez souhaitété un détébat à ce sujet dans les prochains jours, au cours de cette pétériode de session.

ARGMAX

The *arguments of the maxima* function f is defined for a set D as

$$\arg\max_{x \in D} f(x) = x | f(x) \geq f(y), \forall y \in D$$

In other words, the argmax are the points of the domain of some function at which the function values are maximized



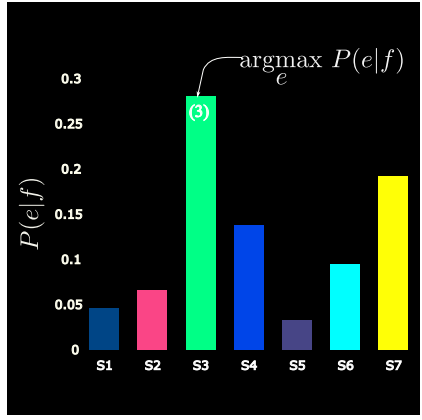
The argmax of the function $\frac{\sin(\pi x)}{\pi x}$ is 0 as the function has the global maximum value of 1

ARGMAX - MT

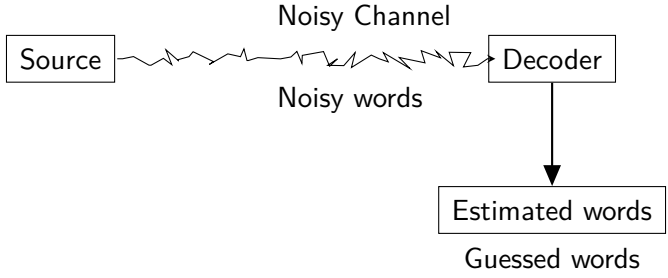
Given a French sentence f, find the most likely English sentence e that maximizes $\mathrm{P}(e|\mathrm{f})$. The *arguments of the maxima* function f is defined as,

$$\arg\max_{e} \mathrm{P}(e|\mathrm{f}) \tag{1}$$

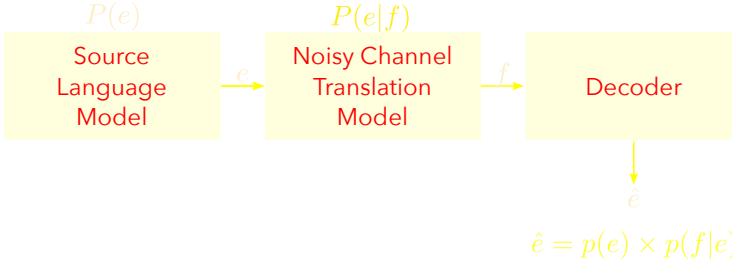
The English sentence e, out of all such sentences, which yields the highest value for $\mathrm{P}(e|\mathrm{f})$. It is possible to have more than one translation for a given sentence. In such cases, *argmax* finds one English sentence e that yields the highest value for $\mathrm{P}(e|\mathrm{f})$.



THE NOISY CHANNEL MODEL



NOISY CHANNEL FOR MT



The Language Model generates an English sentence e. The Translation Model transmits e as the French sentence f. The decoder finds the English sentence hate which is most likely to have given rise to f[3]. $\mathrm{P}(e)$ is the distribution over which sentences are likely in English and $\mathrm{P}(\mathrm{f}|e)$ is the translation model that indicates the likelihood seeing the French sentence f as a translation of e

Many bilinguals, whose mother tongue is not English, may think of the sentence they want to speak in their mother tongue first and then speak out the translated version in English

BAYES' RULE FOR MT

By applying Bayes' Theorem, the translation problem is broken down into two smaller problems. Assume that we have a French sentence f and we would like to translate into an English sentence e.

From the probabilistic perspective, we want to find the English sentence e that has maximal probability given the French sentence f. Using Bayes rule we can write this problem as

$P(e|f) = \frac{P(f|e)P(e)}{P(f)}$ We can find the English sentence using **the** $\arg\max$

$$\begin{aligned} \arg\max_e &= \arg\max_e P(e|f) \\ &= \arg\max_e \frac{P(f|e)P(e)}{P(f)} \\ &\boxed{\hat{e} = \arg\max_e P(f|e)P(e)} \end{aligned}$$

$P(f|e)$ – the translation model and
$P(e)$ – the English Language Model
The problem is reduced to modeling these 2 distributions
Now we have to estimate the parameters of the $P(f|e)$ from the training examples (f^k, e^k) for $k = 1...n$

BIGRAM AND TRIGRAM PROBABILITIES

$$P(w_2|w_1) = \frac{f(w_1, w_2)}{f(w_1)}$$

$f(w1, w2)$ is the number of times w_2 appeared after w_1

$$P(w_3|w_1, w_2) = \frac{f(w_1, w_2, w_3)}{f(w_1, w_2)}$$

$f(w_1, w_2, w_3)$is the number of times w_3 appeared after w_1 and w_2

Or we could use the non-Markovian neurql Language models for LM

SPARSITY AND SMOOTHING

- Newer ways of forming a sentence is common.
- It is possible that a trained model will see a new n-gram
- These new n-grams results in $P(x|y) = 0$
- $P(x|y) = 0$ will propagate through and produce a zero probability for the entire sentence
- Smaller probabilities too create a very small value

To avoid $P(x|y) = 0$, linear interpolation is used.
$P(w_3|w_2, w_1) = \lambda_1 P(w_3|w_2, w_1) + \lambda_2 P(w_2|w_1) + \lambda_3 P(w_1) + \lambda_4$
where $\lambda_1(0.95) + \lambda_2(0.04) + \lambda_3(0.008) + \lambda_4(0.002) = 1$
For new words and n-grams, $P(x|y)$ will always have a small value

KNOWLEDGE CAPTURED BY THE MODEL

I want to eat Chinese food. I want English food. I want to eat english food

$$P_1(english|want) = 0.0011$$
$$P_2(chinese|want) = 0.0065$$
$$P_3(to|want) = 0.66$$
$$P_4(eat|to) = 0.28$$
$$P_4(order|to) = 0.18$$
$$P_5(want|I) = 0.32$$

$$P_6(food|english) = 0.015$$
$$P_7(food|chinese) = 0.15$$
$$P_8(chinese|eat) = 0.34$$
$$P_{10}(english|eat) = = 0.001$$
$$P_{11}(i|<s>) = 0.25$$
$$P_{12}(</s>|food) = 0.12$$

I want _____ food
I want to _____ _____ food.

To avoid underflow values of multiplication to find $P(e)$, one can use log
$\log(P_1 * P_2 * P_3 * P_4 ... P_n) = \log(P_1) + \log(P_2) + \log(P_3) + \log(P_4) ... \log(P_n)$

EVALUATION OF THE LANGUAGE MODEL

Can we apply Bayes rule for evaluation of the model? A model can be evaluated based on the test data

$$\mathrm{P}(\mathrm{model}|\mathrm{testing\,data\,set}) = \frac{\mathrm{P}(\mathrm{model})\mathrm{P}(\mathrm{testing\,data\,set})}{\mathrm{P}(\mathrm{testing\,data\,set})} \tag{2}$$

- A better model is one which assigns a higher probability to the word that actually occurs
- The best model is the one that optimizes the $\mathrm{P}(\mathrm{model})\mathrm{P}(\mathrm{testing\,data\,set})$
- A model that outputs zero probability for any unknown sentence will be discarded

PERPLEXITY

- The tiny numbers of $P(e)$ may underflow any floating point scheme.
- An n-gram model will assign a very tiny $P(e)$ for long sequences.
- Many n-gram conditional probabilities may also be a very small value
- The product for $P(e)$ will be tiny

To compare models, $\mathbb{P} = 2^{-\log_2(P(e))/|V|}$ is computed. $|V|$ is the number of words in the test data. $\mathbb{P}$ is known as the perplexity score.

$$\mathbb{P} \propto \frac{1}{P(e)}$$

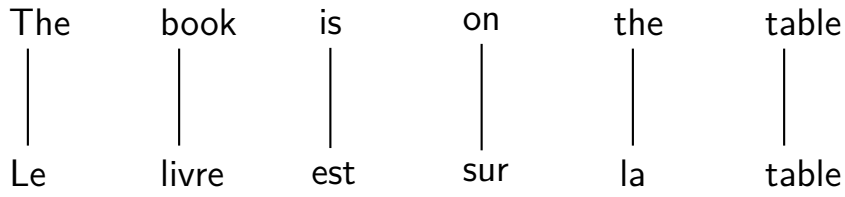
A good model will have a relatively small perplexity score. The lower the perplexity, the better the model is.

$\mathrm{P}(\mathrm{f}|e)$ is the chance that upon seeing e, a translator will produce f.

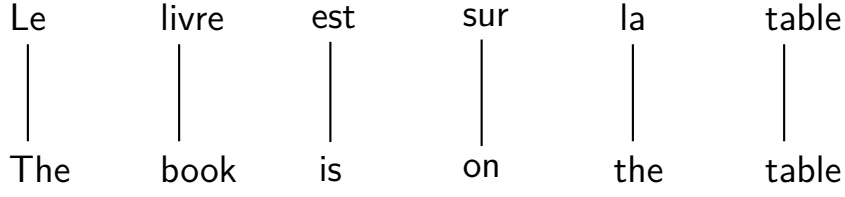
$$\mathrm{P}(\mathrm{f}|e) = \frac{\text{Count of (f,e)}}{\text{Count of } (e)}$$

In simple terms, translating from French to English is to identify the bag of words in English and later form syntactically correct sentences.

In this model, there is no need to use any French to English translated corpus to train the language model.
Is this correct and will it work?



$$P(french|english)$$



$$P(English|French)$$

HOW CAN WE TRANSLATE?

- What steps do we take to translate a language?
- As non-native speakers, how do we frame English sentences?
- Do we have a BoW for English, before writing any English sentences?
- Do we assemble word-for-word translation in mind before writing any English sentences?
- Do we assemble BoW in both languages before writing?
- Can it be thought of string rewriting?
- Identify a corresponding word in the other language and use its language model to build the sentence?

By fixing the size of the French sentence to m words, we will assume that there is some distribution $P(m|n)$ that models the conditional distribution of French sentence length m conditioned on the English sentence length n. We could also choose a set of words $(f_1, f_2, f_3, ...f_m)$

Now, we can write – the conditional probability of the French sentence is conditioned on the English words of length n and the French sentence of length m.

$$P(f_1, f_2, f_3, ...f_m|e_1, e_2, e_3, ...e_n, m) \tag{3}$$

Is it easy or hard to estimate the distribution of equation (3)?

It is hard to estimate $\mathrm{P}(\mathrm{f}|e,\mathrm{m})$ directly. Let us introduce the concept of alignment variables

ALIGNMENT

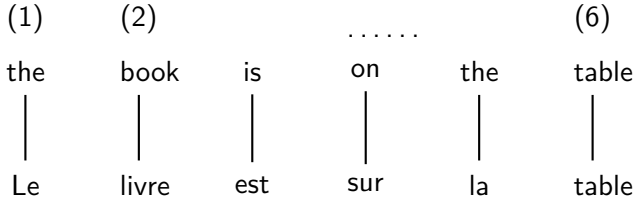
- Consider a seed word in English that starts the translation process
- Assume this seed word, a_j, as the alignment word at the position j^{th} in the English sentence
- The alignment a is $\{a_1, a_2, a_3, ...a_m\}$, where $a_j \in \{0, n\}$
- The possible alignments are $(n+1)^m$
- The idea is to find the most likely alignment

Alignment probability depends on positions of the words, and position relative to neighbors. The likelihood of an alignment depends on how many words align to a certain position

Automatic alignment is the backbone of SMT

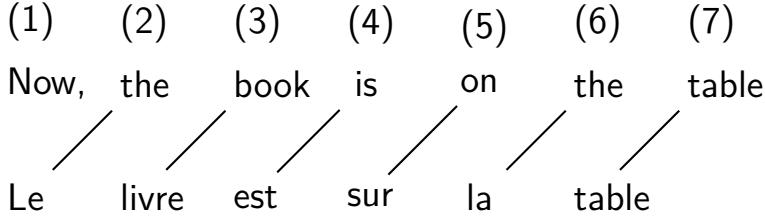
BIJECTIVE ALIGNMENT

- Every word in each text is coupled to exactly one word in the other text.
- No word remains uncoupled or left out



ALIGNMENT - EXAMPLE 1

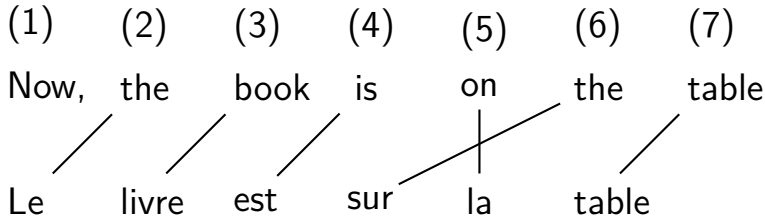
Alignment with respect to the translation model $P(f|e,m,n)$



$n = 7$ and $m = 6$

The alignment $(a_1, a_2, a_3, a_4, a_5, a_6) = \{2,3,4,5,6,7\}$

Alignment with respect to the translation model $P(f|e,m,n)$



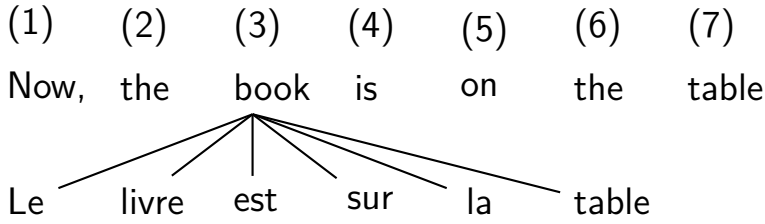
$n = 7$ and $m = 6$

The alignment $(a_1, a_2, a_3, a_4, a_5, a_6) = \{2, 3, 4, 6, 5, 7\}$

The index of the alignment refers to the location of the French word and the value refers to the location of the English word

ALIGNMENT - EXAMPLE 3

Alignment with respect to the translation model $P(f|e,m,n)$



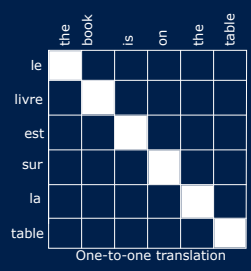
$n = 7$ and $m = 6$

The alignment $(a_1, a_2, a_3, a_4, a_5, a_6) = \{3, 3, 3, 3, 3, 3\}$

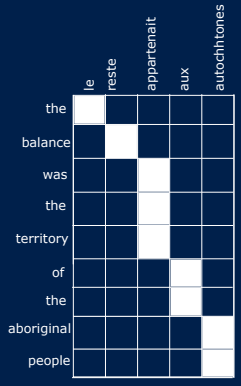
The index of the alignment refers to the location of the French word and the value refers to the location of the English word

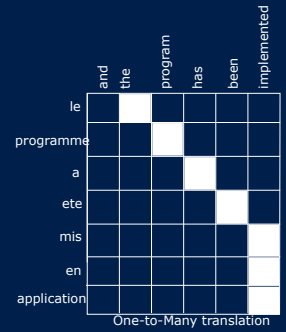
ALIGNMENT - ANOTHER REPRESENTATION

The alignment for the translation model $P(f|e,m,n)$



One-to-one translation





One-to-Many translation

Some examples are from the paper "The Mathematics of Statistical Machine Translation: Parameter Estimation"

https://www.aclweb.org/anthology/J93-2003

ALIGNMENTS

- Insertion - A NULL token is inserted if the target language does not have the equivalent source language word
- One2Many - A source word may translate into more than one target word
- Many2One - Many source words translate into one target word

SAMPLE TABLE FOR TRANSLATION PROBABILITY - $P(f|e,m,n)$

e = Now the book is on the table

f = Le livre est sur la table

	Now	the	book	is	on	the	table
Le	0.006	**0.47**	0.341	0.018	0.128	0.023	0.014
livre	0.108	0.076	**0.416**	0.046	0.048	0.241	0.065
est	0.194	0.101	0.03	**0.421**	0.15	0.057	0.047
sur	0.035	0.116	0.075	0.197	**0.434**	0.121	0.022
la	0.244	0.023	0.289	0.013	0.159	**0.289**	0.242
table	0.108	0.136	0.099	0.035	0.136	0.05	**0.436**

$$p(le|the) > p(le|on) > \ldots > p(le|book) > p(le|now)$$

The parameter, $p(f|e)$, is the conditional probability of generating a French word f from an English word e.

IBM MODEL 1

IBM models [2] are statistical machine translation models. They learn the model parameters by using bilingual corpus. The were part of many SMT systems for more than 20 years

- Lexical translation model (word2word)
- Alignment decisions are independent
- All alignments are equally likely
- The length of the source language sentence is fixed, m
- More than one source language word, (f_j), can be aligned to a single target language word (e_{a_j})

IBM MODEL 1 - TRANSLATION PROBABILITY

The goal is to estimate ***P(e|f)***. This is broken into two small distributions
(1) Translation model - ***P(f,a|e,m)***
(2) Language Model - ***P(e)***

English sentence - $e_1, e_2, e_3, \ldots, e_n$
French Sentence - $f_1, f_2, f_3, \ldots, f_m$
$a = \{a_1, a_2, a_3, \ldots, a_m\}$ - alignment indicates that from which English word each French word originated from - each alignment, $a_j \in [0, m]$. Estimate the translation probability

$$P(f, a|e, m) = P(a|e, m) \times P(f|a, e, m) \tag{4}$$

where $P(a|e, m)$ is the probability distribution of possible alignments

$$\begin{aligned} P(f|e, m) &= \sum_{a \in A} P(f, a|e, m) \\ &= \sum_{a \in A} P(a|e, m) \times P(f|a, e, m) \end{aligned} \tag{5}$$

IBM MODEL 1 - TRANSLATION PROBABILITY

1. Find the alignment - $P(a|e,m) = \frac{1}{(1+n)^m}$
2. Find the French word alignment probability, given the alignment variable, English word and fixed length of French Sentence $P(f|a,e,m) = \prod_{j=1}^{m} p(f_j|e_{a_j})$
3. Find the most probable alignment variables for every pair of e and f

using,

$$P(f,a|e,m) = P(a|e,m) \times P(f|a,e,m)$$
$$= \frac{1}{(1+n)^m} \times \prod_{j=1}^{m} p(f_j|e_{a_j})$$
$$p(f_j|e_{a_j}) = \frac{C(f_j,e_{a_j})}{\sum_{a\in A} C(f_j,e_{a_j})}$$

where $C(f_j,e_{a_j})$ is the count of the french word f_j aligned with the english word e_{aj}

Finally, $\hat{e} = \arg\max_{e\in E} = P(e)P(f,a|e,m)$

IBM MODEL 1 - EXAMPLE

$n = 7$ and $m = 6$
$e =$Now the book is on the table
$f =$Le livre est sur la table
$a = \{2,3,4,5,6,7\}$

$$\begin{aligned} P(f|a,e,m) =& p(\text{Le}|\text{the}) \times p(\text{livre}|\text{book}) \\ & \times p(\text{est}|\text{is}) \times p(\text{sur}|\text{on}) \times p(\text{la}|\text{the}) \\ & \times p(\text{table}|\text{table}) \\ p(\text{le}|\text{the}) =& \frac{\text{Count}(\text{the},\text{Le})}{\text{Count}(\text{the})} \dots \\ P(f,a|e,6) =& \frac{1}{(1+7)^6} \times P(f|a,e,6) \end{aligned}$$

IBM MODEL 1 - TRAINING

- If the alignments are known, then it is possible to estimate the translation probabilities by counting the aligned words
- If the translation probabilities are known, then it is possible to estimate the alignments
- We do not know both - Incomplete data
- Hence an iterative approach with refinement of these values over time is used

If we had complete data, we could estimate model
If we had the model, we could fill in the missing information
To solve this incomplete problem, we use ***Expectation maximization*** algorithm

1. Initialize model parameters (equally likely)
2. Assign probabilities to the missing data
3. Estimate model parameters from completed data
4. Iterate steps 2-3 until convergence

PYTHON CODE FOR EM - SWAHILI → ENGLISH

1. Initialization

```
import numpy as np

# Define the source and target vocabularies
src_vocab = ['my', 'house', 'cycle', 'his','dog']
tgt_vocab = ['mbwa', 'wangu', 'nyumba', 'yangu','mzunguko', 'wake']

# Define the parallel corpus
src_sents = [['my', 'dog'], ^^I['my', 'house'],
^^I^^I^^I^^I       ['my', 'cycle'],^^I['his', 'dog']]
tgt_sents = [['mbwa', 'wangu'], ^^I['nyumba', 'yangu'],
^^I^^I^^I^^I^^I^^I['mzunguko', 'wangu'],^^I['mbwa','wake']]

# Initialize the translation probability matrix
# The matrix has shape (num_source_words, num_target_words)
trans_prob = np.ones((len(src_vocab), len(tgt_vocab))) / len(tgt_voc

# Define the number of iterations
num iterations = 10
```

PYTHON CODE FOR EM - SWAHILI → ENGLISH - E-STEP

```
for iter in range(num_iterations):
^^I#initialize count_matrix and total probability- not shown here
^^Ifor i in range(len(src_sents)):
^^I^^I^^Inorm_factor = np.zeros(len(tgtt_sents[i]))
^^I# Compute the posterior probability of each alignment
^^Ipost_prob = np.zeros((len(src_sents[i]), len(tgt_sents[i])))
^^Ifor j in range(len(src_sents[i])):
^^I^^Ifor k in range(len(tgt_sents[i])):
^^I^^I^^Inorm_factor[k] += trans_prob[k, tgt_vocab.index(tgt_sents[i
^^I^^I^^Iif norm_factor[k]  == 0 :^^Inorm_factor[k] = 1
^^I^^I^^Ipost_prob[j, k] = trans_prob[src_vocab.index(src_sents[i][j
                              tgt_vocab.index(tgt_sents[i][k])] / no
^^I# Update the count matrix and total probability vector
^^Ifor j in range(len(src_sents[i])):
^^I^^Ifor k in range(len(tgt_sents[i])):
^^I^^I^^Icount_matrix[src_vocab.index(src_sents[i][j]),\
            target_vocab.index(tgt_sents[i][k])] += post_prob[j, k]
^^I^^I^^Itotal_prob[src_vocab.index(src_sents[i][j])] += post_prob[j
^^I
```

PYTHON CODE FOR EM - SWAHILI → ENGLISH - M-STEP

```
# Maximization step
for i in range(len(source_vocab)):
^^Ifor j in range(len(target_vocab)):
^^I^^Iif total_prob[i] == 0: # Avoid division by zero
^^I^^I^^Itranslation_prob[i, j] = 1 / len(target_vocab)
^^I^^Ielse:
^^I^^I^^Itranslation_prob[i, j] = count_matrix[i, j] / total_prob[i]
^^I
```

SAMPLE TRACE OF TRANSLATION TABLE - ITERATION 1

	my	dog	house	cycle	your	this	is	a	cat
mbwa	0.50	0.50	0.0000	0.00	0.0000	0.000	0.000	0.0000	0.0000
wangu	0.50	0.25	0.0000	0.25	0.0000	0.000	0.000	0.0000	0.0000
nyumba	0.15	0.00	0.4000	0.00	0.1500	0.100	0.100	0.1000	0.0000
yangu	0.50	0.00	0.5000	0.00	0.0000	0.000	0.000	0.0000	0.0000
mzunguko	0.25	0.00	0.0000	0.50	0.2500	0.000	0.000	0.0000	0.0000
hii	0.00	0.00	0.2500	0.00	0.0000	0.250	0.250	0.2500	0.0000
wako	0.00	0.00	0.0000	0.25	0.3750	0.125	0.125	0.0000	0.1250
yako	0.00	0.00	0.5000	0.00	0.5000	0.000	0.000	0.0000	0.0000
ni	0.00	0.00	0.1429	0.00	0.1071	0.250	0.250	0.1429	0.1071
huu	0.00	0.00	0.0000	0.00	0.2500	0.250	0.250	0.0000	0.2500
paka	0.00	0.00	0.0000	0.00	0.2500	0.250	0.250	0.0000	0.2500

SAMPLE TRACE OF TRANSLATION TABLE - ITERATION 5

	my	dog	house	cycle	your	this	is	a	cat
mbwa	0.1724	0.8276	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
wangu	0.9410	0.0368	0.0000	0.0222	0.0000	0.0000	0.0000	0.0000	0.0000
nyumba	0.0066	0.0000	0.9541	0.0000	0.0066	0.0068	0.0068	0.0190	0.0000
yangu	0.8499	0.0000	0.1501	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
mzunguko	0.0179	0.0000	0.0000	0.9638	0.0183	0.0000	0.0000	0.0000	0.0000
hii	0.0000	0.0000	0.0442	0.0000	0.0000	0.1999	0.1999	0.5559	0.0000
wako	0.0000	0.0000	0.0000	0.0200	0.9800	0.0000	0.0000	0.0000	0.0000
yako	0.0000	0.0000	0.1501	0.0000	0.8499	0.0000	0.0000	0.0000	0.0000
ni	0.0000	0.0000	0.0023	0.0000	0.0000	0.4842	0.4842	0.0292	0.0000
huu	0.0000	0.0000	0.0000	0.0000	0.0000	0.1206	0.1206	0.0000	0.7588
paka	0.0000	0.0000	0.0000	0.0000	0.0000	0.1206	0.1206	0.0000	0.7588

SAMPLE TRACE OF TRANSLATION TABLE - ITERATION 10

	my	dog	house	cycle	your	this	is	a	cat
mbwa	0.0819	**0.9181**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0
wangu	**0.9938**	0.0045	0.0000	0.0017	0.0000	0.0000	0.0000	0.0000	0.0
nyumba	0.0002	0.0000	**0.9966**	0.0000	0.0002	0.0005	0.0005	0.0020	0.0
yangu	**0.9278**	0.0000	0.0722	0.0000	0.0000	0.0000	0.0000	0.0000	0.0
mzunguko	0.0012	0.0000	0.0000	**0.9976**	0.0012	0.0000	0.0000	0.0000	0.0
hii	0.0000	0.0000	0.0079	0.0000	0.0000	0.1697	0.1697	0.6527	0.0
wako	0.0000	0.0000	0.0000	0.0013	**0.9987**	0.0000	0.0000	0.0000	0.0
yako	0.0000	0.0000	0.0722	0.0000	**0.9278**	0.0000	0.0000	0.0000	0.0
ni	0.0000	0.0000	0.0000	0.0000	0.0000	**0.4993**	**0.4993**	0.0013	0.0
huu	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**1.0**
paka	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**1.0**

Additional Step: Estimating Alignment Probabilities

- IBM Model 1 estimates translation probabilities for each word pair in the parallel corpus.
- IBM Model 2 includes an additional step to estimate alignment probabilities.
- Alignment probabilities represent the probability that a French word aligns with an English word.
- These probabilities are used to re-estimate the translation probabilities in Model 2.
- The alignment probabilities are estimated using the following formula:

$$q(j|i, m, n) = \frac{t(f_i|e_j)}{\sum_{i'=1}^{m} t(f_{i'}|e_j)}$$

where $t(f_i|e_j)$ is the translation probability of French word f_i given English word e_j, and m and n are the lengths of the French and English sentences, respectively.

IBM MODEL 2

Two parameters of the alignment model are defined as

1. The conditional probability of generating a French word f_j, given the English word, e_j - $p(f_j|e_i)$, where n and m are the lengths of the English and French sentences, respectively
2. $q(j|i,n,m)$ is the probability of alignment variable a_i taking the value j, conditioned on the lengths n and m of the English and French sentences, respectively.

$$P(a|e,m) = \prod_{j=1}^{m} q(a_j|j,n,m), \text{ where } a = \{a_1, a_2, a_3, \ldots, a_m\}$$

$$\therefore P(f,a|e,m) = \prod_{j=1}^{m} q(a_j|j,n,m)t(f_j|e_{a_j})$$

$$\tilde{e} = \arg\max_{e \in E} = P(e) \times P(a|e,m) \times P(f,a|e,m)$$

IBM MODEL 2 - SWAHILI-ENGLISH TRANSLATION TABLE

	mbwa	mzunguko	nyumba	paka	wake	wangu	yangu
cat	0.143	0.143	0.143	1.0	0.143	0.0	0.143
cycle	0.143	1.0	0.143	0.143	0.143	0.0	0.143
dog	1.0	0.143	0.143	0.143	0.0	0.0	0.143
his	0.0	0.143	0.143	0.143	1.0	0.143	0.143
house	0.143	0.143	1.0	0.143	0.143	0.143	0.0
my	0.0	0.0	0.0	0.0	0.143	0.75	0.25

IBM MODEL 2 - FRENCH → ENGLISH EXAMPLE

$n = 7$ and $m = 6$
e =Now the book is on the table
f =Le livre est sur la table
$a = \{2,3,4,5,6,7\}$

$$
\begin{aligned}
P(a|e,m) = & q(2|1,7,6) \\
& \times q(3|2,7,6) \\
& \times q(4|3,7,6) \\
& \times q(5|4,7,6) \\
& \times q(6|5,7,6) \\
& \times q(7|6,7,6)
\end{aligned}
$$

$$
\begin{aligned}
P(f|a,e,m) = & P(\text{Le}|\text{the}) \\
& \times t(\text{livre}|\text{book}) \\
& \times t(\text{est}|\text{is}) \\
& \times t(\text{sur}|\text{on}) \\
& \times t(\text{la}|\text{the}) \\
& \times t(\text{table}|\text{table})
\end{aligned}
$$

$$P(\text{le}|\text{the}) = \frac{\text{Count}(\text{the},\text{Le})}{\text{Count}(\text{the})} \ldots$$

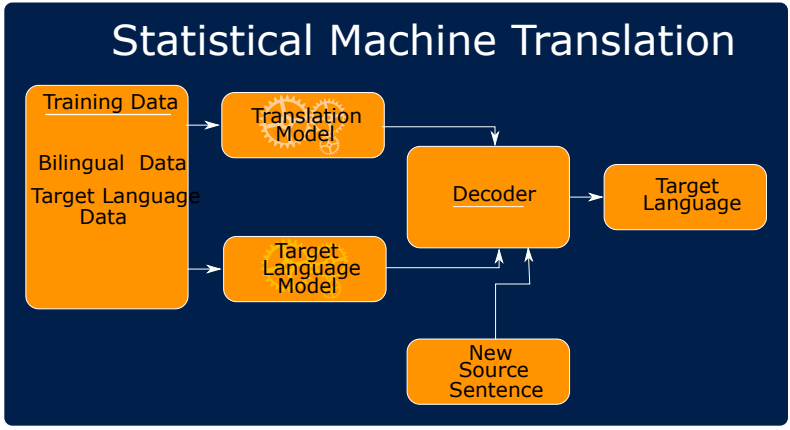
$$P(f,a|e,6) = P(a|e,6) \times P(f|a,e,m)$$

If we know the parameters q and t, it is easy to find the most probable alignment sequence a for any pair of French and English sentences.

$$a_j = \arg\max_{e \in E} q(a|j,l,m) \times p(f_j|e_a), \qquad \text{for } j = 1..m$$

There other models that improve the translation probability. These model are no longer used, but they are used in state of the art NMT models

- To estimate the lexical probability $\mathrm{t}(\mathrm{f}|e)$
- To derive alignments



The translation model represents the probable word translations. The language model encodes the generative model that computes the sentence confidence in terms of probability. The decoder searches for the most likely target word sequence from a large amount of hypotheses using these two models

DECODING PROCESS - 1

the

0.07781586
0.19699646
0.05338291
0.27595864
0.2202764
0.17556973

What next?

A phrase-based translation system can consider inputs and outputs in terms of sequences of phrases and can handle more complex syntaxes than word-based systems. However, long-term dependencies are still difficult to capture in phrase-based systems

PHRASE-BASED TRANSLATION

- Uses Noisy-channel model
- Uses phrase (contiguous subsequence of a sentence or a span of tokens) as the atomic unit - not to be confused with the Linguistic phrases
- Four stages
  1. Use IBM model to align words
  2. Phrase-to-Phrase alignments
  3. Extraction of phrases
  4. Construct phrase probability table

Reference: Statistical Phrase-Based Translation,Philipp Koehn, et al, 2003

Let e be the target language and f be the foreign language. Let e_i be the i^{th} word and f_j be the j^{th} word for e, f, respectively

$$\hat{e} = \arg\max_{e \in E} P(e)P(f|e) \tag{6}$$

$\arg\max$ is a search operation to predict the English sentence with the highest probability

ADVANTAGES OVER WORD2WORD TRANSLATION

Syntactic models such as IBM models map source tokens into foreign tokens does not account for phrases. Hence, they do not lead to better translations

- Many to many translation possible - can handle non-compositional phrases and idioms
- Use of local context - using nearest neighbors
- The number words in the phrase may dictate the correct word order
- If the learned phrases are longer, the whole sentence is translated

HOW TO FIND PHRASE ALIGNMENTS?

Use symmetrization of the alignments -

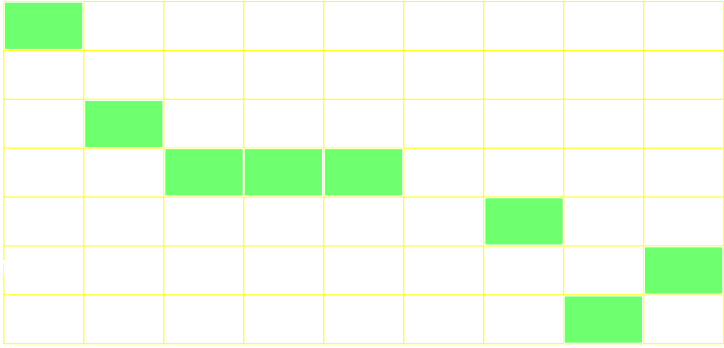
- Use alignment in both directions

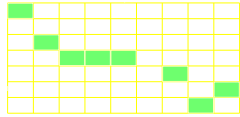
  Find (Source $\rightarrow$ Target) and (Target $\rightarrow$ Source) alignments
- Apply "Intersect" to get precise alignments
- Apply "Union to find intermediate points

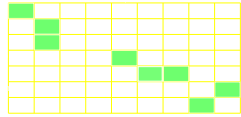
SYMMETRIZATION OF ALIGNMENTS

A method for aligning phrase-to-phrase alignments for a pair of sentences (F,E)is called as ***symmetrization***

Figure

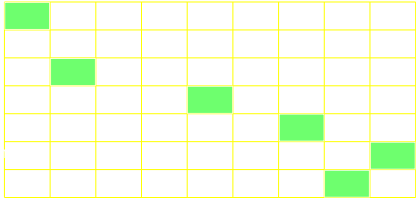


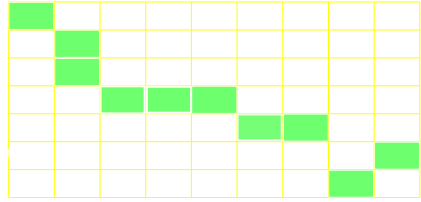




$(f \rightarrow e) \cap (e \rightarrow f)$

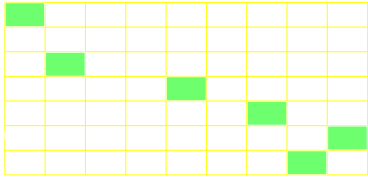
$(e \rightarrow f) \cup (f \rightarrow e)$

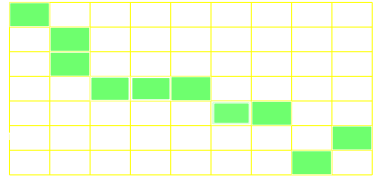


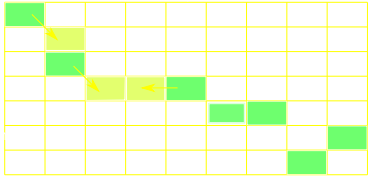


HEURISTICS FOR GROWING ALIGNMENTS

1. To insert new alignment point, search for the alignment points in $\mathrm{P}(e|\mathrm{f}) \cup \mathrm{P}(\mathrm{f}|e)$ alignments
2. If not available in (1), do not fill alignment points
3. Check for points that are not aligned already
4. Start filling the diagonal neighbors and adjacent points



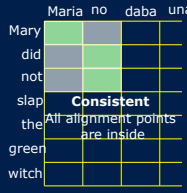


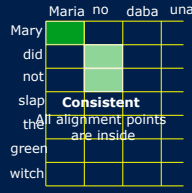


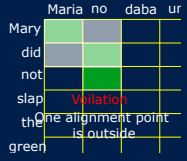
Sta

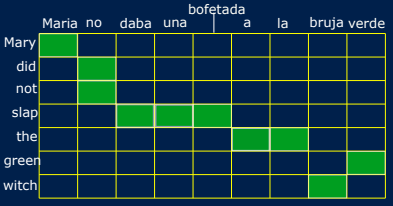
EXTRACTION OF PHRASES

The goal is to extract every possible pair of (f, e)









A phrase-pair (e, f) is consistent only when

- There is at least one word in e aligned to a word in f
- There are no words in f aligned to words outside e
- There are no words in e aligned to words outside f

- (Maria, Mary)
- (no, did not)
- (Maria no, Mary did not)

x (no daba, did not slap)

- (no dabaunabof', did not slap)
- (daba una bof', slap)
- (a la, the)
- (verde, green)
- (bruja, witch)
- (brujaverde, green witch)

x (Maria no daba una bofetada, Mary did not slap)

- (Maria no daba una bofetada a la, Mary did not slap the)
- (daba una bofetada a la bruja verde, slap the green witch)
- (Maria no daba una bofetada a la bruja verde, Mary did not slapthe green witch)

EXAMPLE - TAMIL2ENGLISH

SIZE OF THE PHRASE TABLE

- Very large size - bigger than the parallel corpora - reside in memory
- Extract all the phrases and store them in a database or disk

- Collect all the phrase pairs from the parallel corpora
- Assign probabilities to phrase translations

$$\text{Relative frequency} = p(\bar{f}|\bar{e}) = \frac{count(\bar{e}, \bar{f})}{\sum_i count(\bar{e}, \bar{f}_i)} \tag{7}$$

$$\text{Example}$$

$$p(\text{daba una bofetada}|\text{slap}) = \frac{C(\text{daba una bofetada}, \text{slap})}{C(\text{slap})} \tag{8}$$

$$\begin{aligned}\hat{e} &= \arg\max_{e \in E} P(e|f) \\ &= P(e) \times P(f|e) \qquad (9) \\ &= \arg\max_{e \in E} \prod_{j=1}^{J} p(\bar{f}_j|\bar{e}_j) d(a_j - b_{j-1}) P(e) \qquad (10)\end{aligned}$$

- $p(\bar{f}_j|\bar{e}_j)$ is the probability score for the translation of the phrase f, given e
- $d(a_j - b_{j-1})$ is the reordering score for the phrase which is modeled by the distortion probability distribution. a_j denotes the start position of the foreign word and b_{j-1} denotes the end position of the foreign phrase translated into the $j-1$ English phrase.
- This could be simplified by $\alpha^{|a_j - b_{j-1} - 1|}$
- $P(e)$ is the language model - could be a trigram/fourgram model $p(w_i|w_{i-(n-1)}, \ldots w_{i-1})$

- Start with an empty hypothesis
- A sequence of untranslated foreign words and a possible set of phrases for English are chosen
- The foreign words are marked as translated and the probability cost of the hypothesis is updated
  - $\text{cost} = \text{p}(e) \times \text{p}(\bar{\text{f}}_\text{i}|\bar{e}_\text{i}) \times \text{d}(.)$

NEURAL MACHINE TRANSLATION

Neural Machine Translation (NMT) is the mechanism of modeling the Machine translation process using artificial neural network. Let F and E be the source and the target sentences in a parallel corpora, respectively.

Translation as Probabilistic Decoding:

$$\hat{E} = \arg\max_{E} p_\theta(E \mid F) \tag{11}$$

Neural Training Objective:

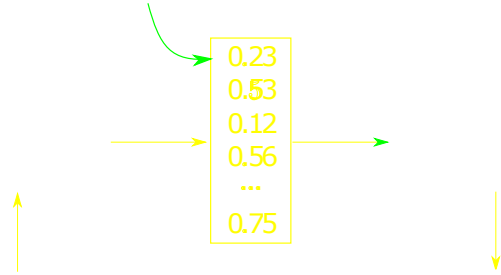
$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(E^{(i)}|F^{(i)}) \tag{12}$$

- Given a source sentence F, the goal is to find the most probable target sentence $\hat{E}$.
- The neural model is trained to maximize the conditional likelihood of target sentences given source sentences over the training corpus.

Once the conditional distribution is learned by a translation model, given a source sentence a corresponding translation can be generated by searching for the sentence that maximizes the conditional probability.

Unlike the phrase-based SMT models, NMT attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation[4]

ENCODER-DECODER MODEL



- All sentences (of varying length) are encoded into fixed sized vector
- Uses fraction of the memory needed by traditional SMT models[1]
- Performance of this model decreases as the length of a source sentence increase

[1] Cho et al, On the Properties of Neural Machine Translation: Encoder-Decoder Approaches, 2014

RNN-BASED TRANSLATION MODEL

- Uses RNN for both encoding and decoding
- Encoder maps the variable length sentence into a fixed-length vector
- Decoder translates the vector representation back to a variable-length target sequence
- Two networks are trained jointly to maximize the conditional probability of the target sentence given the source sentence, namely $P(E|F)$
- This model learns a continuous space representation of a phrase that preserves both the semantic and syntactic structure of the phrase[5].

RECURRENT NEURAL NETWORK

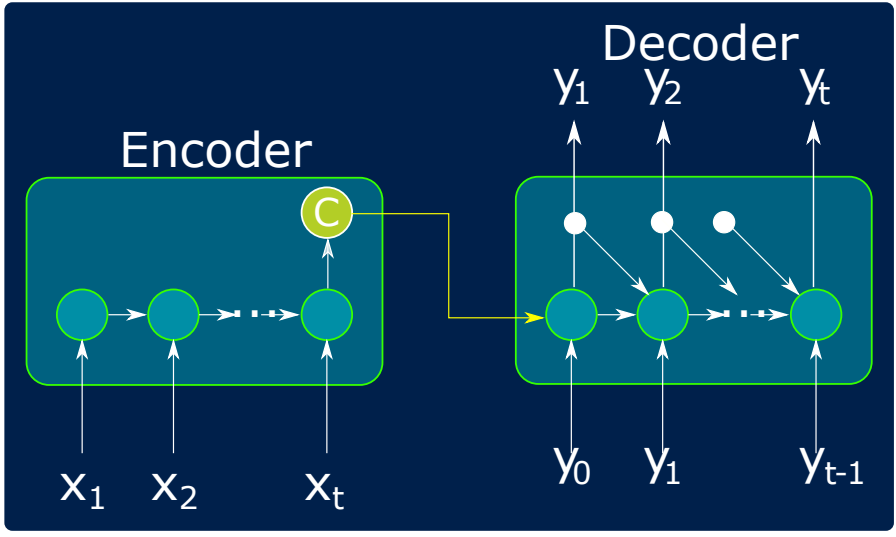
- Input units - variable length source sequence $x = (x_1, x_2, \ldots, x_T)$
- Output units - variable length target sequence $y = (y_1, y_2, \ldots, y'_T)$
- Hidden units for each input state,

$$h_t = f(h_{(t-1)}, x) \tag{13}$$

where f is a simple non-linear activation function (sigmoid or `tanh`) or a complex LSTM/GRU cell

- RNN is trained to predict the next word in the sequence or RNN learns a probability distribution over a sequence
- The output at each time step $t = p(x_t|x_{t-1}, \ldots x_1)$
- The output distribution (Softmax layer) size is equal to the size of the vocabulary V at every unit
- Then, $p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \ldots x_1)$

RNN-BASED ENCODER-DECODER



- RNN learns to map an input sentence of variable length into a fixed-dimensional vector representation.
- It learns to decode a fixed length vector representation back into a variable length sequence
- This model learns to predict a sequence given a sequence $p(y_1, y_2, \ldots, y'_T | x_1, x_2, \ldots, x_T)$. T and T′ may differ
- Encoder reads every symbol in **x**, sequentially
- Hidden state changes according to Eq.(13)
- C is the summary of the hidden states at time T and has encoded all the symbols in the sequence

- This is trained to predict the next symbol y_t and generate the output sequence, given the previous state h_t
- y_t and $\mathbf{h}_t$ are conditioned on the summary from the encoder, C and its previous hidden state
- Decoder's hidden state is given by
- Conditional distribution for the next symbol is

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, y_{t-1}, \mathbf{C}) \tag{14}$$

$$P(y_t | y_{t-1}, y_{t-2} \ldots, y_1, \mathbf{C}) = g(h_{t-1}, y_{t-1}, \mathbf{C}) \tag{15}$$

Both encoder and decoder are jointly trained to maximize the conditional likelihood

$$J(\theta) = \max_{\theta} \frac{1}{N} \log p_\theta(\mathbf{y}_n|\mathbf{x}_m) \tag{16}$$

where θ is the set of model parameters that will be learned during the BPTT and $(\mathbf{x}_m, \mathbf{y}_n)$ is the source sentence sequence and target sequence pair

BPTT - DERIVATIVE FOR V

$$h_t = (Wx_t + Uh_{t-1})$$
$$s_t = \tanh(h_t)$$
$$z_t = Vs_t$$
$$\hat{y}_t = softmax(z_t)$$
$$E_t = -y_t \log(\hat{y}_t)$$
$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial V} \tag{17}$$
$$\text{Let } \delta^t_{out} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}}{\partial z_t}$$
$$\frac{\partial E_t}{\partial V} = \delta^t_{out} s_t \tag{18}$$

Here δ^t_{out} is the loss for each of the units in the output layer

BPTT - DERIVATIVE FOR W

$$\frac{\partial E_t}{\partial W} = \underbrace{\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}}{\partial z_t}}\frac{\partial z_t}{\partial s_t}\frac{\partial s_t}{\partial h_t}\frac{\partial h_t}{\partial W} \tag{19}$$

$$= \delta^t_{out} V \sigma^{'}(h_t) x_t \tag{20}$$

Since the hidden layer activation depends on the previous time state, we have another similar term δ_{t-1} that get added to (20)

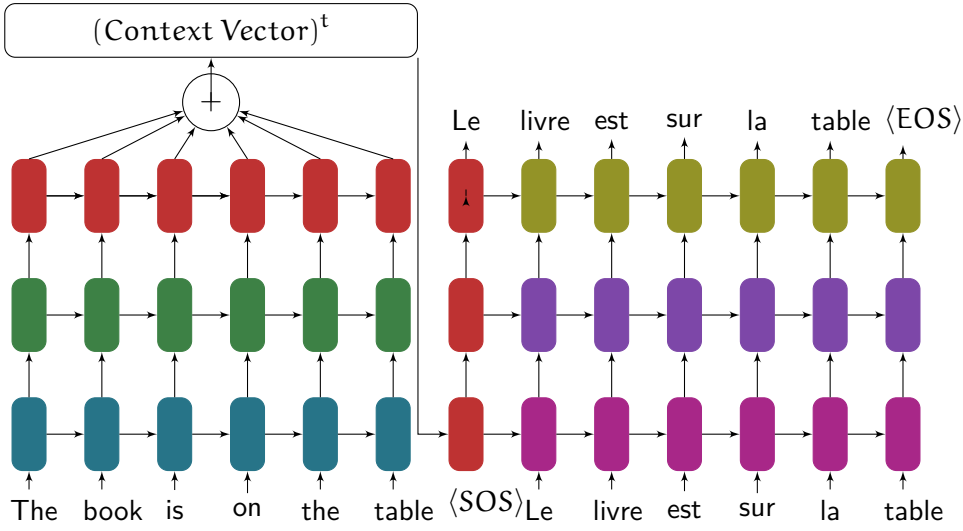
BPTT - DERIVATIVE FOR U

$$\frac{\partial E_t}{\partial U} = \underbrace{\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}}{\partial z_t}\frac{\partial z_t}{\partial s_t}\frac{\partial s_t}{\partial h_t}}\frac{\partial h_t}{\partial U} \tag{21}$$

$$= \delta^t_{out} V \sigma^{'}(h_t) h_{t-1} \tag{22}$$

Since we are back propagating the error from the current state to the previous state, $\delta_{next} = \sigma(h_t) U \delta^t_{out} V \sigma^{'}(h_t)$ needs to be added

SEQ2SEQ TRANSLATOR

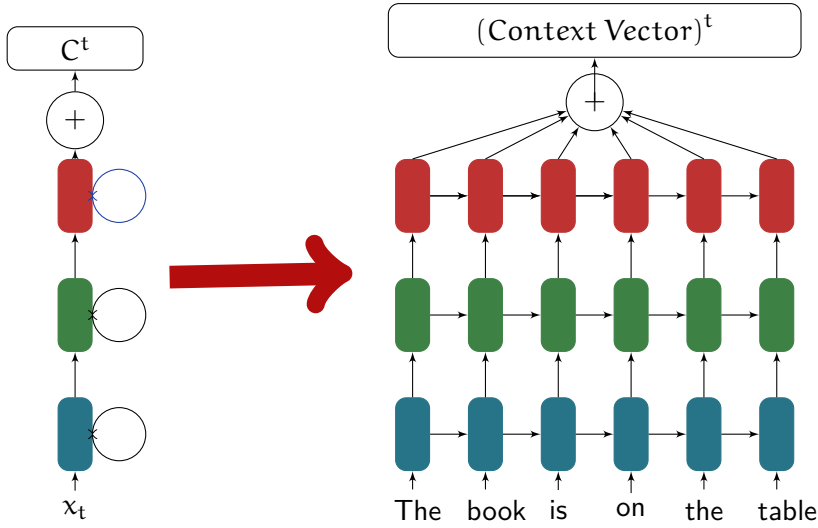


VARIATIONS IN RNN MODELS

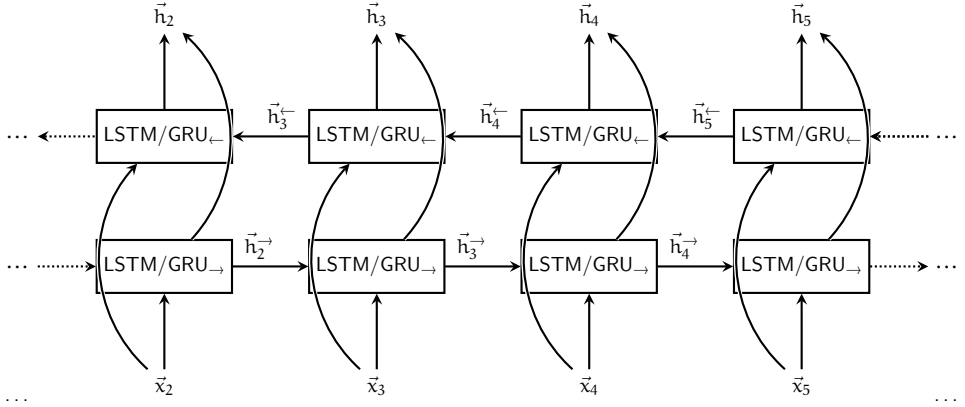
Choices vary in picking the Translation Architecture

- Directionality - Unidirectional or bidirectional
- number of hidden layers and units
- Plain vanilla RNN
- Long Short-term Memory units
- Gated Recurrent Unit
- Choice of Learning Algorithm

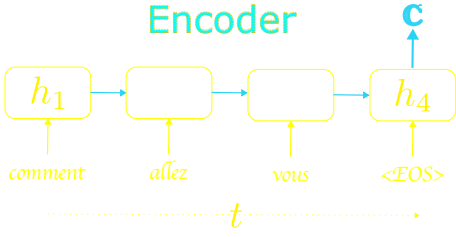
SEQ2SEQ ENCODER

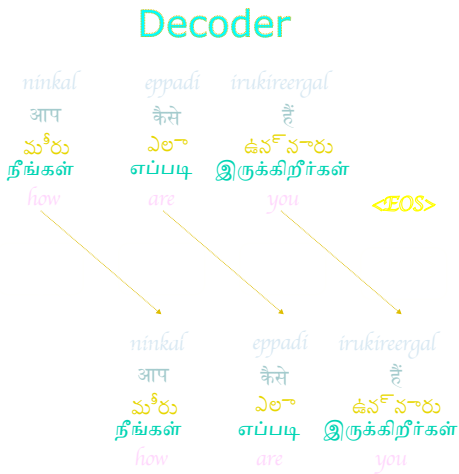


BIDIRECTIONAL RNN

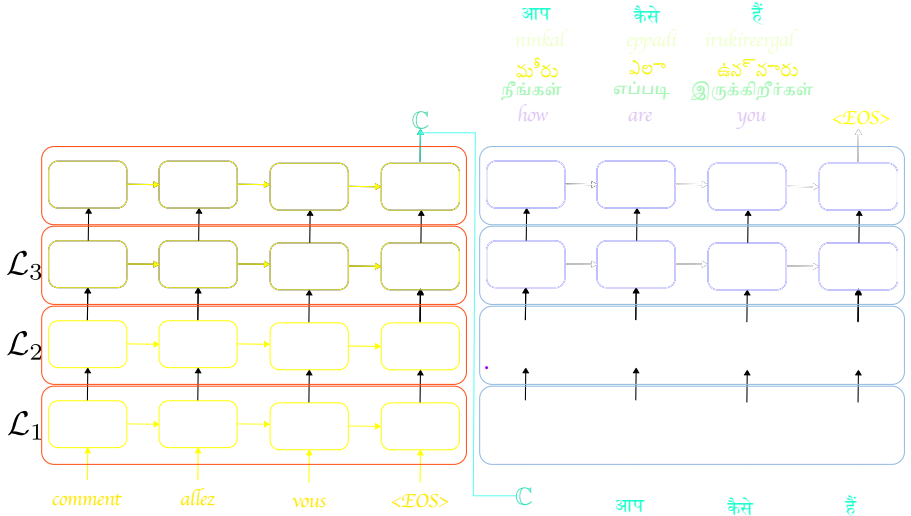


SEQUENCE TO SEQUENCE TRANSLATION - NMT





SEQUENCE TO SEQUENCE TRANSLATION - DEEP RNN



APPLICATIONS

- Translation
- Dialog
- Code generation!

RNN WITH ALIGNMENT

- The objective of attention is to capture the information from the passage tokens that is relevant to the contents of the translation
- Different parts of an input have different levels of significance
- Different parts of the output may even consider different parts of the input as "important"
- The purpose of the attention mechanism is to let the decoder *peek* at the relevant information encapsulating the source sentence as it generates the answer
- Attention mechanisms provide the decoder network with the entire input sequence at every decoding step; the decoder can then decide what input words are important at any point in time

- The attention-based model learns to assign significance to different parts of the input for each step of the output.
- In the context of translation, attention can be thought of as "alignment."
- Bahdanau et al [4] argue that the attention scores α_{ij}, at decoding step i, signify the words in the source sentence that align with word j in the target.
- We can use attention scores to build an alignment table. It is a table mapping of words in the source to corresponding words in the target sentence - based on the learned encoder and decoder from our Seq2Seq NMT system.

ENCODER

Let $x(x_1, x_2, \ldots, x_n)$ and $y(y_1, y_2, \ldots, y_m)$ be the source and target sentences
The encoder reads the input sentence x and converts into a context vector c

$$h_t = f(x_t, h_{t-1}) - \text{hidden values calculated at time } t \quad (23)$$

$$c = g(h_1, h_2, \ldots, h_n) - \text{context vectors computed using all } h_t \text{ values} \quad (24)$$

where functions f and g are non-linear functions.

How does c differ from the *context* of an n-gram language model?

DECODER

Decoder is trained to predict the next word using the c computed by the encoder

$$p(y) = p(y_t|c,\{y_1,y_2,\ldots,y_{t-1}\}) \quad (25)$$

For the RNN, the probability of the next word $p(y_t)$ is computed using

$$p(y_t) = g(y_{t-1}, s_t, c) \quad (26)$$

The hidden states s of the decoder are computed using a recursive formula of the form $s_i = f(s_{i-1}, y_{i-1}, c_i)$, where s_{i-1} is the previous hidden vector, y_{i-1} is the generated word at the previous step, and c_i is a context vector that capture the context from the original sentence that is relevant to the time step i of the decoder.

Figure

Conditional probability for each output neuron

$$p(y_i|y_1, y_2, \ldots, x) = g(y_{i-1}, s_i, c_i) \tag{27}$$

where s_i if the RNN hidden neuron at time i and

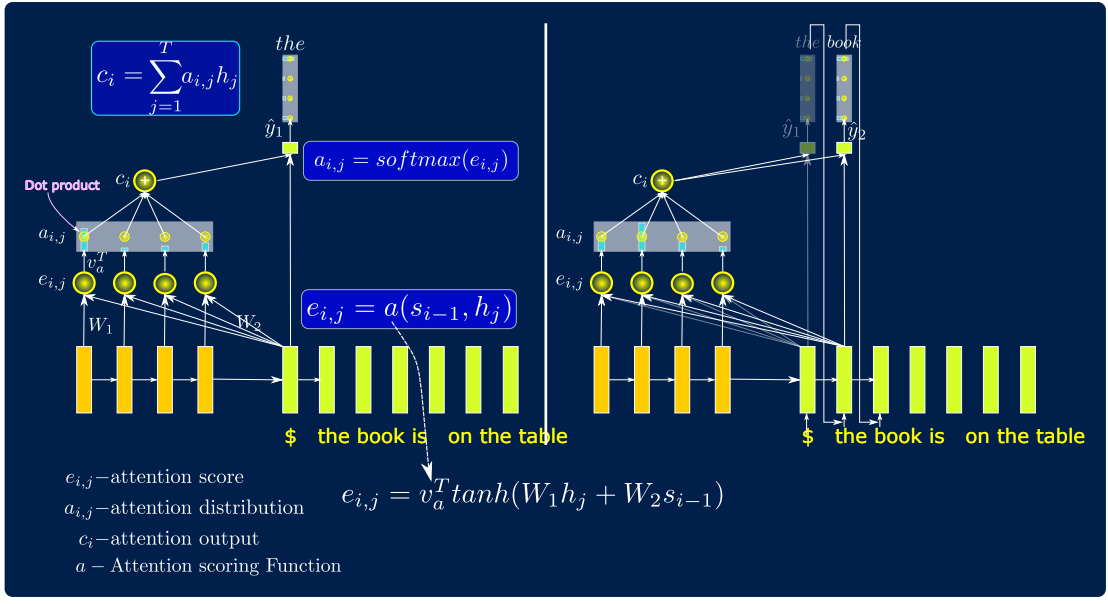
$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{28}$$

The context vector c_i depends on the sequence of annotations $(h_1, h_2, \ldots . h_{T_x})$[4].
Each h_i contains information about every word with a strong focus on context words surrounding the i^{th} word of the input sequence.
The context vector c_i is computed as the weighted sum of these annotations h_i

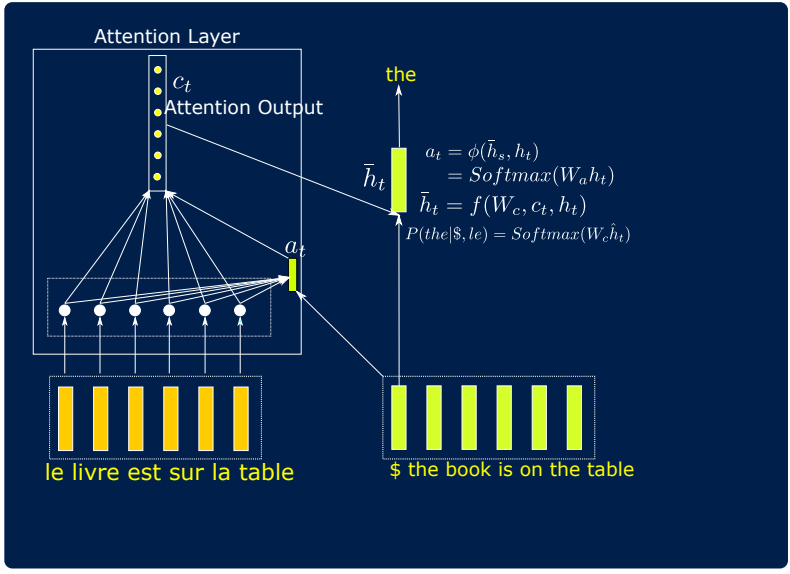
$$c_i = \sum_{j=1}^{T_x} \alpha_{i,j} h_j \qquad \alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{t_x} \exp(e_{i,k})} \qquad e_{i,j} = a(s_{i-1,h_j})$$

α_{ij} of each annotation h_j is computed by is the alignment model. This learns how well the inputs surrounding position j and the output at position i match

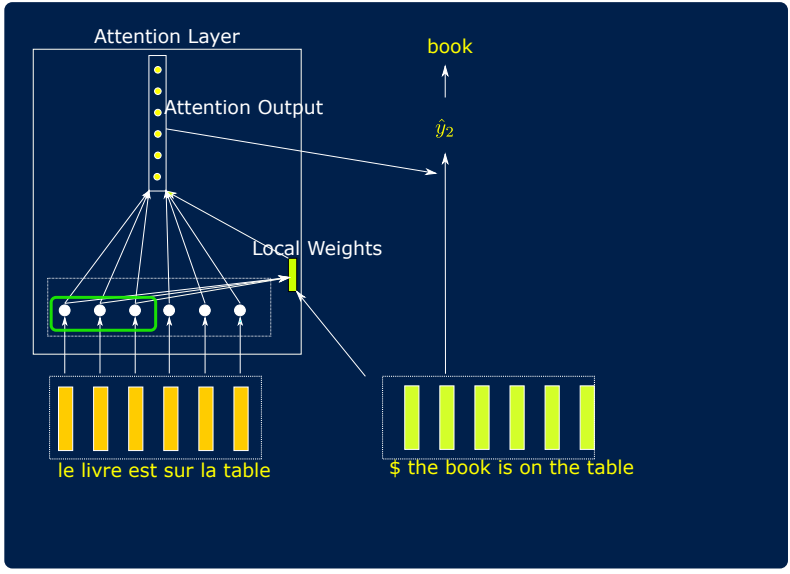
- The alignment is explicitly computed and not latent
- This alignment model is also trained along with the translation model
- α_{ij} is the probability that the target word y_i is aligned to the source x_j
- c_i is the expected annotation over all possible annotations α_{ij}
- α_{ij} or e_{ij} reflects the importance of the annotation h_j wrt to the previous hidden state s_{i-1} of the target. This enables the next state s_i to generate y_i
- The decoder decides which part of the input is important to generate a respective translation rather than depending on the encoded vector of the entire sentence
- Decoder has control over the input sequence and selectively learns to align words/phrases automatically

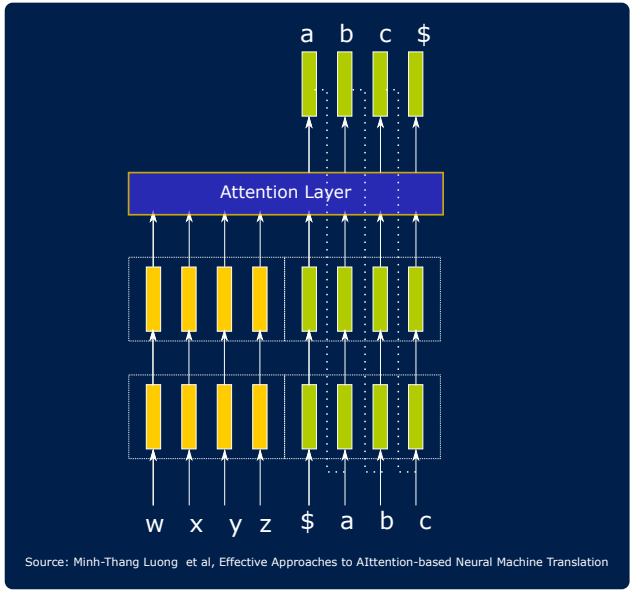

$c_i = \sum_{j=1}^{T} a_{i,j} h_j$
the
$\hat{y}_1$
$a_{i,j} = softmax(e_{i,j})$
Dot product
c_i
$a_{i,j}$
v_a^T
$e_{i,j}$
W_1
W_2
$e_{i,j} = a(s_{i-1}, h_j)$
$ the book is on the table
$e_{i,j}$−attention score
$a_{i,j}$−attention distribution
c_i−attention output
a − Attention scoring Function
$e_{i,j} = v_a^T tanh(W_1 h_j + W_2 s_{i-1})$
the book
$\hat{y}_2$
$ the book is on the table

TRANSLATION WITH GLOBAL ATTENTION



TRANSLATION WITH LOCAL ATTENTION




a b c $
Attention Layer
w x y z $ a b c
Source: Minh-Thang Luong et al, Effective Approaches to AIttention-based Neural Machine Translation

INFERENCE: INPUTTING A NEW SENTENCE

- The goal is to generate the most probable target sequence $\mathtt{y}^*$ (French) for a new source sentence $\mathtt{x}$ (English).
- The decoding strategy, typically **Beam Search**, determines the sequence quality.

PHASE 1: ENCODING THE SOURCE SENTENCE

1. **Input Tokenization & Embedding:** The English source sentence, e.g., "The cat sat on the mat," is tokenized and each word is converted into its embedding vector.
2. **Bidirectional RNN Run:** The trained Bidirectional Encoder (Bi-GRU/LSTM) processes the sequence.
3. **Output Annotations (H):** The encoder produces a sequence of rich hidden states (annotations) for every input word.

$$\mathrm{H} = (\mathrm{h}_1, \mathrm{h}_2, \ldots, \mathrm{h}_m)$$

   - H serves as the **Encoder Memory** that the decoder will reference at every subsequent step.
4. **Initialization:** The Decoder's initial hidden state (s_0) is set, usually derived from the final encoder states.

PHASE 2: ITERATIVE DECODING WITH ATTENTION I

The process starts with $\hat{y}_0 = \langle \text{SOS} \rangle$ and previous state s_{t-1}.

1. **Decoder Input:** The decoder takes the embedding of the previous predicted word ($\hat{y}_{t-1}$) as input.
2. **Attention Scoring (Alignment):** The decoder's previous state (s_{t-1}) is compared to **every** encoder annotation (h_i) using the additive attention function to get the relevance score $e_{t,i}$.

$$\text{Score } e_{t,i} = v_a^T \tanh(W_a s_{t-1} + U_a h_i)$$

3. **Attention Weights (α):** Scores are normalized using Softmax to produce the attention weights.

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{m} \exp(e_{t,k})}$$

4. **Context Vector Generation (c_t):** The context vector is computed as a weighted sum of the Encoder Annotations H.

$$c_t = \sum_{i=1}^{m} \alpha_{t,i} h_i$$

- c_t is the dynamic context, summarizing the most relevant English words for the current French word.

5. **Decoder RNN Update:** The new hidden state s_t is computed using s_{t-1}, $E(\hat{y}_{t-1})$, and the fresh context c_t.

$$s_t = \text{RNN}(s_{t-1}, E(\hat{y}_{t-1}), c_t)$$

6. **Prediction and Selection:** The model predicts the probability distribution P_t over the French vocabulary. The decoding strategy (e.g., Beam Search) selects the next word(s) $\hat{y}_t$ to continue the loop.

PHASE 3: TERMINATION

- The iterative process repeats until the **End-of-Sequence ($\langle \mathrm{EOS} \rangle$)** token is predicted and selected.
- **Final Output:** If using Beam Search, the partial sequence with the highest total cumulative log-probability is output as the final French translation $\mathbf{y}^*$.

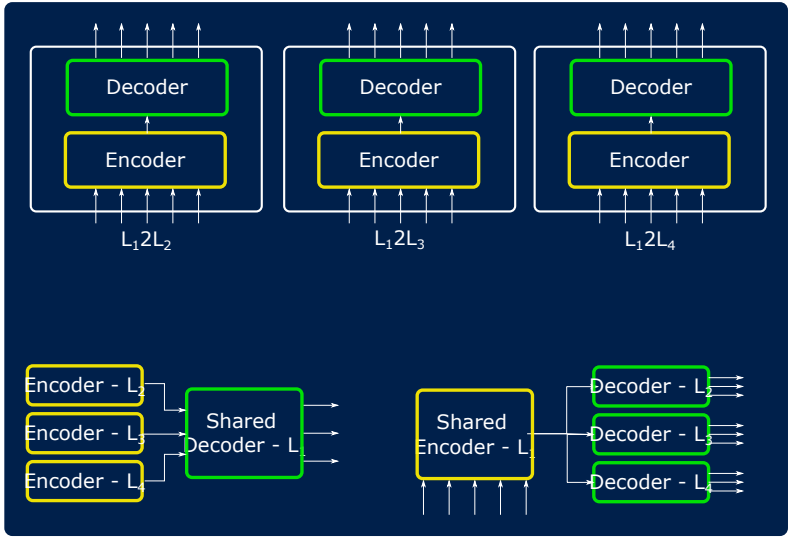
A TYPICAL SETUP

Sentence pairs	3-5M
English words	110M
French words	116M
Vocabulary	≈50K (Source and Target)
Word Embedding size	1000
Hidden layer	1000 LSTM cells
Stacked Hidden Layer	4-8
Learning Rate	Initially as high as 1 and exponential reduction
Training	
Mini batch Gradient Descend size	128
Training Time	1 GPU - about 7-10 days
Evaluation	Bleu - scores ranging from 27-32

ADVANTAGES OF ATTENTION

- Ability to focus on significant part of the sentence
- Ability to peek into source sentence
- Reduces the problem of vanishing gradient
- Alignments are found automatically during the training process
- Improves NMT performance for alignment

DIFFERENT TYPE OF NMT

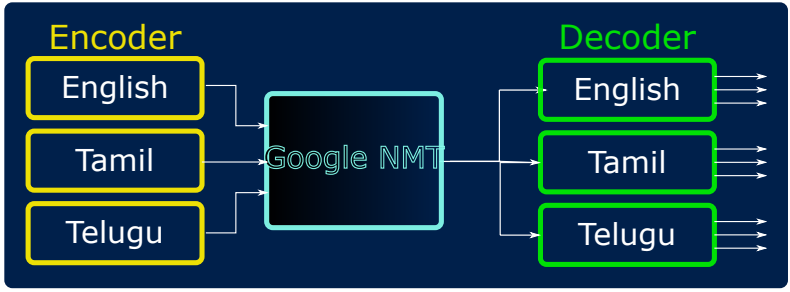


NMT FROM GOOGLE - ZERO-SHOT TRANSLATION

- Moved away from maintaining Seq2Seq model for every pair of languagess
- A single system that translates between any two languages even in the absence of the training corpus for these two languages
  - Assume that only examples of Japanese-English and Korean-English translations are available, Google found hat the multilingual NMT system trained on this data could actually generate reasonable Japanese-Korean translations.
  - Is it trained create the Interlingua?
  - Is the system learning a common representation or a translational knowledge?

Ref: Johnson et el. 2016, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation"

ZERO-SHOT TRANSLATION



FROM RNN ATTENTION TO THE TRANSFORMER

- RNN encoder–decoders with attention (Bahdanau, 2014) established attention as the key ingredient of NMT
- The Transformer (Vaswani et al., 2017, "Attention Is All You Need") removed recurrence entirely. It uses *only* attention plus feed-forward layers
- Benefits: full parallelism over positions, better handling of long-range dependencies, and no vanishing-gradient bottleneck
- Consequences for MT:
  - Transformer-based systems are now the standard for machine translation
  - Massively multilingual models (e.g., NLLB, M2M-100) translate between hundreds of language pairs
  - General-purpose LLMs perform competitive translation via prompting, without a dedicated MT model

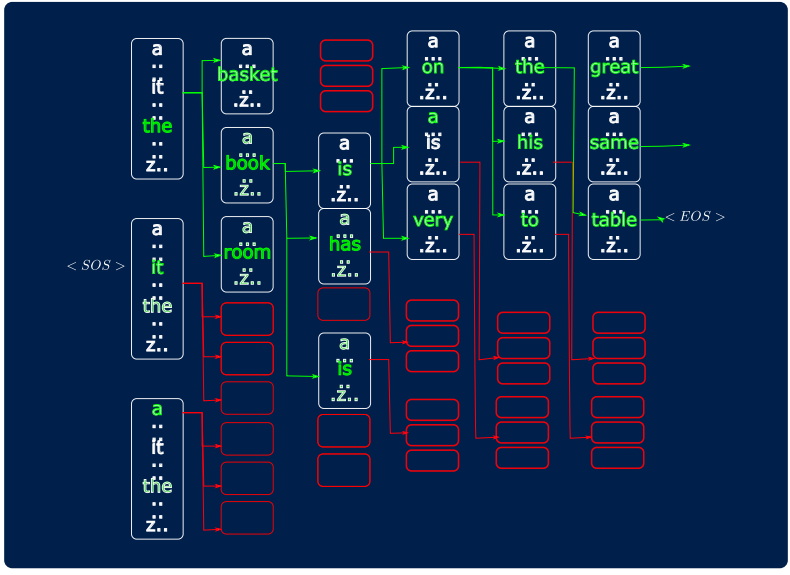
BEAM SEARCH

Beam search is a heuristic search algorithm that selects a few candidate hypothesis from $|V|$. It reduces memory requirement by using only a $M < |V|$ candidates using a score.

- Maintain M candidates/hypothesis at each time step - $C_t = (x_1^1, ..x_t^1)...(x_1^M...x_t^M)$
- Compute C_{t+1} by expanding C_t and keeping the best M candidates
- $\tilde{C} = \bigcup_{i=1}^{M} C_{t-1}^i$

A typical beam width of 5–10 is used in NMT; BLEU scores are comparable across beam widths in this range. (Modern LLM decoding usually prefers sampling methods over beam search for open-ended generation.)

BEAM SEARCH - BEAM WIDTH = 3



Figure

BEAM SEARCH EXAMPLE

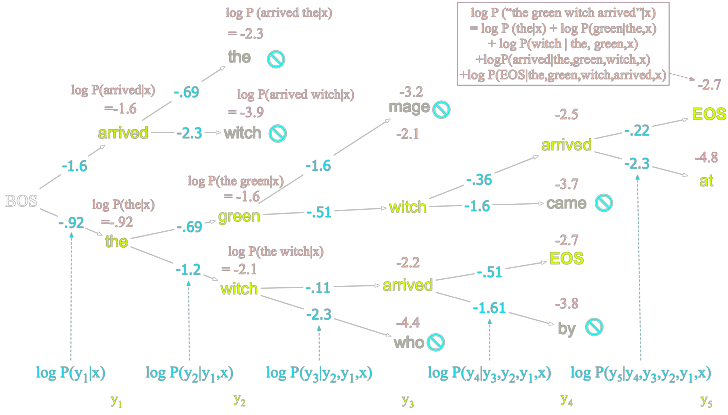


Figure: Scoring for beam search decoding with a beam width of k= 2. We maintain the log probability of each hypothesis in the beam by incrementally adding the logprob of generating each next token. Only the top k paths are extended to the next step

BEAM SEARCH SUMMARY

1. Use all possible partial translations - exhaustive search
2. Beam size, $b = 1$ - greedy search - Words are predicted until the $< EOS >$ is found
3. $b > 1$ - more than one hypotheses
4. Each hypothesis will be produced until the $< EOS >$ is found
5. Each hypothesis will be the translation of the source sentence
6. The length of all hypothesis may not be the same
7. We could use different ***terminate*** conditions
   - Fixed time steps
   - Compute until $< EOS >$ is reached for each hypothesis
8. Use either log probability or product of conditional probability to find the scores for each hypothesis that maximizes
   - $P(y_1, y_2, ...y_m|\mathbf{X}) = \prod_{t=1}^{T} P(y_t| < SOS >, \ldots, y_{t-1}, \mathbf{X})$
   - $\log P(y_1, y_2, ...y_m|\mathbf{X}) = \sum_{t=1}^{T} \log P(y_t| < SOS >, \ldots, y_{t-1}, \mathbf{X})$

PRETRAINING

- Learns linguistic patterns & world knowledge from massive text datasets
- Trains models to predict text sequences, building foundational language understanding

FUNDAMENTALS OF LLM DECODING

- **Core Components:**
  - Context window (prior generated text)
  - Vocabulary probability scores
  - Decoding strategy algorithm
- Autoregressive generation: $\mathrm{P}(w_t|w_{<t})$
- Key challenge: Balance between:
  - Accuracy vs. Creativity
  - Coherence vs. Diversity

GREEDY DECODING

$$\hat{w}_i = \arg\max_{w \in V} P(w_i | w_{<i})$$

- **Advantages:**
  - Maximizes local probability
  - High coherence
- **Limitations:**
  - Repetitive outputs
  - Lacks creativity
  - No error recovery

TEMPERATURE SCALING[6]

$$P(w_t) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

- **Effects:**
  - $T \to 0$: approaches greedy (deterministic) behavior
  - $T = 1$: normal sampling from the model distribution
  - $T > 1$: flatter distribution, increased randomness
- Applications:
  - $T < 1$: Technical writing
  - $T > 1$: Creative writing

TOP-K & TOP-P SAMPLING

Top-k:[7]

- Fixed candidate count
- $k = 1 \Rightarrow$ greedy
- Risk of including implausible tokens

Comparison:

- Top-p generally more flexible
- ChatGPT uses top-p

Top-p (Nucleus):[8]

- Dynamic candidate selection
- Cumulative probability threshold
- $\sum p_i \geq p$

MIN-P SAMPLING

Adaptive threshold calculation[9]:

$$\text{Threshold}_t = \max(P_t) \times \text{min\_p}$$

- Retain tokens where $p_i \geq$ Threshold
- Advantages:
  - Dynamic vocabulary selection
  - Maintains model confidence
  - Reduces low-quality outputs
- Typical min_p values: 0.05-0.2

DECODING STRATEGY PROPERTIES

Method	Temp.	Stochastic
Greedy	No	No
Beam Search	No	No
Temp. Sampling	Yes	Yes
Top-k	Yes	Yes
Top-p	Yes	Yes
Min-p	Yes	Yes

- Key considerations:
  - Application requirements
  - Output quality needs
  - Computational resources

DIFFICULTIES WITH HUMAN EVALUATION

- Human evaluations are extensive but expensive
- A need for quick, reusable, inexpensive method that correlates highly with human evaluation
- Many aspects of translation,including adequacy and fluency should be considered during the automatic evaluation
- Automatic evaluation is a boon to developers of LLM
- Two important aspects required for automatic evaluation
  1. A good metric
  2. A good/gold standards as references

THE IDEA

- Many possible sentences for a given sentence
- A good evaluator identifies a good candidate using adequacy and fluency

The main idea is to use a weighted average of variable length phrase matches against the reference translations[10]

Candidate 1: **It is a guide to action which ensures that the military always obeys the commands of the party**

Candidate 2: **It is to insure the troops forever hearing the activity guidebook that party direct**

Reference: **It is a guide to action that ensures that the military will for ever heed Party commands**

If many words and phrases are shared between the candidate and the reference translations, then it is a good candidate

Can n-grams help in matching the words and phrases?

UNIGRAM PRECISION

C1: It is a guide to action which ensures that the military always obeys the commands of the party

R1: It is a guide to action that ensures that the military will forever heed Party commands

R2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

R3: It is the practical guide for the army to heed the directions of the party.

Unigram precision $= \frac{17}{18}$

C2: It is to insure the troops forever hearing the activity guidebook that party direct

R1: It is a guide to action that ensures that the military will forever heed Party commands

R2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

R3: It is the practical guide for the army always to heed the directions of the party.

Unigram precision $= \frac{8}{14}$

BLEU EVALUATION

```
from nltk.translate.bleu_score import sentence_bleu
references = [
['It', 'is', 'a', 'guide', 'to', 'action', 'that',
  'ensures', 'that', 'the', 'military', 'will',
  'forever', 'heed', 'Party', 'commands'],
['It', 'is', 'the', 'guiding', 'principle',
 'which', 'guarantees', 'the', 'military',
  'forces', 'always', 'being', 'under',
  'the', 'command', 'of', 'the', 'Party.'],
['It', 'is', 'the', 'practical', 'guide', 'for',
 'the', 'army', 'to', 'heed', 'the',
  'directions', 'of', 'the', 'party.']
]
hypothesis = ['It', 'is', 'a', 'guide', 'to',
 'action', 'which', 'ensures', 'that', 'the',
 'military', 'always', 'obeys', 'the',
 'commands', 'of', 'the', 'party']
score = sentence_bleu(references, hypothesis, weights=(1, 0, 0, 0))
print(score)
```

MODIFIED- N-GRAM PRECISION

Compare the number of n-grams in the candidate and in the reference translation
Penalize models that produces many words of the same type

- Count the number of times a word occurs in any single reference translation
- $\text{Count}_{\text{clip}} = \min(\text{Candidate Count}, \text{Maximum Reference Count})$

Refer the previous slide for the examples

$$\text{Modified unigram precision for C1} = \frac{17}{18} \qquad \text{Modified unigram precision for C2} = \frac{8}{14}$$

C3: the the the the the the the
R4: the cat is on the mat

$$\text{Unigram precision} = \frac{7}{7}$$
$$\text{Modified unigram precision} = \frac{2}{7}$$
$$\text{Modified bigram precision} = 0$$

Modified Unigram precision defines the adequacy of the translation, while modified bigram precision matches the fluency of the translation

MODIFIED BIGRAM PRECISION

(It,is),(is,a),(a,guide),
(guide,to),(to,action),
(action,which),(which,ensures),
(ensures,that),(that,the),
(the,military),(military,always),
(always,obeys),(obeys,the),
(the,commands),(commands,of),
(of,the),(the,party)

Modified bigram precision for C1 $= \frac{10}{17}$

```
(It,is),(is,a),(a,guide),(guide,to),
(to,action),(action,that),(that,ensures),
(ensures,that),(that,the),(the,military),
(military,will),(will,forever),(forever,heed),
(heed,Party),(Party,commands)

(It,is),(is,the),(the,guiding),
(guiding,principle),(principle,which),
(which,guarantees),(guarantees,the),
(the,military),(military,forces),(forces,always),
(always,being),(being,under),(under,the),
(the,command),(command,of),
(of,the),(the,Party)

(It,is),(is,the),(the,practical),(practical,guide),
(guide,for),(for,the),(the,army),
(army,always),(always,to),(to,heed),
(heed,the),(the,directions),
(directions,of),(of,the),(the,party)
^^I^^I
```

MODIFIED BIGRAM PRECISION - CANDIDATE 2

(It,is),(is,to),(to,insure),
(insure,the),(the,troops),
(troops,forever),(forever,hearing),
(hearing,the),(the,activity),
(activity,guidebook),
(guidebook,that),(that,party),
(party,direct)

Modified bigram precision for C2 $= \frac{1}{13}$

```
(It,is),(is,a),(a,guide),(guide,to),
(to,action),(action,that),(that,ensures),
(ensures,that),(that,the),(the,military),
(military,will),(will,forever),(forever,heed),
(heed,Party),(Party,commands)

(It,is),(is,the),(the,guiding),
(guiding,principle),(principle,which),
(which,guarantees),(guarantees,the),
(the,military),(military,forces),(forces,always),
(always,being),(being,under),(under,the),
(the,command),(command,of),
(of,the),(the,Party)

(It,is),(is,the),(the,practical),(practical,guide),
(guide,for),(for,the),(the,army),
(army,always),(always,to),(to,heed),
(heed,the),(the,directions),
(directions,of),(of,the),(the,party)
^^I^^I
```

COMBINING N-GRAM PRECISIONS

- Modified n-gram precisions decay exponentially as n increases[10]
- BLEU averages the log-precisions with uniform weights (the geometric mean of the modified n-gram precisions) to handle this decay
- A short candidate ($c < r$) can inflate precision, so a brevity penalty (BP) is applied when $c \leq r$

$$BP = \begin{cases} 1, & \text{if } c > r \\ \exp(1 - \frac{r}{c}), & \text{if } c \leq r \end{cases}$$

where r is the effective length of the reference corpus and c is the length of the candidate sentence

BLEU score is obtained by

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{29}$$

where N is the n-gram size (BLEU uses 4-gram by default),w_n is the weights associated with unigram,bigram,trigram and 4-grams,and p_n is the modified precision score of the test corpus. Here,$\sum_{n=1}^{N} w_n = 1$. One option for $w_n = \frac{1}{N}$

$$p_n = \frac{\sum_{c \in C} \sum_{ngrams \in C} \text{Count}_{clip}(ngrams)}{\sum_{c \in C} \sum_{ngrams \in C} \text{Count}(ngrams)} \tag{30}$$

BLEU - DEMO

BLEU Demo

APPLICATIONS OF BLEU

BLEU is designed as a corpus measure

- Machine translation
- Image labeling
- Text summarization
- Speech recognition

OTHER METRICS

- NIST - National Institute of Standards and Technology - based on BLEU
- METEOR - Metric for Evaluation of Translation with Explicit ORdering
  - Uses stemming and synonymy matching
- WER - Word Error Rate
  - Uses edit distance (Levenshtein distance)
  - Finds minimum number of edit operations such as insertion,deletions or substitutions,needed to change the candidate sentence into the reference sentence
- GLEU - Google BLEU
  - Correlates well with BLEU, and works at the sentence level
- Embedding / model-based metrics (increasingly preferred for fluent LLM output)
  - BERTScore measures token similarity using contextual embeddings
  - BLEURT and COMET are learned metrics trained to match human judgments
  - LLM-as-a-judge prompts a strong LLM to rate or compare outputs

BARTSCORE[11]

- Reframe text evaluation as a text generation task.
- Utilize pre-trained BART model to assess the likelihood of reference text given the generated text.
- Leverage conditional generation capabilities of sequence-to-sequence models.
- Capture semantic similarity and fluency by modeling generation probabilities.

BARTSCORE: MATHEMATICAL FORMULATION

- Given generated text y and reference text x, BARTSCORE computes:
- Forward Score:

$$\text{BARTSCORE}_{\text{forward}}(y, x) = \frac{1}{|x|} \sum_{i=1}^{|x|} \log P(x_i | x_{<i}, y)$$

- Reverse Score:

$$\text{BARTSCORE}_{\text{reverse}}(x, y) = \frac{1}{|y|} \sum_{i=1}^{|y|} \log P(y_i | y_{<i}, x)$$

- Overall BARTSCORE:

$$\text{BARTSCORE}(y, x) = \frac{1}{2} [\text{BARTSCORE}_{\text{forward}}(y, x) + \text{BARTSCORE}_{\text{reverse}}(x, y)]$$

where $P(x_i | x_{<i}, y)$ is the probability of token x_i given previous tokens $x_{<i}$ and generated text y.

BARTSCORE: ADVANTAGES AND RESULTS

- Outperforms existing metrics (e.g., BLEU, ROUGE, BERTScore) in various tasks.
- Strong correlation with human judgments across different evaluation dimensions.
- Versatile: can be adapted to different evaluation tasks through prompting or fine-tuning.
- Efficient: leverages pre-trained models, reducing the need for task-specific training data.
- EXPLAINABOARD leaderboard: fosters transparency and reproducibility in text evaluation.

REFERENCES I

[1] Bernard Vauquois. "A survey of formal grammars and algorithms for recognition and transformation in mechanical translation.". In: *Ifip congress (2)*. Vol. 68. 1968, pp. 1114–1122.

[2] Peter F. Brown et al. "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Comput. Linguist.* 19.2 (June 1993), pp. 263–311. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972470.972474.

[3] Manning et al. *Foundations of statistical natural language processing*. Mit Press. MIT Press, 1999. ISBN: 9780262133609. URL: https://books.google.co.in/books?id=YiFDxbEX3SUC.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

REFERENCES II

[5] KyungHyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259 (2014). arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

[6] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. "A learning algorithm for boltzmann machines". In: *Cognitive Science* 9.1 (1985), pp. 147–169. ISSN: 0364-0213. DOI: https://doi.org/10.1016/S0364-0213(85)80012-4. URL: https://www.sciencedirect.com/science/article/pii/S0364021385800124.

[7] Angela Fan, Mike Lewis, and Yann Dauphin. "Hierarchical Neural Story Generation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 889–898. DOI: 10.18653/v1/P18-1082. URL: https://aclanthology.org/P18-1082/.

REFERENCES III

[8] Ari Holtzman et al. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL]. URL: https://arxiv.org/abs/1904.09751.

[9] Nguyen Nhat Minh et al. "Turning Up the Heat: Min-p Sampling for Creative and Coherent LLM Outputs". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=FBkpCyujtS.

[10] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://www.aclweb.org/anthology/P02-1040.

[11] Weizhe Yuan, Graham Neubig, and Pengfei Liu. *BARTScore: Evaluating Generated Text as Text Generation*. 2021. arXiv: 2106.11520 [cs.CL]. URL: https://arxiv.org/abs/2106.11520.