BIOMEDICAL
COMPUTER
VISION

Natural Language Processing

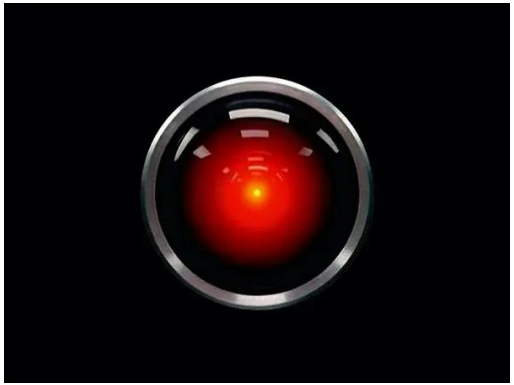
Valentina Massô, Sergio Leal, Javier Pérez
Tutor: Edgar A. Margffoy-Tuay

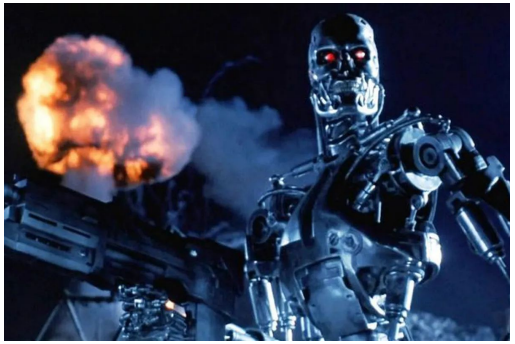
Let’s talk about linguistics

- Language is important for us to communicate and interact with each other
  - Language models and influences the way on which we think
    - It is not the same thinking on Turkish logic (without gender) and Spanish (with gender)
  - Societies, civilizations and empires base their power and domain on promoting a single language
- In principle, we might think that language has no structure at all, as we may talk as we want.
  - However, we need to structure our phrases, such that everybody understands everyone else
  - In fact, linguists think that at some point, humans spoke some sort of “proto” language from which some languages were derived
    - Thus, it may be valuable to understand the structure of language, in order to examine societies and human history itself.

But, what about machines?

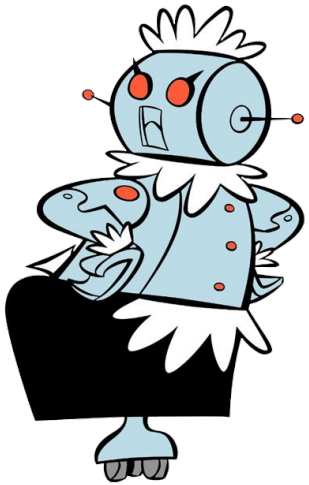
- In popular culture, future is all about talking machines
  - Either machines that would want to dominate us

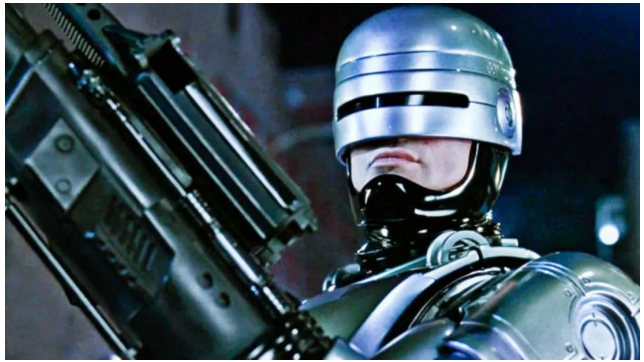




But, what about machines?

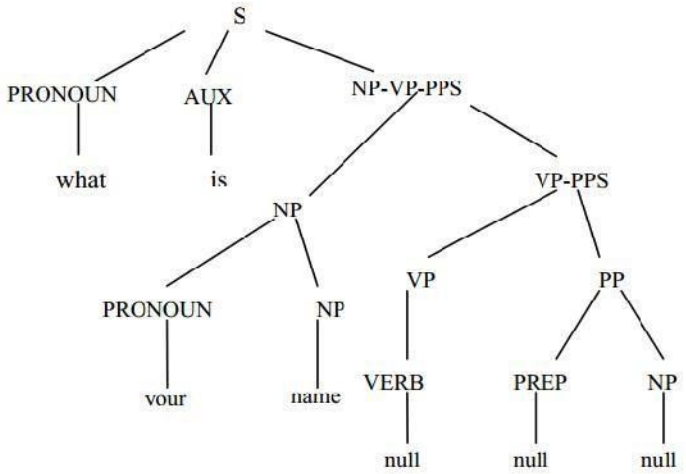
- In popular culture, future is all about talking machines
  - Or machines that would simplify our lives or help us with cumbersome tasks

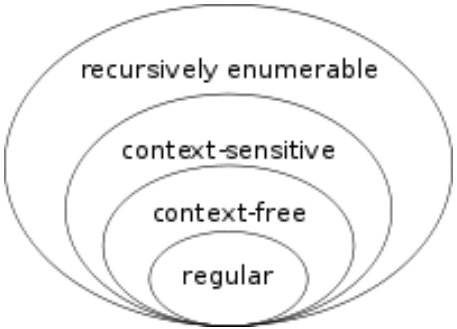




Linguistics and Machines

- Of course, we want machines to talk in order to interact with us, otherwise they would be just plain “dirty-job” machines
  - Sorry WALL-E
- We know something about language structure and analysis
- And we think we know something about machines
  - Why not bridge both areas?
  - Actually, language theory is the basis of theoretical computer science
    - Chomsky hierarchy



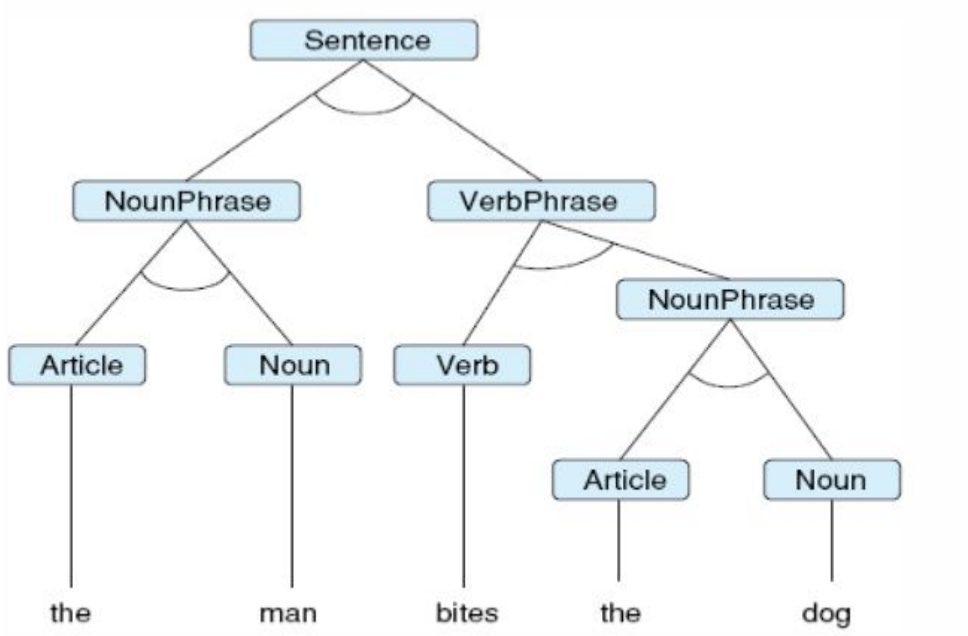


A problem of representation

- Since the beginning of computers, language has been one of the main catalysts to research on AI and CS.
  - We want to build a computer that does not need any manual programming and it executes the instructions dictated by a person.
- Machine translation has been always the problem to look at.
  - We know from linguistics that a large set of languages are related by a common proto language that should have existed and had the basic structure that characterizes the language family
    - Therefore, in theory, we should be able to switch representations among parse trees
  - We also know that programs can be modelled as grammars that can be understood by a Turing Machine
  - Why not try to make programs that understand language?

Deterministic language modelling

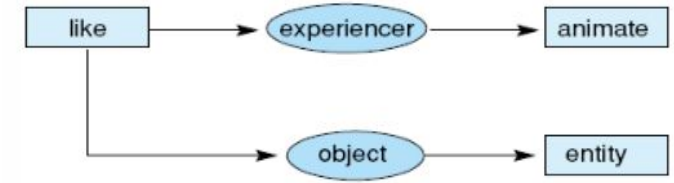
- Lets model language as a program with rules and constraints!

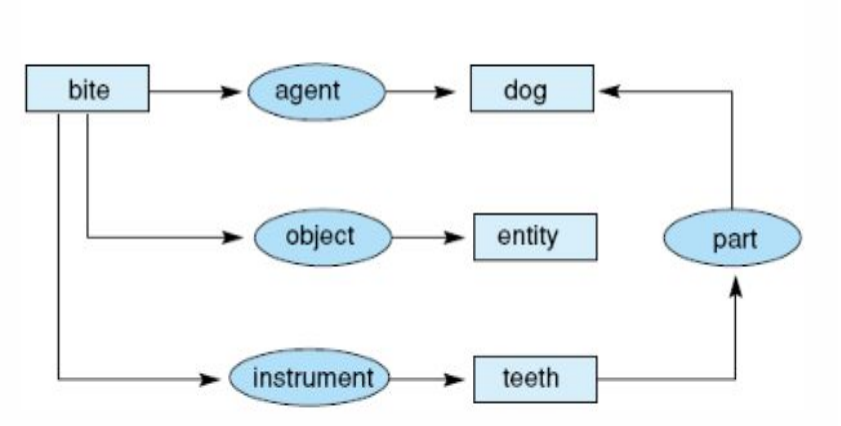


- A parse tree describes the parts of a sentence.
- Nodes on a tree represent the sub part of the phrase
  - They are also given as tokens. E.g., S, NP, VP
- Each language has their own language rules
  - Given by linguistic studies and consensus

Deterministic language modelling

- Let’s start by introducing verb-specific rules

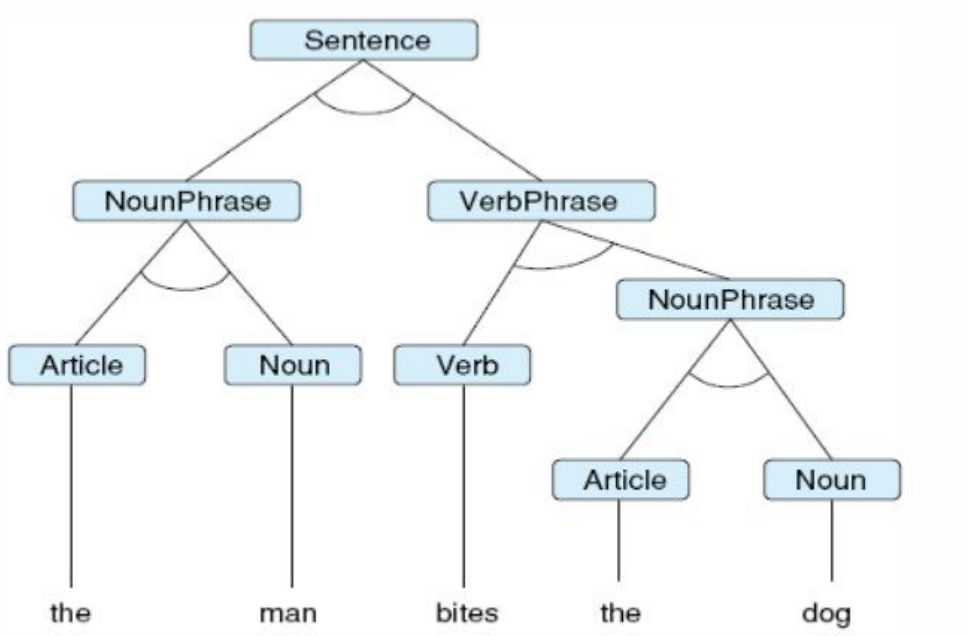




We could go doing this with all verbs on all languages...

Deterministic language modelling

- Now, back to our tree and write a grammar for it:

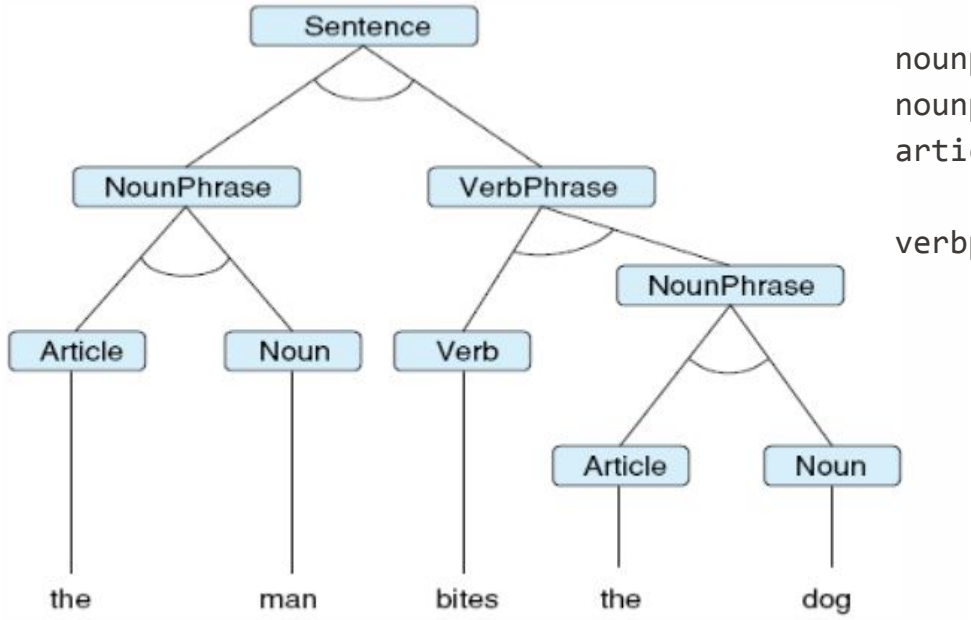



```
Sentence → NounPhrase VerbPhrase
NounPhrase → Noun | Article Noun
VerbPhrase → Verb | Verb NounPhrase
```

Deterministic language modelling

Beware: Prolog ahead!

- Now, some logic programming:




```
utterance(X) :- sentence(X, []).
sentence(Start, End) :- nounphrase(Start, Rest),
                        verbphrase(Rest, End).
nounphrase([Noun | End], End) :- noun(Noun).
nounphrase([Article, Noun | End], End) :-
article(Article),
     noun(Noun).
verbphrase([Verb | End], End) :-verb(Verb).
```

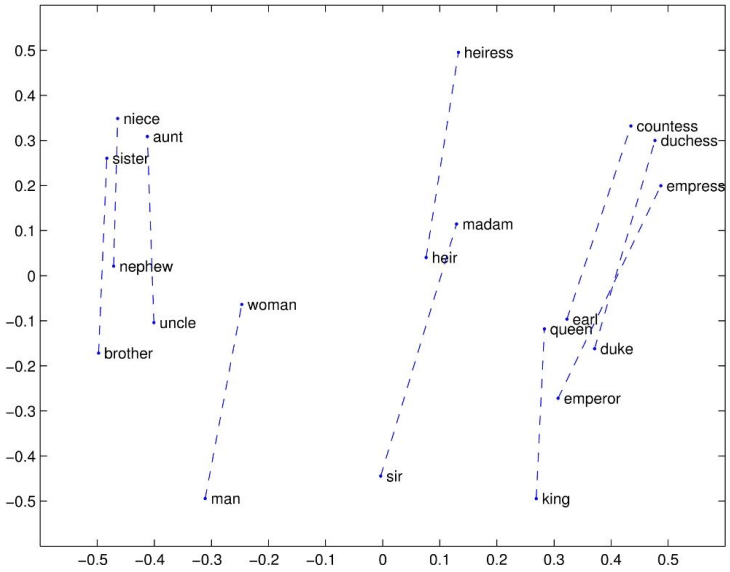
```
Article(a)
Article(the)
Noun(man)
Noun(dog)
Verb(likes)
Verb(bites)
```

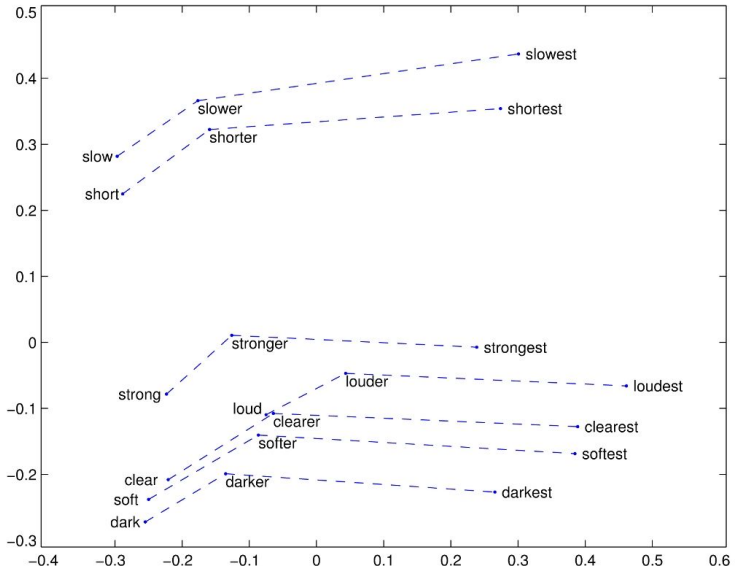
Deterministic language modeling

- We found a simple way of representing a subset of English phrases, but language is more expressive and has more edge cases and exceptions than rules.
  - What about paraphrases, synonyms, context-switching words, and other tenses?
  - This is worse on other languages that have different structures, e.g., German, Tukish, Latin?
  - What about languages with multiple words and ways to express phrases e.g., Almost all romance languages?
  - What happens with variations and dialects? E.g., Arabic
  - What if our alphabet is ideographic? E.g., CJK languages
- We cannot capture all language representations using a simple set of fixed rules
  - This is where the deterministic language modelling failed
  - Welcome to the AI Winter!

Let’s rethink language representation

- Instead of modelling words and phrases using rules, why not try to map them as vectors lying in a higher-order semantic space? E.g., GloVe and Word2vec





Conditional language modelling

- Now that we have numerical entities, let's do some operations over them.
  - Let's describe a phrase as a sequence of word embeddings [e_1, e_2, e_3, ... e_N]
    - We have as many embeddings as words there are on our vocabulary
  - Suppose we want to predict the next word, given the previous ones:

$$\max_{v} \; P\left(e_{n+1}^{v} | e_0, e_1, \cdots, e_n\right)$$

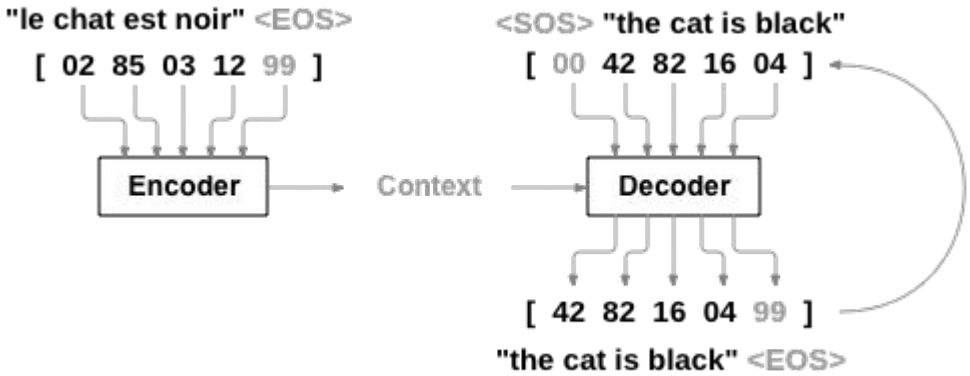
- e_0, e_1, …, e_n is known as a n-gram
- We can model this problem as a many-to-one classification problem
  - Given a n-gram as input, predict the most probable vord on the vocabulary as the next word
- Traditionally this was done using Bayesian Networks and other PGMs
  - But then RNNs attacked

Seq2Seq

What is Seq2Seq?

Sequence-to-sequence (seq2seq) models in NLP are used to **convert sequences of Type A to sequences of Type B** [1]

For example, translation of English sentences to German sentences is a sequence-to-sequence task



NLP tasks related to Seq2Seq

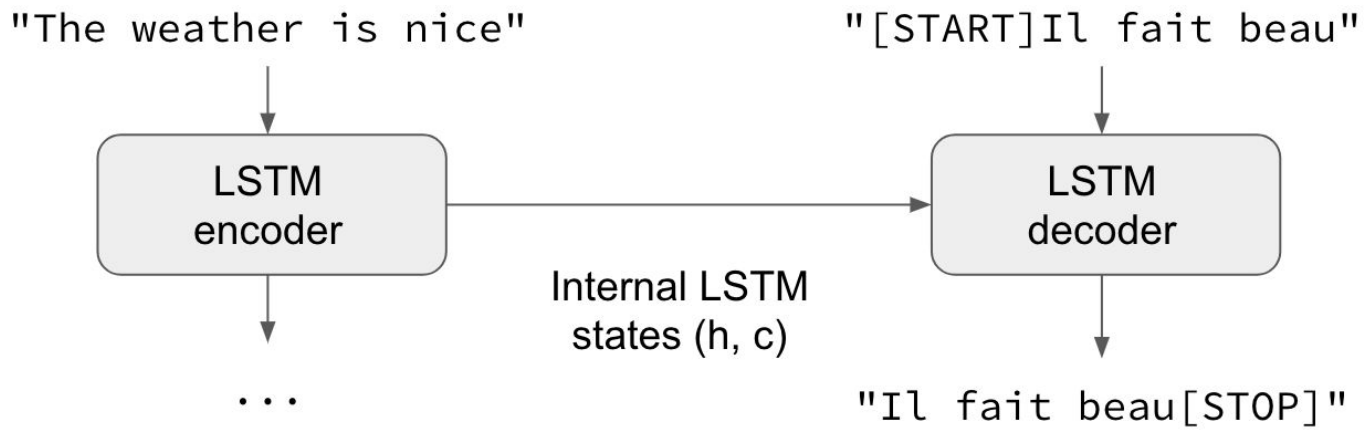
These sequence-to-sequence models are pretty versatile and they are used in a variety of NLP tasks, such as:

- Machine Translation
- Text Summarization
- Question-Answering System, and many more

Machine Translation

What is MT?

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



1990s-2010s: Statistical Machine Translation

We want to find best English sentence y, given French sentence x

Use Bayes Rule to break this down into two components to be learnt separately:

$$= \text{argmax}_y P(x|y) P(y)$$

Translation Model

Models how words and phrases should be translated (*fidelity*). Learnt from parallel data.

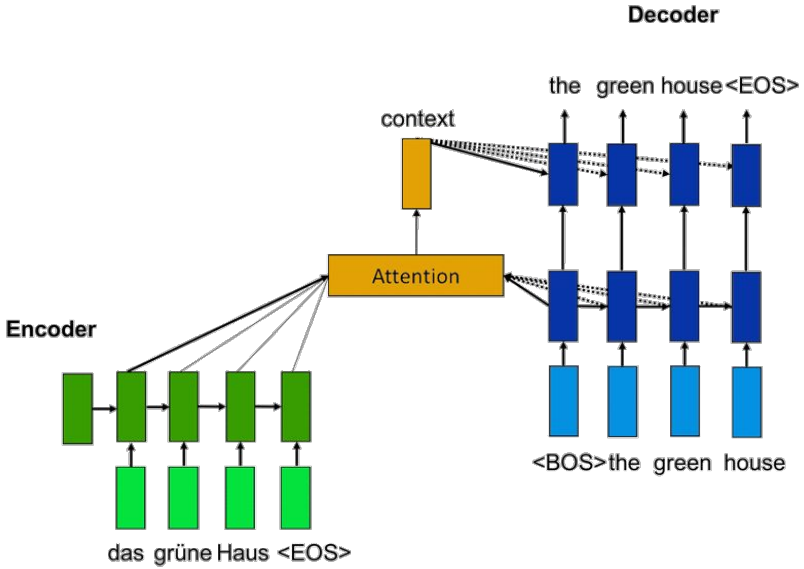
Language Model

Models how to write good English (*fluency*). Learnt from monolingual data.

2014: Neural Machine Translation

Neural Machine Translation (NMT) is a way to do Machine Translation with a single neural network

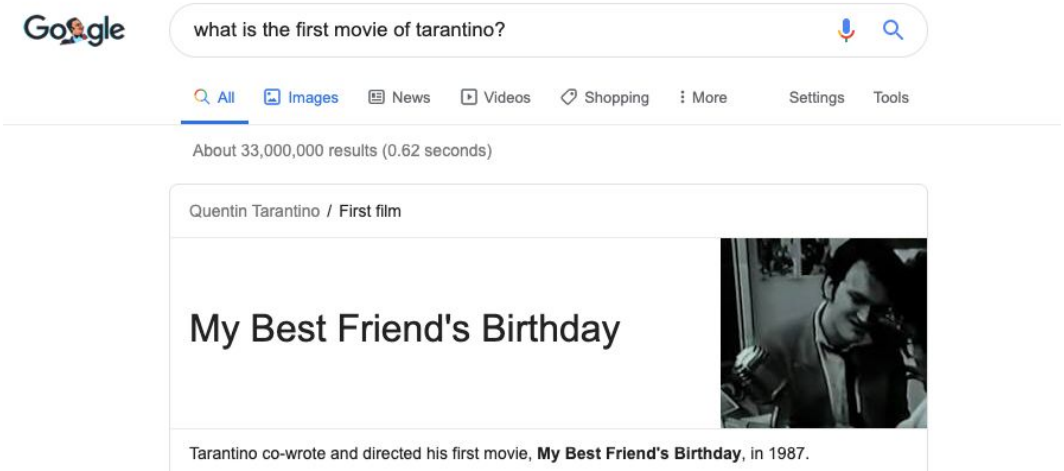
• The neural network architecture is called sequence-to-sequence (seq2seq) and it involves two RNNs.



Question - Answering System

Motivation

- **Finding documents** that (might) contain an answer
  - Which can be handled by traditional information **retrieval/web search**
- **Finding an answer** in a paragraph or a document
  - This problem is often termed **Reading Comprehension**



Passage (*P*) + Question (*Q*) ⟶ Answer (*A*)

P

Alyssa got to the beach after a long trip. She's from Charlotte. She traveled from Atlanta. She's now in Miami. She went to Miami to visit some friends. But she wanted some time to herself at the beach, so she went there first. After going swimming and laying out, she went to her friend Ellen's house. Ellen greeted Alyssa and they both had some lemonade to drink. Alyssa called her friends Kristin and Rachel to meet at Ellen's house…….

Q Why did Alyssa go to Miami?

A To visit some friends

Summarization

Some reasons

1. Summaries reduce reading time.
2. When researching documents, summaries make the selection process easier.
3. Automatic summarization improves the effectiveness of indexing.
4. Automatic summarization algorithms are less biased than human summarizers.
5. ...

Text Summarization

- The process of creating a short and coherent version of a longer document.
- **Task:** given input text *x*, write a summary *y* which is shorter and contains the main information of *x*.
- Summarization can be single-document or multi-document.

Two approaches

Extractive Summarization

- Select parts (typically sentences) of the original text to form a summary.

Abstractive Summarization

- Generate new text using natural language generation techniques.

Extractive Summarization

We identify the important sentences or phrases from the original text and extract only those from the text.

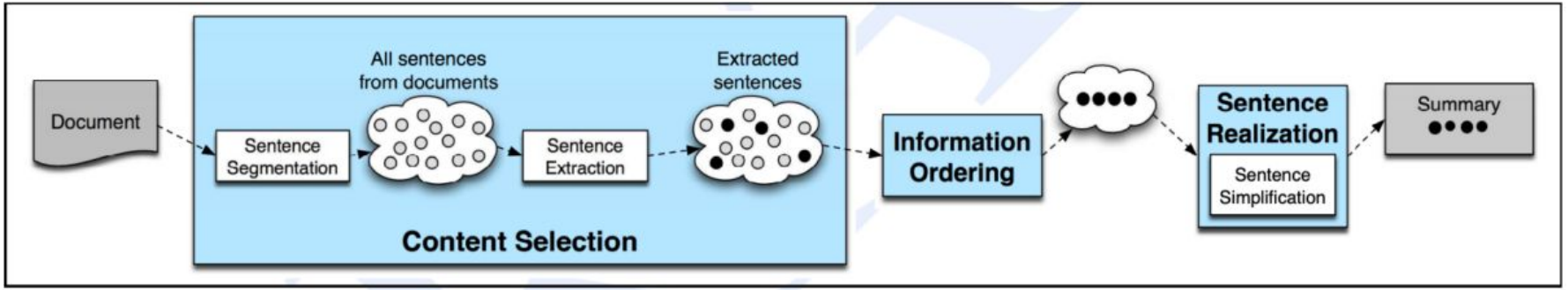


Figure 23.14 The basic architecture of a generic single document summarizer.

Extractive Summarization

Content selection: choose some sentences to include

- Sentence scoring functions
- Graph-based algorithms

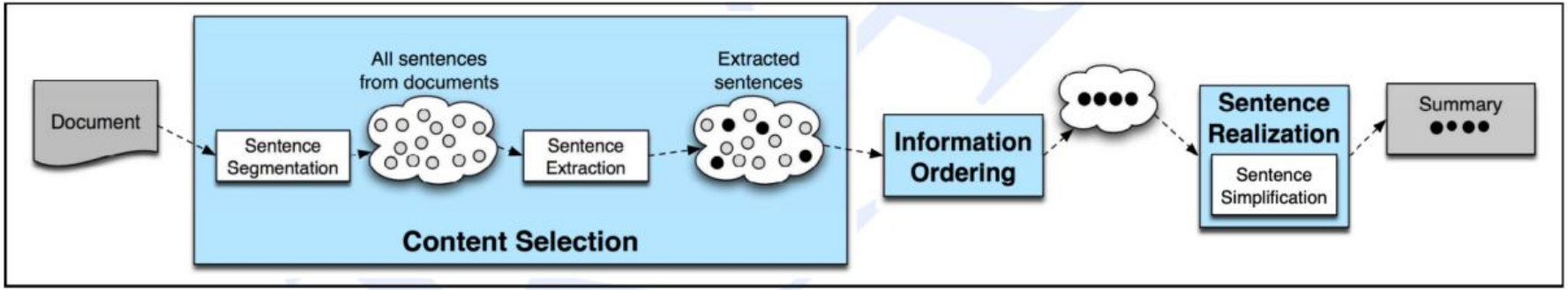


Figure 23.14 The basic architecture of a generic single document summarizer.

Extractive Summarization

Information ordering: choose an ordering of those sentences

Sentence realization: edit the sequence of sentences

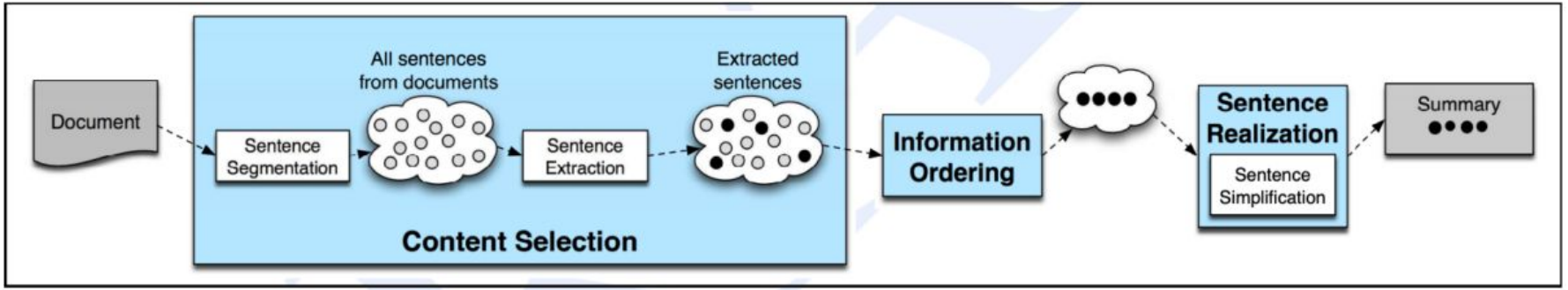


Figure 23.14 The basic architecture of a generic single document summarizer.

Abstractive Summarization

The core of abstractive summarization techniques is to identify the main ideas in the documents and encode them into feature representations.

These encoded features are then passed to natural language generation (NLG) systems for summary generation.

Deep learning techniques.

A Survey on Neural Network-Based Summarization Methods, Dong, 2018 https://arxiv.org/pdf/1804.04589.pdf

Natural Language Generation

NLG

Natural Language Generation refers to any setting in which we **generate new text**.

NLG is a subcomponent of:

- Machine Translation
- (Abstractive) Summarization
- Dialogue (chit-chat and task-based)
- Creative writing: storytelling, poetry-generation
- Question Answering (i.e. answer is generated, not extracted from text or knowledge base)
- Image captioning

NLG

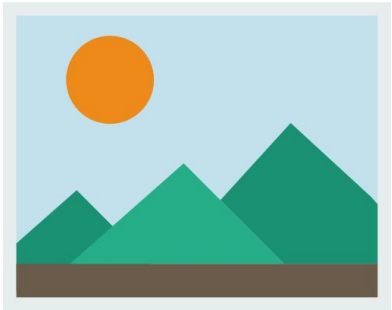
- Language Modeling: the task of **predicting the next word**, given the words so far
- A system that produces this **probability distribution is called a Language Model**
- If that system is an RNN, it’s called a RNN-LM

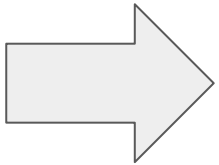
Language + Vision

Image2Text

Given an image generate:

- Story-like paragraphs
- Tags





"It was a sunny day of september and Pete…"

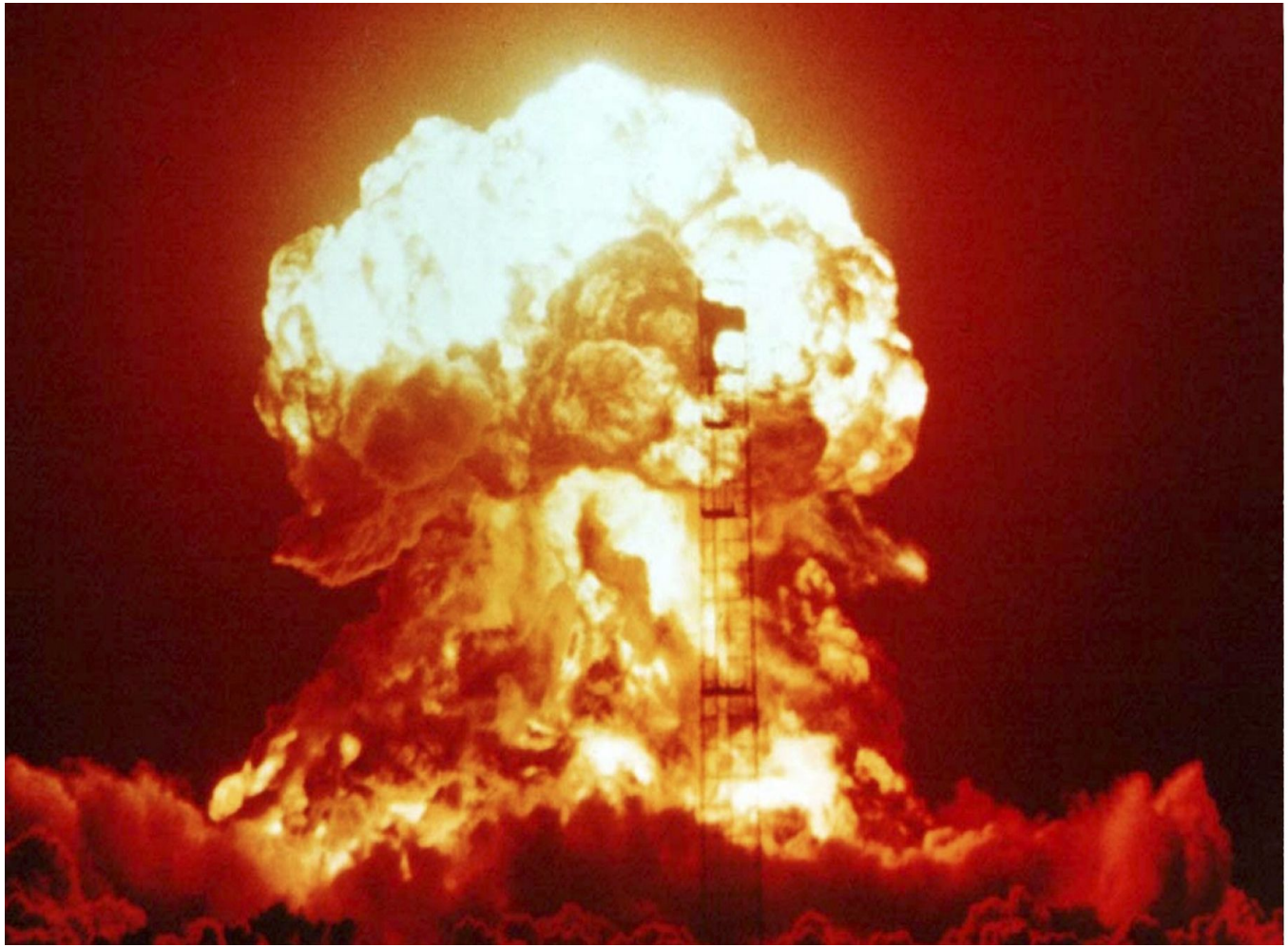
Generated story about image

Model: Romantic Novels

“I had a man in control , and Kate turned her head to look at him. There was no denying it , no matter what she said. In the past few days, I could n't help but notice how beautiful the band was.

The man who stood at the end of the hall told me he was in love with her, and that 's the only way I would ever be able to get my emotions out of my head. Her lips curled into a tight little O, as if waiting for an audience. He seemed to remember something more than a year ago , the man who and I was breaking down .”

Generating Stories about Images, https://medium.com/@samim/generating-stories-about-images-d163ba41e



Generated story about image

Model: Taylor Swift Lyrics

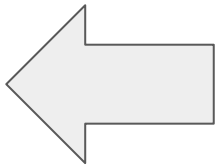
“You have to be the only light bulb in the night sky, I thought, Oh, God, it 's so dark out of me that I missed you , I promise.”

Generating Stories about Images, https://medium.com/@samim/generating-stories-about-images-d163ba41e

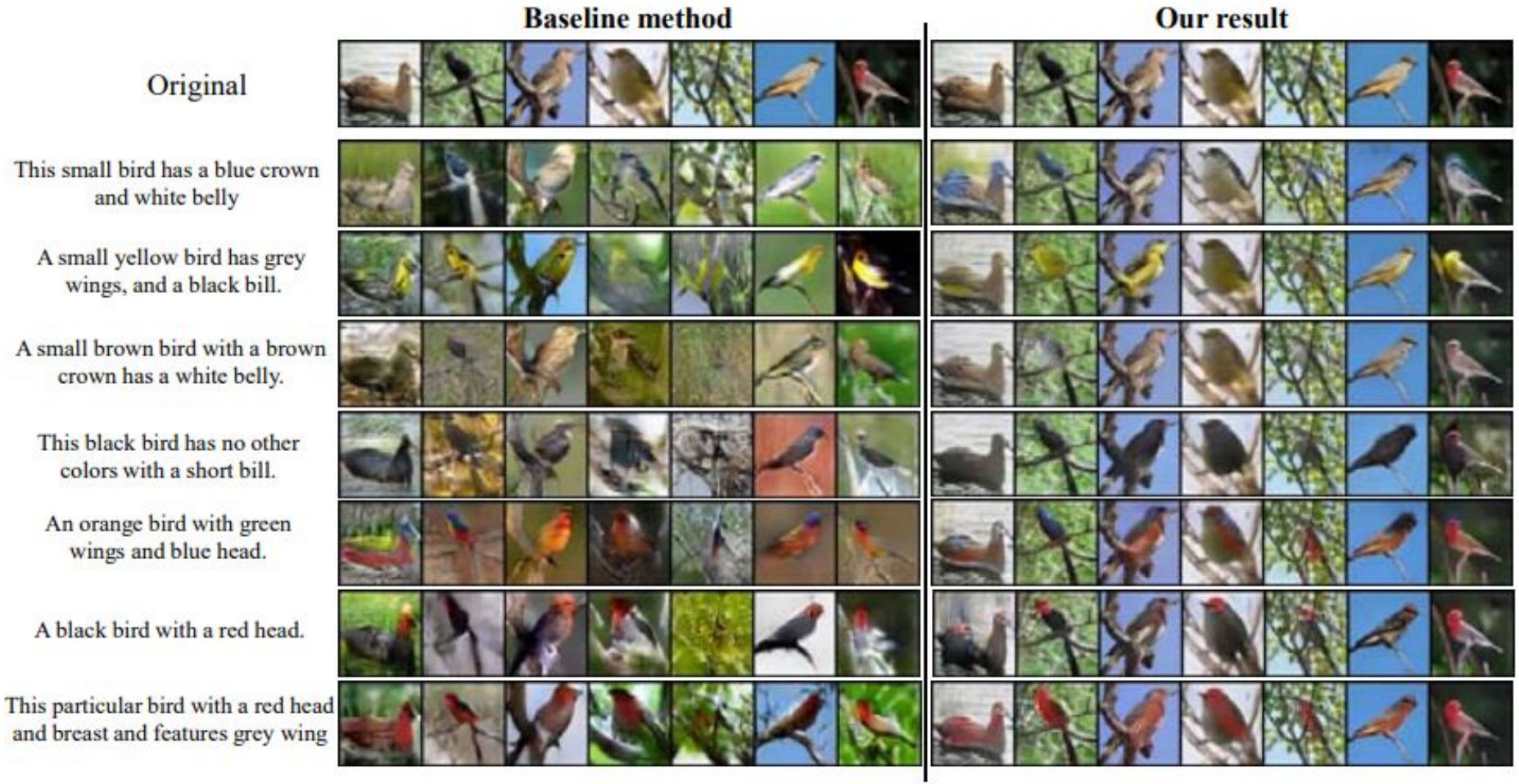
Basically… translate a picture into words (seq2seq)

Highly related with GANs: Conditional Synthesis

It also may happen in the other way (Text2Image)!



"A black cat."

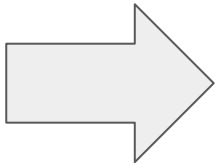


MC-GAN: Multi-conditional Generative Adversarial Network for Image Synthesis: https://arxiv.org/pdf/1805.01123.pdf

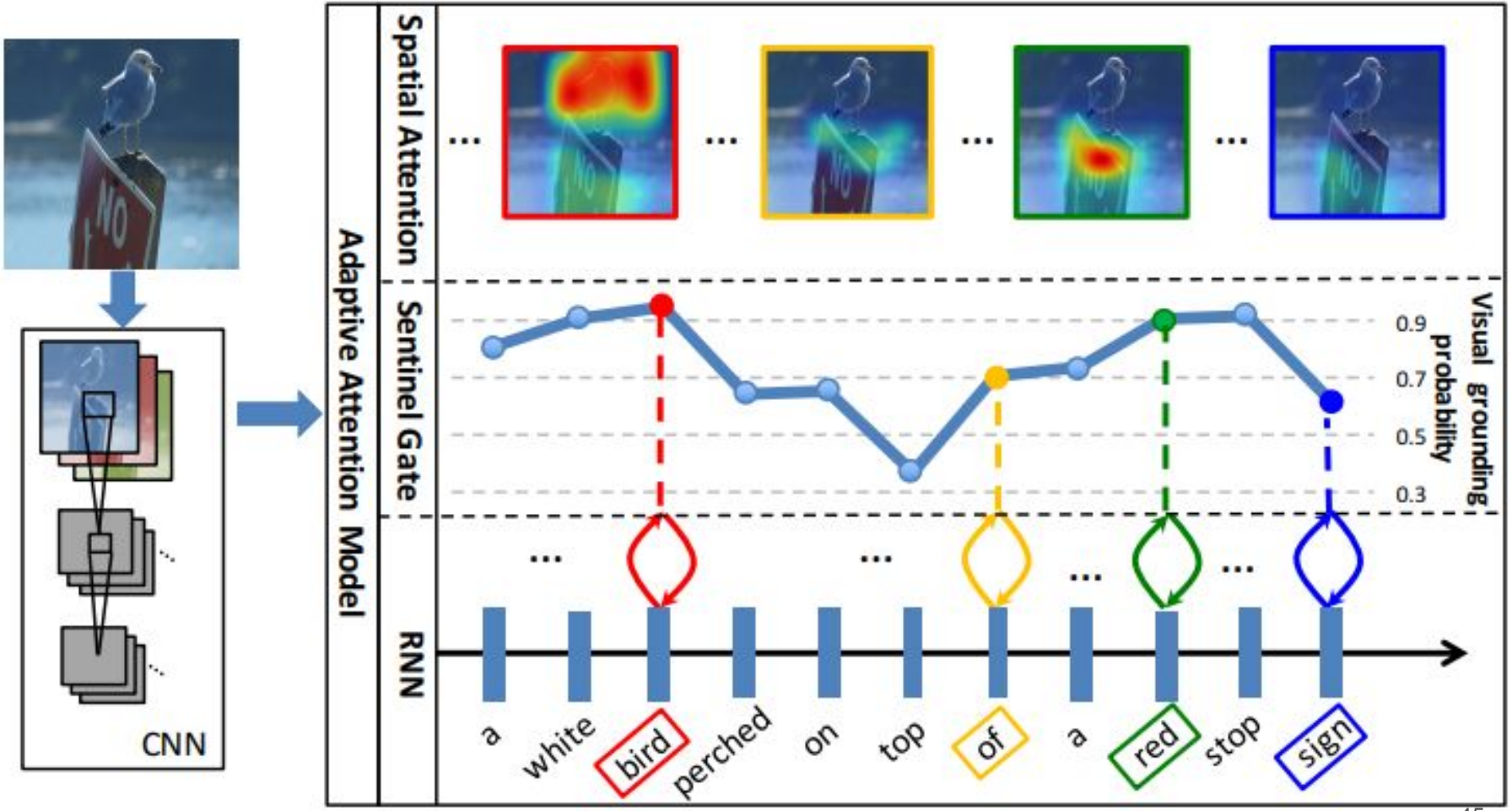
Captioning

Given an image generate:

- Textual descriptions



"Man in blue shirt is running."



Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning https://arxiv.org/pdf/1612.01887v2.pdf

a little girl sitting on a bench holding an umbrella.

a herd of sheep grazing on a lush green hillside.

a close up of a fire hydrant on a sidewalk.

a yellow plate topped with meat and broccoli.

a zebra standing next to a zebra in a dirt field.

a stainless steel oven in a kitchen with wood cabinets.

two birds sitting on top of a tree branch.

an elephant standing next to rock wall.

a man riding a bike down a road next to a body of water.

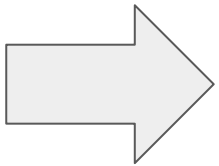
Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning https://arxiv.org/pdf/1612.01887v2.pdf

Retrieval

Given a text:

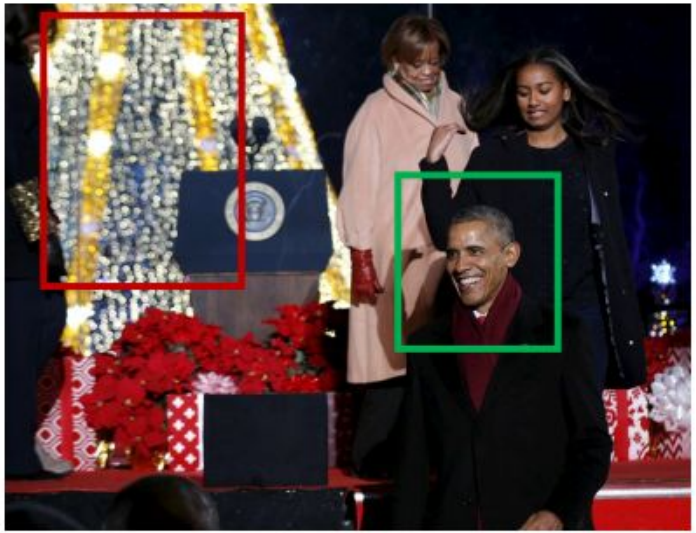
- Recover an images
- (images indexed by query strings or concepts, annotates regions within a image with a word)

Query:
"Blue shirt"

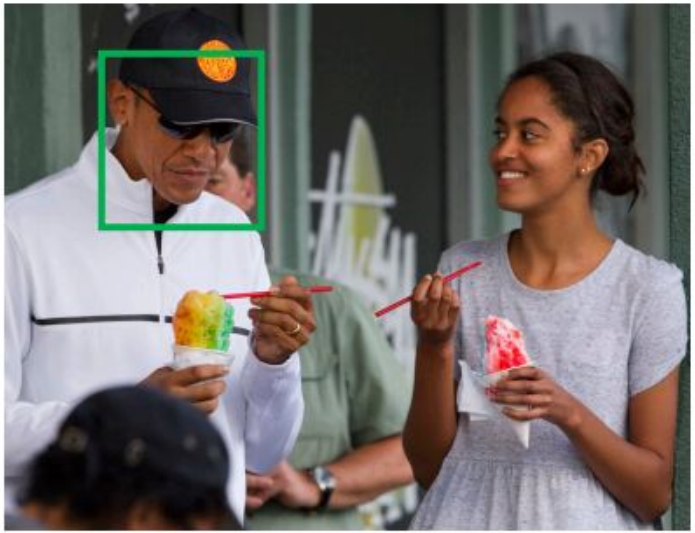


Query1: Barack Obama

Query2: Christmas



Keyword: US president, **Christmas Tree**, ceremony, family …



Keyword: **President Obama**, **Christmas holiday**, Ice-cream, Happy Malia …

AMC: Attention guided Multi-modal Correlation Learning for Image Search https://arxiv.org/pdf/1704.00763.pdf

Referral Expression

Given a text:

- Segment/detect an specific object in context within an image.

"Fourth bottle from right to left"

Visual Question Answering

Given an image generate:

- An answer to a question about that image

Is there a blue shirt man?

Yes!

Who is wearing glasses?

man

woman

Where is the child sitting?

fridge

arms

Is the umbrella upside down?

yes

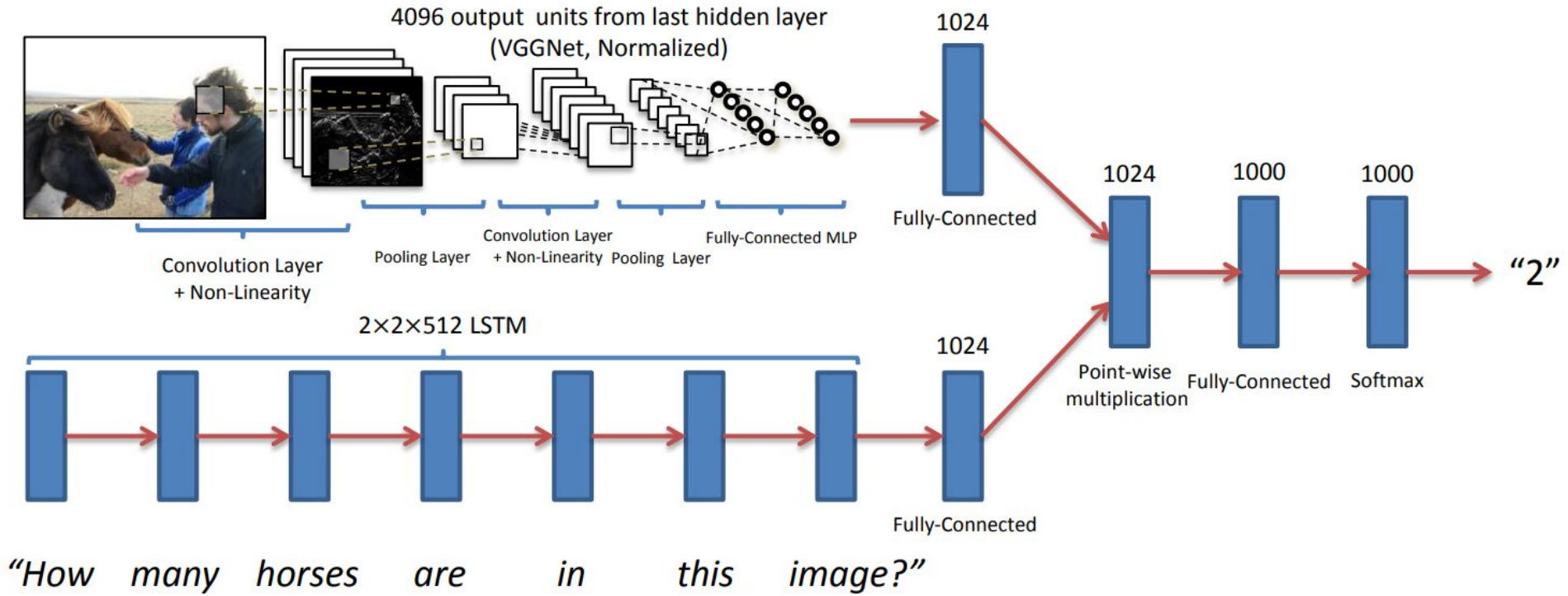
no

How many children are in the bed?

2

1

Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering
https://arxiv.org/pdf/1612.00837.pdf



VQA: Visual Question Answering https://arxiv.org/pdf/1505.00468v6.pdf

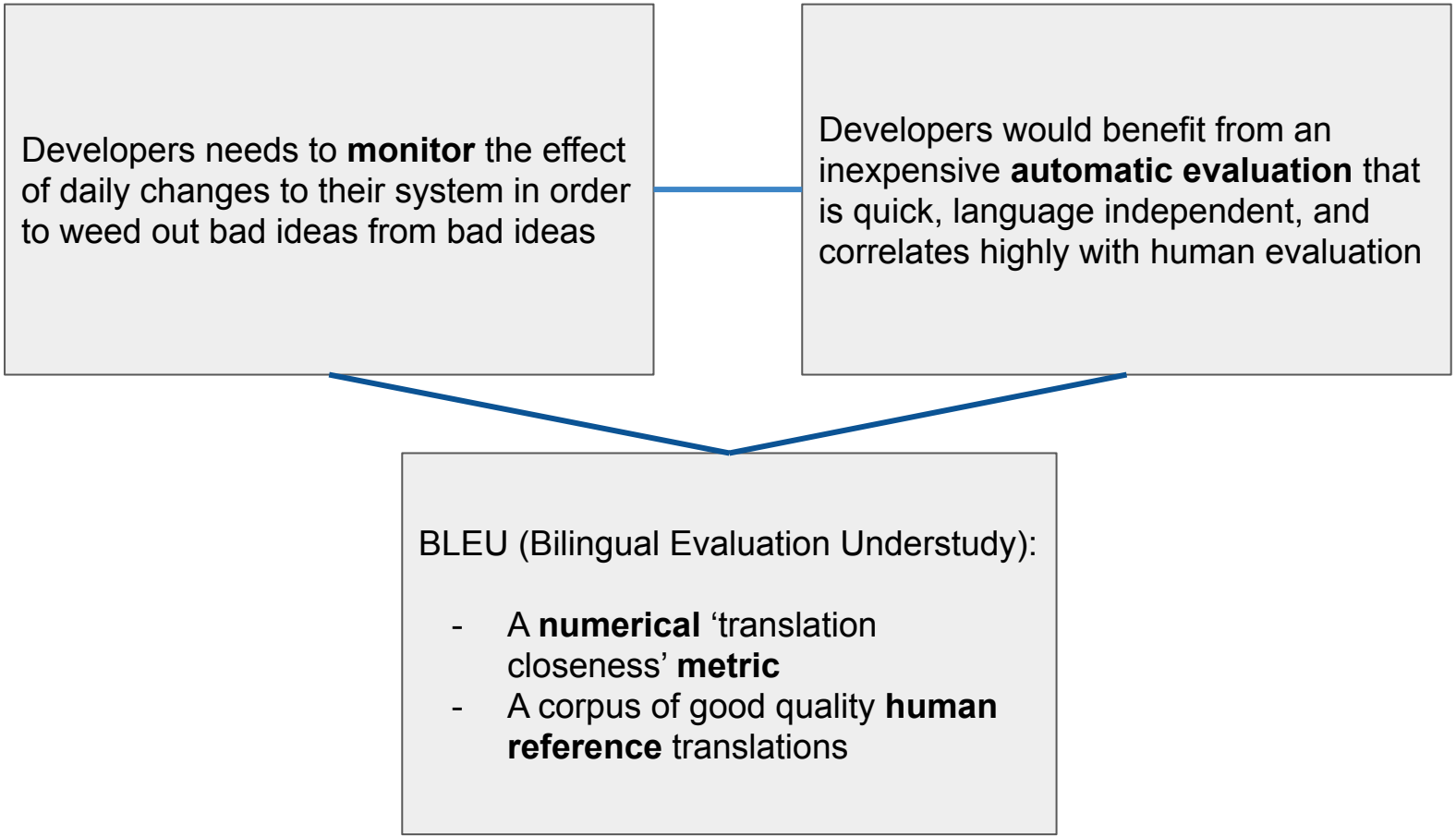
Metrics

Which aspects of translations does human evaluations take into account?

- Adequacy: Is concerned with the internal structure [i.e. grammar]
- Fidelity: how precisely a translated document conforms with its source
- Fluency

How does one measure translation performance?

The closer the machine translation is to a professional human translation, the better it is


Developers needs to **monitor** the effect of daily changes to their system in order to weed out bad ideas from bad ideas
Developers would benefit from an inexpensive **automatic evaluation** that is quick, language independent, and correlates highly with human evaluation
BLEU (Bilingual Evaluation Understudy):
- A **numerical** 'translation closeness' **metric**
- A corpus of good quality **human reference** translations

Not so easy!

Translations may vary in word choice or in word order even when they use the same words

Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.

Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

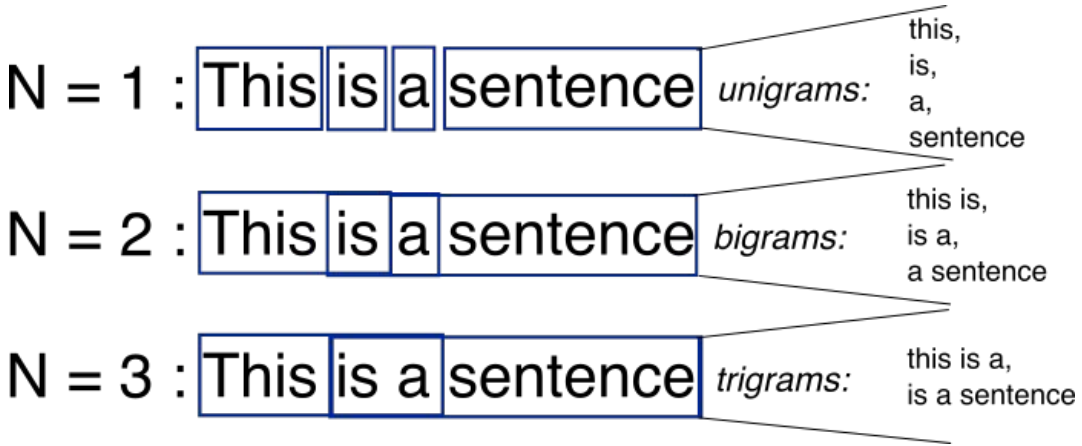
Reference 3: It is the practical guide for the army always to heed the directions of the party.

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.

Tasks of BLEU

- To compare n-grams of the candidate with the n-grams of the reference translation and count the number of matches.
  - Matches are **position independent**
  - The more the matches, the better the candidate translation is



Traditional precision

1. Counts the number of candidate translation words (unigrams) which occur in any reference translation (7)
2. Divides by the total number of words in the candidate translation (7)

```
Candidate: the the the the the the the.
Reference 1: The cat is on the mat.
Reference 2: There is a cat on the mat.
```

Result = 7/7

Traditional precision

1. Counts the number of candidate translation words (unigrams) which occur in any reference translation (7)
2. Divides by the total number of words in the candidate translation (7)

```
Candidate: the the the the the the the.
Reference 1: The cat is on the mat.
Reference 2: There is a cat on the mat.
```

Result = 7/7

Modified n-gram precision

1. Counts the maximum number of times word occurs in any single reference translation (2)
2. Clip the total count of each candidate word by its maximum reference count
3. Add these clipped counts up
4. Divides by the total (unclipped) number of candidate words

Result = 2/7

Candidate: the the the the the the the.

Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

Exercise

What is the modified unigram precision for candidate 1?

A. 17/18
B. 7/18
C. 14/18
D. 16/18

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

Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.

Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference 3: It is the practical guide for the army always to heed the directions of the party.

Exercise

What is the modified unigram precision for candidate 1?

A. 17/18
B. 20/18
C. 19/18
D. 16/18

Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.

Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference 3: It is the practical guide for the army always to heed the directions of the party.

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

Modified n-gram precision

- Modified n-gram precision is computed similarly for any n
- Captures two aspects of translation: **Adequacy** and **fluency**.
  - A translation using the same words (1-grams) as in the references tends to satisfy: adequacy
  - The longer the n-gram matches account for fluency

Modified n-gram precision on blocks of text

Basic unit of evaluation: the sentence

1. Compute the n-gram matches sentence by sentence
2. Add the clipped n-gram counts for all the candidate sentences
3. Divide by the number of candidate n-grams in the test corpus

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}.$$

Sentence length

A candidate translation should be neither too long nor too short

Penalizes:

- N-gram precision **penalizes spurious words** in the candidate that do not appear in any of the reference translation
- Modified precision is penalized if a **word occurs more frequently in a candidate** translation than its maximum reference count

Issues of Modified N-gram Precision : Sentence Length

```
Candidate 3: of the
Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.
Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
Reference 3: It is the practical guide for the army always to heed directions of the party.
```

Modified Unigram Precision : 2/2

Modified Bigram Precision : 1/1

Issues of Modified N-gram Precision : Trouble with Recalls

Good candidate should only use (recall) one possible word choices

Example:

```
Candidate 1: I always invariably perpetually do. (Bad Translation)
Candidate 2: I always do. (A complete Match)
Reference 1: I always do.
Reference 2: I invariably do.
Reference 3: I perpetually do.
```

Sentence brevity penalty

- Candidate translations **longer than their references** are already penalized by the modified n-gram measure
- Multiplicative brevity penalty factor: High-scoring candidate translation must now match the reference translation in **length, in word choice and in word order**!

When a translation matches a reference

BP = 1

When a translation is shorter than the reference

BP < 1

Sentence brevity penalty

- r is the sum of the best match lengths of the candidate sentence in the test corpus
- c is the total length of the candidate translation corpus

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

Then,

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

$$\log \mathrm{BLEU} = \min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} w_n \log p_n.$$

Rouge

Recall-Oriented Understudy for Gisting Evaluation

- For evaluating text summarization.
- To automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans.
- ROUGE measures count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans.

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

Types of ROUGE measures

- ROUGE-N
- ROUGE-L
- ROUGE-W
- ROUGE-S

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

ROUGE-N

N-gram recall between a candidate summary an a set of reference summaries.

An n-gram recall between a candidate summary and a set of reference summaries

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

Example

- System Summary (what the machine produced)

  the cat was found under the bed

- Reference Summary (usually by humans)

  the cat was under the bed

ROUGE-2 = 4/5
ROUGE-3 = 2/4

https://rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/#.XX4-QyhKjIX

Longest Common Subsequence

A sequence Z = [z1, z2, ..., zn] is a subsequence of another sequence X = [x1, x2, ..., xm], if there exists a strict increasing sequence [i1, i2, ..., ik] of indices of X such that for all j = 1, 2, ..., k, we have xij = zj (Cormen et al., 1989).

Given two sequences X and Y, the longest common subsequence (LCS) of X and Y is a common subsequence with maximum length.

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

ROUGE-L

The intuition: the longer the LCS of two summary sentences is, the more similar the two summaries are.

They propose using LCS-based F-measure to estimate the similarity between two summaries *X* of length *m* and *Y* of length *n*, assuming *X* is a reference summary sentence and *Y* is a candidate summary sentence.

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

ROUGE-L

$$R_{lcs} = \frac{LCS(X,Y)}{m}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n}$$

$$F_{lcs} = \frac{(1+\beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$ ← ROUGE-L

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

Example

- S1: police killed the gunman (the reference)
- S2: police kill the gunman
- S3: the gunman kill police

For S2 and S3 ROUGE-2 = 1/3

For S2 and β = 1, ROUGE-L = 3/4

For S3 and β = 1, ROUGE-L = 2/4

https://rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/#.XX4-QyhKjIX

ROUGE-W

Weighted Longest Common Subsequence

We can simply remember the length of consecutive matches encountered so far to a regular two dimensional dynamic program table computing LCS.

Consider a weighting function f that have the property that f(x+y) > f(x) + f(y) for any positive integers x and y.

Consecutive matches are awarded more scores than non-consecutive matches.

ROUGE-W

Weighted Longest Common Subsequence

$$R_{wlcs} = f^{-1}\left(\frac{WLCS(X,Y)}{f(m)}\right)$$

$$P_{wlcs} = f^{-1}\left(\frac{WLCS(X,Y)}{f(n)}\right)$$

$$F_{wlcs} = \frac{(1+\beta^2)R_{wlcs}P_{wlcs}}{R_{wlcs}+\beta^2 P_{wlcs}}$$

← ROUGE-W

ROUGE-S

Skip-Bigram Co-Occurrence Statistics

Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps.

Skip-bigram co-occurrence statistics measure the overlap of skip bigrams between a candidate translation and a set of reference translations.

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

ROUGE-S

$$R_{skip2} = \frac{SKIP2(X,Y)}{C(m,2)}$$

$$P_{skip2} = \frac{SKIP2(X,Y)}{C(n,2)}$$

$$F_{skip2} = \frac{(1+\beta^2)R_{skip2}P_{skip2}}{R_{skip2}+\beta^2 P_{skip2}}$$ ← ROUGE-S

ROUGE: A Package for Automatic Evaluation of Summaries. Lin, 2004. https://www.aclweb.org/anthology/W04-1013

Meteor

What’s the problem with Bleu?

Doesn’t reproduce reliable scores for sentence- level translations (i.e. individual sentences are affected due to averaging)!

In response to this limitation:

- GTM
- TER
- CDER
- **METEOR or (Metric for Evaluation for Translation with Explicit Ordering)**

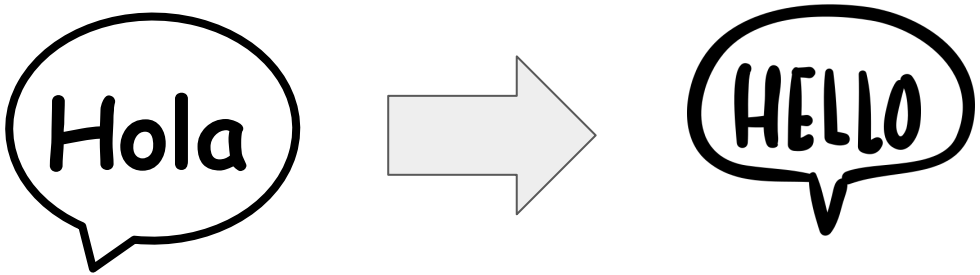
Specifically designed to **increase the correlation with human judgments at segment level.**

Spoiler: It outperforms Bleu!

But… What METEOR does?

- Evaluate translations: **Word to word matches.**
  - If >1 ref. Translation is available, then -> translation vs all references
  - Best score is used.

- Pair of strings -> **word alignment (1 to 1) (system vs reference)**
  (i.e. every word in each string maps to at most one word in the other one)



But… How to align?

Several modules for word-mapping (alignments/) :

1. Exact: Are they the same word?
2. Porter stem: Do they have similar meaning? (removing suffixes)
3. WN synonymy: Are they synonyms?

They might be applied simultaneously!
Order = matching preferences...

Basically: **They generate word matches between pairs of strings! (connect unigram A and unigram B)**

But… then what?

1. Identify the largest subset of the word maps (alignments) previously generated.
2. Is there at least one maximal alignment?
3. Select the alignment in which the two strings are most similar.
   a. Generally: Exact > Porter > WN
4. After the final alignment is done, compute METEOR score:
   4.1. Calculate **unigram** **P**recision = m/t
   4.2. Calculate **unigram** **R**ecall = m/r
      a. m: number of mapped **unigrams** between two strings.
      b. t: total number of **unigrams** in the translation
      c. r: total number of **unigrams** in the reference.

But … What are unigrams?

Consider:

“I have a blue car” ------> “I” “have” “a” “blue” “car” **(Unigrams)**

“I have a blue car” ------> “I have” “have a” “a blue” “blue car” **(Biagram: two adjacent words)**

But… then what?

4.3. Compute a parameterized harmonic mean (**Fmean**) of the **P**recision and the **R**ecall:

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1 - \alpha) \cdot R}$$

Fmean, P & R based on single word matches…

But… *what if matched unigrams are not in the same **word order** in two strings?*

5. Penalty (Pen):

$$Penalty = \gamma \left(\frac{c}{m}\right) \text{^}\beta$$

Fragmentation fraction

c: number of chunks

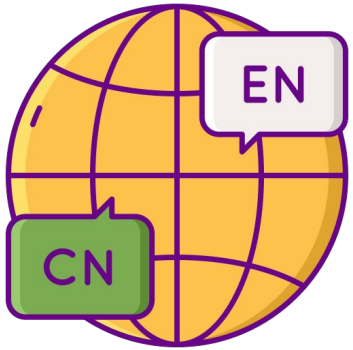
m: number of matches (unigrams that have been mapped)

But …

Chunks:

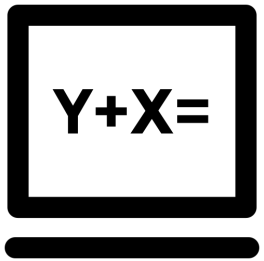
“set of unigrams that are adjacent in the translation and in the reference”

Have in mind that: contiguous matches = lower number of chunks = lower penalty



And finally! 6. The METEOR score:

$$score = (1 - Pen) \cdot F_{mean}$$



But… What are those α , β , γ parameters?

α : Relative weight of precision and recall in Fmean score.

β : Shape of penalty as a function of fragmentation (relation between fragmentation and penalty).

γ : Relative weight of the fragmentation penalty.

Every version of METEOR have used: α = 0.9, β = 3.0 and γ = 0.5.

*This parameters **may vary depending on the languages**.*

But ... what about an example?

Consider:

Reference: the boy was running in the street
Translation: the boy running in the street

m: 6
r: 7
t: 6

So...

Consider:

Reference: the boy was running in the street
Translation: the boy running in the street

P = 6/6 = 1
R = 6/7 = 0.8571

Using the parameters as above.

Consider:

Reference: the boy was running in the street
Translation: the boy running in the street

Fmean = (1*0.8571)/(0.9*1)+(1-0.9)*0.8571=0.8695

Chunks (c): 2

Consider:

Reference: the boy was running in the street
Translation: the boy running in the street

Penalty= 0.5*(2/6)^3 = 0.0185

Consider:

Reference: the boy was running in the street
Translation: the boy running in the street

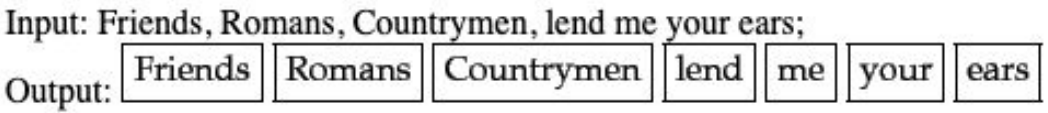
METEOR_score = (1-0.0185)*0.8695 = 0.8534

Intermission
10 minutes

Autoregressive Transformer Models

Relevant words

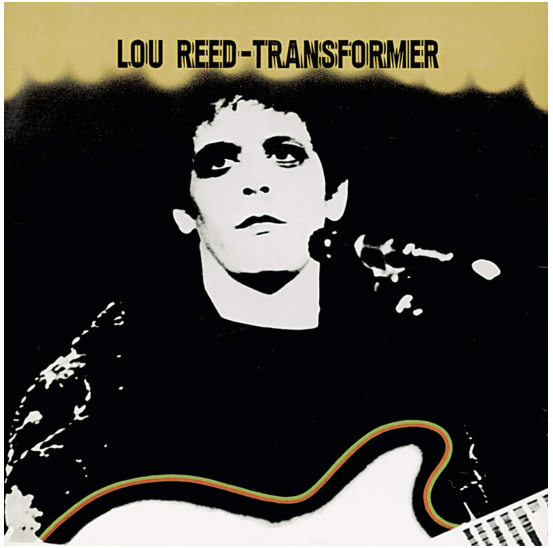
- Tokenization: Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens* [1]



- Source sentence: input sentence
- Target sentence: output sentence

[1] https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

What are Transformers?

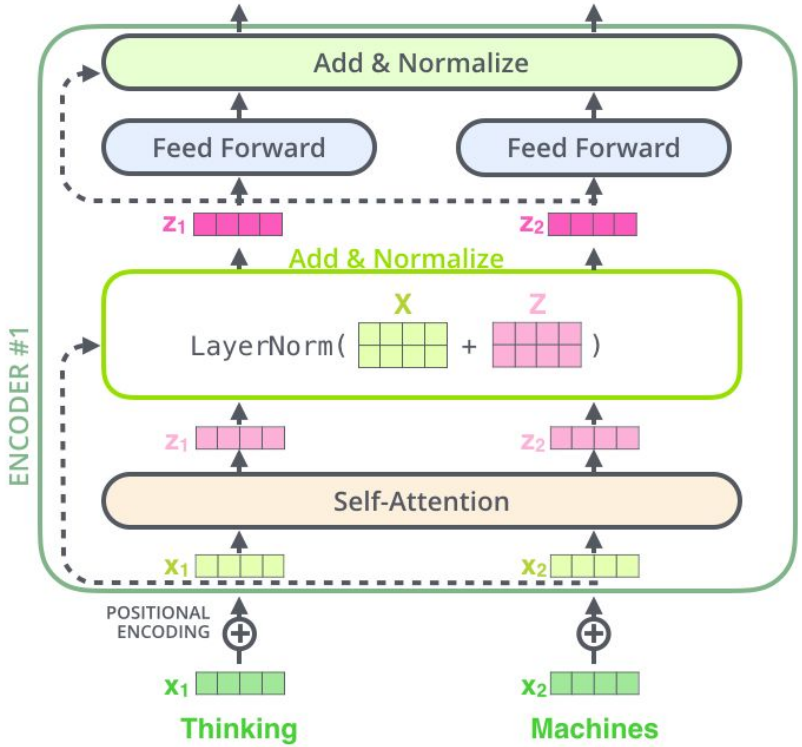


What is a Transformer?

The Transformer in NLP is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease

> *"The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution."*

What is a Transformer?



- A Transformer encoder computes a set of revised representations z from a input sequence x.
- The representation size is equal to the input size
- The encoder is composed of several transformer layers

Papers

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

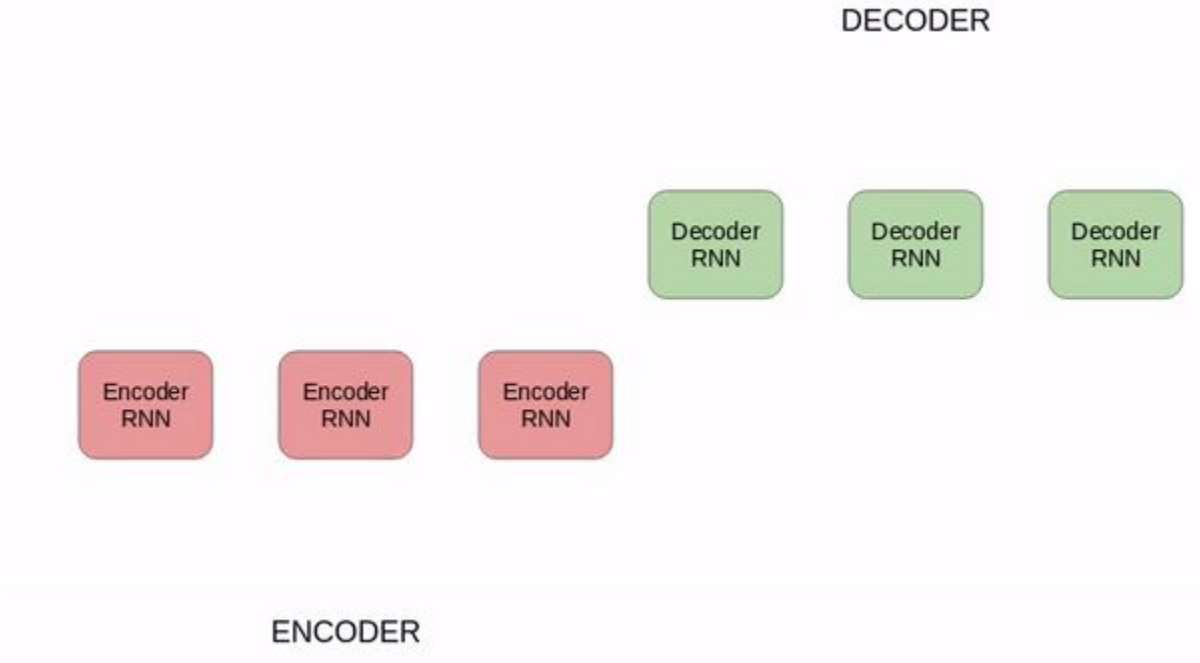
Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Brief recap of RNN



What is attention?

Attention is:

- A mechanism to basically increase the performance of RNN
- A mechanism in which the decoder in the step we are looking at can actually decide to go back and look at particular parts of the input!

Why Attention is all you need?

- We would like to teach the decoder: Hey you need to pay close attention to this step here, because that was the step when the word X was just encoded!
- Path length of info is much **shorter** than going through all the hidden states
- With attention you **wouldn’t need recurrent** things in every step!

Transformer

- This is not an RNN, all happens at once, in consequence:
  - Every step is a training sample
  - There’s no multi step as in RNN

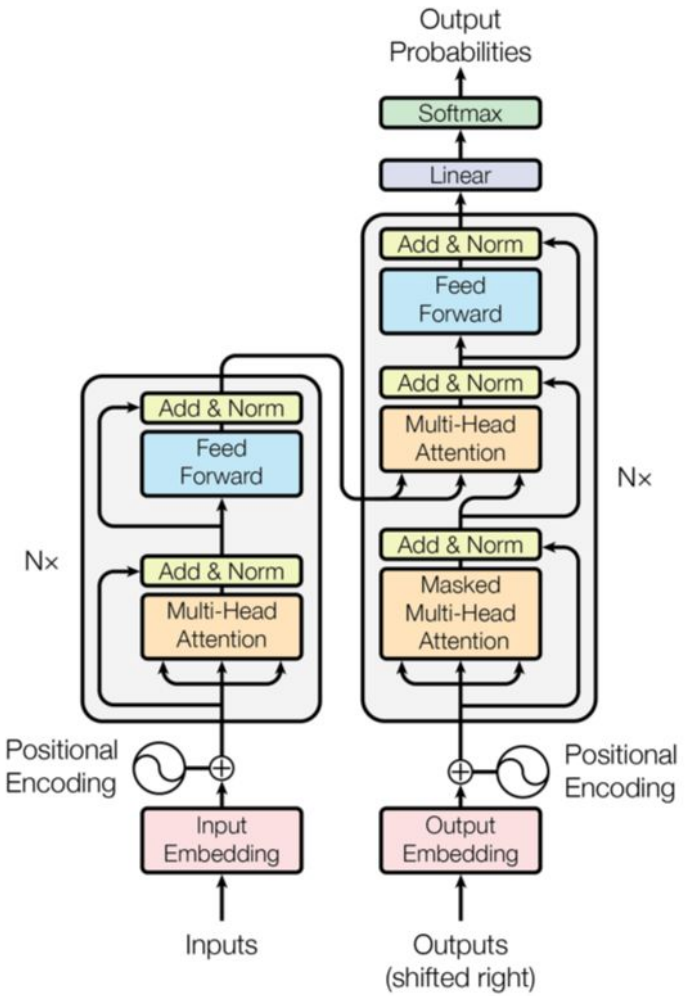


Figure 1: The Transformer - model architecture.

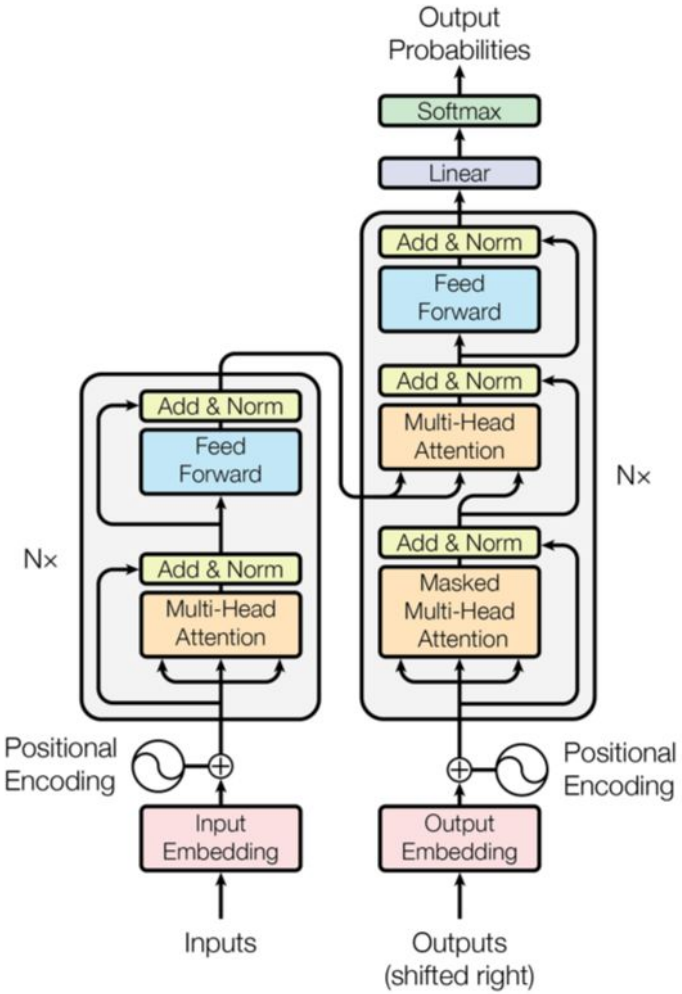
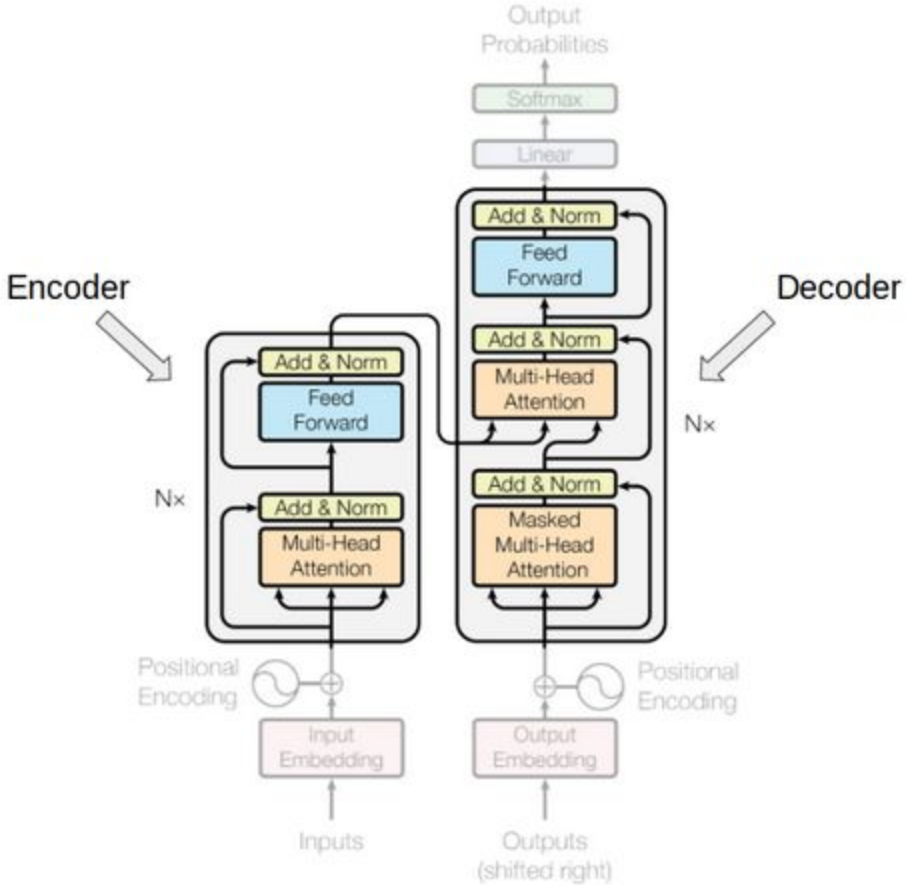


Figure 1: The Transformer - model architecture.


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Encoder
Decoder
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

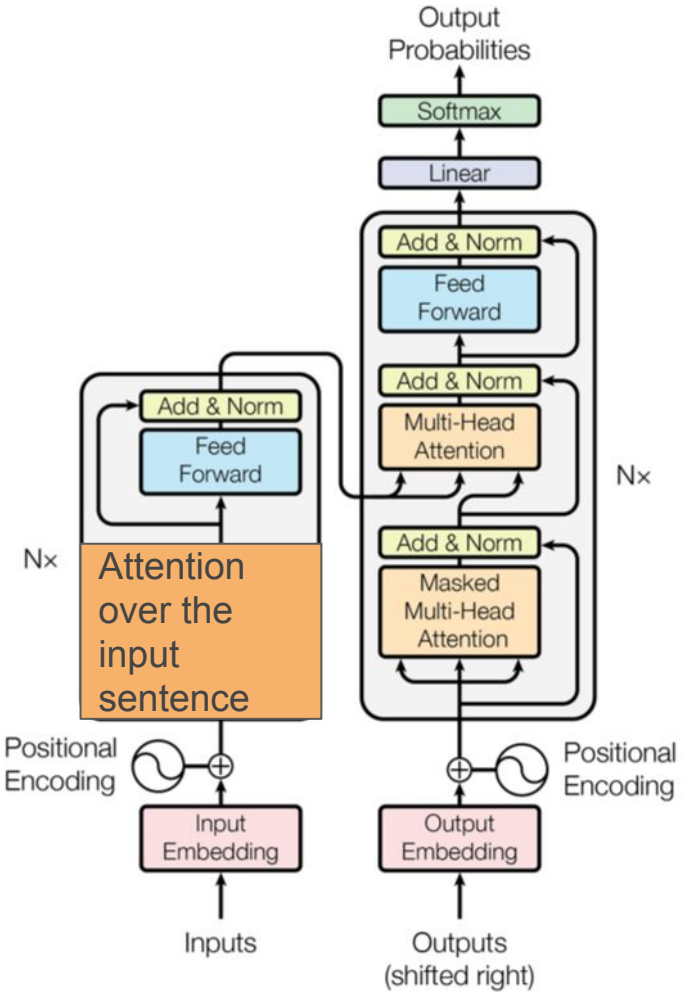


Figure 1: The Transformer - model architecture.

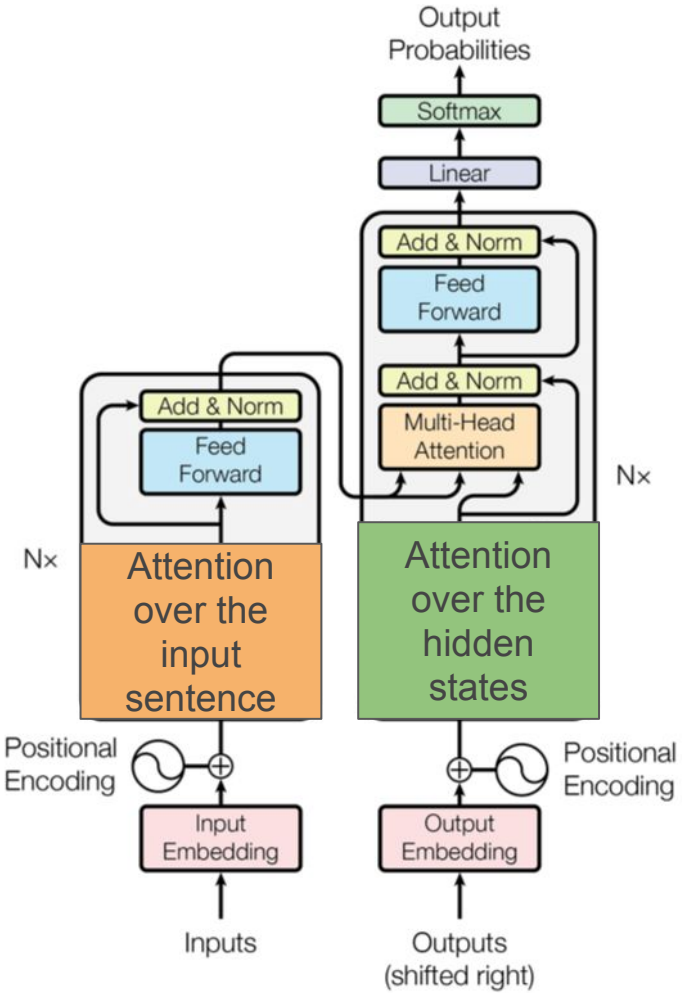


Figure 1: The Transformer - model architecture.

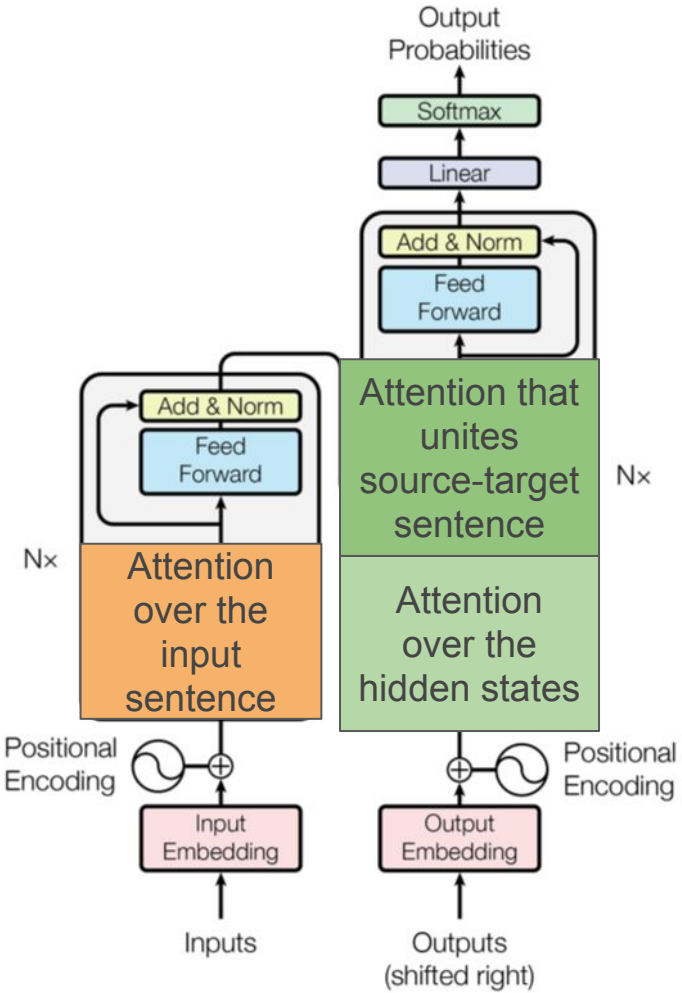
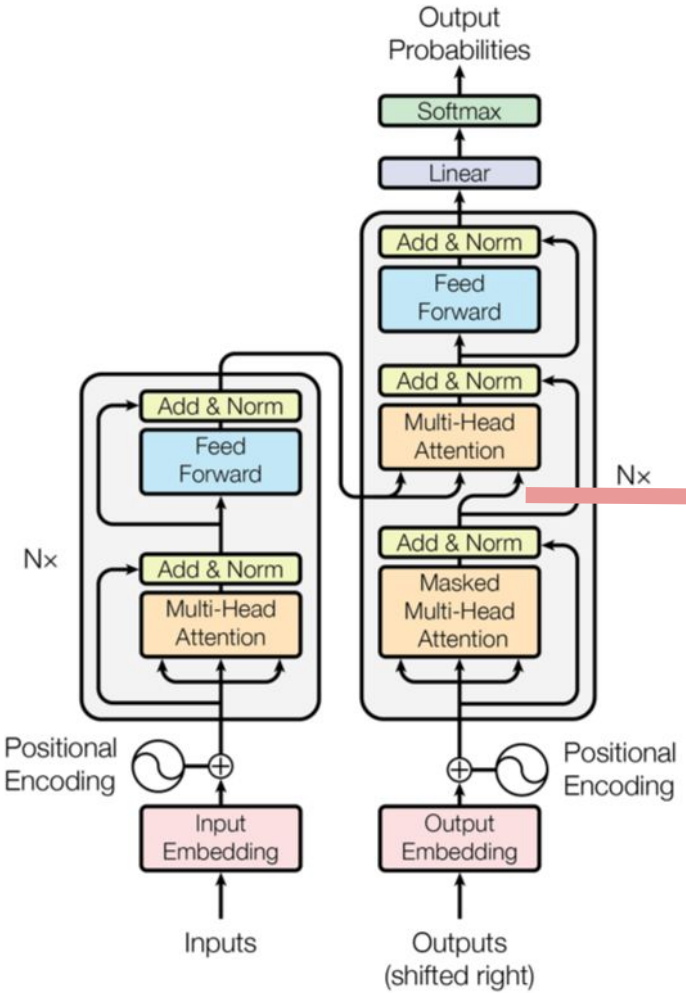


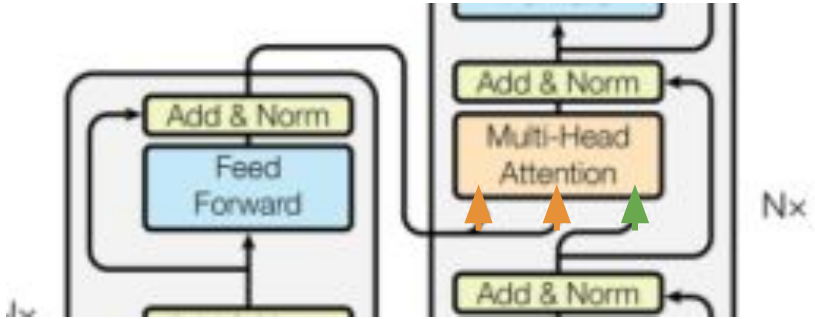
Figure 1: The Transformer - model architecture.



What are this three connections?

Figure 1: The Transformer - model architecture.

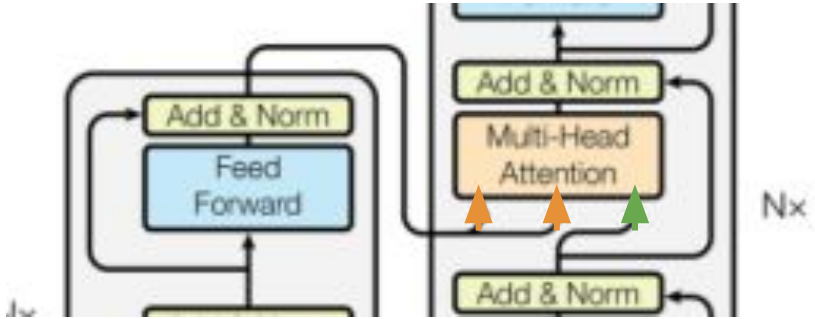
What are these three connections?

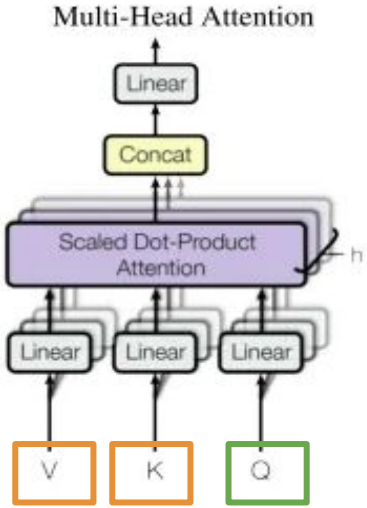


Two first connections are the Key / Values outputs by the encoding part of the source sentence

The last connection is composed of the queries by the encoding part of the target sentence

What are these three connections?





Two first connections are the Key / Values outputs by the encoding part of the source sentence

The last connection is composed of the queries by the encoding part of the target sentence

Scaled Dot-product Attention

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

Q: Query vector
K: Key vector
V: Value vector
Dk: Dimension of k

Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Variations

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

Attentions is all you need!

- Faster than architectures based on recurrent or convolutional layers
- Reduces the path length
- improvement over the RNN based seq2seq models

Limitations

- Attention can only deal with fixed-length text strings. **The text has to be split into a certain number of segments** or chunks before being fed into the system as input
- This chunking of text causes **context fragmentation**. For example, if a sentence is split from the middle, then a significant amount of context is lost. In other words, **the text is split without respecting the sentence or any other semantic boundary**

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Google AI

Bidirectional Encoder Representations from Transformers

Training general purpose language representation models using the enormous amount of unannotated text on the web.

The pre-trained model can then be fine-tuned on small-data NLP tasks like question answering and sentiment analysis, resulting in substantial accuracy improvements compared to training on these datasets from scratch (Google AI blog).

"We'll use transformer encoders", said BERT.

"This is madness", replied Ernie, "Everybody knows bidirectional conditioning would allow each word to indirectly see itself in a multi-layered context."

"We'll use masks", said BERT confidently.

http://jalammar.github.io/illustrated-bert/

Problem with Previous Methods

Problem: Language models only use left context or right context, but language understanding is bidirectional.

Why are LMs unidirectional?

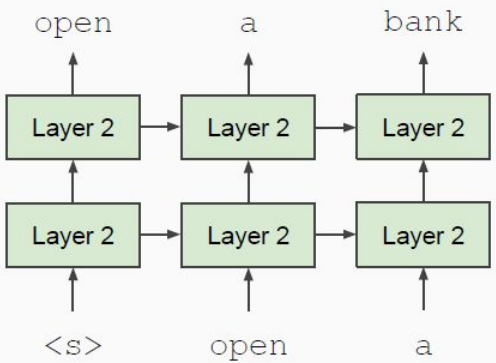
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
- Reason 2: Words can “see themselves” in a bidirectional encoder.

Based on slides from Jacob Devlin

Unidirectional vs. Bidirectional Models

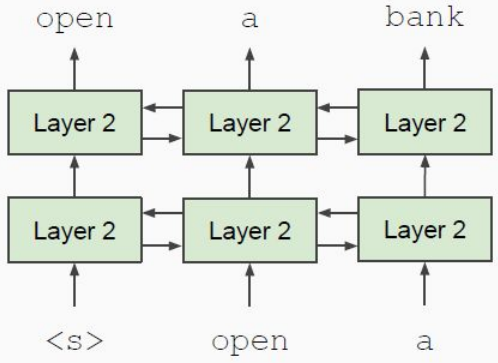
Unidirectional context

Build representation incrementally



Bidirectional context

Words can “see themselves”

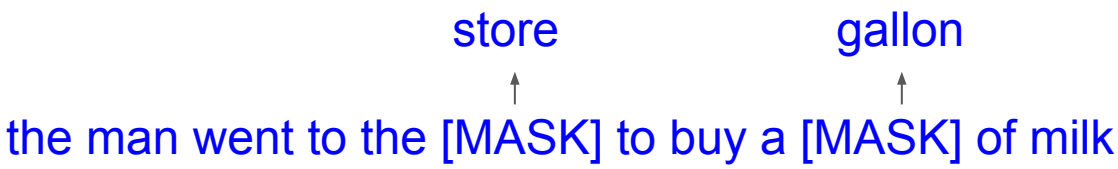


Based on slides from Jacob Devlin

Task 1: Masked LM (Cloze task)

Solution: Mask out *k*% of the input words, and then predict the masked words

- k = 15% (1 in 7 words)



- Too little masking: Too expensive to train
- Too much masking: Not enough context

Based on slides from Jacob Devlin

Task 2: Next Sentence Prediction

To learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence.

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

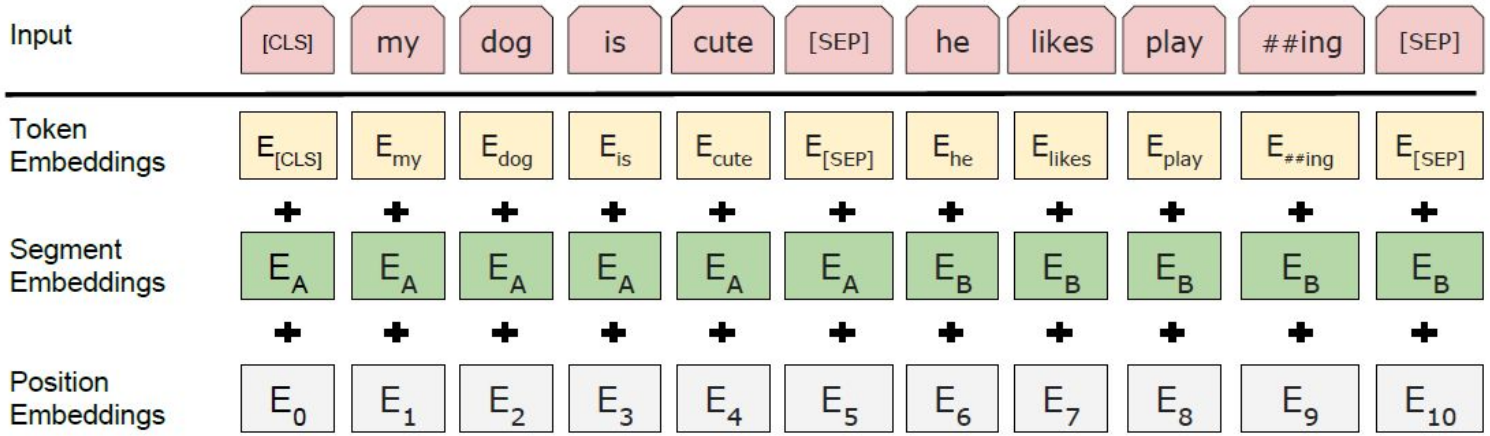
Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

Based on slides from Jacob Devlin

Input Representation

Use 30,000 WordPiece vocabulary on input.

Each token is sum of three embeddings



Based on slides from Jacob Devlin

Model Architecture

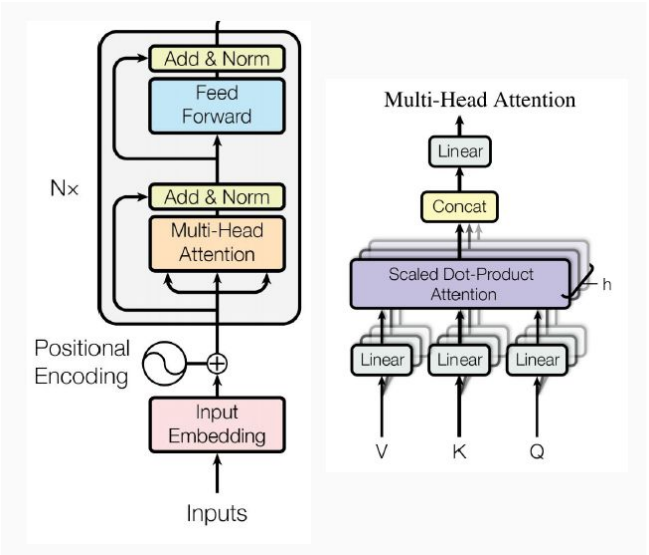
BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation.

Notation

L: the number of layers (i.e., Transformer blocks).

H: the hidden size.

A: the number of self-attention heads.



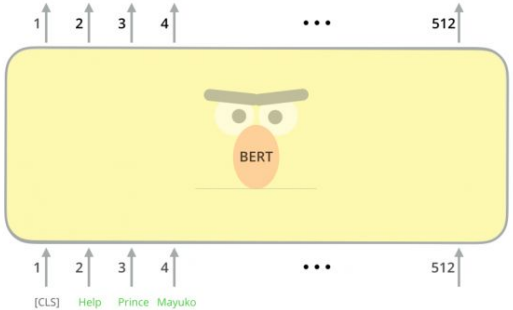
Based on slides from Jacob Devlin

Model Architecture

They report results on two model sizes:

- $\text{BERT}_{\text{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M)

- $\text{BERT}_{\text{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M)



Based on slides from Jacob Devlin

http://jalammar.github.io/illustrated-bert/

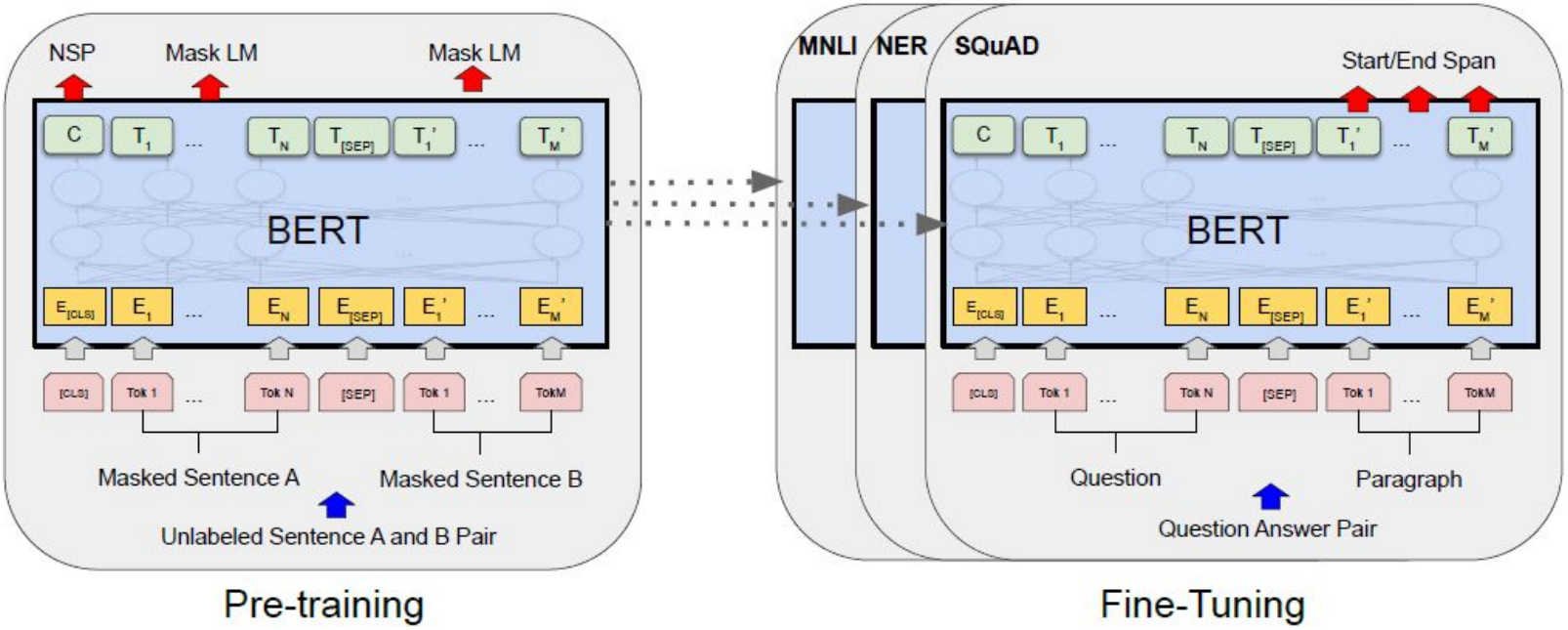
Model Details

Data: Wikipedia (2.5B words) + BookCorpus (800M words)

- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- Optimizer: AdamW, 1e-4 learning rate, linear decay
- Trained on 4x4 or 8x8 TPU slice for 4 days

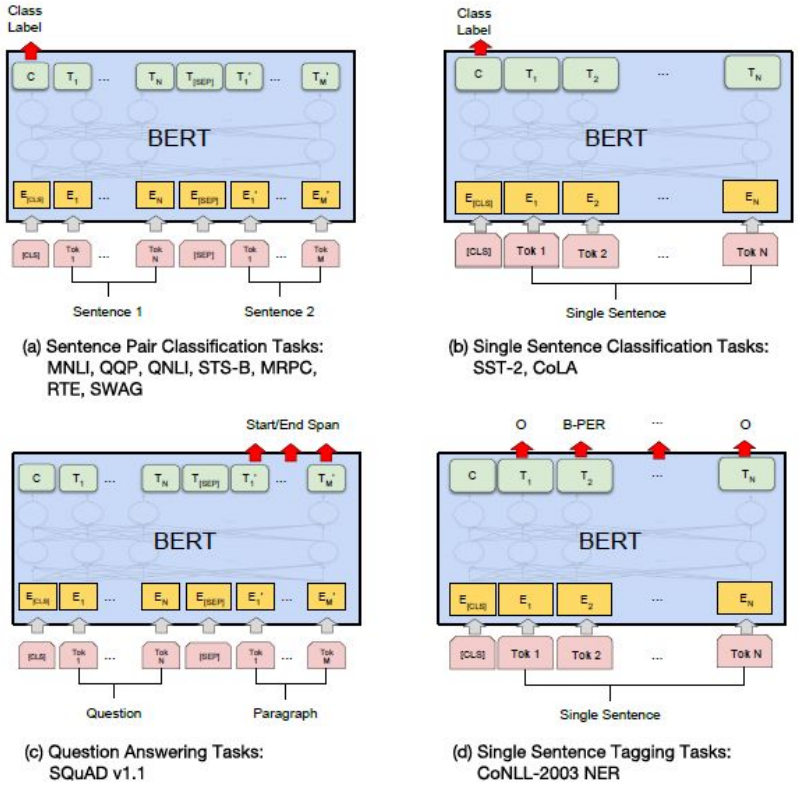
Based on slides from Jacob Devlin

Fine-Tuning Procedure



Based on slides from Jacob Devlin

Fine-tuning BERT on Different Tasks



Comparison of BERT, ELMo ,and OpenAI GPT

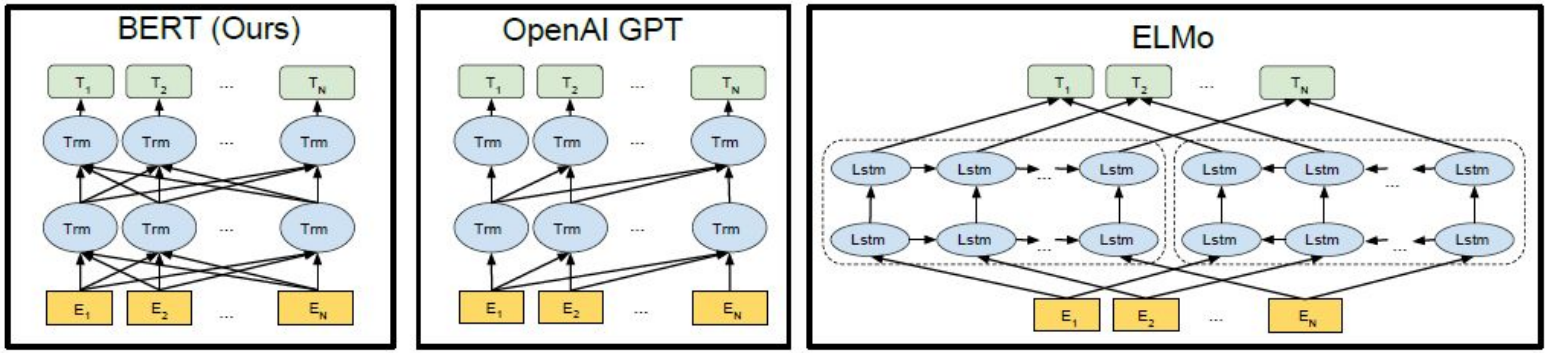


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

GLUE Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MultiNLI

Premise: Hills and mountains are especially sanctified in Jainism.
Hypothesis: Jainism hates nature.
Label: Contradiction

CoLa

Sentence: The wagon rumbled down the road.
Label: Acceptable

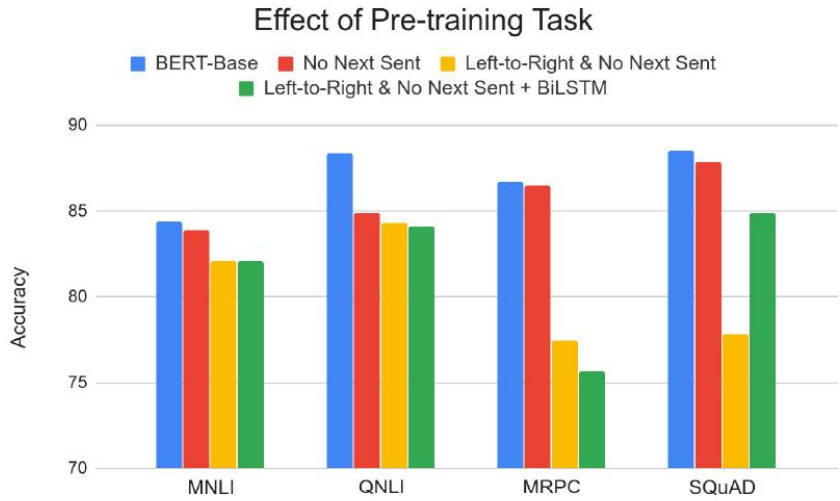
Sentence: The car honked down the road.
Label: Unacceptable

Based on slides from Jacob Devlin

Effect of Pre-training Task

Masked LM (compared to left-to-right LM) is very important on some tasks, Next Sentence Prediction is important on other tasks.

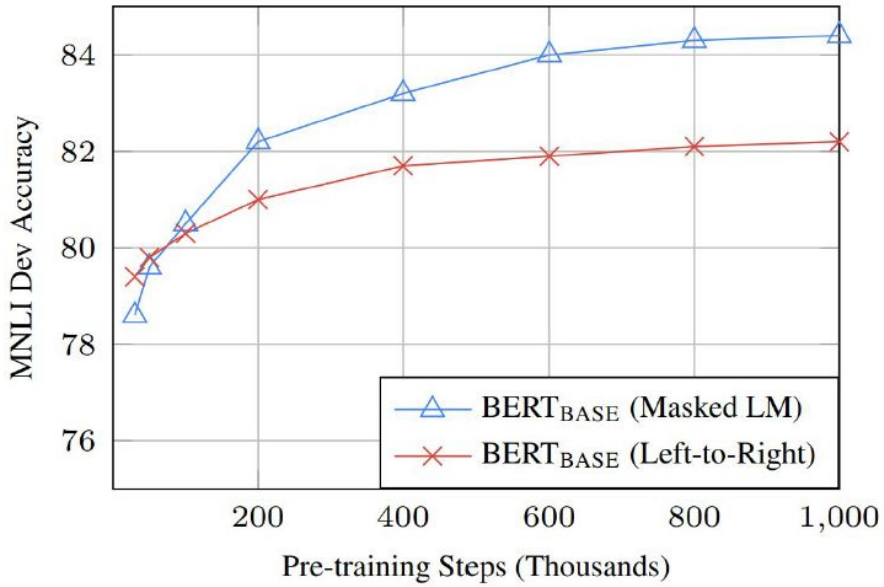
Left-to-right model does very poorly on word-level task (SQuAD), although this is mitigated by BiLSTM



Based on slides from Jacob Devlin

Effect of Directionality and Training Time

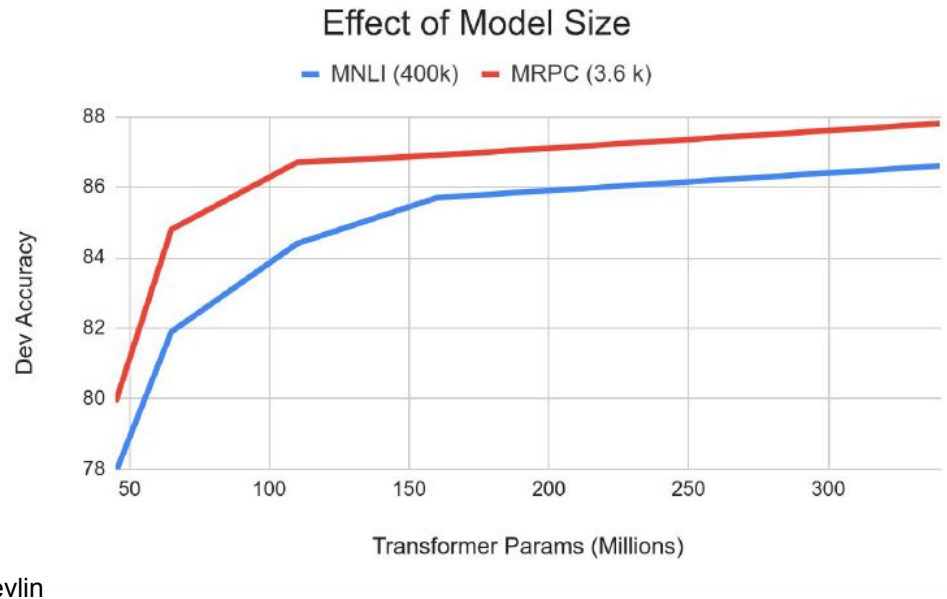
- Masked LM takes slightly longer to converge because we only predict 15% instead of 100%
- But absolute results are much better almost immediately



Based on slides from Jacob Devlin

Effect of Model Size

- Big models help a lot
- Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples



Based on slides from Jacob Devlin

Effect of Masking Strategy

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

- Masking 100% of the time hurts on feature-based approach
- Using random word 100% of time hurts slightly

Based on slides from Jacob Devlin

Conclusions

Rich, unsupervised pre-training is an integral part of many language understanding systems.

The bi-directionality and the two pretraining tasks (Masked LM and Next Sentence Prediction) account for the majority of the empirical improvements.

With pre-training, bigger == better, without clear limits (so far).

Based on slides from Jacob Devlin

Tutorial

Transformer A

3.1 Encoder and Decoder Stacks

```python
class EncoderLayer(nn.Module):
    def __init__(self):
        super(EncoderLayer, self).__init__()
        #Recall : " Each layer has two sub-layers. The first is a multi-head
        #self-attention mechanism, and the second is a simple, positionwise
        #fully connected feed-forward network.
        self.enc_self_attn = MultiHeadAttention()
        self.pos_ffn = PoswiseFeedForwardNet()

    def forward(self, enc_inputs, enc_self_attn_mask):
        enc_outputs, attn = self.enc_self_attn(enc_inputs, enc_inputs, enc_inputs, enc_self_attn_mask) # enc_inputs to same Q,K,V
        enc_outputs = self.pos_ffn(enc_outputs) # enc_outputs: [batch_size x len_q x d_model]
        return enc_outputs, attn
```

```python
class Encoder(nn.Module):
    def __init__(self):
        super(Encoder, self).__init__()
        self.src_emb = nn.Embedding(src_vocab_size, d_model)
        self.pos_emb = nn.Embedding.from_pretrained(get_sinusoid_encoding_table(src_len+1, d_model),freeze=True)
        # Iterative creation of Encoder layers
        self.layers = nn.ModuleList([EncoderLayer() for _ in range(n_layers)])

    def forward(self, enc_inputs): # enc_inputs : [batch_size x source_len]
        enc_outputs = self.src_emb(enc_inputs) + self.pos_emb(torch.LongTensor([[1,2,3,4,0]]))
        enc_self_attn_mask = get_attn_pad_mask(enc_inputs, enc_inputs)
        enc_self_attns = []
        for layer in self.layers:
            enc_outputs, enc_self_attn = layer(enc_outputs, enc_self_attn_mask)
            enc_self_attns.append(enc_self_attn)
        return enc_outputs, enc_self_attns
```

3.1 Encoder and Decoder Stacks

```
class DecoderLayer(nn.Module):
    def __init__(self):
        super(DecoderLayer, self).__init__()
        #Recall : In addition to the two sub-layers in each encoder layer, the decoder
        #inserts a third sub-layer, which performs multi-head attention over
        #the output of the encoder stack
        self.dec_self_attn = MultiHeadAttention()
        self.dec_enc_attn = MultiHeadAttention()
        self.pos_ffn = PoswiseFeedForwardNet()
```

```
class Decoder(nn.Module):
    def __init__(self):
        super(Decoder, self).__init__()
        self.tgt_emb = nn.Embedding(tgt_vocab_size, d_model)
        self.pos_emb = nn.Embedding.from_pretrained(get_sinusoid_encoding_table(tgt_len+1, d_model),freeze=True)
        # Iterative creation of Decoder layers
        self.layers = nn.ModuleList([DecoderLayer() for _ in range(n_layers)])

    def forward(self, dec_inputs, enc_inputs, enc_outputs): # dec_inputs : [batch_size x target_len]
        dec_outputs = self.tgt_emb(dec_inputs) + self.pos_emb(torch.LongTensor([[5,1,2,3,4]]))
        dec_self_attn_pad_mask = get_attn_pad_mask(dec_inputs, dec_inputs)
        dec_self_attn_subsequent_mask = get_attn_subsequent_mask(dec_inputs)
        dec_self_attn_mask = torch.gt((dec_self_attn_pad_mask + dec_self_attn_subsequent_mask), 0)
        dec_enc_attn_mask = get_attn_pad_mask(dec_inputs, enc_inputs)

        dec_self_attns, dec_enc_attns = [], []
        for layer in self.layers:
            dec_outputs, dec_self_attn, dec_enc_attn = layer(dec_outputs, enc_outputs, dec_self_attn_mask, dec_enc_attn_mask)
            dec_self_attns.append(dec_self_attn)
            dec_enc_attns.append(dec_enc_attn)
        return dec_outputs, dec_self_attns, dec_enc_attns
```

3.2 Attention
3.2.1 Scaled Dot-Product Attention

```
"""
3.2.1 Scaled Dot-Product Attention:
    Recall: Attention(Q,K,V)=softmax(QK^T/dk^1/2)V

"""
class ScaledDotProductAttention(nn.Module):
    def __init__(self):
        super(ScaledDotProductAttention, self).__init__()

    def forward(self, Q, K, V, attn_mask):
        #(QK^T/dk^1/2)
        scores = torch.matmul(Q, K.transpose(-1, -2)) / np.sqrt(d_k) # scores : [batch_size x n_heads x len_q(=len_k) x len_k(=len_q)]
        scores.masked_fill_(attn_mask.type(torch.bool), -1e9) # Fills elements of self tensor with value where mask is one.
        #Softmax
        attn = nn.Softmax(dim=-1)(scores)
        #Softmax(QK^T/dk^1/2)V
        context = torch.matmul(attn, V)
        return context, attn
```

3.2 Attention
3.2.2 Multi-Head Attention

```
"""
3.2.2 Multi-Head Attention:
    Recall: MultiHead(Q,K,,V)= Concat(Attention(QWi^Q,KWi^K,VWi^V),...,Attention(QWh^Q,KWh^K,VWh^V))W^O

"""
class MultiHeadAttention(nn.Module):
    def __init__(self):
        super(MultiHeadAttention, self).__init__()
        self.W_Q = nn.Linear(d_model, d_k * n_heads)
        self.W_K = nn.Linear(d_model, d_k * n_heads)
        self.W_V = nn.Linear(d_model, d_v * n_heads)
    def forward(self, Q, K, V, attn_mask):
        # q: [batch_size x len_q x d_model], k: [batch_size x len_k x d_model], v: [batch_size x len_k x d_model]
        residual, batch_size = Q, Q.size(0)
        # (B, S, D) -proj-> (B, S, D) -split-> (B, S, H, W) -trans-> (B, H, S, W)
        q_s = self.W_Q(Q).view(batch_size, -1, n_heads, d_k).transpose(1,2)  # q_s: [batch_size x n_heads x len_q x d_k]
        k_s = self.W_K(K).view(batch_size, -1, n_heads, d_k).transpose(1,2)  # k_s: [batch_size x n_heads x len_k x d_k]
        v_s = self.W_V(V).view(batch_size, -1, n_heads, d_v).transpose(1,2)  # v_s: [batch_size x n_heads x len_k x d_v]

        attn_mask = attn_mask.unsqueeze(1).repeat(1, n_heads, 1, 1) # attn_mask : [batch_size x n_heads x len_q x len_k]

        # context: [batch_size x n_heads x len_q x d_v], attn: [batch_size x n_heads x len_q(=len_k) x len_k(=len_q)]
        context, attn = ScaledDotProductAttention()(q_s, k_s, v_s, attn_mask)
        context = context.transpose(1, 2).contiguous().view(batch_size, -1, n_heads * d_v) # context: [batch_size x len_q x n_heads * d_v]
        output = nn.Linear(n_heads * d_v, d_model)(context)
        return nn.LayerNorm(d_model)(output + residual), attn # output: [batch_size x len_q x d_model]
```

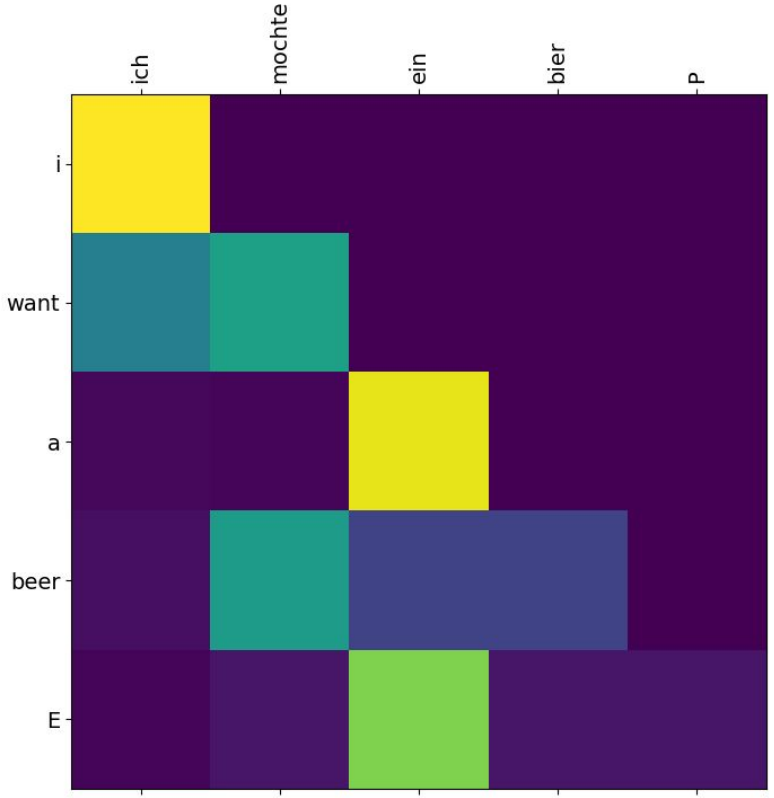
3.3 Position-wise Feed-Forward Networks

```
"""
3.3 Position-wise Feed-Forward Networks:
    Recall: FFN(x) = max(0,xW1+b1)W2+b2

"""
class PoswiseFeedForwardNet(nn.Module):
    def __init__(self):
        # In the paper fully connected layers are used.
        # But recall: "Another way of describing this is as two convolutions with kernel size 1."
        super(PoswiseFeedForwardNet, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=d_model, out_channels=d_ff, kernel_size=1)
        self.conv2 = nn.Conv1d(in_channels=d_ff, out_channels=d_model, kernel_size=1)

    def forward(self, inputs):
        residual = inputs # inputs : [batch_size, len_q, d_model]
        output = nn.ReLU()(self.conv1(inputs.transpose(1, 2)))
        output = self.conv2(output).transpose(1, 2)
        return nn.LayerNorm(d_model)(output + residual)
```

3.5 Positional Encoding

```
"""
3.5. Positional Encoding:
    Recall: PE(pos,2i) = sin(pos/10000^(2i/d_model))
            PE(pos,2i+1) = cos(pos/10000^(2i/d_model))
"""
def get_sinusoid_encoding_table(n_position, d_model):
    def cal_angle(position, hid_idx):
        return position / np.power(10000, 2 * (hid_idx // 2) / d_model)
    def get_posi_angle_vec(position):
        return [cal_angle(position, hid_j) for hid_j in range(d_model)]

    sinusoid_table = np.array([get_posi_angle_vec(pos_i) for pos_i in range(n_position)])
    sinusoid_table[:, 0::2] = np.sin(sinusoid_table[:, 0::2])  # dim 2i
    sinusoid_table[:, 1::2] = np.cos(sinusoid_table[:, 1::2])  # dim 2i+1
    return torch.FloatTensor(sinusoid_table)
```

Attention plot



Transformer B

Layer normalization

```
class LayerNormalization(nn.Module):

    def __init__(self, features_count, epsilon=1e-6):
        super(LayerNormalization, self).__init__()

        self.gain = nn.Parameter(torch.ones(features_count))
        self.bias = nn.Parameter(torch.zeros(features_count))
        self.epsilon = epsilon

    def forward(self, x):

        mean = x.mean(dim=-1, keepdim=True)
        std = x.std(dim=-1, keepdim=True)

        return self.gain * (x - mean) / (std + self.epsilon) + self.bias
```

Dropouts

```
class PointwiseFeedForwardNetwork(nn.Module):

    def __init__(self, d_ff, d_model, dropout_prob):
        super(PointwiseFeedForwardNetwork, self).__init__()

        self.feed_forward = nn.Sequential(
            nn.Linear(d_model, d_ff),
            nn.Dropout(dropout_prob),
            nn.ReLU(),
            nn.Linear(d_ff, d_model),
            nn.Dropout(dropout_prob),
        )

    def forward(self, x):
        """

        Args:
             x: (batch_size, seq_len, d_model)
        """
        return self.feed_forward(x)
```

Label Smoothing

```
class LabelSmoothingLoss(nn.Module):
    """
    With label smoothing,
    KL-divergence between q_{smoothed ground truth prob.}(w)
    and p_{prob. computed by model}(w) is minimized.
    """
    def __init__(self, label_smoothing, vocabulary_size, pad_index=0):
        assert 0.0 < label_smoothing <= 1.0

        super(LabelSmoothingLoss, self).__init__()

        self.pad_index = pad_index
        self.log_softmax = nn.LogSoftmax(dim=-1)
        self.criterion = nn.KLDivLoss(reduction='sum')

        smoothing_value = label_smoothing / (vocabulary_size - 2)  # exclude pad and true label
        smoothed_targets = torch.full((vocabulary_size,), smoothing_value)
        smoothed_targets[self.pad_index] = 0
        self.register_buffer('smoothed_targets', smoothed_targets.unsqueeze(0))  # (1, vocabulary_size)

        self.confidence = 1.0 - label_smoothing
```

Label Smoothing

```
def forward(self, outputs, targets):
    """
    outputs (FloatTensor): (batch_size, seq_len, vocabulary_size)
    targets (LongTensor): (batch_size, seq_len)
    """
    batch_size, seq_len, vocabulary_size = outputs.size()

    outputs_log_softmax = self.log_softmax(outputs)
    outputs_flat = outputs_log_softmax.view(batch_size * seq_len, vocabulary_size)
    targets_flat = targets.view(batch_size * seq_len)

    smoothed_targets = self.smoothed_targets.repeat(targets_flat.size(0), 1)
    # smoothed_targets: (batch_size * seq_len, vocabulary_size)

    smoothed_targets.scatter_(1, targets_flat.unsqueeze(1), self.confidence)
    # smoothed_targets: (batch_size * seq_len, vocabulary_size)

    smoothed_targets.masked_fill_((targets_flat == self.pad_index).unsqueeze(1), 0)
    # masked_targets: (batch_size * seq_len, vocabulary_size)

    loss = self.criterion(outputs_flat, smoothed_targets)
    count = (targets != self.pad_index).sum().item()

    return loss, count
```

Parameters

```
parser.add_argument('--d_model', type=int, default=128)
parser.add_argument('--layers_count', type=int, default=1)
parser.add_argument('--heads_count', type=int, default=2)
parser.add_argument('--d_ff', type=int, default=128)
parser.add_argument('--dropout_prob', type=float, default=0.1)

parser.add_argument('--label_smoothing', type=float, default=0.1)
parser.add_argument('--optimizer', type=str, default="Adam", choices=["Noam", "Adam"])
parser.add_argument('--lr', type=float, default=0.001)

parser.add_argument('--batch_size', type=int, default=4)
parser.add_argument('--epochs', type=int, default=10)
```

Dataset

Source: It is not acceptable that , with the help of the national bureaucracies , Parliament. legislative prerogative should be made null and void by means of implementing provisions whose content , purpose and extent are not laid down in advance .

Target: Es geht nicht an , dass über Ausführungsbestimmungen , deren Inhalt , Zweck und Ausmaß vorher nicht bestimmt ist , zusammen mit den nationalen Bürokratien das Gesetzgebungsrecht des Europäischen Parlaments ausgehebelt wird .

Homework

1. Using **Transformer A**, write a **detailed pseudocode** of what is done in ***"Attention is all you need"*** and how the transformer works. At the same time, try training with different words (not just: "i want a beer") and analyze what the **attention plot** suggests about the training.

2. Using **Transformer B**, implement the **METEOR** metric and vary the parameters of **d_model, dff, h, dk** and **dv**. Then, discuss about the metric (METEOR compared to BLEU, **is it really better?**) and whether the behavior you obtain is similar to the presented in the paper (for example: **increasing the d_model** and **dff** parameters seems to **improve the BLEU** obtained, **why does it happen? did it happen to you?**). Have in mind that this code is using a different dataset from the one used in ***"Attention is all you need"***, so keep the distances.

3. **BERT Fine-Tuning for Sentence Classification** in Jupyter Notebook: report the Matthews correlation coefficient. Briefly discuss the algorithm.

https://colab.research.google.com/drive/1xj364W2xuP6eSLVTTrwSw856KwhTJ3HB