# CS772: Deep Learning for Natural Language Processing (DL-NLP)

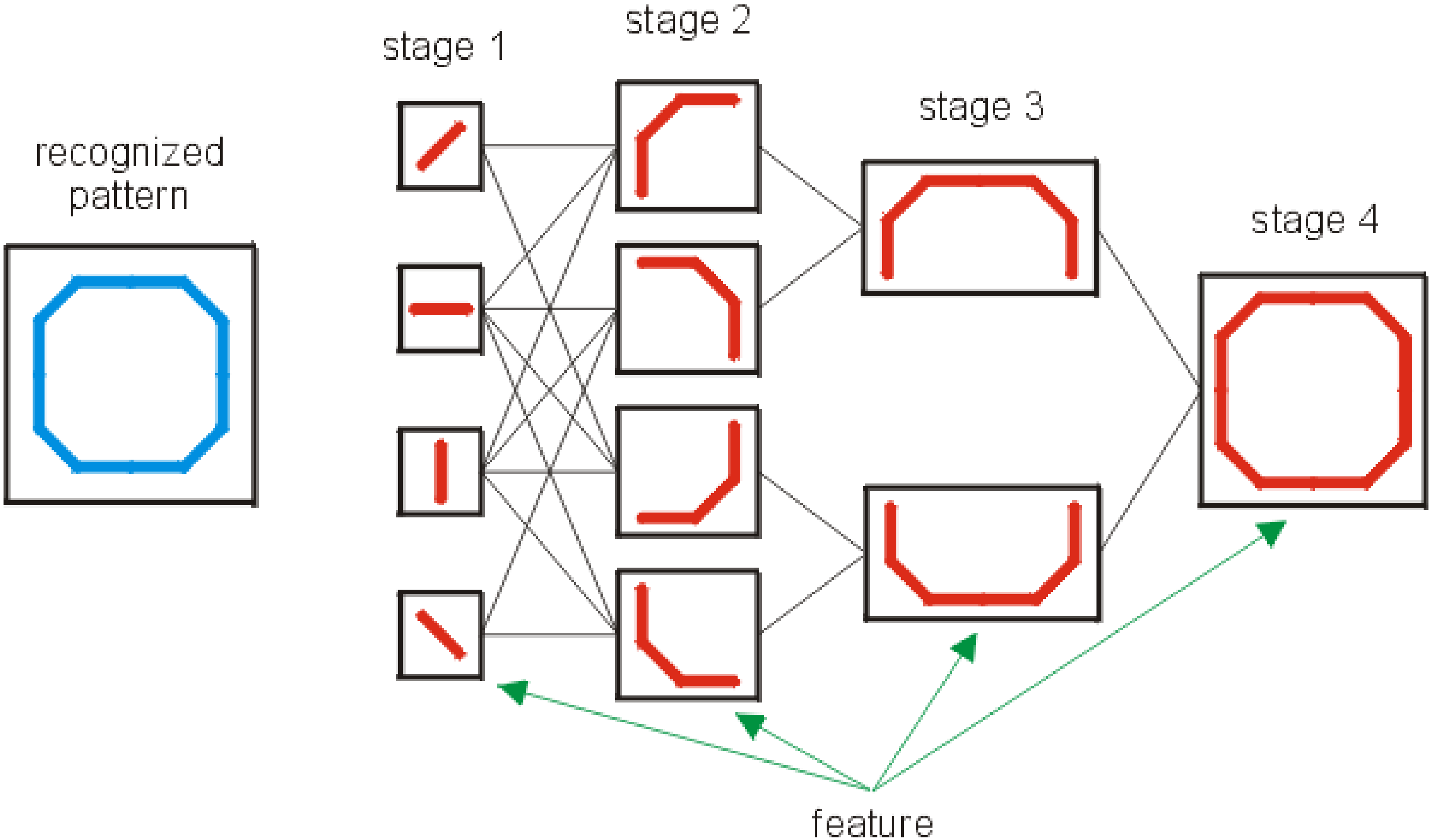
***Attention, Positional Embedding and Transfromer***

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

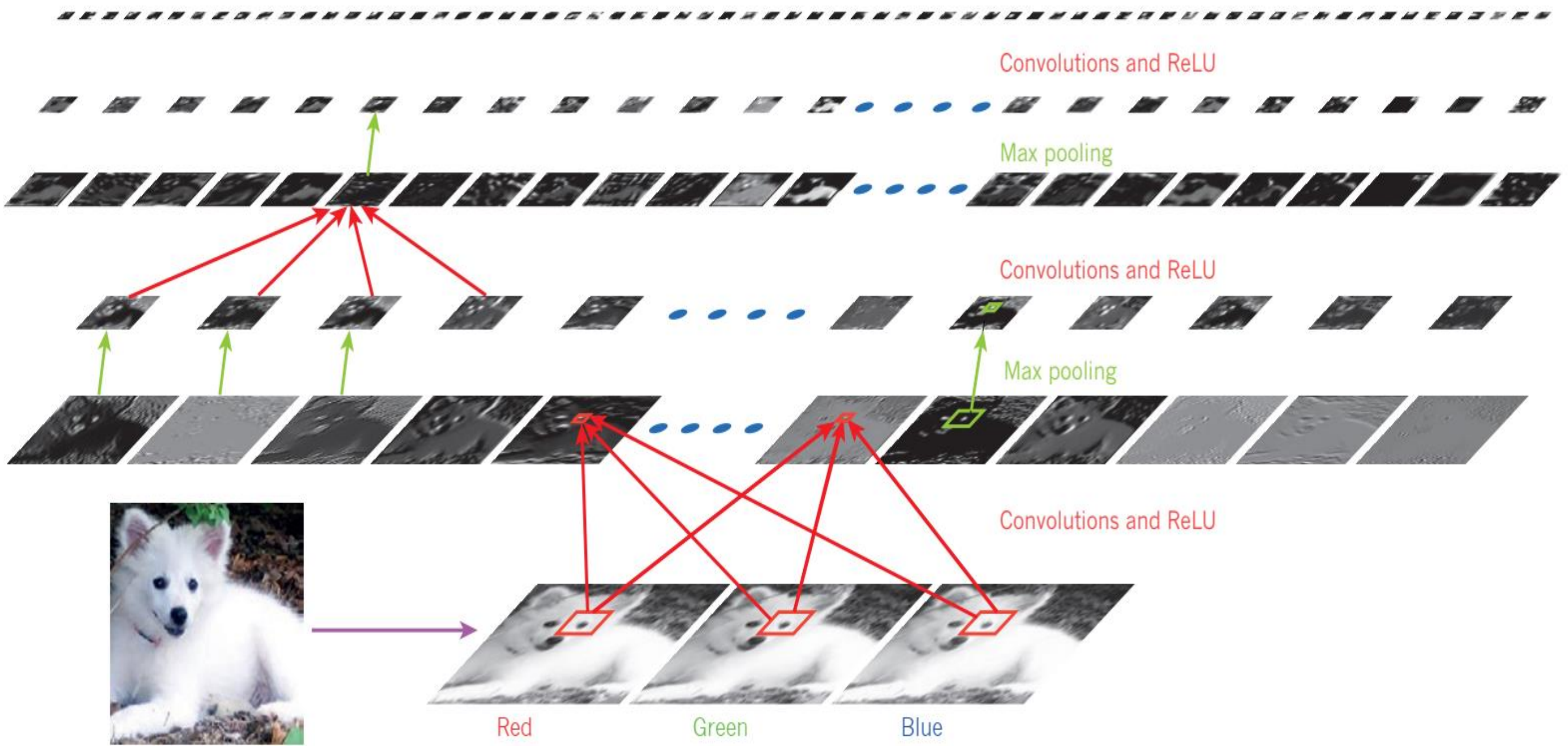
*Week 9 of 6$^{th}$ March, 2023*

# Re-cap

# CNN Genesis: Neocognitron (Fukusima, 1980)



# A typical ConvNet




Lecun, Bengio, Hinton, Nature, 2015

# Why CNN became a rage: image

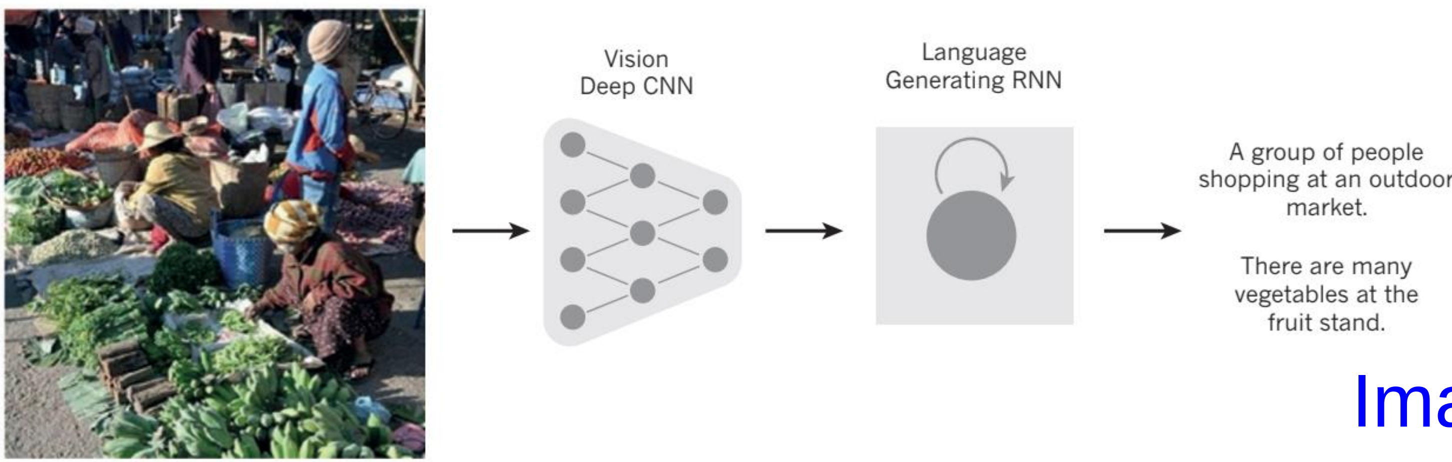



Image Captioning-1


A **stop** sign is on a road with a mountain in the background

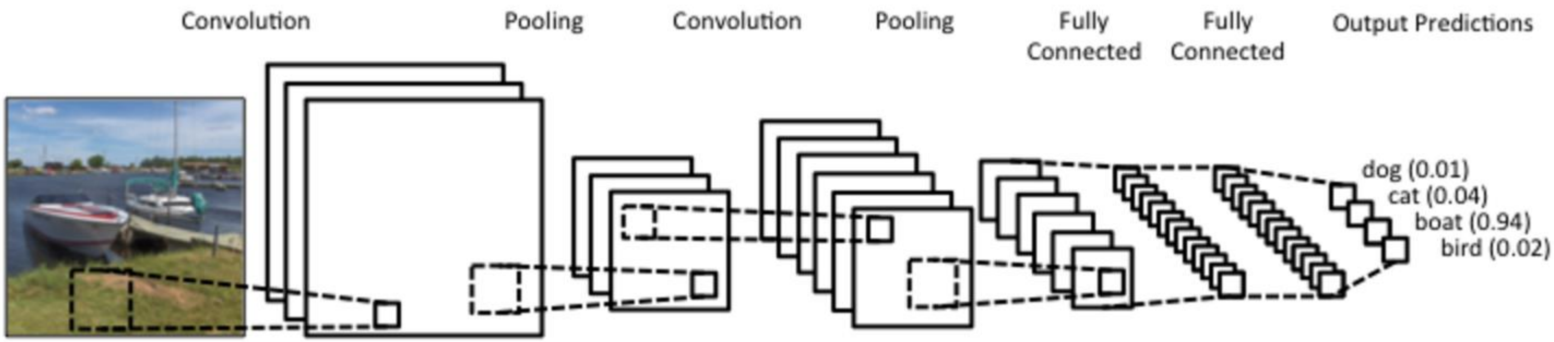
Image Captioning-2

# Role of ImageNet

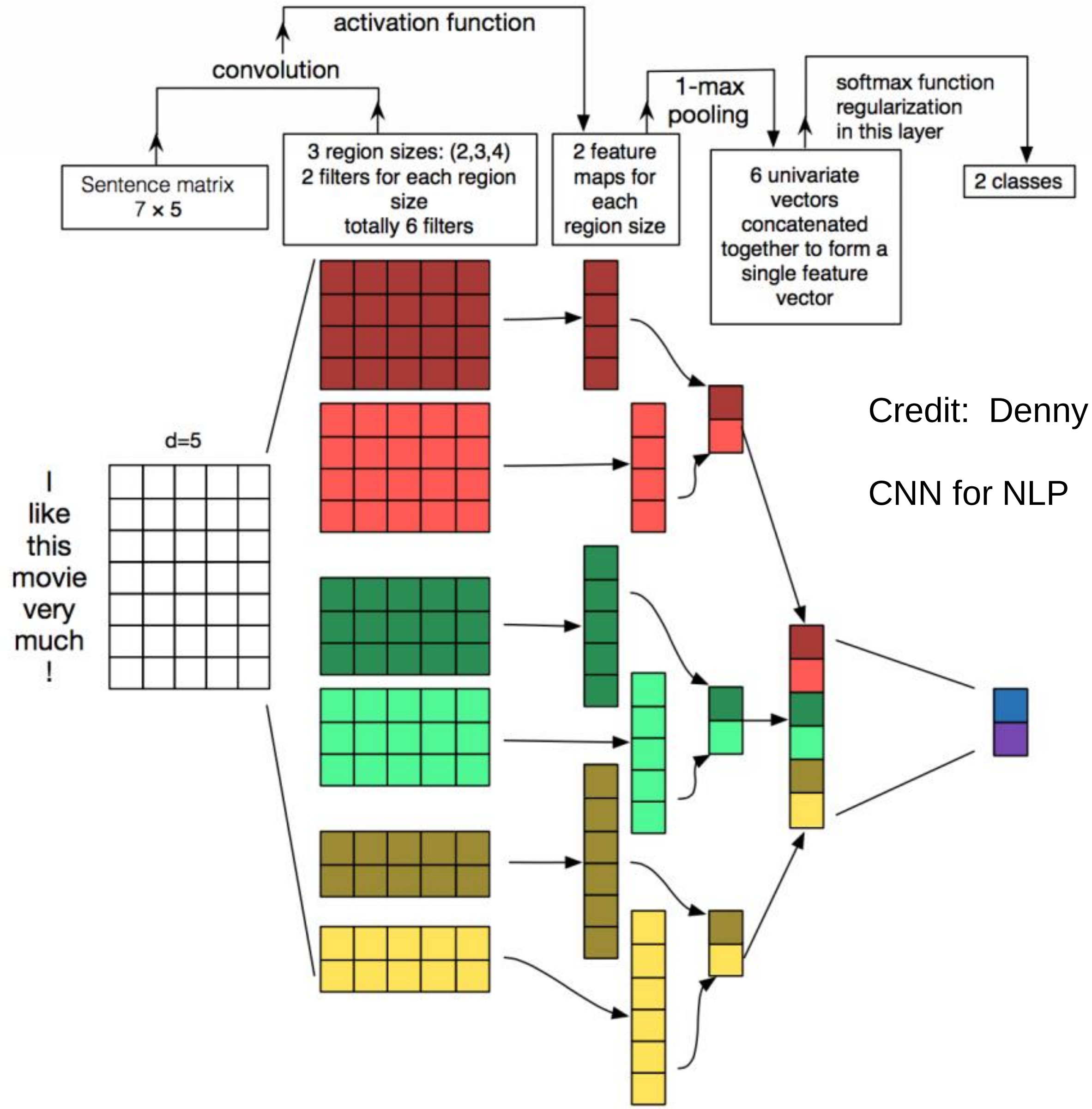
- Million images from the web
- 1,000 different classes
- Spectacular results!
- Almost halving the error rates of the best competing approaches1.

# Learning in CNN

- **Automatically learns the values of its filters**
- For example, in Image Classification learn to
  - detect edges from raw pixels in the first layer,
  - then use the edges to detect simple shapes in the second layer,
  - and then use these shapes to deter higher-level features, such as facial shapes in higher layers.
  - The last layer is then a classifier that uses these high-level features.



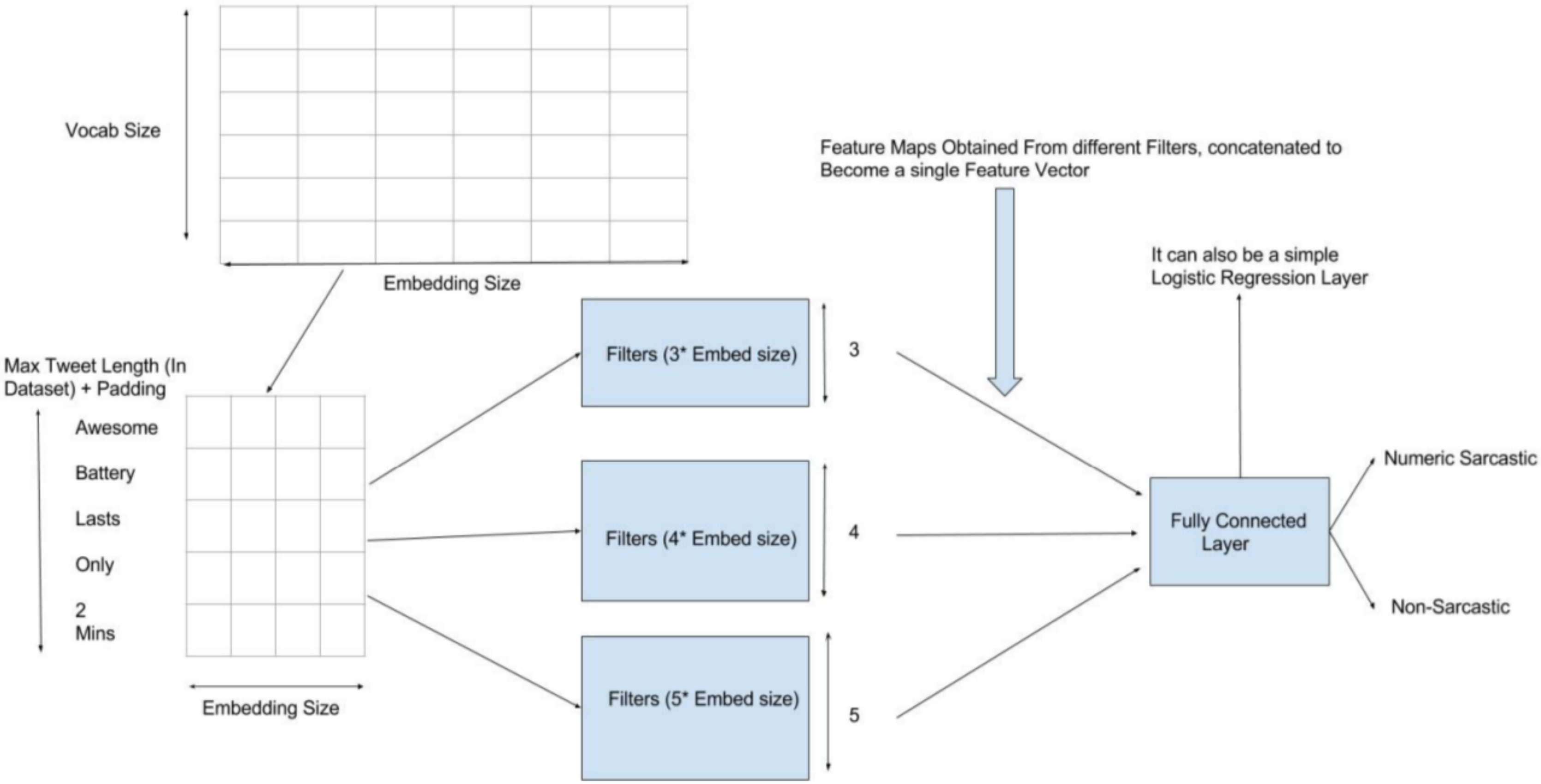

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/




Credit: Denny Britz

CNN for NLP

# CNN-FF for Sarcasm

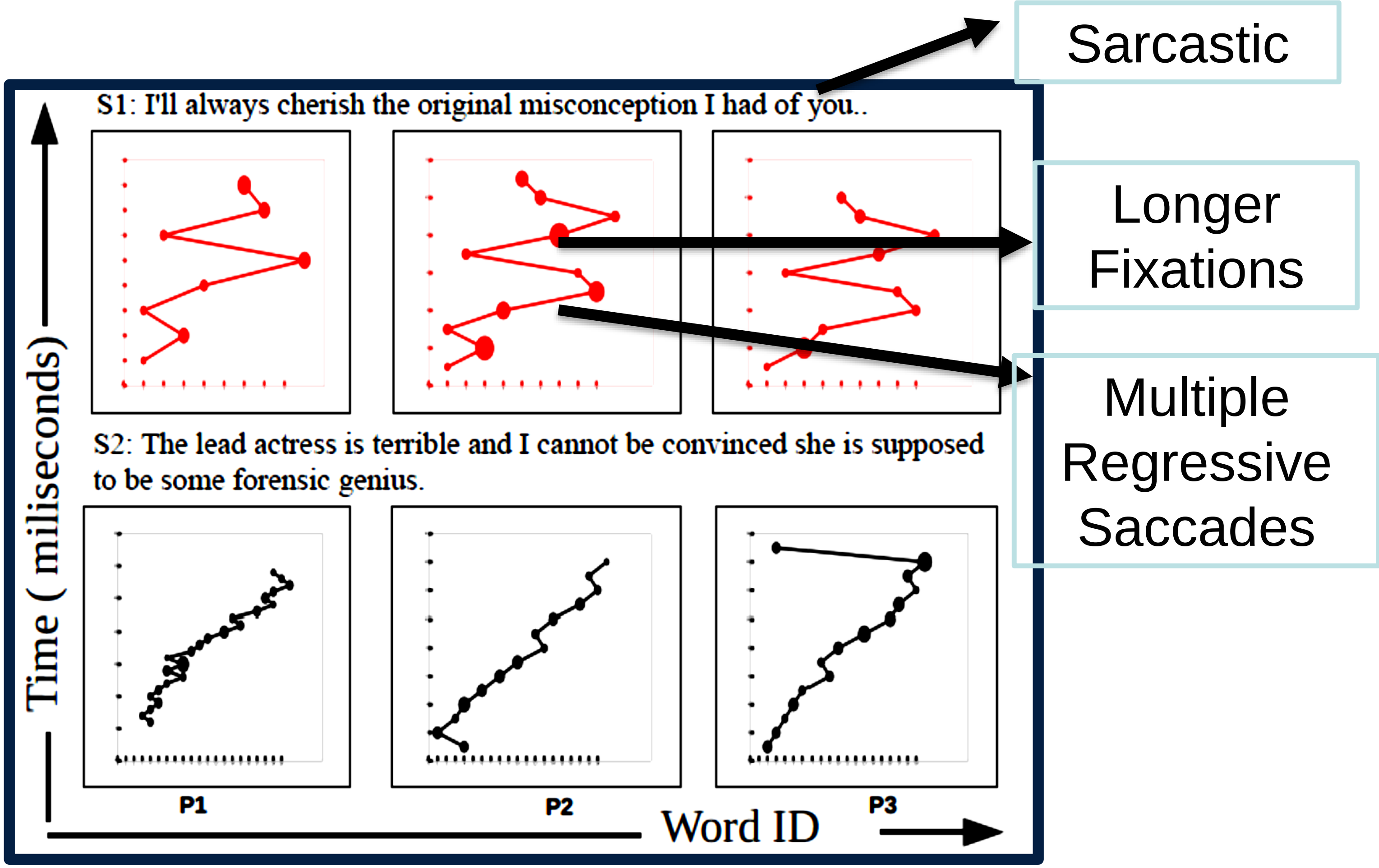


# Comparison of results (1: sarcastic, 0: non-sarcastic)

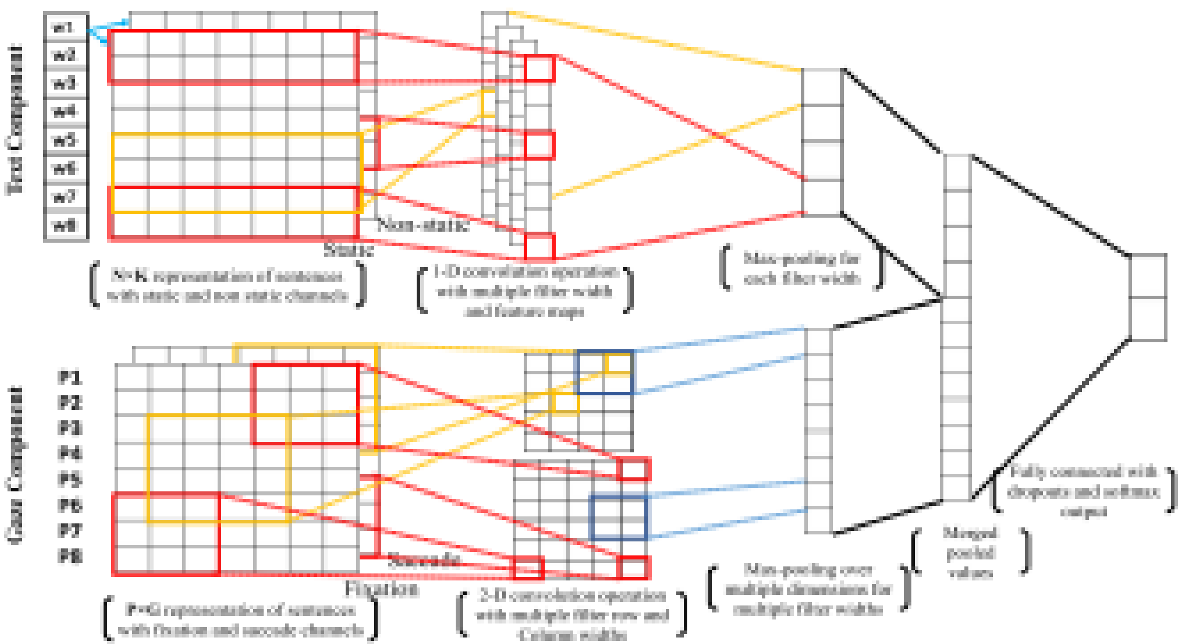
| Approaches | Precision | | | Recall | | | F-score | | |
|---|---|---|---|---|---|---|---|---|---|
| | **P(1)** | **P(0)** | **P(avg)** | **R(1)** | **R(0)** | **R(avg)** | **F(1)** | **F(0)** | **F(avg)** |
| **Past Approaches** | | | | | | | | | |
| Buschmeier et.al. | 0.19 | 0.98 | 0.84 | 0.99 | 0.07 | 0.24 | 0.32 | 0.13 | 0.16 |
| Liebrecht et.al. | 0.19 | 1.00 | 0.85 | 1.00 | 0.07 | 0.24 | 0.32 | 0.13 | 0.17 |
| Gonzalez et.al. | 0.19 | 0.96 | 0.83 | 0.99 | 0.06 | 0.23 | 0.32 | 0.12 | 0.15 |
| Joshi et.al. | 0.20 | 1.00 | 0.86 | 1.00 | 0.13 | 0.29 | 0.33 | 0.23 | **0.25** |
| **Rule-Based Approaches** | | | | | | | | | |
| Approach-1 | 0.53 | 0.87 | 0.81 | 0.39 | 0.92 | 0.83 | 0.45 | 0.90 | **0.82** |
| Approach-2 | 0.44 | 0.85 | 0.78 | 0.28 | 0.92 | 0.81 | 0.34 | 0.89 | 0.79 |
| **Machine-Learning Based Approaches** | | | | | | | | | |
| SVM | 0.50 | 0.95 | 0.87 | 0.80 | 0.82 | 0.82 | 0.61 | 0.88 | **0.83** |
| KNN | 0.36 | 0.94 | 0.84 | 0.81 | 0.68 | 0.70 | 0.50 | 0.79 | 0.74 |
| Random Forest | 0.47 | 0.93 | 0.85 | 0.74 | 0.81 | 0.80 | 0.57 | 0.87 | 0.82 |
| **Deep-Learning Based Approaches** | | | | | | | | | |
| **CNN-FF** | **0.88** | **0.94** | **0.93** | **0.71** | **0.98** | **0.93** | **0.79** | **0.96** | **0.93** |
| CNN-LSTM-FF | 0.82 | 0.94 | 0.92 | 0.72 | 0.96 | 0.92 | 0.77 | 0.95 | 0.92 |
| LSTM-FF | 0.76 | 0.93 | 0.90 | 0.68 | 0.95 | 0.90 | 0.72 | 0.94 | 0.90 |

back

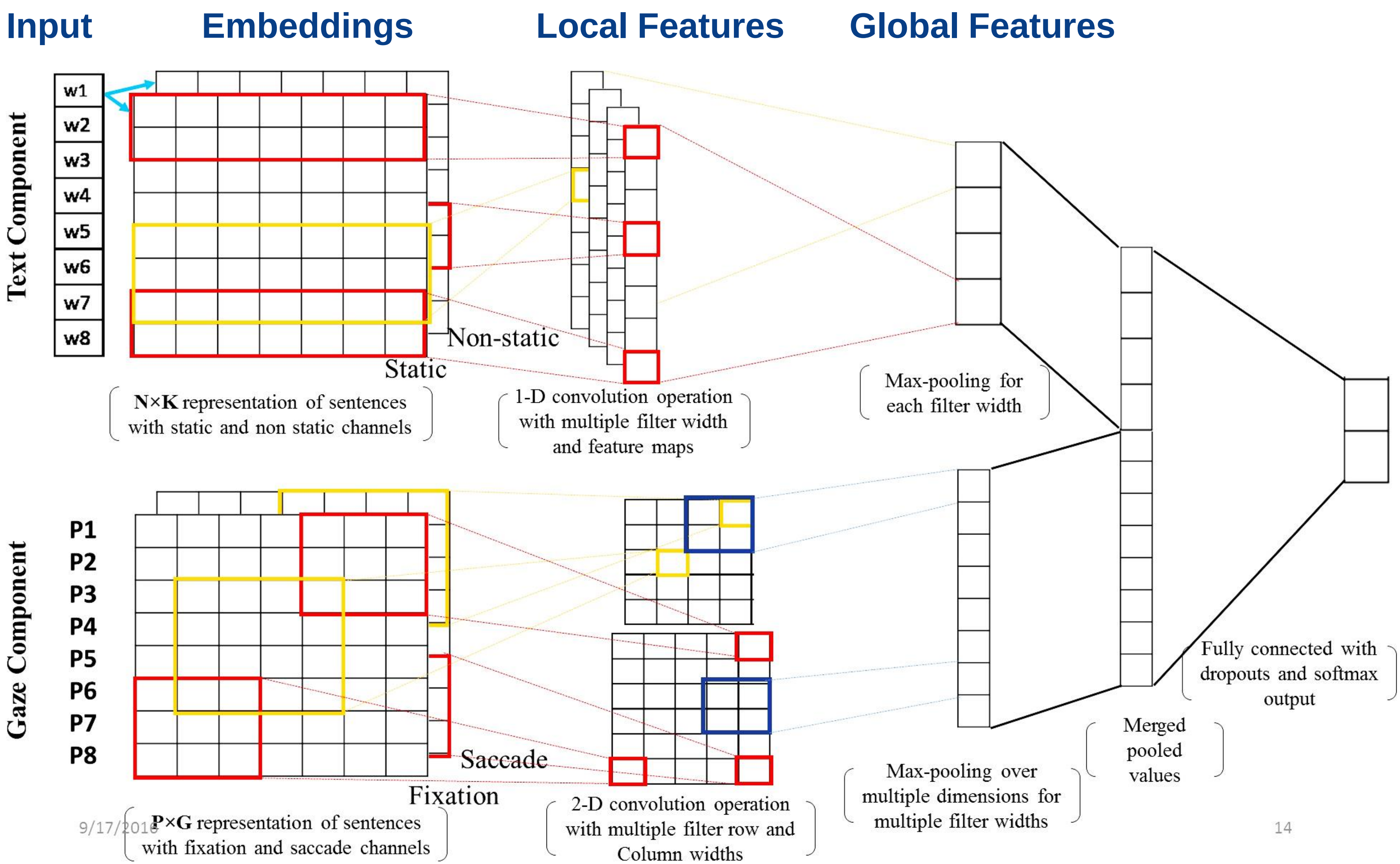
# Sentiment Annotation and Eye Movement



Abhijit Mishra, Kuntal Dey and Pushpak Bhattacharyya, *Learning Cognitive Features from Gaze Data for Sentiment and Sarcasm Classification Using Convolutional Neural Network*, **ACL 2017**, Vancouver, Canada, July 30-August 4, 2017.



# Neural Network Architecture



# Results – Sarcasm Detection

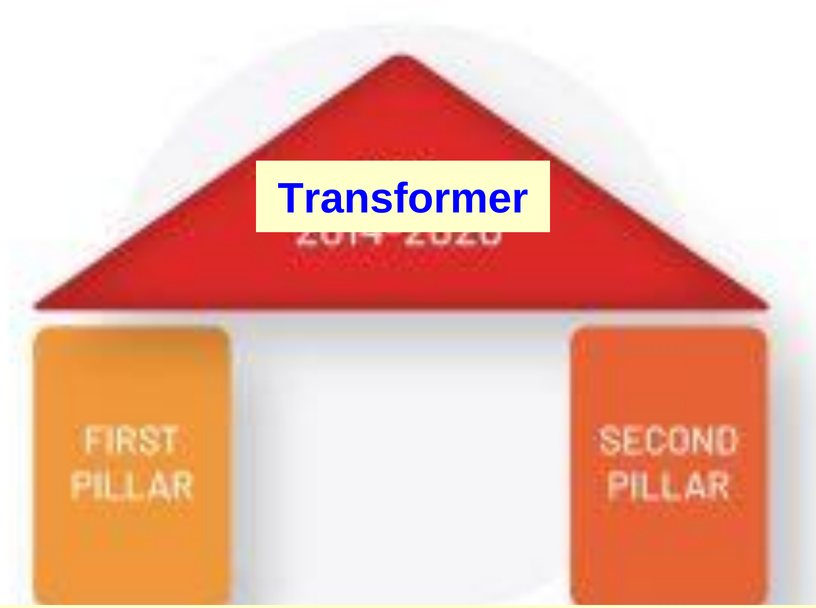
| | Configuration | P | R | F |
|---|---|---|---|---|
| Traditional systems based on textual features | Näive Bayes | 69.1 | 60.1 | 60.5 |
| | Multi-layered Perceptron | 69.7 | 70.4 | 69.9 |
| | SVM (Linear Kernel) | 72.1 | 71.9 | 72 |
| Systems by Riloff et al. (2013) | Text based (Ordered) | 49 | 46 | 47 |
| | Text + Gaze (Unordered) | 46 | 41 | 42 |
| System by Joshi et al. (2015) | Text based (best) | 70.7 | 69.8 | 64.2 |
| Systems by Mishra et al. (2016b) | Gaze based (Best) | 73 | 73.8 | 73.1 |
| | Text based (Best) | 72.1 | 71.9 | 72 |
| | Text + Gaze (Best) | 76.5 | 75.3 | 75.7 |
| CNN with only text input (Kim, 2014) | StaticText | 67.17 | 66.38 | 66.77 |
| | NonStaticText | 84.19 | **87.03** | 85.59 |
| | MultiChannelText | 84.28 | **87.03** | 85.63 |
| CNN with only gaze input | Fixation | 74.39 | 69.62 | 71.93 |
| | Saccade | 68.58 | 68.23 | 68.40 |
| | MultiChannelGaze | 67.93 | 67.72 | 67.82 |
| CNN with both text and gaze Input | StaticText + Fixation | 72.38 | 71.93 | 72.15 |
| | StaticText + Saccade | 73.12 | 72.14 | 72.63 |
| | StaticText + MultiChannelGaze | 71.41 | 71.03 | 71.22 |
| | NonStaticText + Fixation | **87.42** | 85.2 | 86.30 |
| | NonStaticText + Saccade | 84.84 | 82.68 | 83.75 |
| | NonStaticText + MultiChannelGaze | 84.98 | 82.79 | 83.87 |
| | MultiChannelText + Fixation | 87.03 | 86.92 | **86.97** |
| | MultiChannelText + Saccade | 81.98 | 81.08 | 81.53 |
| | MultiChannelText + MultiChannelGaze | 83.11 | 81.69 | 82.39 |

# Attention and Transformer

Arguably, the most important application-

**MACHINE TRANSLATION**

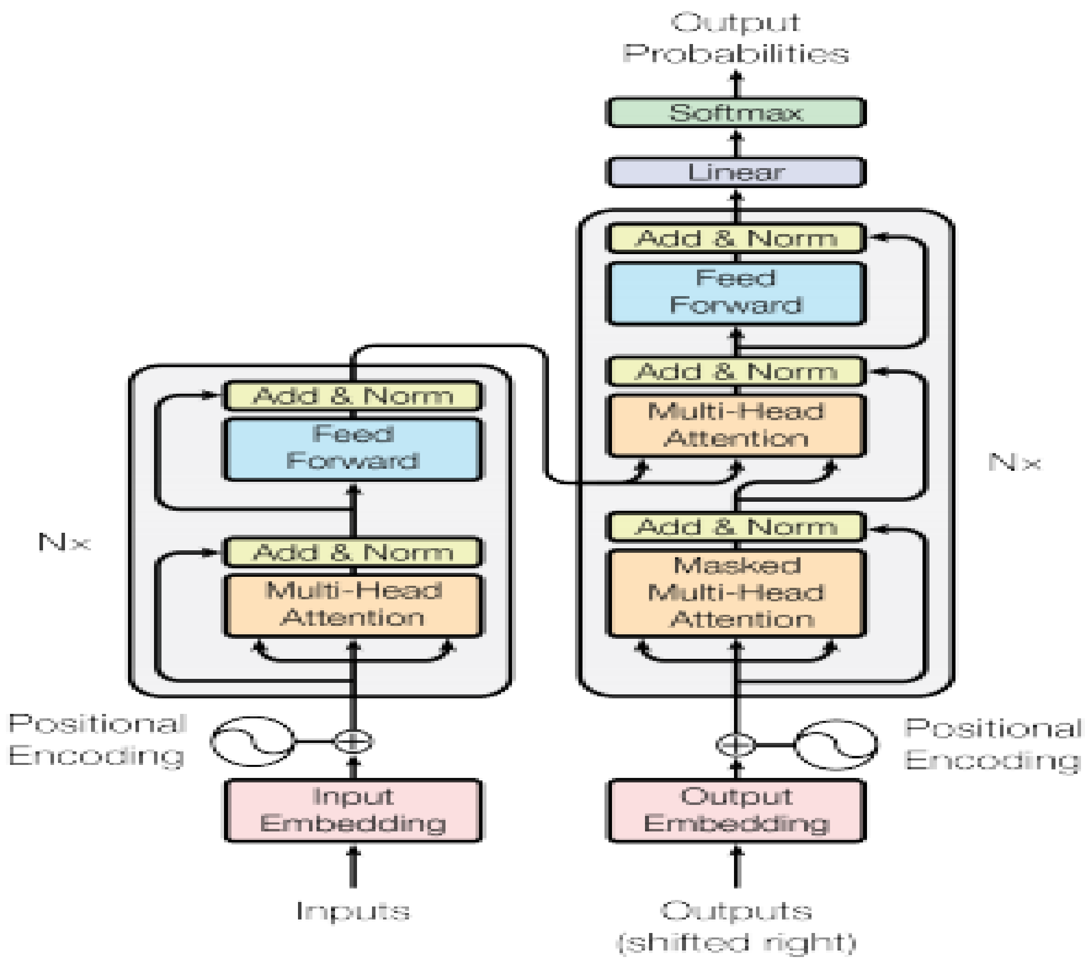
# Two Pillars of Transformer




**Attention + Positional Encoding**

**= Transformer**

# A classic diagram and a classic paper

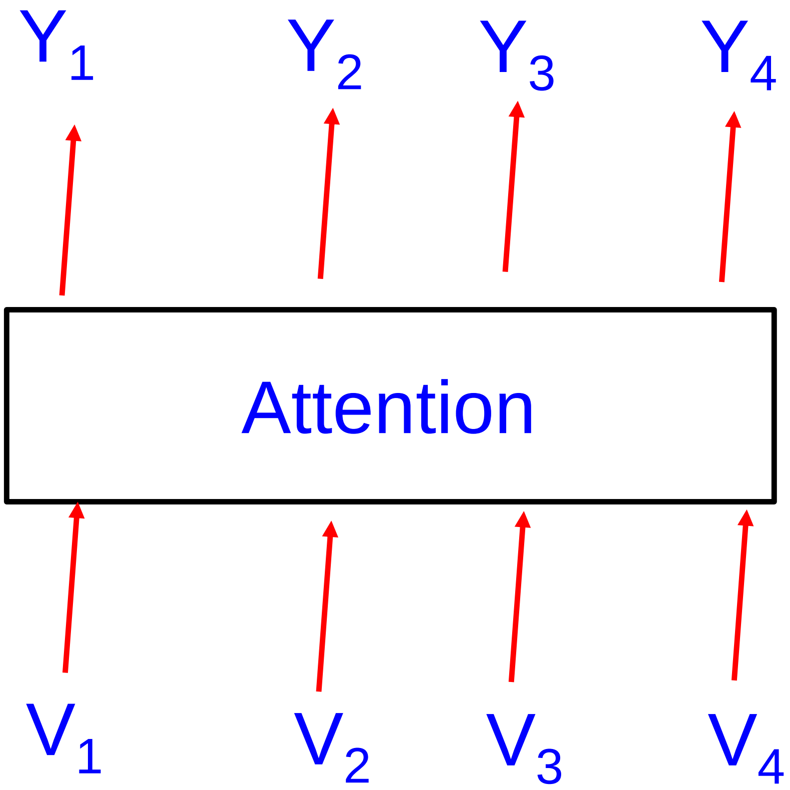



Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." NeurIPS (2017).

http://nlp.seas.harvard.edu/2018/04/03/attention.html
http://jalammar.github.io/illustrated-transformer/

# Attention: Self, Multi-headed, Cross

# Self Attention Block




Bank of the river

# Word Embedding and Contextual Word Embedding

- Consider the phrase "*bank of the river*"
- Word embeddings of '*bank*', '*of*', '*the*', '*river*': $V_1$, $V_2$, $V_3$, $V_4$
- Now create a 'score' vector $S_i$ for each word vector
- $S_1$: ($V_1.V_1$, $V_1.V_2$, $V_1.V_3$, $V_1.V_4$)
- Similarly, $S_2$, $S_3$, $S_4$

# S-matrix

$$S = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix}$$

# S-scaled matrix

$$S-scaled = \frac{1}{\sqrt{d_k}} \times \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix}$$

# W-matrix

$$W = \begin{bmatrix} w_{11}\, w_{12}\, w_{13}\, w_{14} \\ w_{21}\, w_{22}\, w_{23}\, w_{24} \\ w_{31}\, w_{32}\, w_{33}\, w_{34} \\ w_{41}\, w_{42}\, w_{43}\, w_{44} \end{bmatrix}$$

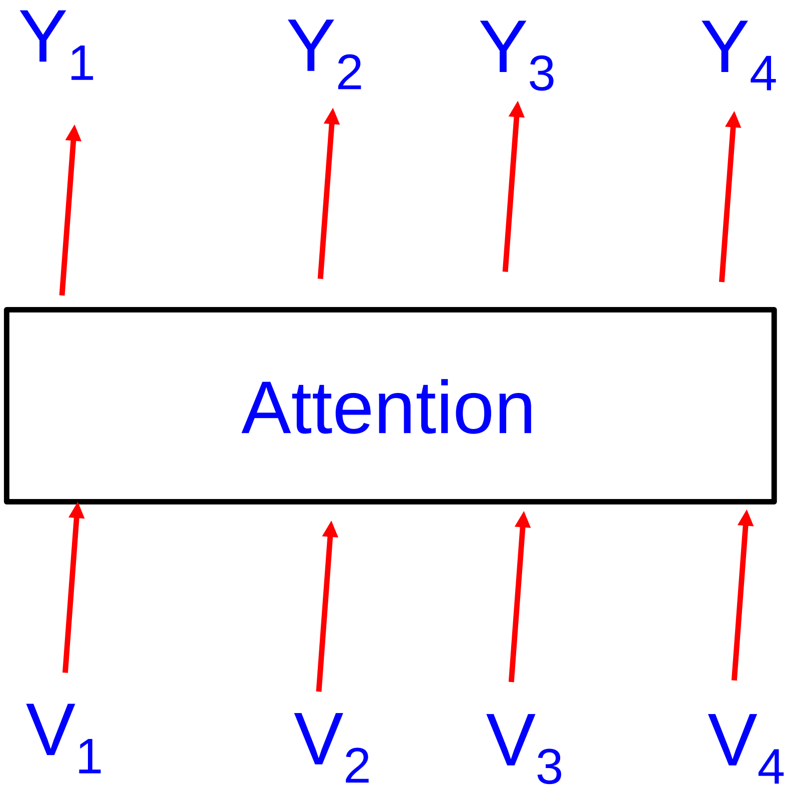
$$W_i - vector = soft\max\left( \frac{S_i - vector}{\sqrt{d_k}} \right)$$

# Y-matrix

$$Y = \begin{bmatrix} y_{11}\, y_{12}\, y_{13}\, y_{14} \\ y_{21}\, y_{22}\, y_{23}\, y_{24} \\ y_{31}\, y_{32}\, y_{33}\, y_{34} \\ y_{41}\, y_{42}\, y_{43}\, y_{44} \end{bmatrix}$$

$$Y_i - vector = w_{11}.V_1 + w_{12}.V_2 + w_{13}.V_3 + w_{14}.V_4$$

# Attention Block



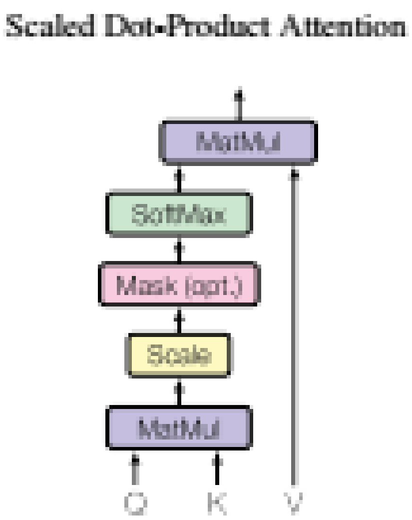

Bank of the river

# *Query, Key and Value*

$$attention(Q, K, V) = soft\max\left(\frac{Q \, . K^{T}}{\sqrt{d_k}}\right) . V$$

# *Query, Key and Value* with LEANABLE Parameter (1/2)

$$attention(Q,K,V) = soft\max\left(\frac{W^{Q}Q\ .W^{K}K^{T}}{\sqrt{d_k}}\right).W^{V}V$$




$W^Q$, $W^K$ and $W^V$ can be the weights of 3 linear layers of neurons which can be learnt by gradient descent

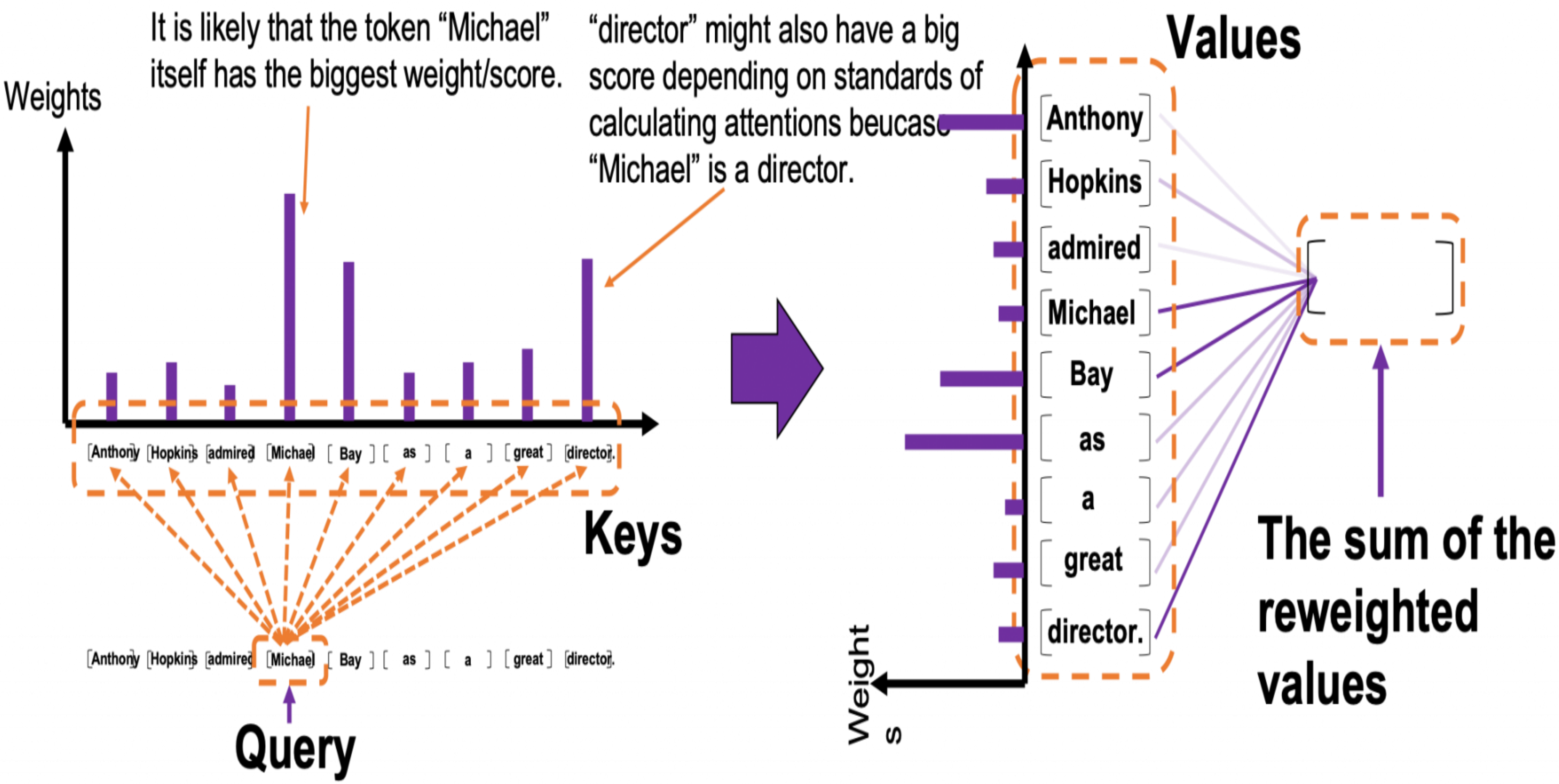
# Important observations on self attention

- In the input sequence, pairs of words differ in their strength of association
- For example for an adjective-noun combination, adjective's attention should be stronger for the noun than for other words in the sentence
- So the key questions are:
  - What to attend to
  - With how much attention to attend to

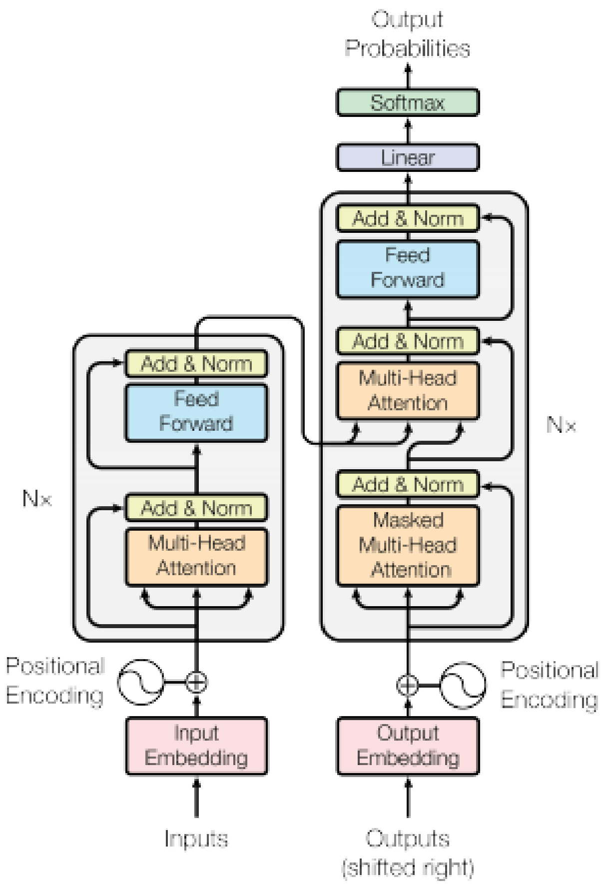
# Attention that is non-self

- When the decoder generates the output sequence, attention is a 2-part attention
- Each output token should attend to whatever token has been output before
- Additionally, it should attend to the tokens in the input sequence

# Fundamental concepts- “Attention”, “query”, “key”, “value”



# Putting it all together




Decoder layer also has a cross-attention layer

Decoder ➔ masking for future time-steps while computing self-attention

There are residual connections & layer-normalization between layers

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." NeurIPS (2017).

http://nlp.seas.harvard.edu/2018/04/03/attention.html
http://jalammar.github.io/illustrated-transformer/

Transformer has led to tremendous advances in MT

Encoder architectures like BERT based on Transformer have yielded large improvements in NLU tasks

Transformer models are the de-facto standard models for many NLP tasks

# Back to attention

# What is "Attention"

- **Attention** enhances the important parts of the input data and fades out the rest

- The network should devote more computing power on that small part of the data that matters

# Sentence-1

- Ram who is a good student and lives in London which is a large metro, will go to the University for higher studies.
- राम जो एक अच्छा छात्र है और लंदन में रहता है जो एक बड़ी मेट्रो है, उच्च अध्ययन के लिए विश्वविद्यालय जाएगा।

# Sentence-2

- Sita who is a good student and lives in London which is a large metro, will go to the University for higher studies.
- सीता जो एक अच्छी छात्रा है और लंदन में रहती है जो एक बड़ी मेट्रो है, उच्च अध्ययन के लिए विश्वविद्यालय जाएगी।

# Learning “Attention”

- Which part of the data is more important than others depends on the context

- Learned through training data by **gradient descent**

# Two kinds of Attention

- Dot Product Attention

- Multihead Attention

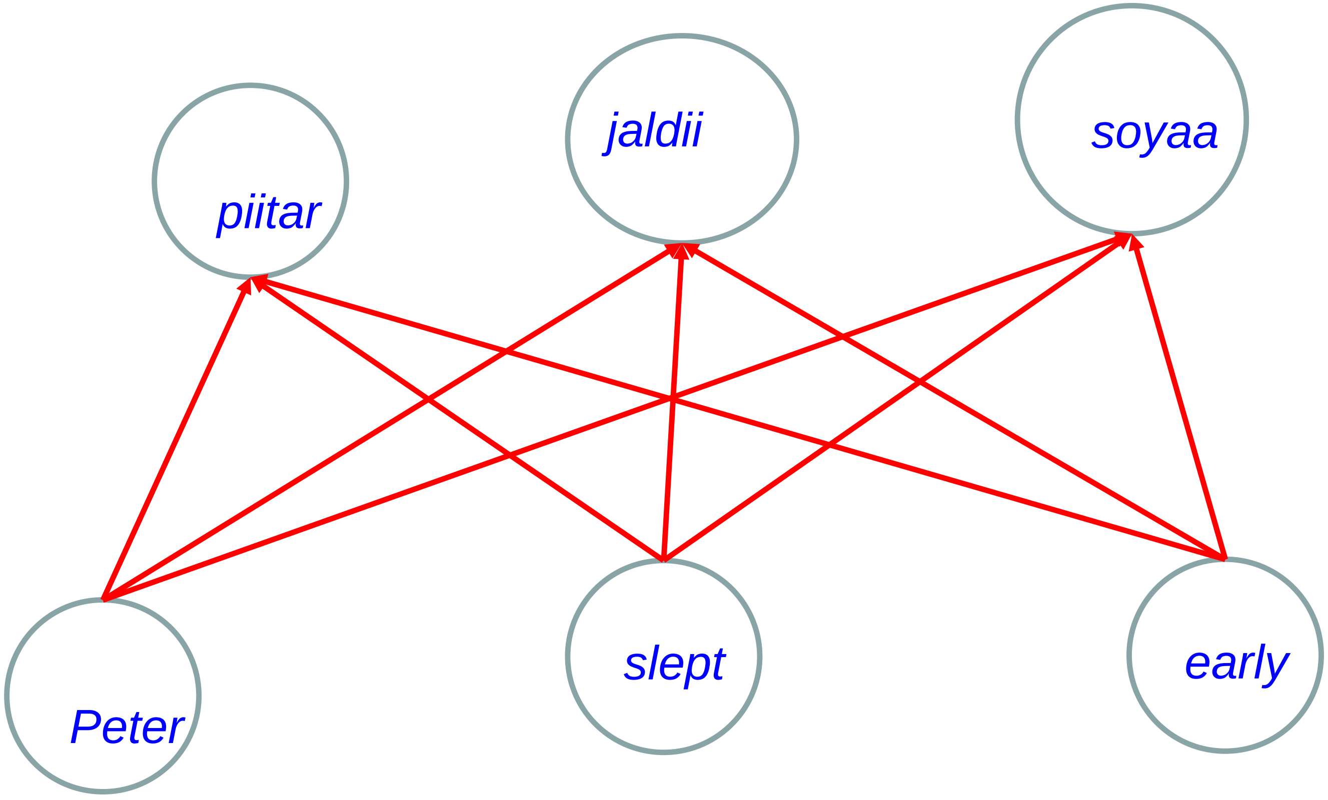
# Dependency Parse- Attention by Parsing

- root(ROOT-0, go-18)
- nsubj(go-18, Ram-1)
- nsubj(student-6, who-2)
- cop(student-6, is-3)
- det(student-6, a-4)
- amod(student-6, good-5)
- acl:relcl(Ram-1, student-6)
- cc(lives-8, and-7)
- conj(student-6, lives-8)
- case(London-10, in-9)
- nmod(lives-8, London-10)
- nsubj(metro-15, which-11)
- cop(metro-15, is-12)
- det(metro-15, a-13)
- amod(metro-15, large-14)
- acl:relcl(student-6, metro-15)
- aux(go-18, will-17)
- case(University-21, to-19)
- det(University-21, the-20)
- obl(go-18, University-21)
- case(studies-24, for-22)
- amod(studies-24, higher-23)
- nmod(University-21, studies-24)

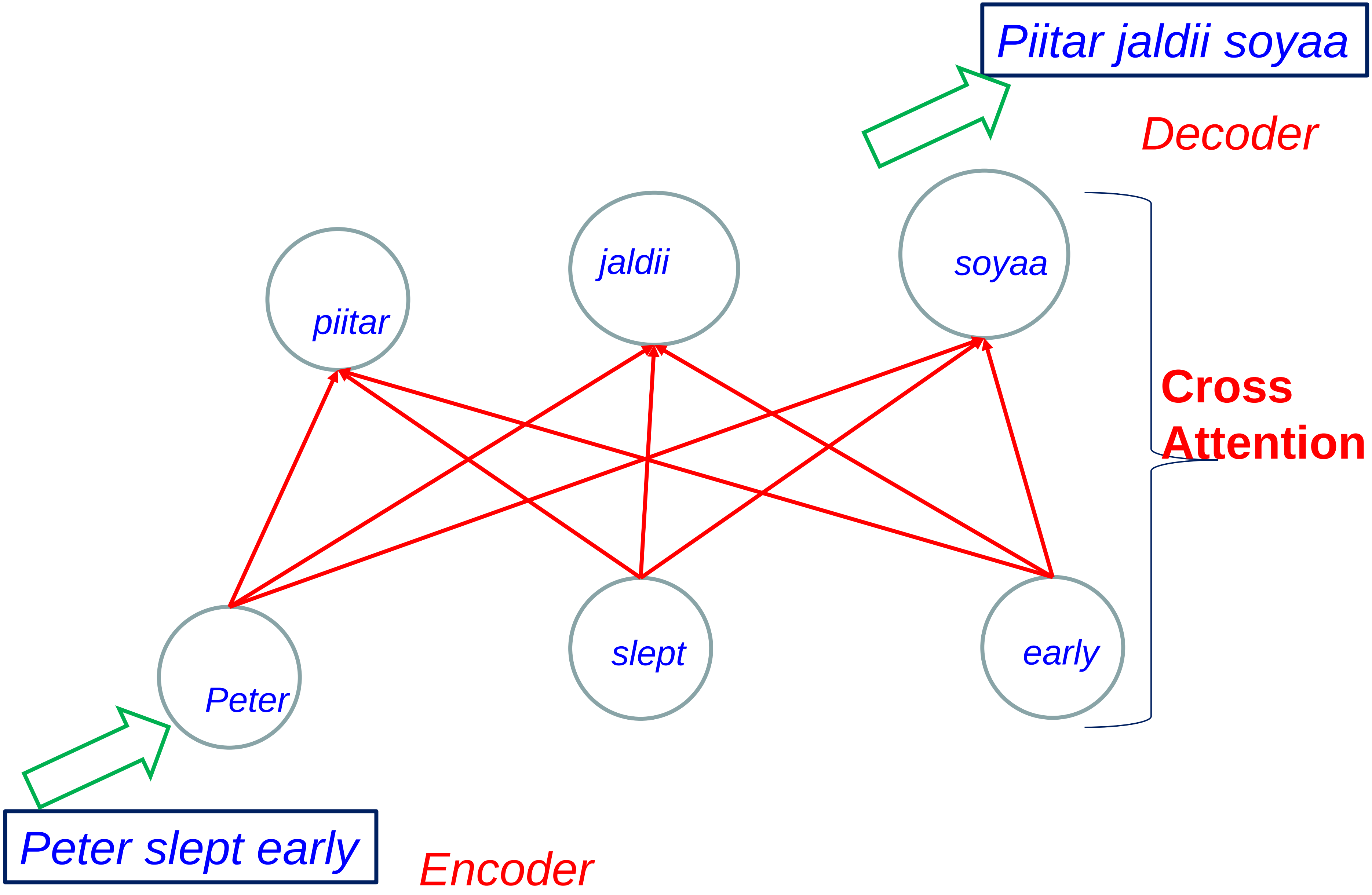
# Attention and Alignment

| Hindi (col) --><br><br>English (row) \|<br>\/ | PIITAR (पीटर) | | JALDII (जल्दी) | | SOYAA (सोया) |
|---|---|---|---|---|---|
| | | | | | |
| **PETER** | 1 | | 0 | | 0 |
| | | | | | |
| **SLEPT** | 0 | | 0 | | 1 |
| | | | | | |
| **EARLY** | 0 | | 1 | | 0 |

# FFNN for alignment: *Peter slept early → piitar jaldii soyaa*

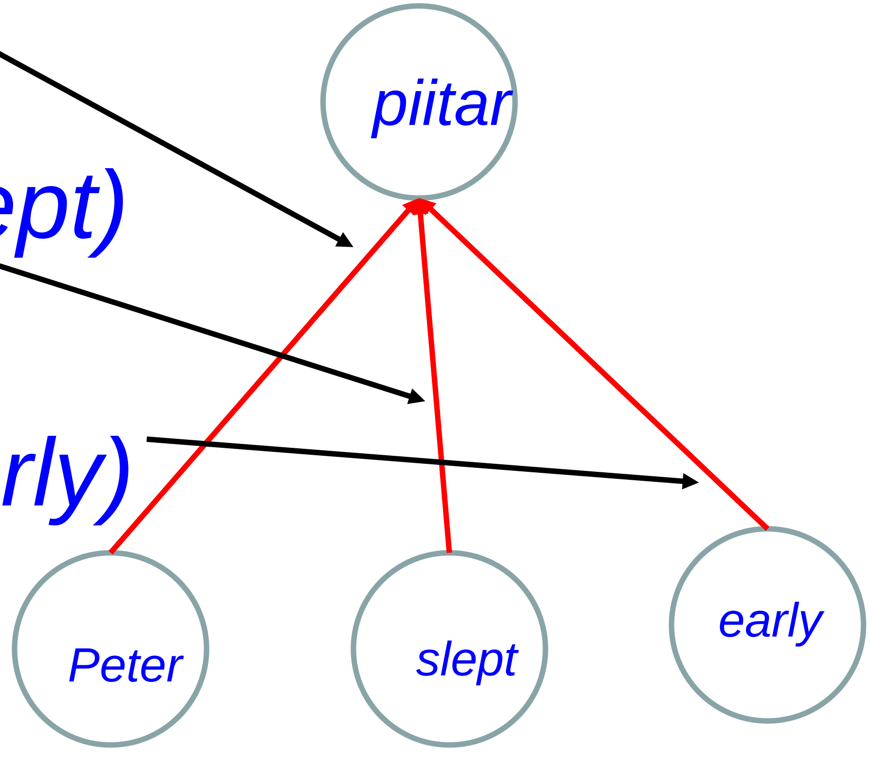


# Introduce Attention Layer between Encoder and Decoder



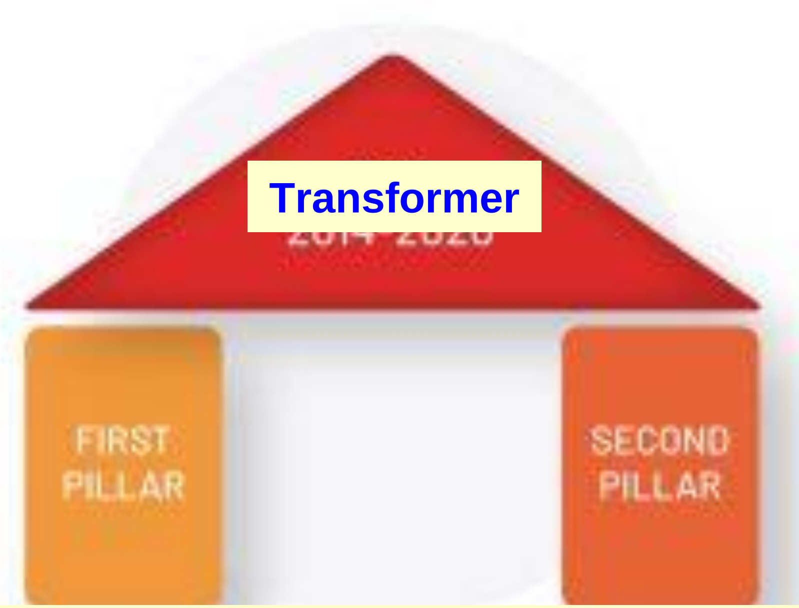
# How to learn the attention weights?

- Weight (*piitar*, *peter*)
- Weight (*piitar*, *slept*)
- Weight (*piitar*, *early*)



# Positional Encoding

# Two Pillars of Transformer




**Attention + Positional Encoding**

**= Transformer**

# Limitation of RNN

- Encoder-decoder RNN generates a sequence of hidden states $h_t$, $t$ varying from 0 to $L$, where $L$ is the sentence length.
- Each $h_t$ is a function of previous hidden state $h_{t-1}$ and the input at position $t$.
- So, to process the input at $t^{th}$ step, the encoder or decoder has to wait for $t-1$ steps.
- This sequential nature of RNN makes the training time very large.

# Inspiration from Shakespeare

- “All the world's a stage,/ And all the men and women merely players”- As You Like It- Shakespeare

- All the sentence’s a stage./And all the words and punctuations are merely players *“children saw a big lion in the zoo in the morning”*

- main verb: *saw*;
- who (agent): *children*
- what (object): *lion*
- where (locative): *zoo*
- when (temporal): *evening*

# Position Sensitivity: “Jack saw Jill” vs. “Jill saw Jack”

*IF*

*the main verb (MV) is transitive and in past tense*

*THEN*

*the NP to the left of MV should get the ‘ ne’ postposition mark*

*and*

*The NP to the right of MV should get the ‘ko’ postposition mark*

# Transformer's major contribution- Positional Encoding (1/2)

- Word positions as additional disambiguation signals.
- Words influence one another by virtue of their properties and positions
- Such influences manifest in translations as morphological transformations, lexical choices, pragmatic markers and so on.
- Tenet of ML-NLP: *with sufficient data all these mutual influences can be learnt*.

# Transformer’s $2^{nd}$ major contribution after attention- Positional Encoding (2/2)

- Positions are encoded as embeddings and positional embeddings are supplied along with input word embeddings.

- The training phase teaches the transformer to condition the output by paying attention to not only input words, but also their positions.

# Position Vector components

Let the $k^{th}$ component of the $t^{th}$ position vector be denoted as *pos(t,k)*, *k* varying from 0 to *d-1, d* being the dimension of the PE vector. Then for even and odd positions (I varies from 0 to d/2-1):

$$pos(t,2i) = \sin\left(\frac{1}{10000^{\frac{2i}{d}}}t\right)$$

$$pos(t,2i+1) = \cos\left(\frac{1}{10000^{\frac{2i}{d}}}t\right)$$

# Challenges in designing PEs

- Cannot append decimal integers as position values; words later in the sentence will dominate, by the force of their positions being large integers

- Cannot normalize too: Word relations changing with the length of sentences- linguistically untenable

- "*Oh, what a beautiful day!!*"- which expresses (i) delight, (ii) the nature of the '*day*' being '*beautiful*', (iii) '*Oh*', being an exclamatory prefix to the rest of the phrase and so on, should be invariant with respect to the sentence length

# Binary values also will not do!

- *0*s will contribute nothing, and *1*s will influence completely.
- Such *black-and-white* (0-1) hard decisions go against the grain of NLP whose other name is ambiguity.
- A language object represented by a vector must allow *soft choices* in its components, preferably represented by values in the closed range [0,1].

# Criteria PEs should satisfy

- Should be *added* component by component to the word vector.
- Components should range from 0 to 1, both included.
- Components should be periodic, since they represent consecutive integers.
- Ingenious on the part of the creators of transformers to spot that *sine* and *cosine* functions meet the above requirements.

# Position Vector components (reminding)

Let the $k^{th}$ component of the $t^{th}$ position vector be denoted as *pos(t,k)*, *k* varying from 0 to *d-1, d* being the dimension of the PE vector. Then for even and odd positions (I varies from 0 to d/2-1):

$$pos(t,2i) = \sin\left(\frac{1}{10000^{\frac{2i}{d}}} t\right)$$

$$pos(t,2i+1) = \cos\left(\frac{1}{10000^{\frac{2i}{d}}} t\right)$$

# Example: “Jack saw Jill” (1/2)

Three positions indexed as 0, 1 and 2.

Assume word vector dimension *d* to be *4*

Assume the frequency to be $1/(10^{2i/d})$

*i varies from 0 to (4/2-1)=1*

Then (cntd. next slide)

# Example: “Jack saw Jill” (2/2)

$$pos\_vector('Jack') = < pos(0,0), pos(0,1), \cancel{pos(0,0)}, pos(0,2), pos(0,3) >$$

$$= < \sin(0), \cos(0), \sin(0), \cos(0) >$$

$$= < 0, 1, 0, 1 >$$

$$pos\_vector('saw') = < pos(1,0), pos(1,1), pos(1,2), pos(1,3) >$$

$$= \left\langle \sin\left(\frac{1}{10^{\frac{2 \cdot 0}{4}}}\right), \cos\left(\frac{1}{10^{\frac{2 \cdot 0}{4}}}\right), \sin\left(\frac{1}{10^{\frac{2 \cdot 1}{4}}}\right), \cos\left(\frac{1}{10^{\frac{2 \cdot 1}{4}}}\right) \right\rangle$$

$$= \left\langle \sin(1), \cos(1), \sin\left(\frac{1}{10^{0.5}}\right), \cos\left(\frac{1}{10^{0.5}}\right) \right\rangle$$

$$pos\_vector('Jill') = < pos(2,0), pos(2,1), pos(2,2), pos(2,3) >$$

$$= < \left\langle \sin(2), \cos(2), \sin\left(\frac{2}{10^{0.5}}\right), \cos\left(\frac{2}{10^{0.5}}\right) \right\rangle$$

# Machine Translation

# The tricky case of ‘have’ translation

- *Peter has a house*
- *Peter has a brother*
- *This hotel has a museum*

# The tricky case of 'have' translation

**English**

- *Peter has a house*
- *Peter has a brother*
- *This hotel has a museum*

**Marathi**

- पीटरकडे एक घर आहे/ piitar kade ek ghar aahe
- पीटरला एक भाऊ आहे/ piitar laa ek bhaauu aahe
- ह्या हॉटेलमध्ये एक संग्रहालय आहे/ hyaa hotel madhye ek saMgrahaalay aahe

# RBMT

*If*

syntactic subject is animate AND syntactic object is owned by subject

*Then*

“have” should translate to “kade … aahe”

*If*

syntactic subject is animate AND syntactic object denotes kinship with subject

*Then*

“have” should translate to “laa … aahe”

*If*

syntactic subject is inanimate

*Then*

“have” should translate to “madhye …  aahe”

# EBMT

*X have Y →*

*Xkade Y aahe /*

*Xlaa Y aahe /*

*Xmadhye Y aahe*

# SMT

- *has a house ←→ kade ek ghar aahe*
- *has a car ←→ kade ek gaadii aahe*
- *has a brother ←→ laa ek bhaau aahe*
- *has a sister ←→ laa ek bahiin aahe*
- *hotel has ←→ hotel madhye*
- *hospital has ←→ hospital madhye*

# SMT: new sentence

"This hospital has 100 beds"

- *n*-grams (*n=1, 2, 3, 4, 5*) like the following will be formed:
  - *"This", "hospital",… (unigrams)*
  - *"This hospital", "hospital has", "has 100",… (bigrams)*
  - *"This hospital has", "hospital has 100", … (trigrams)*

*DECODING !!!*

## Why is MT difficult?

# Language divergence

# Why is MT difficult: Language Divergence

- Languages have different ways of expressing meaning
  - Lexico-Semantic Divergence
  - Structural Divergence

Our work on English-IL Language Divergence with illustrations from Hindi
*(Dave, Parikh, Bhattacharyya, Journal of MT, 2002)*

# Languages differ in expressing thoughts: Agglutination

Finnish*:* “istahtaisinkohan”

English: "I wonder if I should sit down for a while“

Analysis:

- ist + "sit", verb stem
- ahta + verb derivation morpheme, "to do something for a while"
- isi + conditional affix
- n + 1st person singular suffix
- ko + question particle
- han a particle for things like reminder (with declaratives) or "softening" (with questions and imperatives)

# Language Divergence Theory:

## *Lexico-Semantic Divergences* (few examples)

- Conflational divergence
  - F: vomir; E: to be sick
  - E: *stab;* H: *chure se maaranaa (knife-with hit)*
  - S: *Utrymningsplan;* E: *escape plan*

- Categorial divergence
  - Change is in POS category:
  - *The play is on_PREP (vs. The play is Sunday)*
  - *Khel chal_rahaa_haai_VM (vs. khel ravivaar ko haai)*

# Language Divergence Theory:

## *Structural Divergences*

- SVO→SOV
  - E: *Peter plays basketball*
  - H: *piitar basketball kheltaa haai*

- Head swapping divergence
  - E: *Prime Minister of India*
  - H: *bhaarat ke pradhaan mantrii (India-of Prime Minister)*

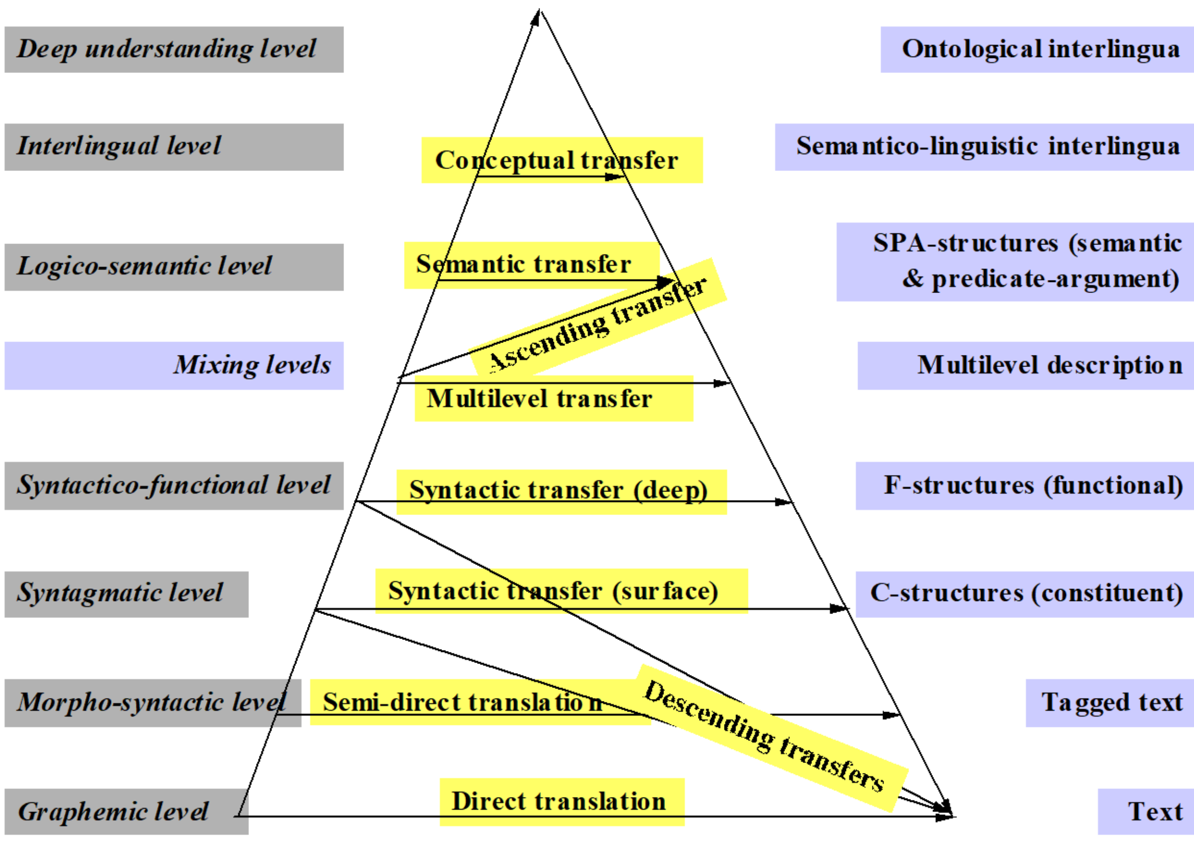
# Language Divergence Theory: *Syntactic Divergences* (few examples)

- Constituent Order divergence
  - E: *Singh, the PM of India, will address the nation today*
  - H: *bhaarat ke pradhaan mantrii, singh, … (India-of PM, Singh…)*
- Adjunction Divergence
  - E: *She will visit here in the summer*
  - H: *vah yahaa garmii meM aayegii (she here summer-in will come)*
- Preposition-Stranding divergence
  - E: *Who do you want to go with?*
  - H: *kisake saath aap jaanaa chaahate ho? (who with…)*

# Vauquois Triangle

# Kinds of MT Systems

*(point of entry from source to the target text)*



# Simplified Vauquois Triangle

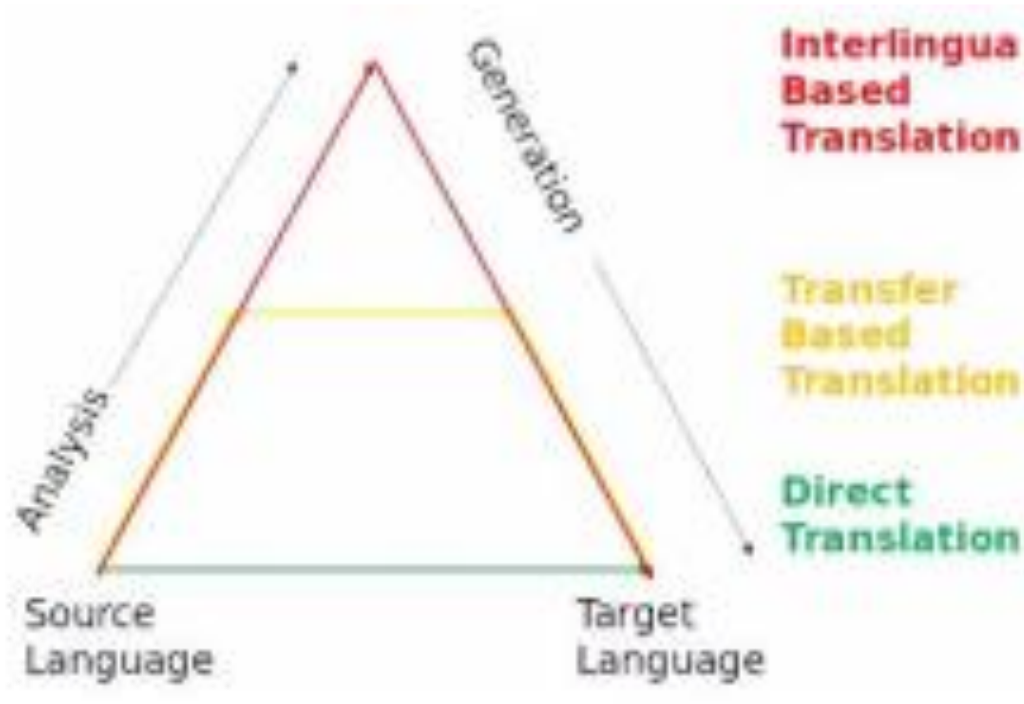



**Figure X.6**: Abridged Vauquois Triangle

# ATG and NMT

- Analysis-Transfer-Generation, the foundation of MT

- NMT addresses this by
  - (a) encoding the input
  - (b) encoded vector is enriched by self attention
  - (c) cross attention and
  - (d) auto regression