INTRODUCTION OF NATURAL LANGUAGE PROCESSING

SHIJU SS

HUMAN LANGUAGE

- The most difficult problem
  - Why ?
- Still not completely solved
- Is data available ?

AMBIGUITY OF LANGUAGE

Classic Example :

Boy saw a man on a hill with a telescope

Look at the dog with one eye

A program

make

HISTORY

- 1950s
  - Early MT: word translation + re-ordering
  - Chomsky's Generative grammar
  - Bar-Hill's argument
- 1960-80s
  - Applications
    - BASEBALL: use NL interface to search in a database on baseball games
    - LUNAR: NL interface to search in Lunar
    - ELIZA: simulation of conversation with a psychoanalyst
    - SHREDLU: use NL to manipulate block world
    - Message understanding: understand a newspaper article on terrorism
    - Machine translation
  - Methods
    - ATN (augmented transition networks): extended context-free grammar
    - Case grammar (agent, object, etc.)
    - DCG – Definite Clause Grammar
    - Dependency grammar: an element depends on another

- 1990s-2010s
  - Statistical methods
  - Speech recognition
  - Question-answering
- 2010s - now
  - Translation
  - Chabot
  - Conversational Bots
  - Question-answering

CLASSICAL STAGES OF NLP

- Morphological Analysis
  - Individual words are analyzed into their components
- Syntactic Analysis
  - Linear sequence of words are transformed into structures that show how the words relate to each other.
- Semantic Analysis
  - Transformation is made from the input text to an internal representation that reflects the meaning – ANALYSE CONTEXT and MEANING
- Pragmatic Analysis
  - To reinterpret what was said to what was actually meant
- Discourse Analysis
  - Resolving references between sentences. (It. He, they etc.)

TOKENIZATION AND SENTENCE SEGMENTATION

- Cells of a language.
- Same words occur many times in a sentence
- Is word count a relevant feature ?
- How many tokens in "India is my country"
- Words may represent the object which we are looking for.
  - "Total budget for this year is approximately 25000 USD.
- After tokenization need stemming in some analysis.
  - What is stemming?
    - Life -> life
    - life -> life
    - Lives -> life

PART OF SPEECH

- Grammatical tagging
- Word category disambiguation.
- Based on definition and context
  - Make cake
  - Make of a car

- Approach :-
  - mid:1980s – HMM
  - Dynamic Programming – similar to Viterbi
  - Mostly variations of HMM.

CHUNKING

- Identifies phrase level constituents in sentences.
  - Noun phrases – "Prime minister Modi"
  - Verb Phrases – "playing cricket"
- Phrase extraction
- Detecting the mentions of key entities or other relevant information.
- Uses POS tag information along with words
- Most relevant in Named Entity Recognition

RULES VS. STATISTICS

- Rules and categories do not fit a sentence equally
  - Some are more likely in a language than others
  - E.g.
    - hardcopy: noun or verb?
      - $P(N \mid \text{hardcopy}) \gg P(V \mid \text{hardcopy})$
    - the training …
      - $P(N \mid \text{training}, \text{Det}) > P(V \mid \text{training}, \text{Det})$
- Idea: use statistics to help

STATISTICAL ANALYSIS TO HELP SOLVE AMBIGUITY

- Choose the most likely solution

  solution* = $\text{argmax}_{solution}$ P(solution | word, context)

  e.g. argmax_{cat} P(cat | word, context)

  argmax_{sem} P(sem | word, context)

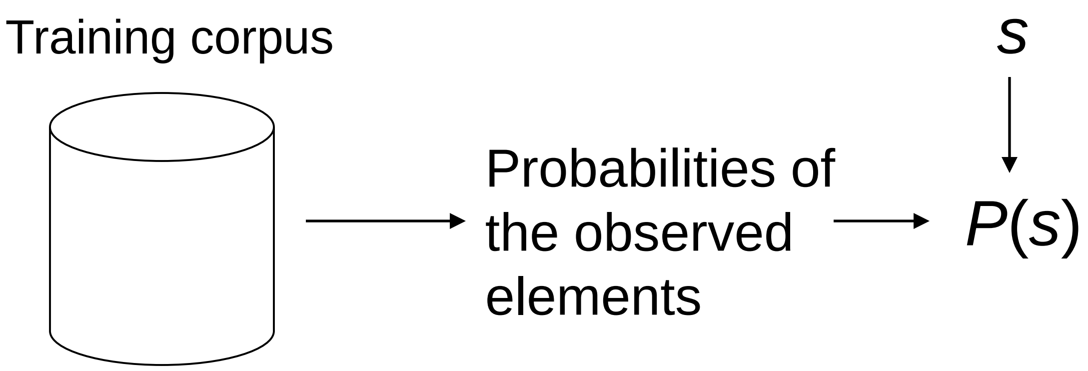
  Context varies largely (precedent work, following word, category of the precedent word, …)

- How to obtain P(solution | word, context)?
  - Training corpus

STATISTICAL LANGUAGE MODELING

Goal: create a statistical model so that one can calculate the probability of a sequence of tokens $s = w_1, w_2,\ldots, w_n$ in a language.

- General approach:



PROB. OF A SEQUENCE OF WORDS

$$P(s) = P(w_1, w_2, \ldots w_n)$$

$$= P(w_1)P(w_2 \mid w_1) \ldots P(w_n \mid w_{1,n-1})$$

$$= \prod_{i=1}^{n} P(w_i \mid h_i)$$

Elements to be estimated: $$P(w_i \mid h_i) = \frac{P(h_i w_i)}{P(h_i)}$$

- If h_i is too long, one cannot observe (h_i, w_i) in the training corpus, and (h_i, w_i) is hard generalize
- Solution: limit the length of h_i

N-GRAMS

$$P(s)=\prod_{i=1}^{n}P(w_i)$$

$$P(s)=\prod_{i=1}^{n}P(w_i \mid w_{i-1})$$

$$P(s)=\prod_{i=1}^{n}P(w_i \mid w_{i-2}w_{i-1})$$

JACCARD COEFFICIENT

- A commonly used measure of overlap of two sets *A* and *B*
- jaccard(A,B) = $|A \cap B| / |A \cup B|$
- jaccard(A,A) = 1
- jaccard(A,B) = 0 if $A \cap B = 0$
- *A* and *B* don't have to be the same size.
- Always assigns a number between 0 and 1.

JACCARD COEFFICIENT: SCORING EXAMPLE

- What is the query-document match score that the Jaccard coefficient computes for each of the two documents below?
- Query: *ides of march*
- Document 1: *caesar died in march*
- Document 2: *the long march*

ISSUES WITH JACCARD FOR SCORING

- It doesn't consider *term frequency* (how many times a term occurs in a document)
- Rare terms in a collection are more informative than frequent terms. Jaccard doesn't consider this information
- We need a more sophisticated way of normalizing for length

$$|A \cap B| / \sqrt{|A \cup B|}$$

BINARY TERM-DOCUMENT INCIDENCE MATRIX

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

TERM-DOCUMENT COUNT MATRICES

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^{v}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

BAG OF WORDS MODEL

- Vector representation doesn't consider the ordering of words in a document
- *John is quicker than Mary* and *Mary is quicker than John* have the same vectors
- This is called the bag of words model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
- For now: bag of words model

TERM FREQUENCY TF

- The term frequency $\mathrm{tf}_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- We want to use tf when computing query-document match scores. But how?
- Raw term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

LOG-FREQUENCY WEIGHTING

- The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.
- Score for a document-query pair: sum over terms *t* in both *q* and *d*:
- score

$$= \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$$

- The score is 0 if none of the query terms is present in the document.

DOCUMENT FREQUENCY

- Rare terms are more informative than frequent terms
  - Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
- → We want a high weight for rare terms like *arachnocentric.*

DOCUMENT FREQUENCY, CONTINUED

- Frequent terms are less informative than rare terms
- Consider a query term that is frequent in the collection (e.g., *high, increase, line*)
- A document containing such a term is more likely to be relevant than a document that doesn't
- But it's not a sure indicator of relevance.
- → For frequent terms, we want high positive weights for words like *high, increase, and line*
- But lower weights than for rare terms.
- We will use document frequency (df) to capture this.

IDF WEIGHT

- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the idf (inverse document frequency) of t by

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

TF-IDF WEIGHTING

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$\mathrm{w}_{t,d} = \log(1+\mathrm{tf}_{t,d}) \times \log_{10}(N/\mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
  - Note: the "-" in tf-idf is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

SCORE FOR A DOCUMENT GIVEN A QUERY

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

- There are many variants
  - How "tf" is computed (with/without logs)
  - Whether the terms in the query are also weighted
  - …

BINARY → COUNT → WEIGHT MATRIX

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

FEATURE ENGINEERING

- Conventional hand crafted features → Representation Learning
- Representation Learning
  - We deals with wide variety of data
    - Signals, Images, Text etc.
  - Which is the best way of representing the data
  - How to represent the data, so that machine can learn from it
  - Def : A bunch of **techniques** which **automatically** detect / **learn** the **representation** for feature selection from natural / raw data.

REPRESENTATIONS IN NLP

- Word Representation
- Sentence Representation
- Document Representation

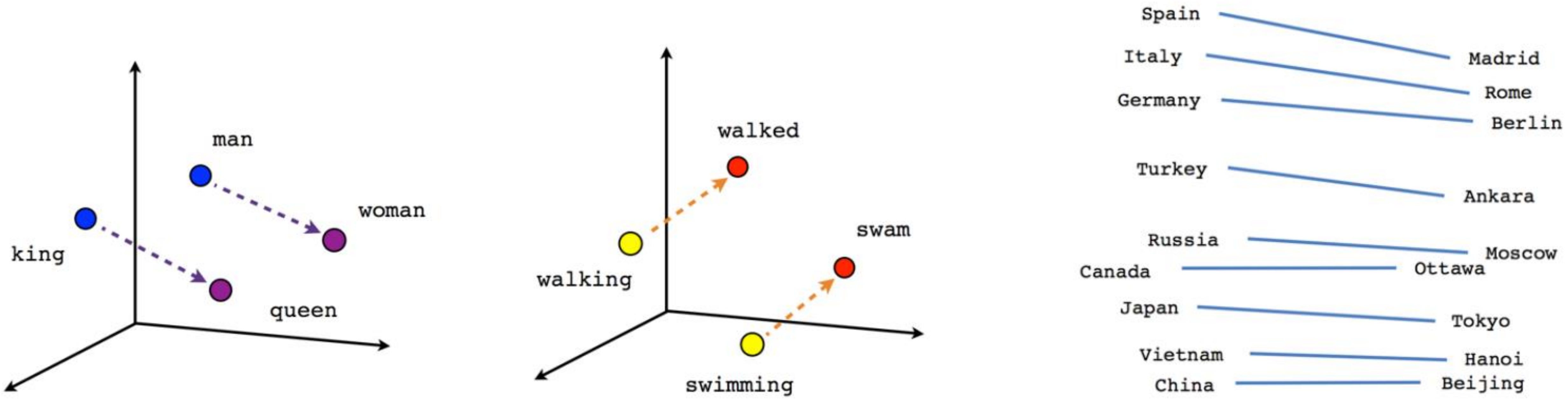
WORD REPRESENTATIONS

Traditional Method - Bag of Words Model	Word Embeddings
• Uses one hot encoding • Each word in the vocabulary is represented by one bit position in a HUGE vector. • For example, if we have a vocabulary of 10000 words, and "Hello" is the 4^{th} word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . 0 0 0 0 • Context information is not utilized	• Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300) • Unsupervised, built just by reading huge corpus • For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02] • Dimensions are basically projections along different axes, more of a mathematical concept.

THE POWER OF WORD VECTORS

- They provide a fresh perspective to ***almost ALL*** problems in NLP.
- Technological Improvement
  - Rise of deep learning since 2006 (Big Data + GPUs + Work done by Andrew Ng, Yoshua Bengio, Yann Lecun and Geoff Hinton)
  - Application of Deep Learning to NLP – led by Yoshua Bengio, Christopher Manning, Richard Socher, Tomas Mikalov
- The need for unsupervised learning . (Supervised learning tends to be excessively dependant on hand-labelled data and often does not scale)

EXAMPLES



Male-Female

Verb tense

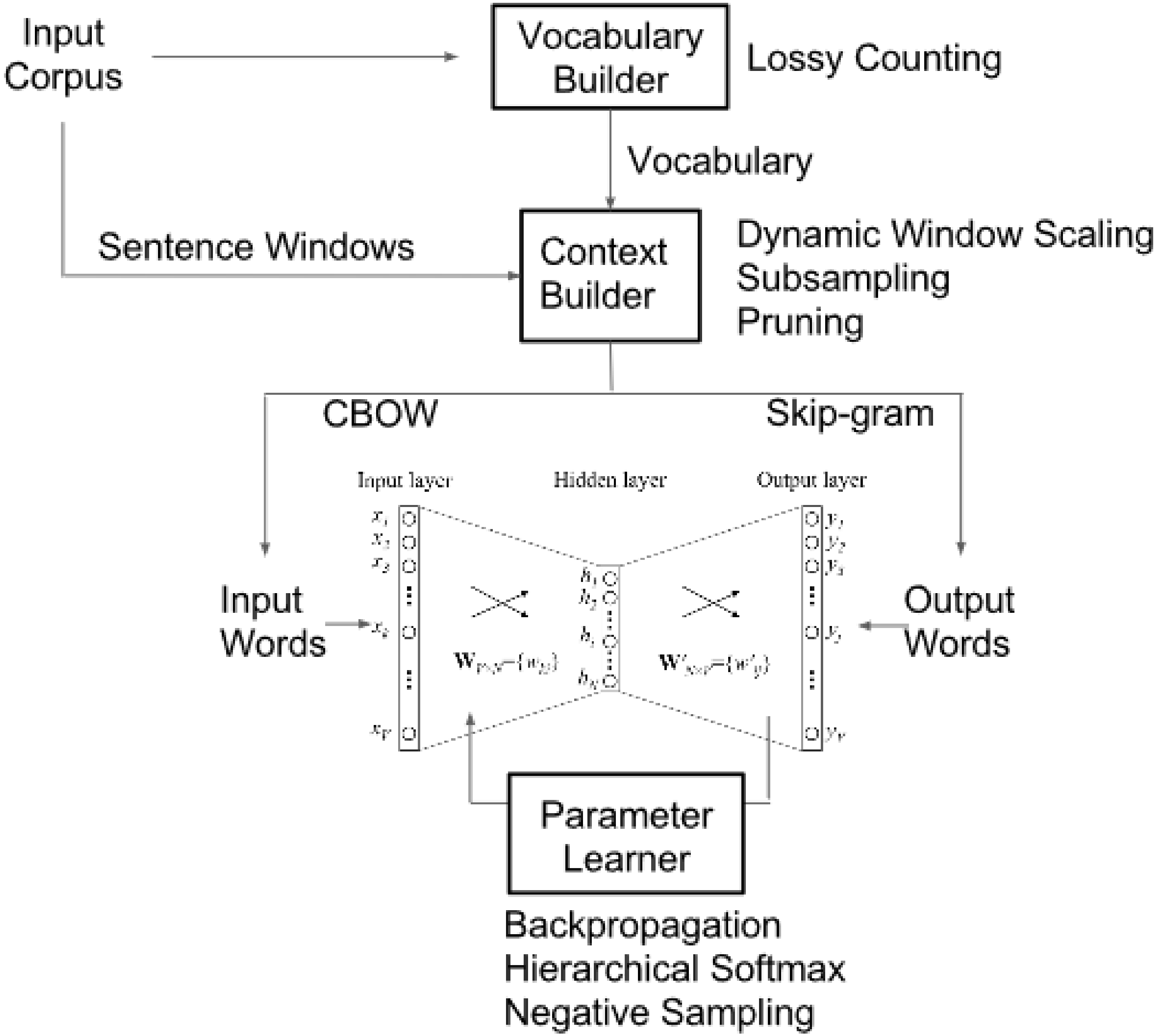
Country-Capital

vector[Queen] = vector[King] - vector[Man] + vector[Woman]

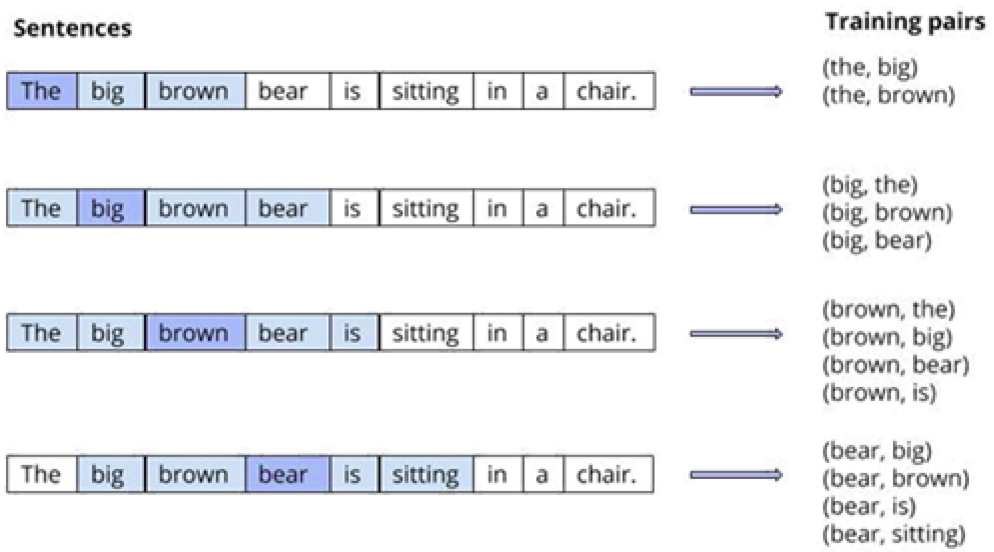
ARCHITECTURE

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p(w_{t+j}|w_t)$$

$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}$$



TRAINING – SKIP GRAM



APPLICATIONS

<u>Word Similarity</u>

Classic Methods : Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries

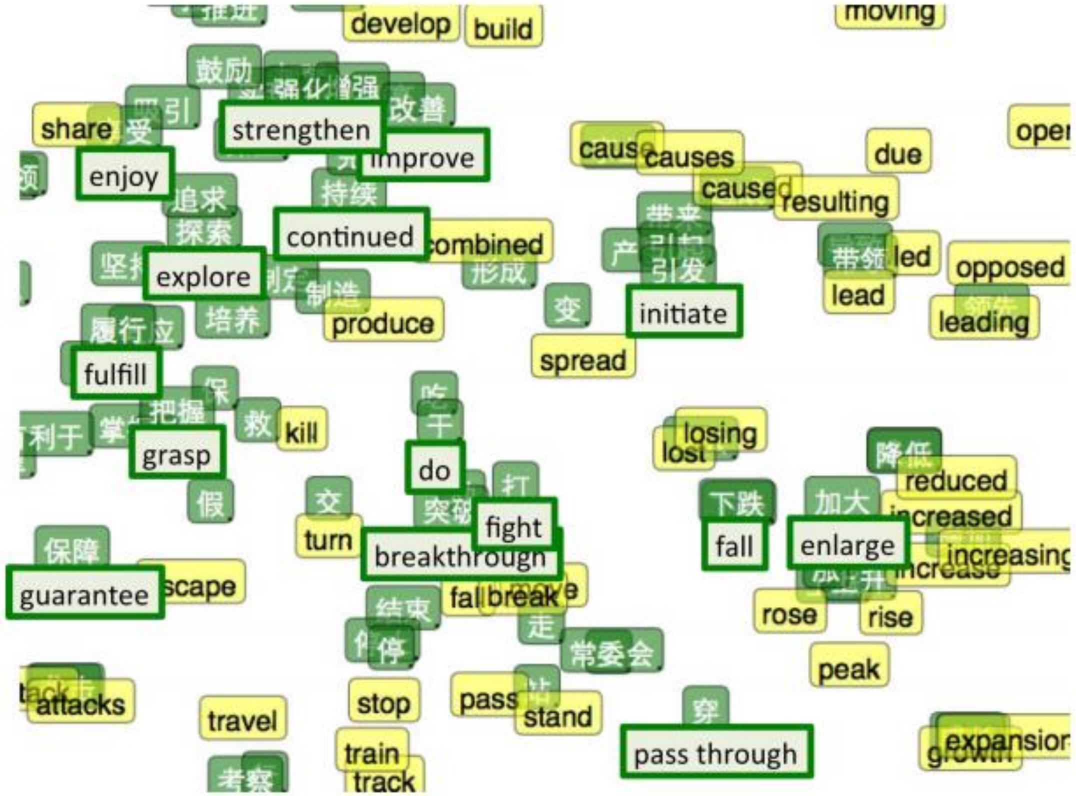
- Easily identifies similar words and synonyms since they occur in similar contexts
- Stemming (thought -> think)
- Inflections, Tense forms
- *eg. Think, thought, ponder, pondering,*
- *eg. Plane, Aircraft, Flight*

NAMED ENTITY RECOGNITION

- Identifies and classifies strings of characters representing proper nouns.
- Useful for filtering documents
- Linguistic grammar based models
  - Months of work by linguistic expert
  - High precision
- Machine learning
  - Supervised as well semi-supervised
  - Require large amount of annotated data
  - Conditional Random Fields

MACHINE TRANSLATION

Classic Methods : Rule-based machine translation, morphological transformation



LANGUAGE TRANSLATION

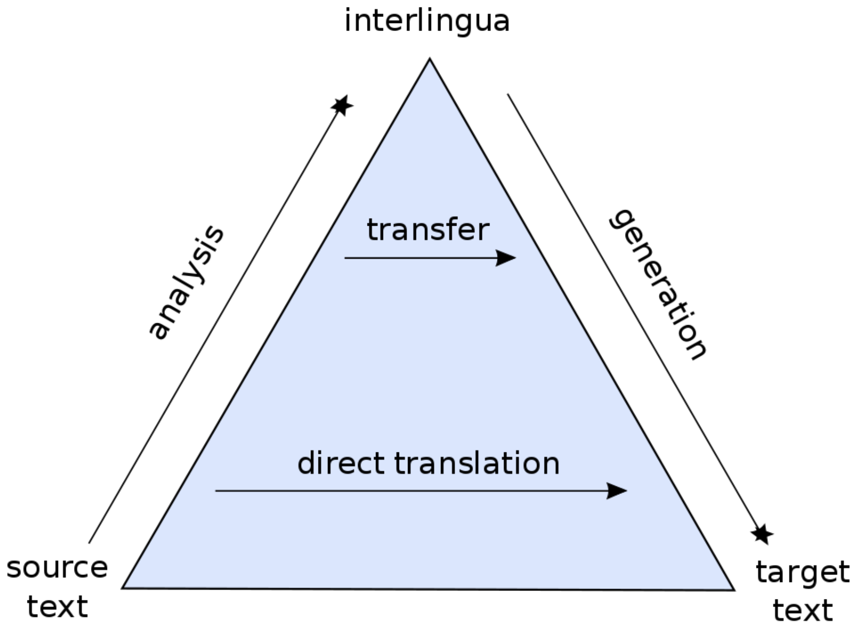
- Languages differs in
  - Script
  - Vocabulary
  - Grammar

These difference can be considered as a measure of language distance

Machine Translation Problem

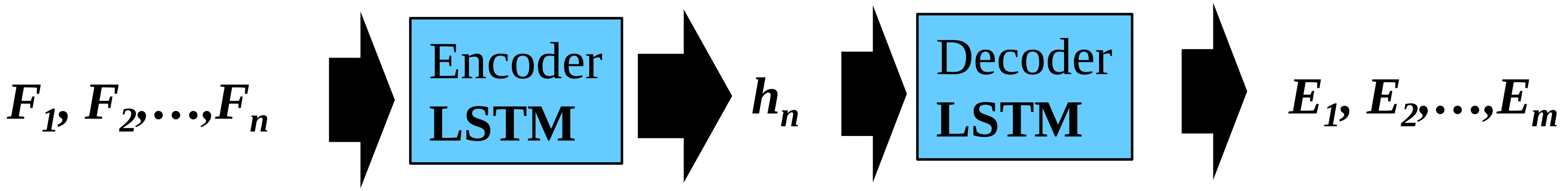
Language Encoding -> representation and analysis

Language Decoding -> synthesis



NEURAL MACHINE TRANSLATION (NMT)

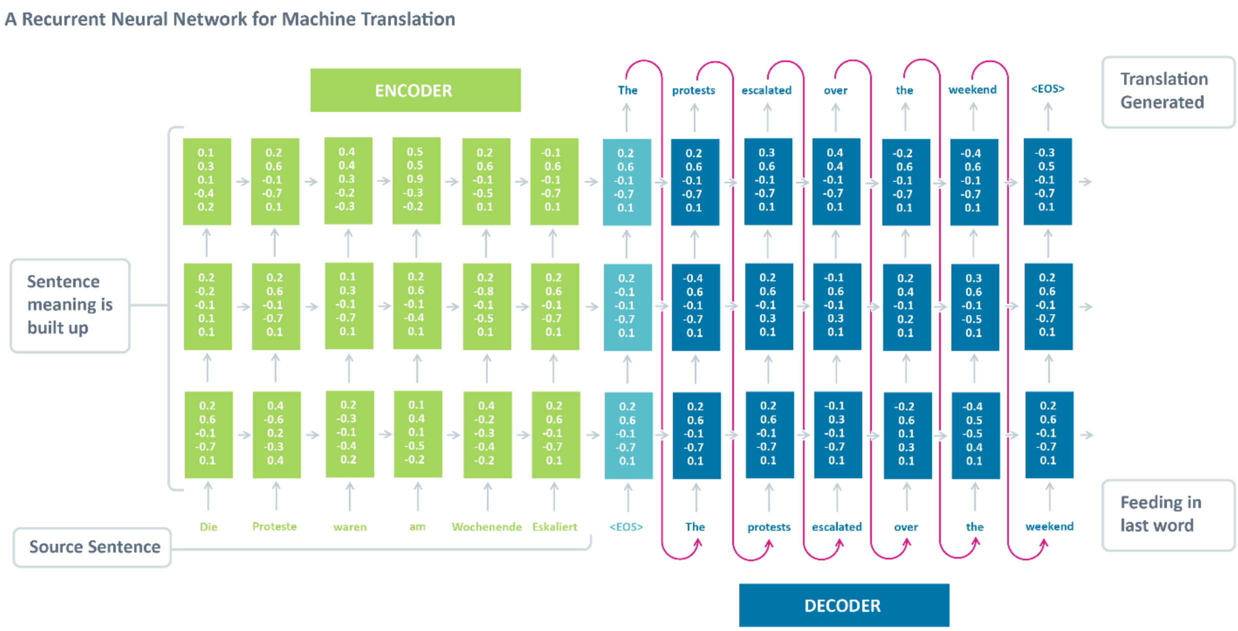
- Encoder/Decoder framework maps sentence in source language to a "deep vector" then another LSTM maps this vector to a sentence in the target language



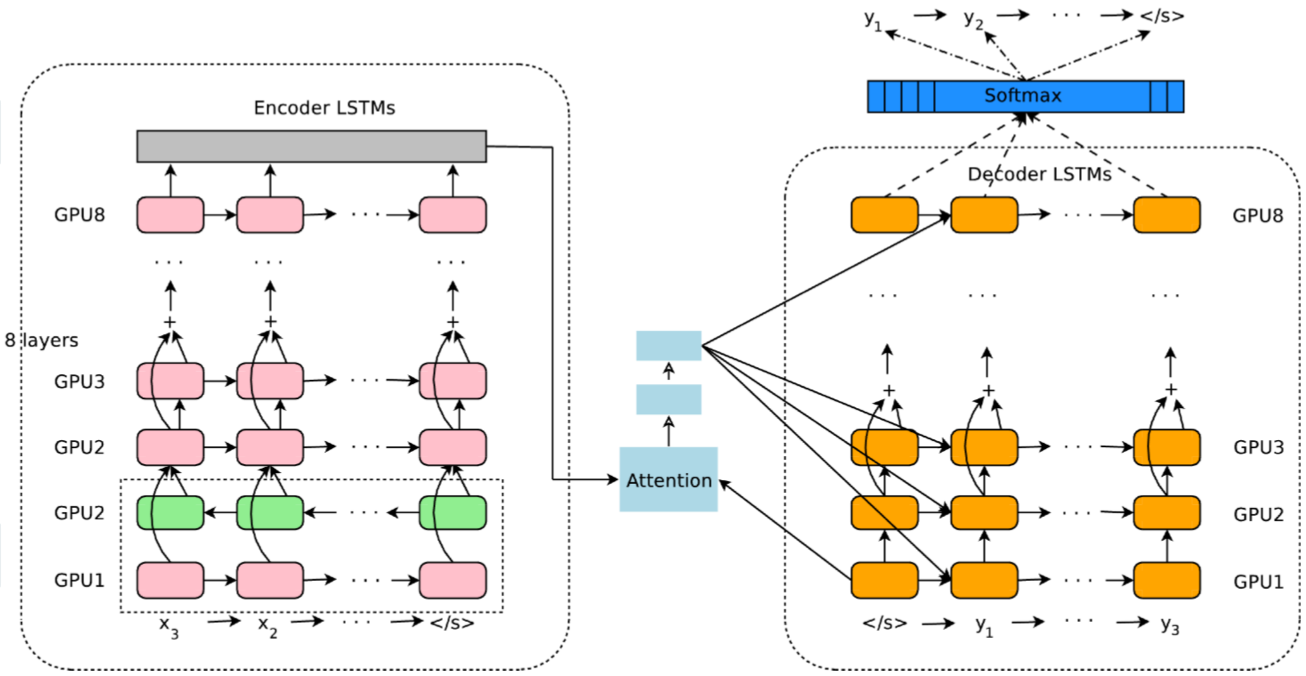
Train model "end to end" on sentence-aligned parallel corpus.

NMT

- Without Attention



- With Attention



CHATBOT

- Chat Bot: A computer program that can ***talk*** to humans in natural language!
- Uses Artificial Intelligence Markup Language (AIML) to represent knowledge.
- Can replace a human for monotonous jobs of answering queries, e.g. E-help desk.

ELIZA

- Eliza – the first chat bot made by *Joseph Weizenbaum*.
- Eliza Effect
  - tendency of humans to attach associations to terms from prior experience.
- Working of Eliza is based on
  - Knowledge Representation
  - Pattern Recognition
  - Substitution of key words into known phrases.

ELIZA

- Looks for certain patterns of words in the user's input.
- Replies with pre-determined output, if the pattern is matched.
- Needs to have an idea of what the user will chat
- Has suitable responses defined in the AIML file

AIML WORKING

- Contains patterns that does not have wildcards "*" or "_".
- Example:

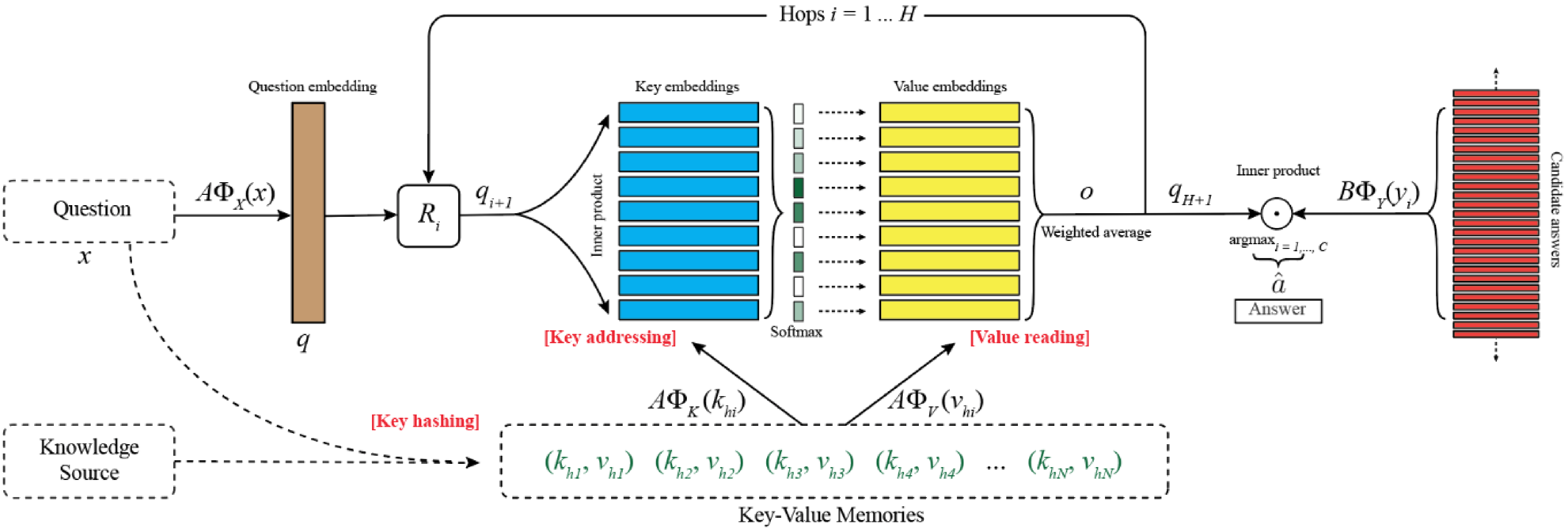
```
<category>
    <pattern>10 DOLLARS</pattern>
    <template> wow, what a cheap  </template>
</category>
```

- Conversation:

User: This watch is for 10 dollars

Chat Bot: Wow, what a cheap watch!

MEMORY NETWORKS FOR QA



CONCLUSION

- Started with one hot encoding,
- RF-IDF
- Word2Vec
- Encoder-Decoder
- Memory Networks & Neural Turing Machine
- And goes on.

Questions ?