

Machine Translation



Dan Klein
UC Berkeley

Many slides from John DeHoro and Philip Koehn

Translation Task

- Text is both the input and the output.
- Input and output have roughly the same information content.
- Output is more predictable than a language modeling task.
- Lots of naturally occurring examples (but not much metadata).

Translation Examples

English-German News Test 2013 (a standard dev set)

Republican leaders justified their policy by the need to combat electoral fraud.

Die Führungskräfte der Republikaner
The Executives of the republican
rechtfertigen ihre Politik mit der
justify your politics With of the
Notwendigkeit, den Wahlbetrug zu
need, the election fraud to
bekämpfen.
fight.

Variety in Translations?

Human-generated reference translation

A small planet, whose is as big as could destroy a middle sized city, passed by the earth with a distance of 463 thousand kilometers. This was not found in advance. The astronomers got to know this incident 4 days later. This small planet is 50m in diameter. The astronomers are hard to find it for it comes from the direction of sun.

A commercial system from 2002

A volume enough to destroy a medium city small planet is big, flit earth within 463,000 kilometres of close however were not in advance discovered, astronomer just knew this matter after four days. This small planet diameter is about 50 metre, from the direction at sun, therefore astronomer very hard to discovers it.

Google Translate, 2020

An asteroid that was large enough to destroy a medium-sized city, swept across the earth at a short distance of 463,000 kilometers, but was not detected early. Astronomers learned about it four days later. The asteroid is about 50 meters in diameter and comes from the direction of the sun, making it difficult for astronomers to spot it.

Evaluation

BLEU Score

BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty (harshly penalizes translations shorter than the reference).

$$Matched_i = \sum_{t_i} \min \{ C_h(t_i), \max_j C_j(t_i) \}$$

If "of the" appears twice in hypothesis h but only at most once in a reference, then only the first is "correct".

$$P_i = \frac{Matched_i}{H_i}$$

"Clipped" precision of n-gram tokens

$$B = \exp \left\{ \min \left(0, \frac{n-L}{n} \right) \right\}$$

Brevity penalty only matters if the hypothesis corpus is shorter than the sum of (shortest) references.

$$BLUE = B \left(\prod_{i=1}^4 P_i \right)^{\frac{1}{4}}$$

BLEU is a mean of clipped precisions, scaled down by the brevity penalty.

Evaluation with BLEU

In this sense, the measures will partially undermine the American democratic system.

In this sense, these measures partially undermine the democratic system of the United States.

BLEU = 26.52, 75.0/40.0/21.4/7.7 (BP=1.000, ratio=1.143, hyp_len=16, ref_len=14)

Epstein et al., 2020 BLEU is a method for automatic evaluation of machine translation.

Corpus BLEU Correlations with Average Human Judgments

There are ecological correlations over multiple segments; segment-level BLEU scores are noisy.

Commercial machine translation providers seem to all perform human evaluations of some sort.

$R^2 = 0.25$
 $R^2 = 0.12$

(Ma et al., 2019) Results of the WMT19 Metrics Shared Task: Segment-Level and Strong MT Systems Pose Big Challenges

Human Evaluations

Direct assessment: adequacy & fluency

- Monolingual: Ask humans to compare machine translation to a human-generated reference. (Easier to source annotators)
- Bilingual: Ask humans to compare machine translation to the source sentence that was translated. (Compares to human quality)
- Annotators can assess segments (sentences) or whole documents.
- Segments can be assessed with or without document context.

Ranking assessment:

- Raters are presented with 2 or more translations.
- A human-generated reference may be provided, along with the source.
- "In a pairwise ranking experiment, human raters assessing adequacy and fluency show a stronger preference for human over machine translation when evaluating documents as compared to isolated sentences." (Laußli et al., 2018)

Editing assessment: How many edits required to reach human quality

Leahy et al., 2016 The Machine Translation Assessment Human Parity Scale for Document Translation

Epstein et al., 2020 Proceedings of the 2020 Conference on Machine Translation

Translationese and Evaluation

Translated text can: (Baker et al., 1993; Graham et al., 2019)

- be more explicit than the original source
- be less ambiguous
- be simplified (lexical, syntactically and stylistically)
- display a preference for conventional grammaticality
- avoid repetition
- exaggerate target language features
- display features of the source language

"If we consider only original source text (i.e. not translated from another language, or translationese), then we find evidence showing that human parity has not been achieved." (Torral et al., 2018)

Baker et al., 1993. English-Japanese and French-French: Implications and applications. In Proceedings of the 1993 Conference on Machine Translation Evaluation.


Torral et al., 2018. Missing the Untranslatable? Reassessing Claims of Human Parity in Human-Machine Translation.

How are We Doing? Example: WMT 2019 Evaluation

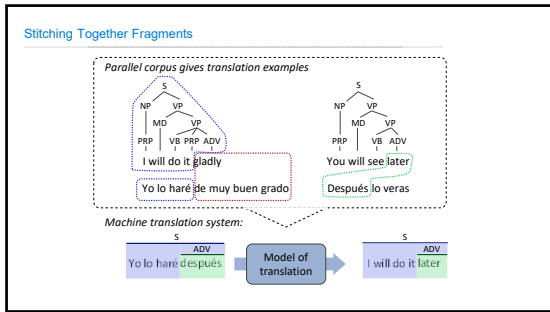
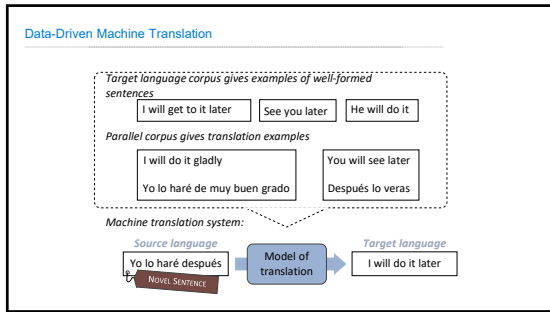
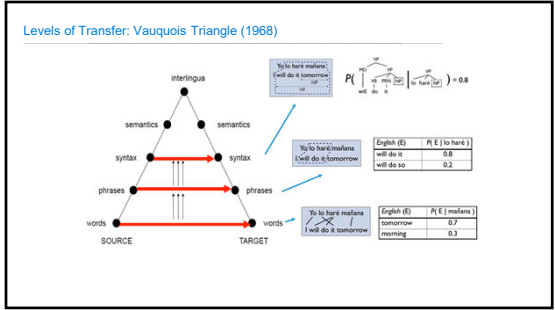
2019 segment-in-context direct assessment (Barraut et al. 2019):

- ✓ German to English: many systems are tied with human performance;
- ✗ English to Chinese: all systems are outperformed by the human translator;
- ✗ English to Czech: all systems are outperformed by the human translator;
- ✗ English to Finnish: all systems are outperformed by the human translator;
- ✓ English to German: Facebook-FAIR achieves super-human translation performance; several systems are tied with human performance;
- ✗ English to Gujarati: all systems are outperformed by the human translator;
- ✗ English to Kazakh: all systems are outperformed by the human translator;
- ✗ English to Lithuanian: all systems are outperformed by the human translator;
- ✓ English to Russian: Facebook-FAIR is tied with human performance.

Statistical Machine Translation
(1990 - 2015)



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 (1949)



Evolution of the Noisy Channel Model

$$P(e|f) \propto P(f|e) \cdot P(e)$$

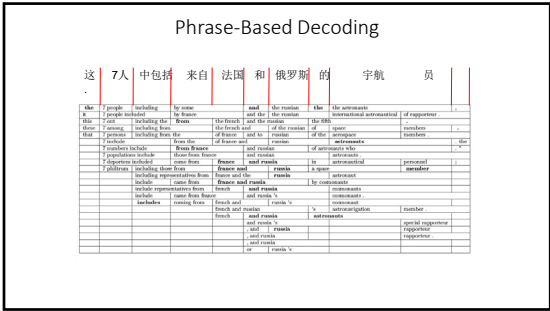
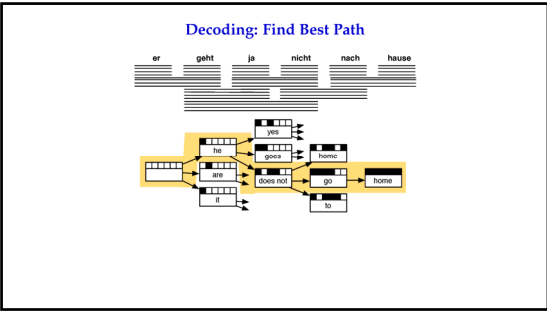
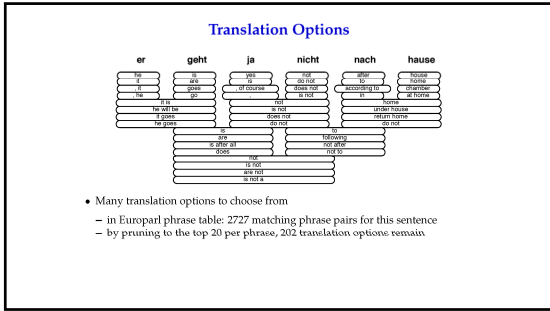
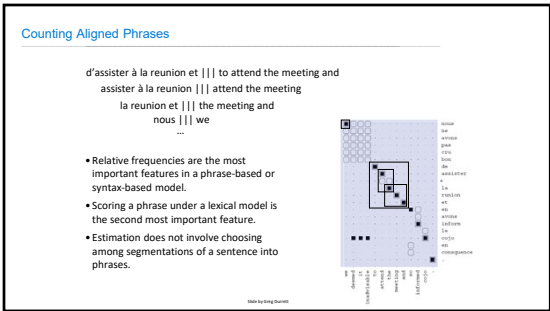
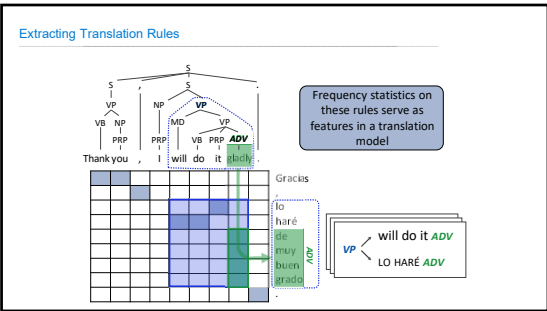
$$P(e|f) \propto P(f|e)^{\phi_{fm}} \cdot P(e)^{\phi_{fm}}$$

$$P(e|f) \propto \exp\left\{\sum_i w_i \cdot f_i(e, f)\right\}$$

Chosen to minimize loss

E.g., log P(e)

Word Alignment and Phrase Extraction



Machine Translation



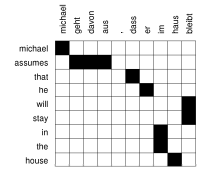
Dan Klein
UC Berkeley

Many slides from John DeHoro and Philip Koehn

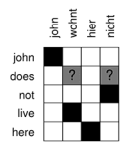
Word Alignments

Word Alignment

Given a sentence pair, which words correspond to each other?

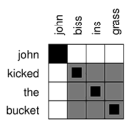


Word Alignment?



Is the English word **does** aligned to the German **wohnt** (verb) or **nicht** (negation) or neither?

Word Alignment?



How do the idioms **kicked the bucket** and **biss ins grass** match up?
Outside this exceptional context, **bucket** is never a good translation for **grass**

Lexical Translation / Word Alignment Models

Unsupervised Word Alignment

- Input: a *bitext*: pairs of translated sentences

nous acceptons votre opinion .
we accept your view .
- Output: *alignments*: pairs of translated words
 - When words have unique sources, can represent as a (forward) alignment function a from French to English positions

Word Alignment

- Even today models are often built on the IBM alignment models
- Create probabilistic word-level translation models
- The models incorporate latent (unobserved) word alignments
- Optimize the probability of the observed words
- Use the imputed alignments to reveal word-level correspondence
- Throw out the translation models themselves

Alignment

- In a parallel text (or when we translate), we align words in one language with the words in the other

1	2	3	4
das	Haus	ist	klein
the	house	is	small
1	2	3	4
- Word positions are numbered 1-4

Alignment Function

- Formalizing alignment with an alignment function
- Mapping an English target word at position i to a German source word at position j with a function $a: i \rightarrow j$
- Example

$$a: \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\}$$

Reordering

Words may be reordered during translation

1	2	3	4
klein	ist	das	Haus
the	house	is	small
1	2	3	4

$$a: \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$$

One-to-Many Translation

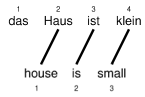
A source word may translate into multiple target words

1	2	3	4	5
das	Haus	ist	klitzeklein	klein
			/	\
the	house	is	very	small
1	2	3	4	5

$$a: \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}$$

Dropping Words

Words may be dropped when translated
(German article *das* is dropped)



$$a: [1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4]$$

Inserting Words

- Words may be added during translation
 - The English *just* does not have an equivalent in German
 - We still need to map it to something: special NULL token

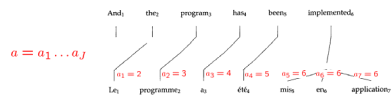


$$a: [1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4]$$

IBM Model 1: Allocation

IBM Model 1 (Brown 93)

- Alignments: a hidden vector called an *alignment* specifies which English source is responsible for each French target word.



$$\begin{aligned}
 P(f, a|e) &= \prod_j P(a_j = i) P(f_j | e_i) \\
 &= \prod_j \frac{1}{I+1} P(f_j | e_i) \\
 P(f|e) &= \sum_a P(f, a|e)
 \end{aligned}$$

Example

das		Haus		ist		klein	
e	t(e f)	e	t(e f)	e	t(e f)	e	t(e f)
the	0.7	house	0.8	is	0.8	small	0.4
that	0.15	building	0.16	's	0.16	little	0.4
which	0.075	home	0.02	exists	0.02	short	0.1
who	0.05	household	0.015	has	0.015	minor	0.06
this	0.025	shell	0.005	are	0.005	petty	0.04

$$\begin{aligned}
 p(e, a|f) &= \frac{e}{I+1} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein}) \\
 &= \frac{e}{I+1} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\
 &= 0.0028e
 \end{aligned}$$

IBM Models 1/2

E: Thank you . . . I shall do so gladly .

A: ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨

F: Gracias . lo haré de muy buen grado .

Model Parameters

Translation: $P(F_1 = \text{Gracias} | E_1 = \text{Thank})$ Alignment: $P(A_2 = 3)$

Expectation Maximization

- ### EM Algorithm
- Incomplete data
 - if we had *complete data*, would could estimate *model*
 - if we had *model*, we could fill in the *gaps in the data*
 - Expectation Maximization (EM) in a nutshell
 1. initialize model parameters (e.g. uniform)
 2. assign probabilities to the missing data
 3. estimate model parameters from completed data
 4. iterate steps 2-3 until convergence

EM Algorithm

... la maison ... la maison blue ... la fleur ...

... the house ... the blue house ... the flower ...

- Initial step: all alignments equally likely
- Model learns that, e.g., *la* is often aligned with *the*

EM Algorithm

... la maison ... la maison blue ... la fleur ...

... the house ... the blue house ... the flower ...

- After one iteration
- Alignments, e.g., between *la* and *the* are more likely

EM Algorithm

... la maison ... la maison bleu ... la fleur ...

... the house ... the blue house ... the flower ...

- After another iteration
- It becomes apparent that alignments, e.g., between *fleur* and *flower* are more likely (pigeon hole principle)

EM Algorithm

... la maison ... la maison bleu ... la fleur ...

 ... the house ... the blue house ... the flower ...

- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm

... la maison ... la maison bleu ... la fleur ...

 ... the house ... the blue house ... the flower ...

↓

$p(la|the) = 0.453$
 $p(le|the) = 0.334$
 $p(maison|house) = 0.876$
 $p(bleu|blue) = 0.563$
 ...

- Parameter estimation from the aligned corpus

IBM Model 1 and EM

- EM Algorithm consists of two steps
- Expectation-Step: Apply model to the data
 - parts of the model are hidden (here: alignments)
 - using the model, assign probabilities to possible values
- Maximization-Step: Estimate model from data
 - take assign values as fact
 - collect counts (weighted by probabilities)
 - estimate model from counts
- Iterate these steps until convergence

IBM Model 1 and EM

- We need to be able to compute:
 - Expectation-Step: probability of alignments
 - Maximization-Step: count collection

IBM Model 1 and EM

- Probabilities
 - $p(the|la) = 0.7$ $p(house|la) = 0.05$
 - $p(the|maison) = 0.1$ $p(house|maison) = 0.8$
- Alignments
 - la • • the la • • the la • • the la • • the
 - maison • • house maison • • house maison • • house maison • • house

$p(e, a|f) = 0.56$ $p(e, a|f) = 0.035$ $p(e, a|f) = 0.08$ $p(e, a|f) = 0.005$
 $p(a|e, f) = 0.824$ $p(a|e, f) = 0.052$ $p(a|e, f) = 0.118$ $p(a|e, f) = 0.007$
- Counts
 - $c(the|la) = 0.824 + 0.052$ $c(house|la) = 0.052 + 0.007$
 - $c(the|maison) = 0.118 + 0.007$ $c(house|maison) = 0.824 + 0.118$

Convergence

e	f	initial	1st it.	2nd it.	3rd it.	...	final
the	das	0.25	0.5	0.6364	0.7479	...	1
book	das	0.25	0.25	0.1818	0.1208	...	0
house	das	0.25	0.25	0.1818	0.1313	...	0
the	buch	0.25	0.25	0.1818	0.1208	...	0
book	buch	0.25	0.5	0.6364	0.7479	...	1
a	buch	0.25	0.25	0.1818	0.1313	...	0
book	ein	0.25	0.5	0.4286	0.3466	...	0
a	ein	0.25	0.5	0.5714	0.6534	...	1
the	haus	0.25	0.5	0.4286	0.3466	...	0
house	haus	0.25	0.5	0.5714	0.6534	...	1

Perplexity

- How well does the model fit the data?
- Perplexity: derived from probability of the training data according to the model

$$\log_2 PP = - \sum_i \log_2 p(e_i | f_i)$$

- Example (e=1)

	initial	1st it.	2nd it.	3rd it.	...	final
$p(\text{the haus:das haus})$	0.0625	0.1875	0.1905	0.1913	...	0.1875
$p(\text{the book:das buch})$	0.0625	0.1406	0.1790	0.2075	...	0.25
$p(\text{a book:en buch})$	0.0625	0.1875	0.1907	0.1913	...	0.1875
perplexity	4095	202.5	153.6	131.6	...	113.8

Problems with Model 1

- There's a reason they designed models 2-5!
- Problems: alignments jump around, align everything to rare words
- Experimental setup:
 - Training data: 1.1M sentences of French-English text, Canadian Hansards
 - Evaluation metric: alignment error Rate (AER)
 - Evaluation data: 447 hand-aligned sentences

IBM Model 2: Global Monotonicity

Monotonic Translation

Japan shaken by two new quakes

Le Japon secoué par deux nouveaux séismes

Local Order Change

Japan is at the junction of four tectonic plates

Le Japon est au confluent de quatre plaques tectoniques

IBM Model 2

- Alignments tend to the diagonal (broadly at least)

$$P(f, a|e) = \prod_j P(a_j = i | j, I, J) P(f_j | e_i)$$

$$P(dist = i - j) = \frac{1}{Z} e^{-\alpha(i-j)^2}$$

EM for Models 1/2

- Model 1 Parameters:
 - Translation probabilities (1+2) $P(f_j|e_i)$
 - Distortion parameters (2 only) $P(a_j = i|j, I, J)$
- Start with $P(f_j|e_i)$ uniform, including $P(f_j|null)$
- For each sentence:
 - For each French position j
 - Calculate posterior over English positions

$$P(a_j = i|f_j, e) = \frac{P(a_j = i|j, I, J)P(f_j|e_i)}{\sum_{i'} P(a_j = i'|j, I, J)P(f_j|e_{i'})}$$
 - (or just use best single alignment)
 - Increment count of word f_j with word e_i by these amounts
 - Also re-estimate distortion probabilities for model 2
- Iterate until convergence

HMM Model: Local Monotonicity

Phrase Movement

The HMM Model

E: Thank you . I shall do so gladly .

A:

F: Gracias , lo haré de muy buen grado .

Model Parameters

Emissions: $P(F_i = Gracias | E_{A_i} = Thank)$ Transitions: $P(A_2 = 3 | A_1 = 1)$

The HMM Model

- Model 2 preferred global monotonicity
- We want local monotonicity:
 - Most jumps are small
- HMM model (Vogel 96)

f	$i(f e)$
nationale	0.469
national	0.418
nationaux	0.054
nationales	0.029

$$P(f, a|e) = \prod_j P(a_j|a_{j-1})P(f_j|e_i)$$

$$P(a_j - a_{j-1})$$
- Re-estimate using the forward-backward algorithm
- Handling nulls requires some care
- What are we still missing?

Models 3+: Fertility

IBM Models 3/4/5

Mary did not slap the green witch

Mary not slap slap the green witch $n(3|slap)$

Mary not slap slap NULL the green witch $P(NULL)$

Mary no daba una botefada a la verde bruja $t(\{a\}the)$

Mary no daba una botefada a la bruja verde $d(j|i)$

[from Al-Onaizan and Knight, 1998]

Examples: Translation and Fertility

the

f	t(f e)	ϕ	n(ϕe)
le	0.497	1	0.746
la	0.207	0	0.254
les	0.155		
l'	0.086		
ce	0.018		
cette	0.011		

not

f	t(f e)	ϕ	n(ϕe)
ne	0.497	2	0.735
pas	0.442	0	0.154
non	0.029	1	0.107
rien	0.011		

farmers

f	t(f e)	ϕ	n(ϕe)
agriculteurs	0.442	2	0.731
les	0.418	1	0.228
cultivateurs	0.046	0	0.039
producteurs	0.021		

Example: Idioms

he is nodding

il hoche la tête

nodding

f	t(f e)	ϕ	n(ϕe)
signe	0.164	4	0.342
la	0.123	3	0.293
tête	0.097	2	0.167
oui	0.086	1	0.163
fait	0.073	0	0.023
que	0.073		
hoche	0.054		
hocher	0.048		
faire	0.030		
me	0.024		
approuve	0.019		
qui	0.019		
un	0.012		
faites	0.011		

Example: Morphology

should

f	t(f e)	ϕ	n(ϕe)
devrait	0.330	1	0.649
devraient	0.123	0	0.336
devrions	0.109	2	0.014
faudrait	0.073		
faut	0.058		
doit	0.058		
aurait	0.041		
doivent	0.024		
devons	0.017		
devrais	0.013		

Machine Translation

Berkeley

Dan Klein
UC Berkeley

Many slides from John DeNero and Philip Koehn

Phrase-Based Model

natuerlich hat john spass am spiel

of course john has fun with the game

- Foreign input is segmented in phrases
- Each phrase is translated into English
- Phrases are reordered

Getting Phrases

Extracting Phrase Pairs

	michael	geht	davon	aus	,	dass	er	im	haus	bleibt
michael	■									
assumes		■	■	■	■					
that					■					
he						■				
will										■
stay										■
in										■
the										■
house										■

extract phrase pair consistent with word alignment:
 assumes that / geht davon aus , dass

Consistent

■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■

ok consistent

■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■

violated one alignment point outside inconsistent

■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■

ok unaligned word is fine consistent

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

Phrase Pair Extraction

	michael	geht	davon	aus	,	dass	er	im	haus	bleibt
michael	■									
assumes		■	■	■	■					
that					■					
he						■				
will										■
stay										■
in										■
the										■
house										■

Smallest phrase pairs:
 michael — michael
 assumes — geht davon aus / geht davon aus ,
 that — dass , dass
 he — er
 will stay — bleibt
 in the — im
 house — haus

unaligned words (here: German comma) lead to multiple translations

Larger Phrase Pairs

	michael	geht	davon	aus	,	dass	er	im	haus	bleibt
michael	■									
assumes		■	■	■	■					
that					■					
he						■				
will										■
stay										■
in										■
the										■
house										■

michael assumes — michael geht davon aus / michael geht davon aus ,
 assumes that — geht davon aus , dass ; assumes that he — geht davon aus , dass er
 that he — dass er / , dass er ; in the house — im haus
 michael assumes that — michael geht davon aus , dass
 michael assumes that he — michael geht davon aus , dass er
 michael assumes that he will stay in the house — michael geht davon aus , dass er im haus bleibt
 assumes that he will stay in the house — geht davon aus , dass er im haus bleibt
 that he will stay in the house — dass er im haus bleibt ; dass er im haus bleibt
 he will stay in the house — er im haus bleibt ; will stay in the house — im haus bleibt

Scoring Phrase Translations

- Phrase pair extraction: collect all phrase pairs from the data
- Phrase pair scoring: assign probabilities to phrase translations
- Score by relative frequency:

$$p(f|\bar{e}) = \frac{\text{count}(\bar{e}, f)}{\sum_f \text{count}(\bar{e}, f)}$$

Real Example

- Phrase translations for den Vorschlag learned from the Europarl corpus:

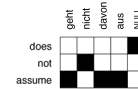
English	$\phi(e f)$	English	$\phi(f e)$
the proposal	0.6227	the suggestions	0.0114
's proposal	0.1068	the proposed	0.0114
a proposal	0.0341	the motion	0.0091
the idea	0.0250	the idea of	0.0091
this proposal	0.0227	the proposal ,	0.0068
proposal	0.0205	its proposal	0.0068
of the proposal	0.0159	it	0.0068
the proposals	0.0159

- lexical variation (proposal vs suggestions)
- morphological variation (proposal vs proposals)
- included function words (the, a, ...)
- noise (it)

Other Scoring Terms

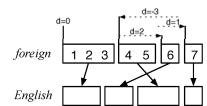
More Feature Functions

- Bidirectional alignment probabilities: $\phi(e|f)$ and $\phi(f|e)$
- Rare phrase pairs have unreliable phrase translation probability estimates
→ lexical weighting with word translation probabilities



$$\text{lex}(e|f, a) = \prod_{i=1}^{\text{length}(e)} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{v(i,j) \in a} w(e_i|f_j)$$

Distance-Based Reordering

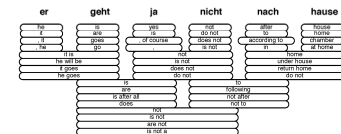


phrase	translates	movement	distance
1	1-3	start at beginning	0
2	6	skip over 4-5	+2
3	4-5	move back over 4-6	-3
4	7	skip over 6	+1

Scoring function: $d(x) = a^{|x|}$ — exponential with distance

Phrase-Based Decoding

Translation Options



- Many translation options to choose from
 - in Europarl phrase table: 2727 matching phrase pairs for this sentence
 - by pruning to the top 20 per phrase, 202 translation options remain

Translation Options

- The machine translation decoder does not know the right answer
 - picking the right translation options
 - arranging them in the right order
- Search problem solved by heuristic beam search

Decoding: Precompute Translation Options

consult phrase translation table for all input phrases

Decoding: Start with Initial Hypothesis

initial hypothesis: no input words covered, no output produced

Decoding: Hypothesis Expansion

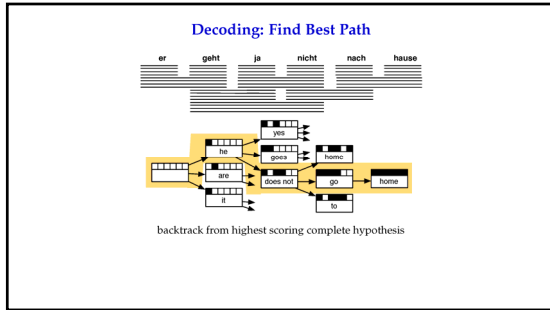
pick any translation option, create new hypothesis

Decoding: Hypothesis Expansion

create hypotheses for all other translation options

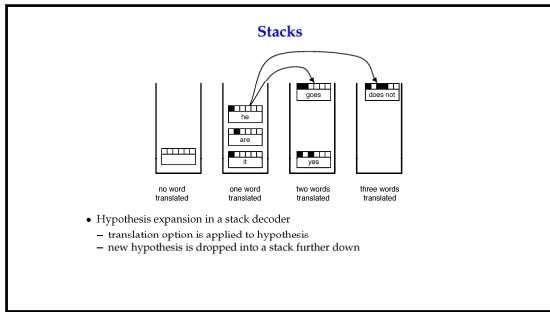
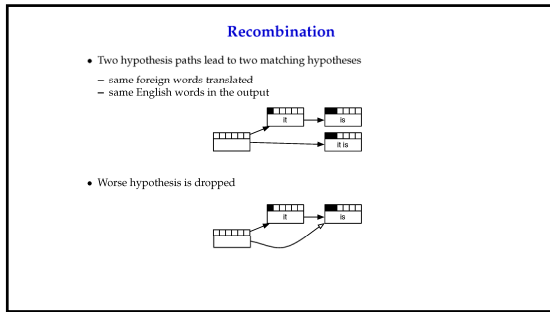
Decoding: Hypothesis Expansion

also create hypotheses from created partial hypothesis



Dynamic Programming

- ### Computational Complexity
- The suggested process creates exponential number of hypothesis
 - Machine translation decoding is NP-complete
 - Reduction of search space:
 - recombination (risk-free)
 - pruning (risky)



Stack Decoding Algorithm

```

1: place empty hypothesis into stack 0
2: for all stacks 0...n - 1 do
3:   for all hypotheses in stack do
4:     for all translation options do
5:       if applicable then
6:         create new hypothesis
7:         place in stack
8:         recombine with existing hypothesis if possible
9:         prune stack if too big
10:      end if
11:    end for
12:  end for
13: end for
    
```


Future Costs

Translating the Easy Part First?

the tourism initiative addresses this for the first time

the

tm: -0.18, tm: -0.4
e: 0, st: -0.85

tourism

tm: -1.10, tm: -2.83
e: 0, st: -4.02

initiative

tm: -1.21, tm: -4.07
e: 0, st: -5.88

addresses

tm: -0.50, tm: -2.81
e: -0.74, st: -4.11

both hypotheses translate 3 words
worse hypothesis has better score

Estimating Future Cost

- Future cost estimate: how expensive is translation of rest of sentence?
- Optimistic: choose cheapest translation options
- Cost for each translation option
 - translation model: cost known
 - language model: output words known, but not context
→ estimate without context
 - reordering model: unknown, ignored for future cost estimation

Cost Estimates from Translation Options

the tourism initiative addresses this for the first time

-1.0

-2.0

-1.0

-2.4

-1.4

-1.0

-1.0

-1.0

-4.0

-2.5

-2.2

-1.3

-2.4

-2.7

-2.8

-2.3

-2.3

cost of cheapest translation options for each input span (log-probabilities)

Cost Estimates for all Spans

- Compute cost estimate for all contiguous spans by combining cheapest options

first word	future cost estimate for n words (from first)								
	1	2	3	4	5	6	7	8	9
the	-1.0	-3.0	-4.5	-6.9	-8.3	-9.3	-9.6	-10.6	-10.6
tourism	-2.0	-3.5	-5.9	-7.3	-8.3	-8.6	-9.6	-9.6	
initiative	-1.5	-3.9	-5.3	-6.3	-6.6	-7.6	-7.6		
addresses	-2.4	-3.8	-4.8	-5.1	-6.1				
this	-1.4	-2.4	-2.7	-3.7	-3.7				
for	-1.0	-1.3	-2.3	-2.3					
the	-1.0	-2.2	-2.3						
first	-1.9	-2.4							
time	-1.6								

- Function words cheaper (the: -1.0) than content words (tourism: -2.0)
- Common phrases cheaper (for the first time: -2.3) than unusual ones (tourism initiative addresses: -5.9)

Combining Score and Future Cost

the tourism initiative

tm: -0.18, tm: -0.4
e: 0, st: -0.85

addresses

tm: -0.50, tm: -2.81
e: -0.74, st: -4.11

the first time

tm: -1.21, tm: -4.07
e: 0, st: -5.88

for the first time

tm: -0.82, tm: -2.98
e: -1.08, st: -4.86

- Hypothesis score and future cost estimate are combined for pruning
 - left hypothesis starts with hard part: the tourism initiative
score: -5.88, future cost: -6.1 → total cost -11.98
 - middle hypothesis starts with easiest part: the first time
score: -4.11, future cost: -9.3 → total cost -13.41
 - right hypothesis picks easy parts: this for ... time
score: -4.86, future cost: -9.1 → total cost -13.96

A* Search

- Uses *admissible* future cost heuristic: never overestimates cost
- Translation agenda: create hypothesis with lowest score + heuristic cost
- Done, when complete hypothesis created

Neural Machine Translation

Dan Klein
UC Berkeley

Slides from Abigail See and John DeNero

1990s-2010s: Statistical Machine Translation

- SMT was a *huge research field*
- The best systems were *extremely complex*
 - Hundreds of important details we haven't mentioned here
 - Systems had many *separately-designed subcomponents*
 - Lots of *feature engineering*
 - Need to design features to capture particular language phenomena
 - Require *compiling and maintaining extra resources*
 - Like tables of equivalent phrases
 - Lots of *human effort to maintain*
 - Repeated effort for each language pair!

Neural Machine Translation

2014

(dramatic reenactment)

2014

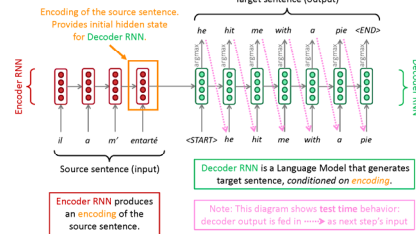
(dramatic reenactment)

What is Neural Machine Translation?

- **Neural Machine Translation (NMT)** is a way to do Machine Translation with a *single neural network*
- The neural network architecture is called **sequence-to-sequence** (aka *seq2seq*) and it involves *two RNNs*.

Neural Machine Translation (NMT)

The sequence-to-sequence model



Sequence-to-sequence is versatile!

- Sequence-to-sequence is useful for *more than just MT*
- Many NLP tasks can be phrased as sequence-to-sequence:
 - **Summarization** (long text → short text)
 - **Dialogue** (previous utterances → next utterance)
 - **Parsing** (input text → output parse as sequence)
 - **Code generation** (natural language → Python code)

Neural Machine Translation (NMT)

- The **sequence-to-sequence** model is an example of a **Conditional Language Model**.
 - **Language Model** because the decoder is predicting the next word of the target sentence y
 - **Conditional** because its predictions are also conditioned on the source sentence x

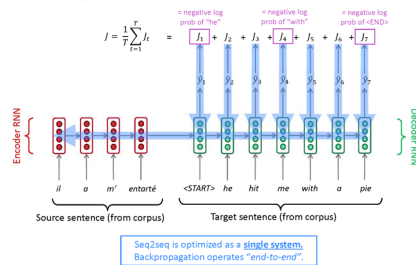
NMT directly calculates $P(y|x)$:

$$P(y|x) = P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots P(y_T|y_1, \dots, y_{T-1}, x)$$

Probability of next target word, given target words so far and source sentence x

- **Question:** How to **train** a NMT system?
- **Answer:** Get a big parallel corpus...

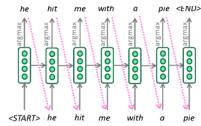
Training a Neural Machine Translation system



NMT Decoding

Greedy decoding

- We saw how to generate (or "decode") the target sentence by taking argmax on each step of the decoder



- This is **greedy decoding** (take most probable word on each step)
- Problems with this method?**

Problems with greedy decoding

- Greedy decoding has no way to undo decisions!
 - Input:** *il a m'entarté* (he hit me with a pie)
 - he _____
 - he hit _____
 - he hit **a** _____ (whoops! no going back now...)
- How to fix this?

Exhaustive search decoding

- Ideally we want to find a (length T) translation y that maximizes

$$P(y|x) = P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots P(y_T|y_1, \dots, y_{T-1}, x)$$

$$= \prod_{t=1}^T P(y_t|y_1, \dots, y_{t-1}, x)$$
- We could try computing **all possible sequences** y
 - This means that on each step t of the decoder, we're tracking V^t possible partial translations, where V is vocab size
 - This $O(V^t)$ complexity is **far too expensive!**

Beam search decoding

- Core idea:** On each step of decoder, keep track of the **k most probable partial translations** (which we call **hypotheses**)
 - k is the **beam size** (in practice around 5 to 10)
- A hypothesis y_1, \dots, y_t has a **score** which is its log probability:

$$\text{score}(y_1, \dots, y_t) = \log P_{LM}(y_1, \dots, y_t|x) = \sum_{i=1}^t \log P_{LM}(y_i|y_1, \dots, y_{i-1}, x)$$
 - Scores are all negative, and higher score is better
 - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is **not guaranteed** to find optimal solution
- But **much more efficient** than exhaustive search!

Beam search decoding: example

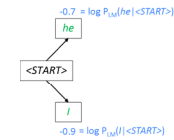
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{LM}(y_i|y_1, \dots, y_{i-1}, x)$

<START>

Calculate prob dist of next word

Beam search decoding: example

Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{LM}(y_i|y_1, \dots, y_{i-1}, x)$



Take top k words and compute scores

Beam search decoding: example
 Beam size = $k = 2$. Blue numbers = $\text{score}(p_1, \dots, p_k) = \sum_{t=1}^k \log P_{LM}(p_t | p_1, \dots, p_{t-1}, x)$

For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example
 Beam size = $k = 2$. Blue numbers = $\text{score}(p_1, \dots, p_k) = \sum_{t=1}^k \log P_{LM}(p_t | p_1, \dots, p_{t-1}, x)$

Of these k^2 hypotheses, just keep k with highest scores

Beam search decoding: example
 Beam size = $k = 2$. Blue numbers = $\text{score}(p_1, \dots, p_k) = \sum_{t=1}^k \log P_{LM}(p_t | p_1, \dots, p_{t-1}, x)$

For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example
 Beam size = $k = 2$. Blue numbers = $\text{score}(p_1, \dots, p_k) = \sum_{t=1}^k \log P_{LM}(p_t | p_1, \dots, p_{t-1}, x)$

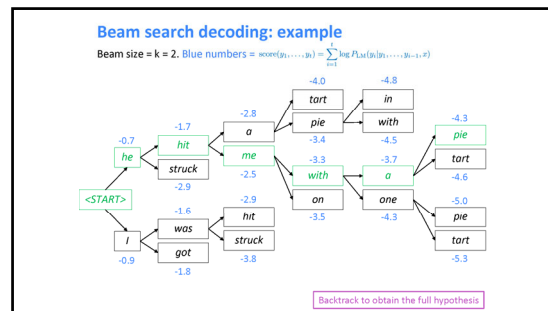
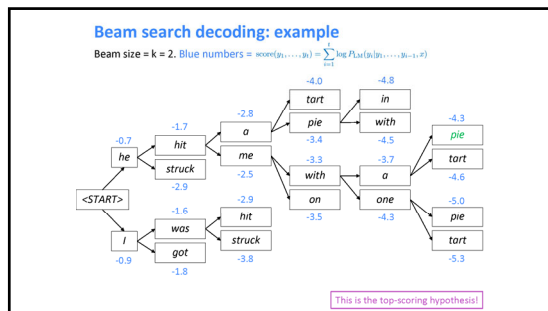
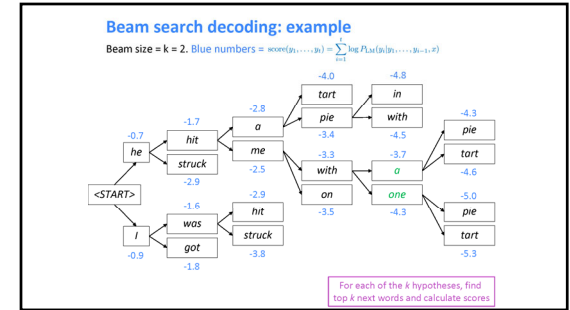
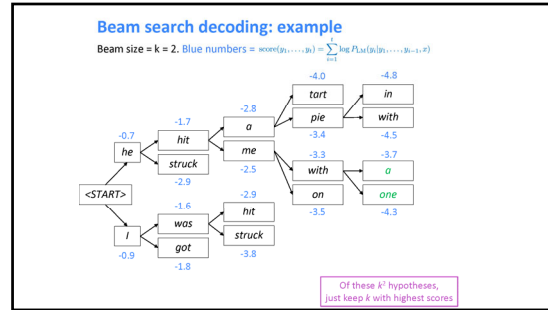
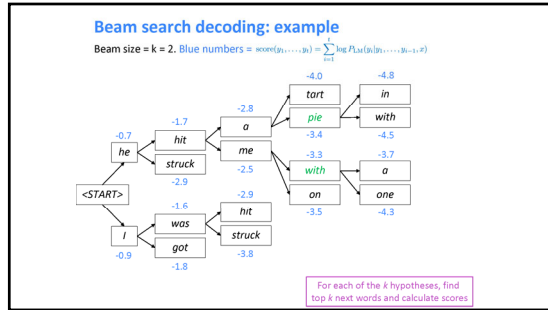
Of these k^2 hypotheses, just keep k with highest scores

Beam search decoding: example
 Beam size = $k = 2$. Blue numbers = $\text{score}(p_1, \dots, p_k) = \sum_{t=1}^k \log P_{LM}(p_t | p_1, \dots, p_{t-1}, x)$

For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example
 Beam size = $k = 2$. Blue numbers = $\text{score}(p_1, \dots, p_k) = \sum_{t=1}^k \log P_{LM}(p_t | p_1, \dots, p_{t-1}, x)$

Of these k^2 hypotheses, just keep k with highest scores



- ### Beam search decoding: stopping criterion
- In **greedy decoding**, usually we decode until the model produces a **<END> token**
 - For example: <START> he hit me with a pie <END>
 - In **beam search decoding**, different hypotheses may produce **<END> tokens on different timesteps**
 - When a hypothesis produces **<END>**, that hypothesis is **complete**.
 - Place it **aside** and continue exploring other hypotheses via beam search.
 - Usually we continue beam search until:
 - We reach timestep T (where T is some pre-defined cutoff), or
 - We have at least n completed hypotheses (where n is pre-defined cutoff)

Beam search decoding: finishing up

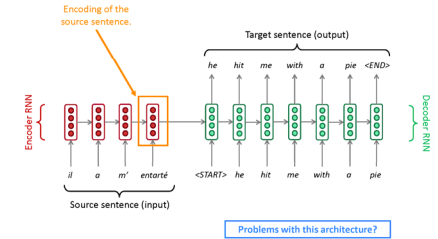
- We have our list of completed hypotheses.
- How to select top one with highest score?
- Each hypothesis y_1, \dots, y_t on our list has a score

$$\text{score}(y_1, \dots, y_t) = \log P_{LM}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$$
- **Problem with this:** longer hypotheses have lower scores
- **Fix:** Normalize by length. Use this to select top one instead:

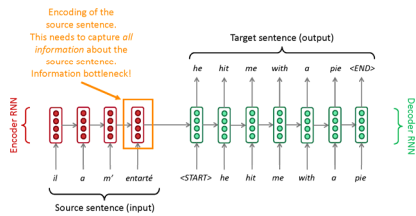
$$\frac{1}{t} \sum_{i=1}^t \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$$

Attention


Sequence-to-sequence: the bottleneck problem



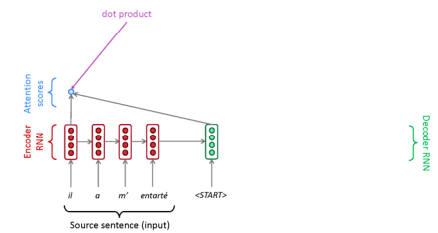
Sequence-to-sequence: the bottleneck problem

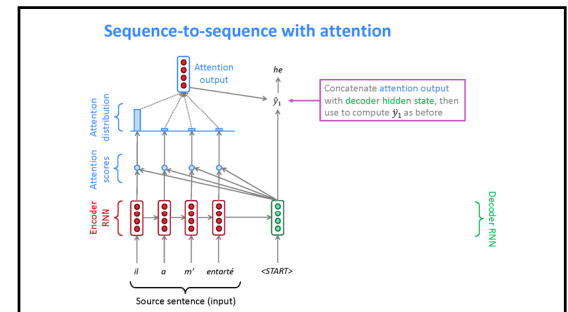
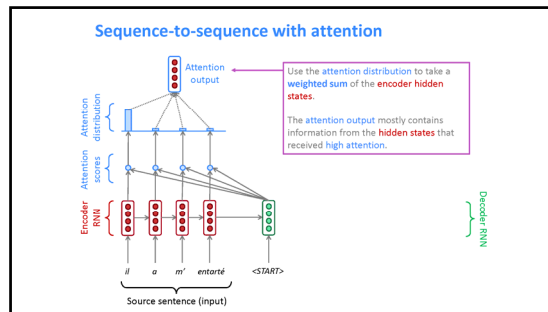
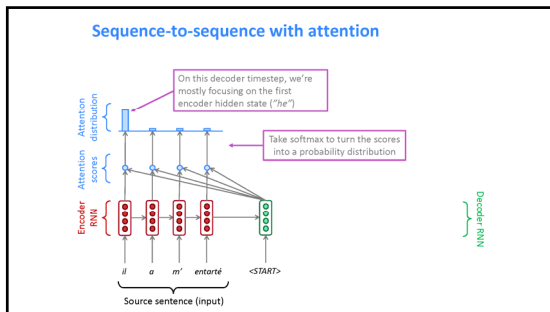
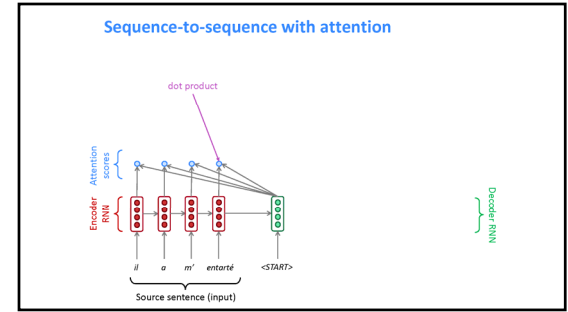
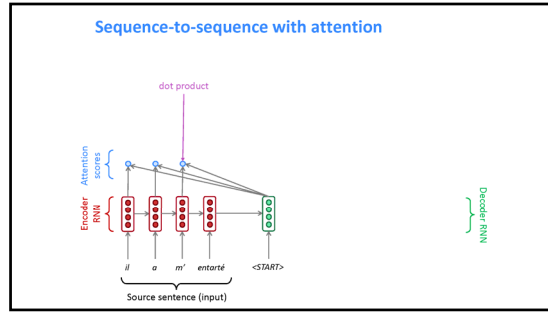
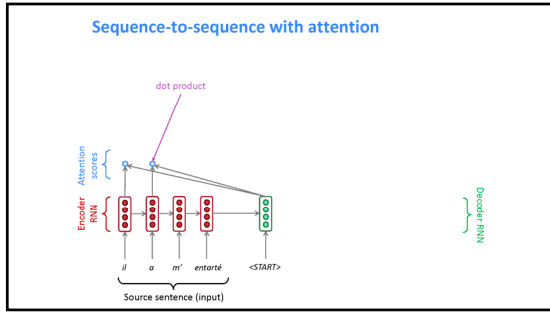


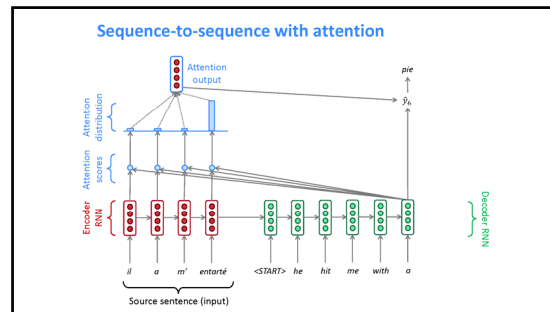
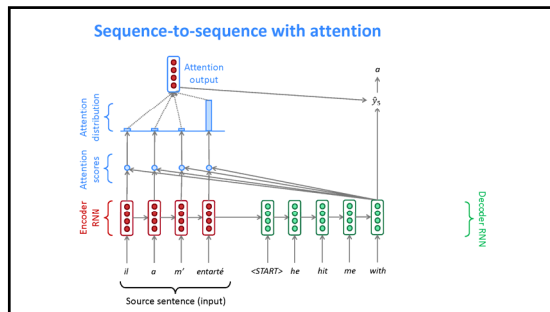
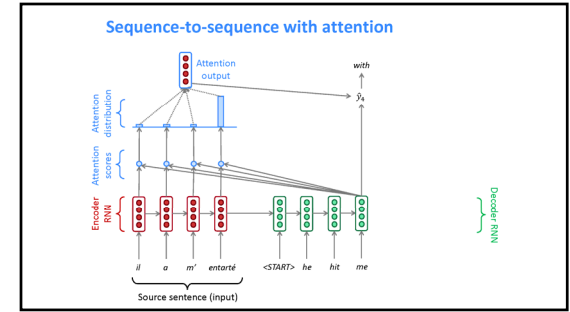
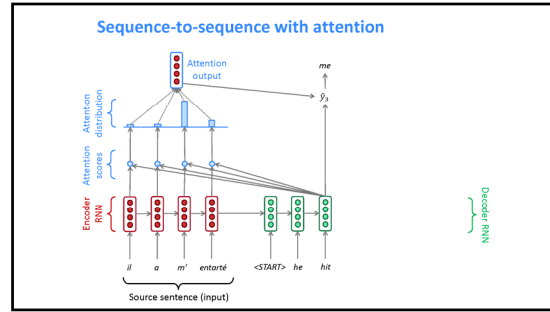
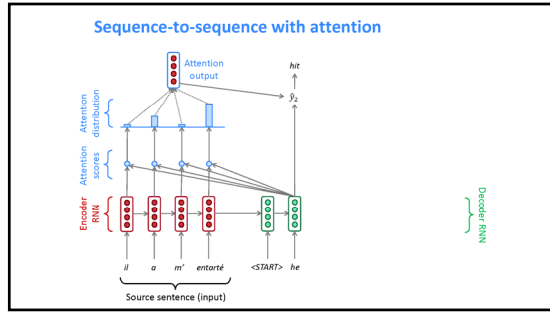
Attention

- **Attention** provides a solution to the bottleneck problem.
 - **Core idea:** on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence
- 
- First we will show via diagram (no equations), then we will show with equations

Sequence-to-sequence with attention







Attention: in equations

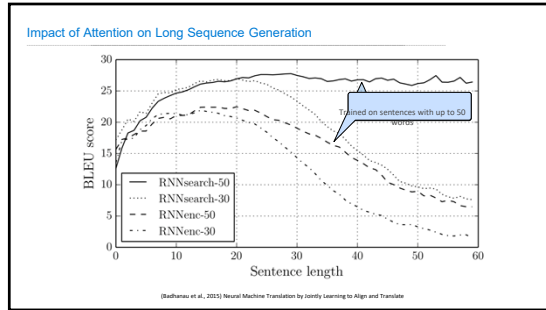
- We have encoder hidden states $h_1, \dots, h_N \in \mathbb{R}^h$
- On timestep t , we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores e^t for this step:

$$e^t = [s_t^T h_1, \dots, s_t^T h_N] \in \mathbb{R}^N$$
- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

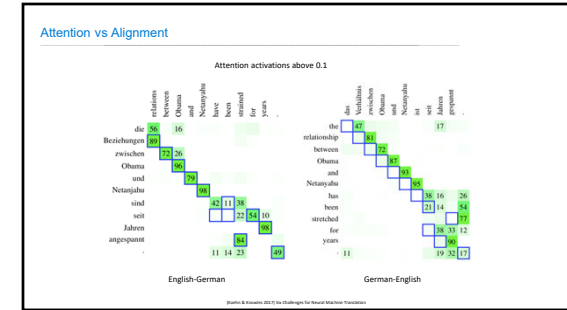
$$\alpha^t = \text{softmax}(e^t) \in \mathbb{R}^N$$
- We use α^t to take a weighted sum of the encoder hidden states to get the attention output a_t

$$a_t = \sum_{i=1}^N \alpha_i^t h_i \in \mathbb{R}^h$$
- Finally we concatenate the attention output a_t with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

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



- ### Attention is great
- Attention significantly **improves NMT performance**
 - It's very useful to allow decoder to focus on certain parts of the source
 - Attention **solves the bottleneck problem**
 - Attention allows decoder to look directly at source; bypass bottleneck
 - Attention **helps with vanishing gradient problem**
 - Provides shortcut to faraway states
 - Attention provides **some interpretability**
 - By inspecting attention distribution, we can see what the decoder was focusing on
 - We get (soft) **alignment for free!**
 - This is cool because we never explicitly trained an alignment system
 - The network just learned alignment by itself
-



- ### Attention is a general Deep Learning technique
- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
 - However: You can use attention in **many architectures** (not just seq2seq) and **many tasks** (not just MT)
- More general definition of attention:**

 - 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).

- ### Attention is a general Deep Learning technique
- More general definition of attention:**

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.
- Intuition:**
- 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).

- ### There are several attention variants
- We have some **values** $h_1, \dots, h_N \in \mathbb{R}^{d_1}$ and a **query** $s \in \mathbb{R}^{d_2}$
 - Attention always involves:
 - Computing the **attention scores** $e \in \mathbb{R}^N$

There are multiple ways to do this
 - Taking softmax to get **attention distribution** α :

$$\alpha = \text{softmax}(e) \in \mathbb{R}^N$$
 - Using attention distribution to take weighted sum of values:

$$\alpha = \sum_{i=1}^N \alpha_i h_i \in \mathbb{R}^{d_1}$$
 thus obtaining the **attention output** σ (sometimes called the **context vector**)

Attention variants

You'll think about the relative advantages/disadvantages of these in Assignment 4!

There are several ways you can compute $e \in \mathbb{R}^N$ from $h_1, \dots, h_N \in \mathbb{R}^{d_1}$ and $s \in \mathbb{R}^{d_2}$:

- Basic dot-product attention:** $e_i = a^T h_i \in \mathbb{R}$
 - Note: this assumes $d_1 = d_2$
 - This is the version we saw earlier
- 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
- Additive 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.
 - d_3 (the attention dimensionality) is a hyperparameter

More information: "Deep Learning by NP Best Practices", Rubin, 2015. <https://github.com/npbestpractices/Deep-Learning-by-NP-Best-Practices>. "Neural Computational Models: The Attention Architecture", Bitts et al., 2017. <https://arxiv.org/abs/1703.07322>

Transformers

Transformer

In lieu of an RNN, use ONLY attention!

High throughput & expressivity: compute queries, keys and values as (different) linear transformations of the input.

Attention weights are queries
 * keys, outputs are sums of weighted values.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Input: Thinking, Machines

Embedding: x_1 (Thinking), x_2 (Machines)

Queries: q_1 (Thinking), q_2 (Machines)

Keys: k_1 (Thinking), k_2 (Machines)

Values: v_1 (Thinking), v_2 (Machines)

Score: $q_1 \cdot k_1 = 112$, $q_1 \cdot k_2 = 90$

Divide by 8 ($\sqrt{d_k}$): 14, 12

Softmax: 0.88, 0.12

Softmax X Value: v_1 (Thinking), v_2 (Machines)

Sum: z_1 (Thinking), z_2 (Machines)

© 2017 OpenAI. All rights reserved.

Transformer Architecture

- Layer normalization ("Add & Norm" cells) helps with RNN-attention architectures as well.
- Positional encodings can be learned or based on a formula that makes it easy to represent distance.

Model	EN-DE
ByteNet [18]	33.75
Deep-Attn + PosUnk [39]	24.6
GNMT + RL [38]	25.16
ConvS2S [9]	26.03
MoE [32]	26.30
Deep-Attn + PosUnk Ensemble [39]	26.36
GNMT + RL Ensemble [38]	27.3
ConvS2S Ensemble [9]	28.4
Transformer (base model)	
Transformer (big)	

Inputs: Input Embedding, Positional Encoding

Outputs: Output Embedding, Positional Encoding (shifted right)

© 2017 OpenAI. All rights reserved.

Some Transformer Concerns

- Problem:** Bag-of-words representation of the input.
Remedy: Position embeddings are added to the word embeddings.
- Problem:** During generation, can't attend to future words.
Remedy: Masked training that zeroes attention to future words.
- Problem:** Deep networks needed to integrate lots of context.
Remedies: Residual connections and multi-head attention.
- Problem:** Optimization is hard.
Remedies: Large mini-batch sizes and layer normalization.

Training Data

Bitexts

Where do bitexts come from?

- Careful, low level / literal translations: organizational translation processes (eg parliamentary proceedings), multilingual newsfeeds, etc
- Discovered translations (ad hoc translations on webpages, etc)
- Loose translations (multilingual Wikipedia, etc)
- Synthetic data (distillation, backtranslation, etc)



Back Translations

Synthesize an en-de parallel corpus by using a de-en system to translate monolingual de sentences.

- Better generating systems don't seem to matter much.
- Can help even if the de sentences are already in an existing en-de parallel corpus!

system	EN→DE		DE→EN	
	dev	test	dev	test
baseline	22.4	26.8	26.4	28.5
+synthetic	25.8	31.6	29.9	36.2
+ensemble	27.5	33.1	31.5	37.5
+r2l reranking	28.1	34.2	32.1	38.6

Table 2: English↔German translation results (BLEU) on dev (newstest2015) and test (newstest2016). Submitted system in bold.

(Sennrich et al., 2015) Improving Neural Machine Translation Models with Monolingual Data
(Sennrich et al., 2016) Improving Neural Machine Translation Systems with Monolingual Data

Subwords

The sequence of symbols that are embedded should be common enough that an embedding can be estimated robustly for each, and all symbols have been observed during training.

Solution 1: Symbols are words with rare words replaced by UNK.

- Replacing UNK in the output is a new problem (like alignment).
- UNK in the input loses all information that might have been relevant from the rare input word (e.g., tense, length, POS).

Solution 2: Symbols are subwords.

- Byte-Pair Encoding is the most common approach.
- Other techniques that find common subwords aren't reliably better (but are somewhat more complicated).
- Training on many sampled subword decompositions improves out-of-domain translations.

(Sennrich et al., 2016) Neural Machine Translation of Rare Words with Subword Units
(BPE, SPM) Subword Segmentation: Improving Neural Network Translation Models with Multiple Subword Outputs

```

vocab = {'low </w>': 5, 'lower </w>': 2,
        'newest </w>': 6, 'widest </w>': 3}

def get_stats(vocab):
    pairs = collections.defaultdict(int)
    for word, freq in vocab.items():
        symbols = word.split()
        for i in range(len(symbols)-1):
            pairs[symbols[i],symbols[i+1]] += freq
    return pairs

def merge_vocab(pair, v_in):
    v_out = {}
    bigram = re.escape(' '.join(pair))
    p = re.compile(r'(?!\S)' + bigram + r'(?!\S)')
    for word in v_in:
        w_out = p.sub(' '.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out
    
```

BPE Example

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
word-level (with back-off)	Forschungsinstitute
character bigrams	Fo rs ch un gs in st it ut io ne n
BPE	Gesundheits forsch lungsinstitute

Example From Rico Sennrich

Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance**
 - More fluent
 - Better use of context
 - Better use of phrase similarities
- A single neural network to be optimized end-to-end**
 - No subcomponents to be individually optimized
- Requires much less human engineering effort**
 - No feature engineering
 - Same method for all language pairs

Disadvantages of NMT?

Compared to SMT:

- NMT is **less interpretable**
 - Hard to debug
- NMT is **difficult to control**
 - For example, can't easily specify rules or guidelines for translation
 - Safety concerns!

NMT: the biggest success story of NLP Deep Learning

Neural Machine Translation went from a **fringe research activity** in 2014 to the **leading standard method** in 2016

- **2014:** First seq2seq paper published
- **2016:** Google Translate switches from SMT to NMT
- **This is amazing!**
 - **SMT** systems, built by **hundreds** of engineers over many **years**, outperformed by NMT systems trained by a **handful** of engineers in a few **months**


So is Machine Translation solved?

- **Nope!**
- Many difficulties remain:
 - **Out-of-vocabulary** words
 - **Domain mismatch** between train and test data
 - Maintaining **context** over longer text
 - **Low-resource** language pairs

Further reading: "Has AI surpassed humans at translation? Not even close!" https://www.skynewstoday.com/editorials/state_of_nmt

So is Machine Translation solved?

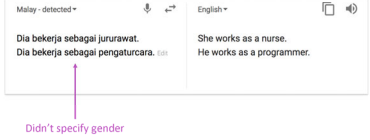
- **Nope!**
- Using **common sense** is still hard



Source: <https://www.vox.com/en-us/article/2016/12/14/13529240/google-translate-why-is-people-translate-anything-out-what-are-these-quirks>

So is Machine Translation solved?

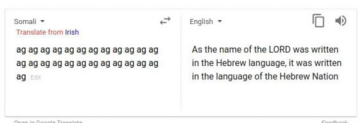
- **Nope!**
- NMT picks up **biases** in training data



Source: <https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-c617c8683c>

So is Machine Translation solved?

- **Nope!**
- **Uninterpretable systems do strange things**



Picture source: <https://www.vox.com/en-us/article/2016/12/14/13529240/google-translate-why-is-people-translate-anything-out-what-are-these-quirks>

Summary

- We learned some history of Machine Translation (MT)
- Since 2014, *Neural MT* rapidly replaced intricate Statistical MT
- *Sequence-to-sequence* is the architecture for NMT (uses 2 RNNs)
- *Attention* is a way to *focus on particular parts* of the input
 - Improves sequence-to-sequence a lot!

