Natural Language Understanding

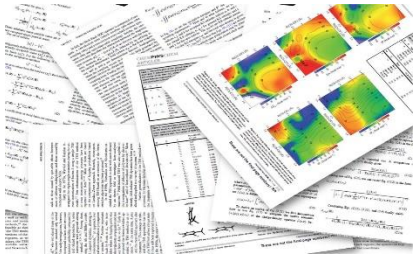
Jian Tang

HEC Montreal

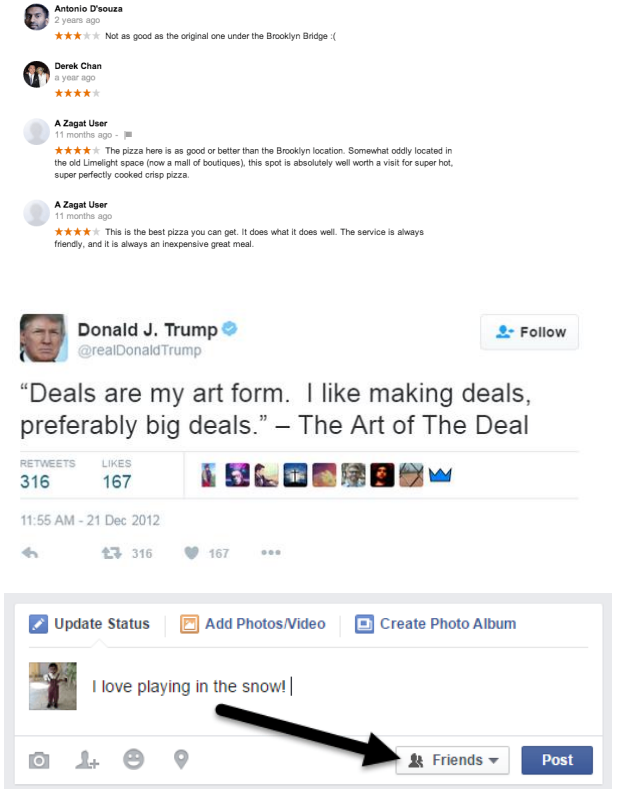
Mila-Quebec AI Institute

Email: jian.tang@hec.ca

A Huge Amount of Unstructured Text



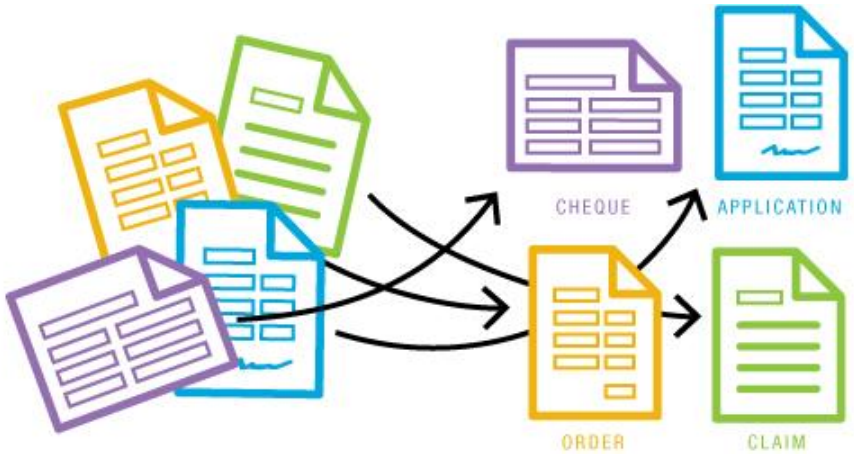
Traditional media



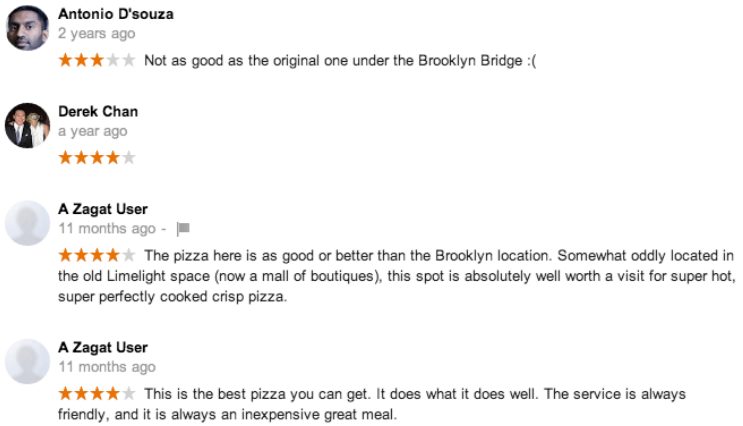
Social media

Electronic Health Records

Document (Sentence) Classification

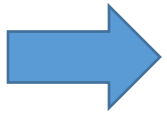


Topic classification

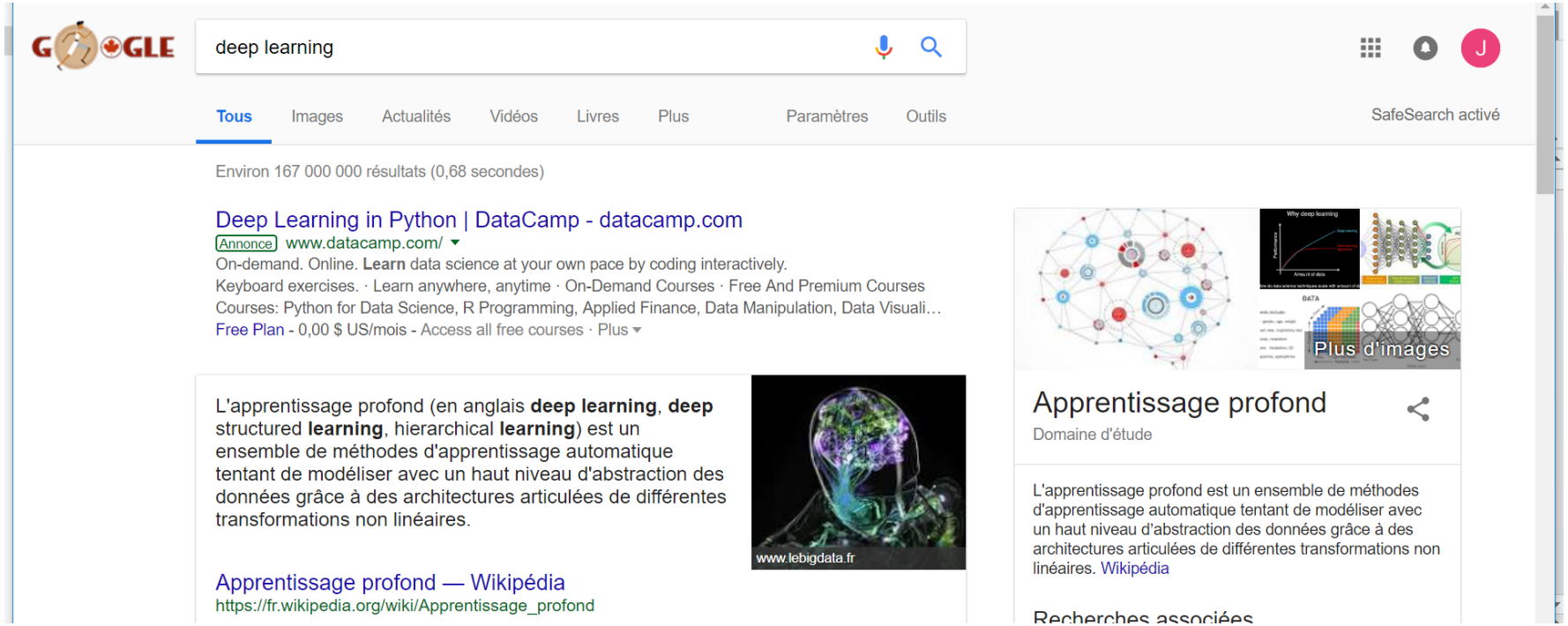


Sentiment classification

Document Clustering



Information Retrieval



Question Answering

Passage: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. **When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901**, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla's breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.

Question: On what did Tesla blame for the loss of the initial money?

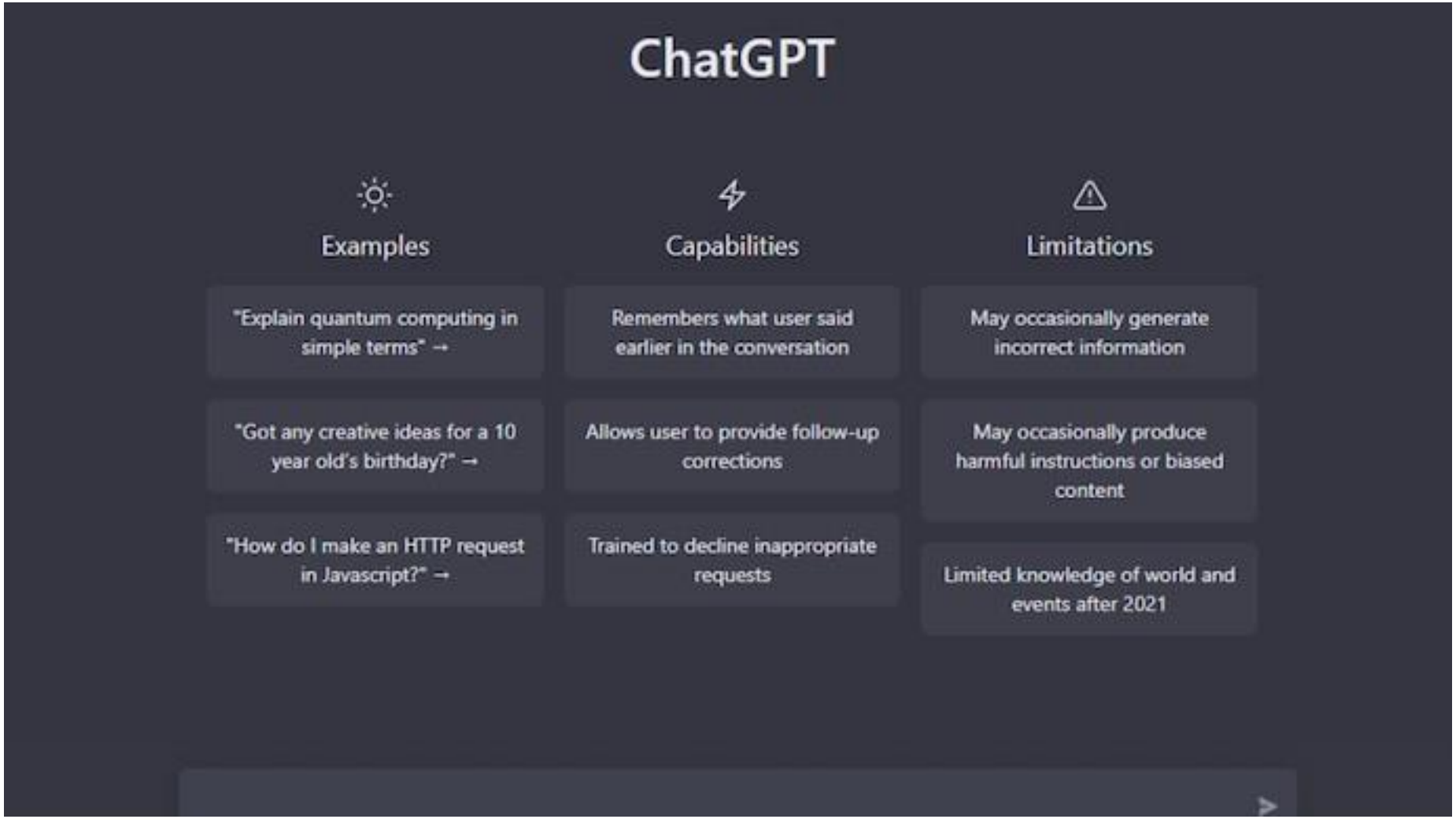
Answer: Panic of 1901

Text Summarization

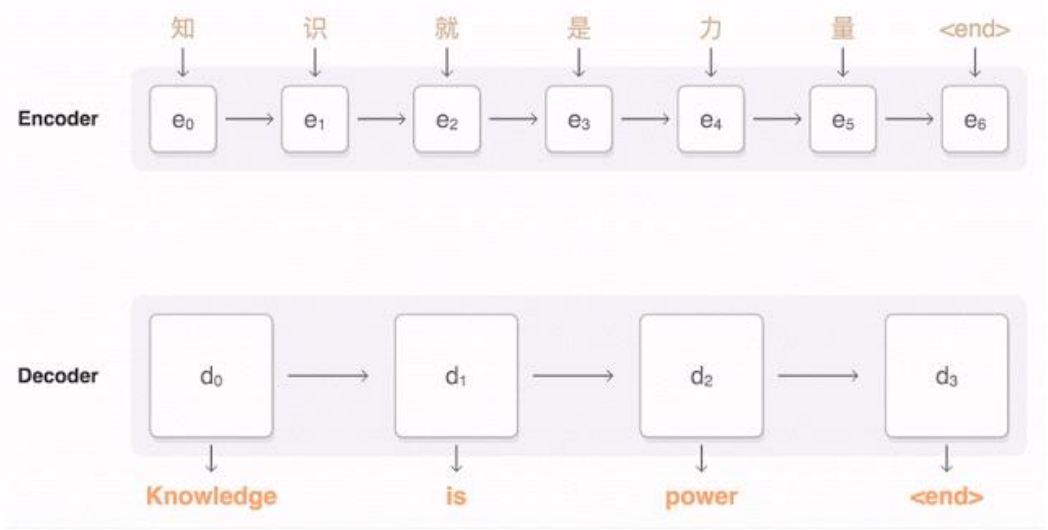
Russian Defense Minister Ivanov called **Sunday** *for the creation of a joint front for combating global terrorism.*

Russia calls for joint front against terrorism.

Dialogue Systems



Machine Translation



Research Questions

- How to effectively learn the representations of **words**, **phrases**, **sentences** and **documents**?

- How to generate nature language?

Classical Word Representations

- Words as atomic symbols: "*One-hot*" representation
- Documents: "*Bag-of-words*"

"network" = [0,1,0,0,0,0,0]

"networks" = [0,0,0,0,1,0,0]

AND

=

0

- Ignore the *semantic relatedness* between words
- The *curse* of dimensionality
  - As large as *millions* in a large text corpus.

Neural Word Embeddings (Bengio et al. 2003)

- Represent each word with a *continuous dense* vector
  - Hundreds or thousands of dimensions
  - Words with similar meanings are represented with similar vectors
- Represent *phrases*, *sentences* and *documents* through word embedding

Distributional Hypothesis

- “You shall know a word by the company it keeps” (J.R. Firth 1957:11)
- The meaning of a word can be represented by its neighbors

A telecommunications network allows computers to exchange data

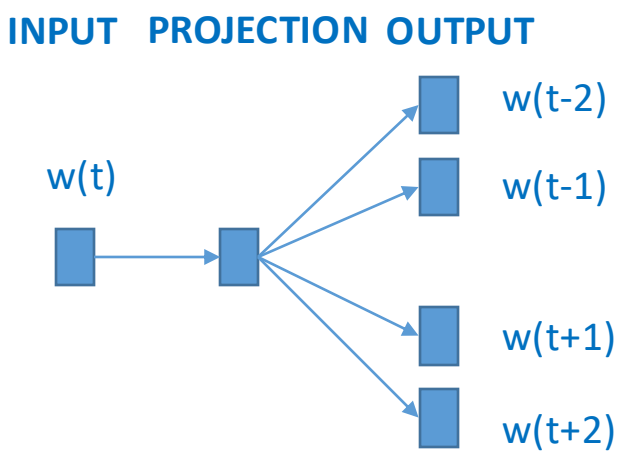
In information technology, a network is a series of points or nodes interconnected...

Represent “*network*” with the neighboring words

Word2VEC (Mikolov et al. 2013)

- **Skip-gram**: finding word representations that are useful for predicting the surrounding words in a sentence or a document

A telecommunications network allows computers to exchange data



Objective of Skip-gram

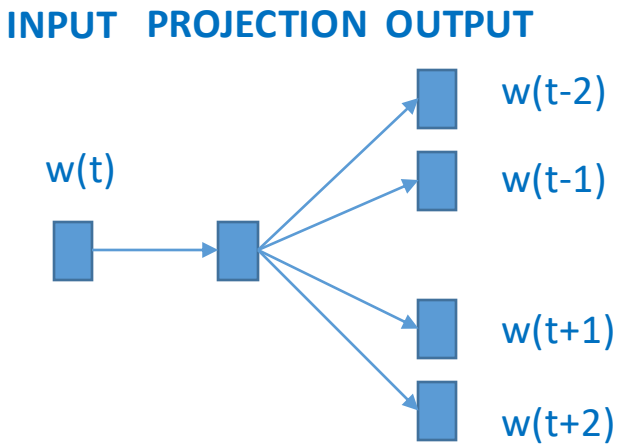
- Given a sequence of training words $w_1 w_2, \ldots, w_T$, the objective of the skip-gram is to maximize the average log probability:

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

- Where c is the size of the training context. $p(w_{t+j}|w_t)$ is defined with a softmax function

$$p(w_{t+j}|w_t) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W} \exp({v'_w}^T v_{w_I})}$$

- Where v_w and v'_w are the "input" and "output" vector representations of w. W is the vocabulary size.
- Calculating $p(w_{t+j}|w_t)$ is very computational expensive



Negative Sampling (Mikolov et al. 2013)

- Modify the objective as:

$$\log \sigma({v'_{w_O}}^T v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_{n(w)}} \log \sigma(-{v'_{w_i}}^T v_{w_I})$$

- It aims to distinguish the target word w_O from draws from the noise distribution $P_n(w)$ using logistic regression. k is the number of negative samples for each input word (k is usually 5-20).
- $P_n(w)$ is usually set as the unigram distribution $\mathrm{U}(w)$ raised to the 3/4rd power, i.e.,

$$P_n(w) = U(w)^{0.75}/Z$$

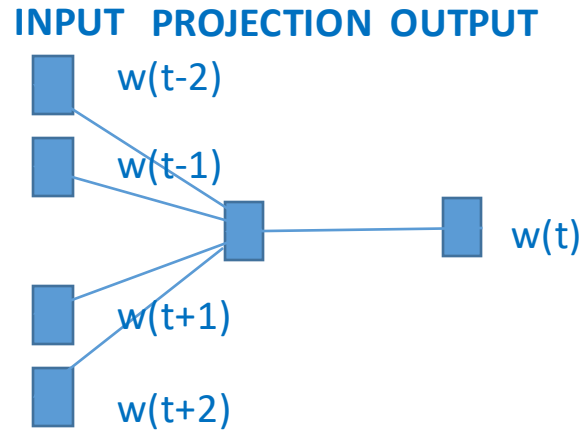
CBOW (Mikolov et al. 2013)

- Instead of using center words to predict nearby words, using nearby words to predict the center words
- Calculating the context embedding

$$v_c = \frac{1}{2c} \sum_{-c \le j \le c, j \ne 0} v_j$$

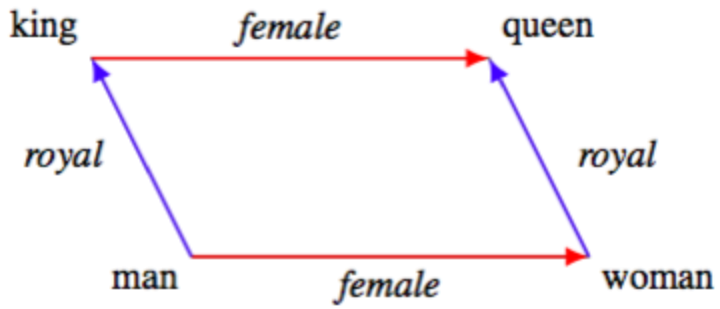
- Predict the center word:

$$p(w_t | w_{t-c}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+c}) = \frac{\exp({v'_{w_t}}^T v_c)}{\sum_{w=1}^{W} \exp({v'_w}^T v_c)}$$



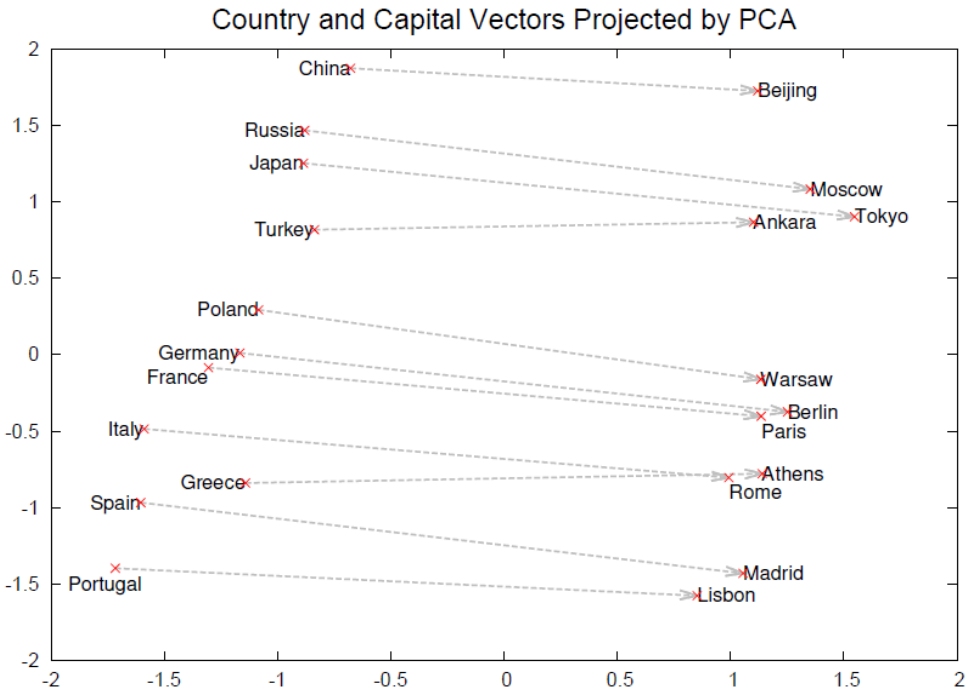
Word Analogy

- Find a word that is similar to *small* in the same sense as *biggest* is similar to *big*.
- Compute vector X = vector(“biggest”)-vector(“big”) + vector(“small”)
- Then search the vector space for the word closest to X measured by cosine distance, and use it as the answer.



Examples

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

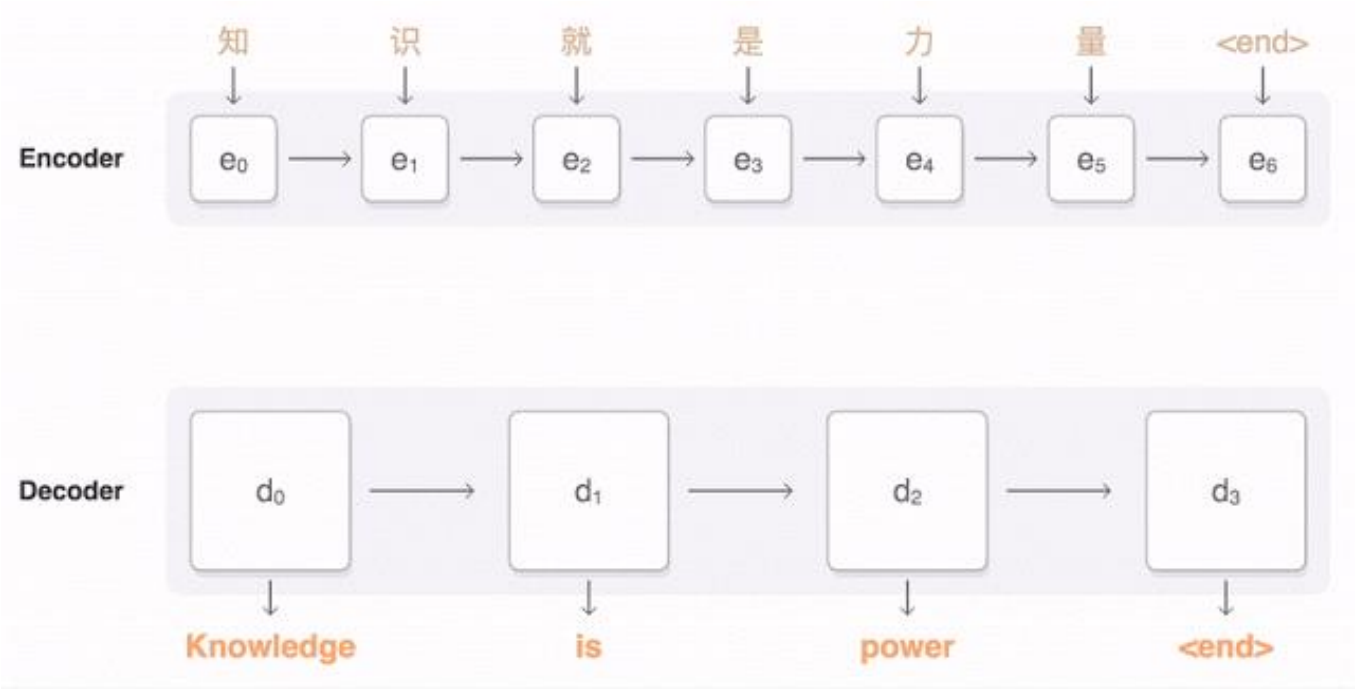


Word Embeddings in Pytorch

- https://github.com/blackredscarf/pytorch-SkipGram
- https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html

Sequence to Sequence

- Machine translation



Sequence to Sequence

- Text Summarization

Russian Defense Minister Ivanov called **Sunday** *for the creation of a joint front for combating global terrorism.*

Russia calls for joint front against terrorism.

Sequence to Sequence

- Dialogue systems

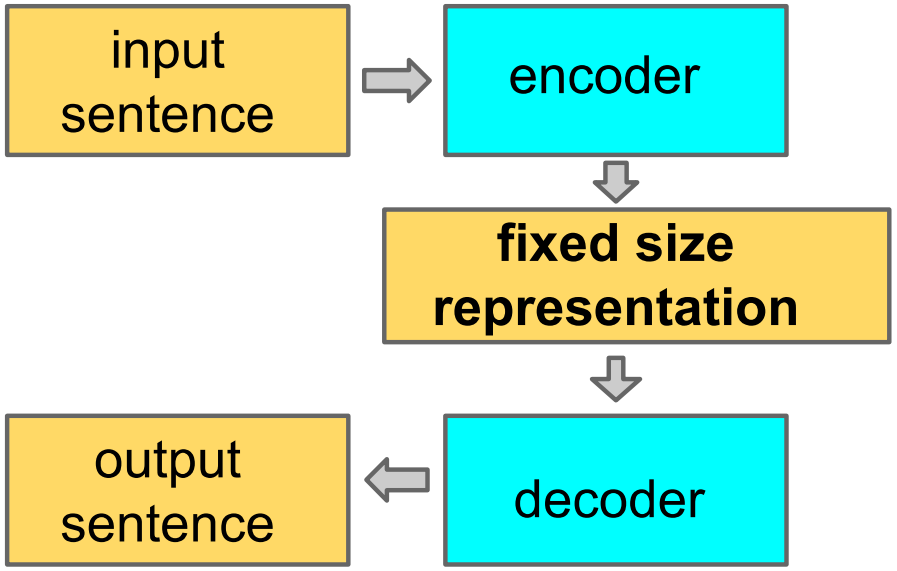
I: Hello Jack, my name is Chandralekha.

R: Nice to meet you, Chandralekha.

I: This new guy doesn't perform exactly as we expected.

R: What do you mean by "doesn't perform exactly as we expected"?

Sequence2Sequence (Encoder-Decoder)



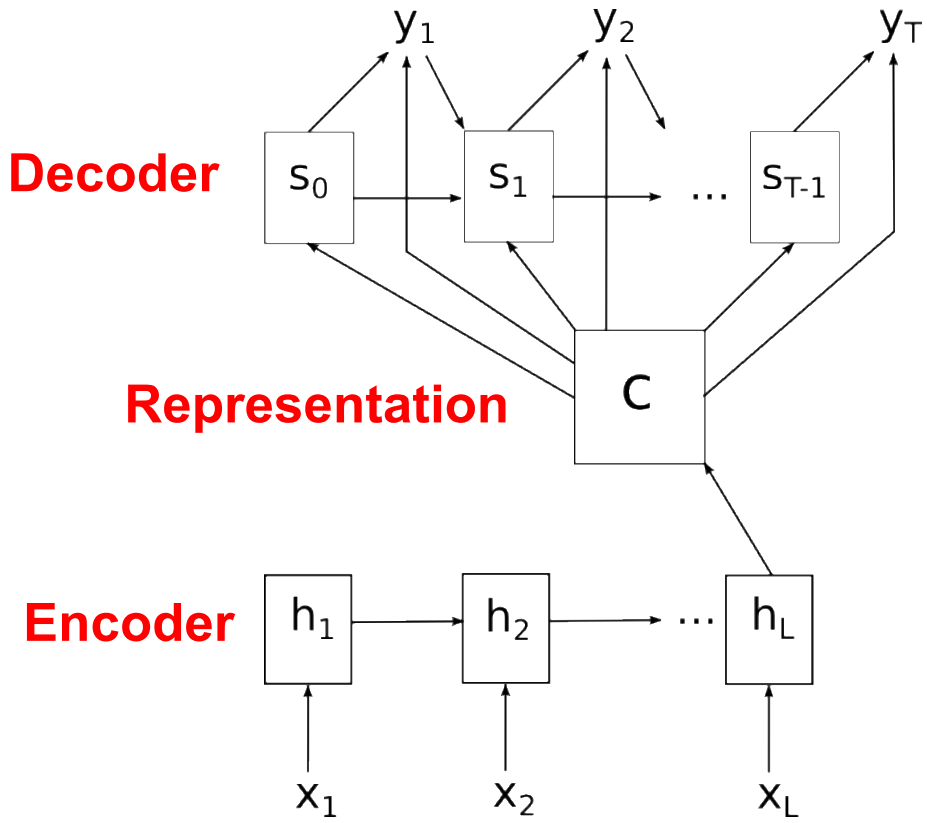
(Ñeco&Forcada, 1997)

(Kalchbrenner et al., 2013)

(Cho et al., 2014)

(Sutskever et al., 2014)

RNN Encoder-Decoder (Cho et al. 2014):



Sequence to Sequence Model

- Given an input sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_L)$, we want to map it to an output sequence $\mathbf{y} = (y_1, y_2, \ldots, y_{\mathrm{T}})$. We are essentially trying to model the conditional probability

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1, y_2, \ldots, y_T|x_1, x_2, \ldots, x_L)$$

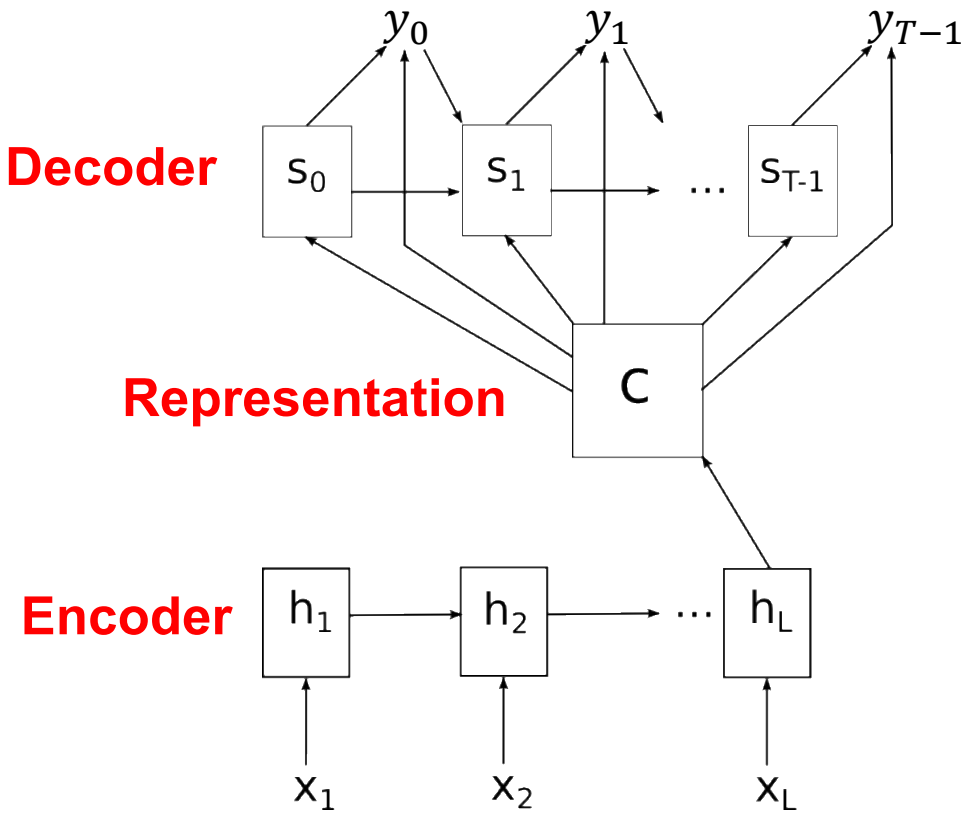
Sequence to Sequence (Encoder-Decoder)

- Encoder
  - One Recurrent Neural Networks
- Decoder
  - Another Recurrent Neural Networks

Encoder: one Recurrent Neural Network

- C is the last hidden state of input sequence
  - summary of the input sequence

RNN Encoder-Decoder (Cho et al. 2014):



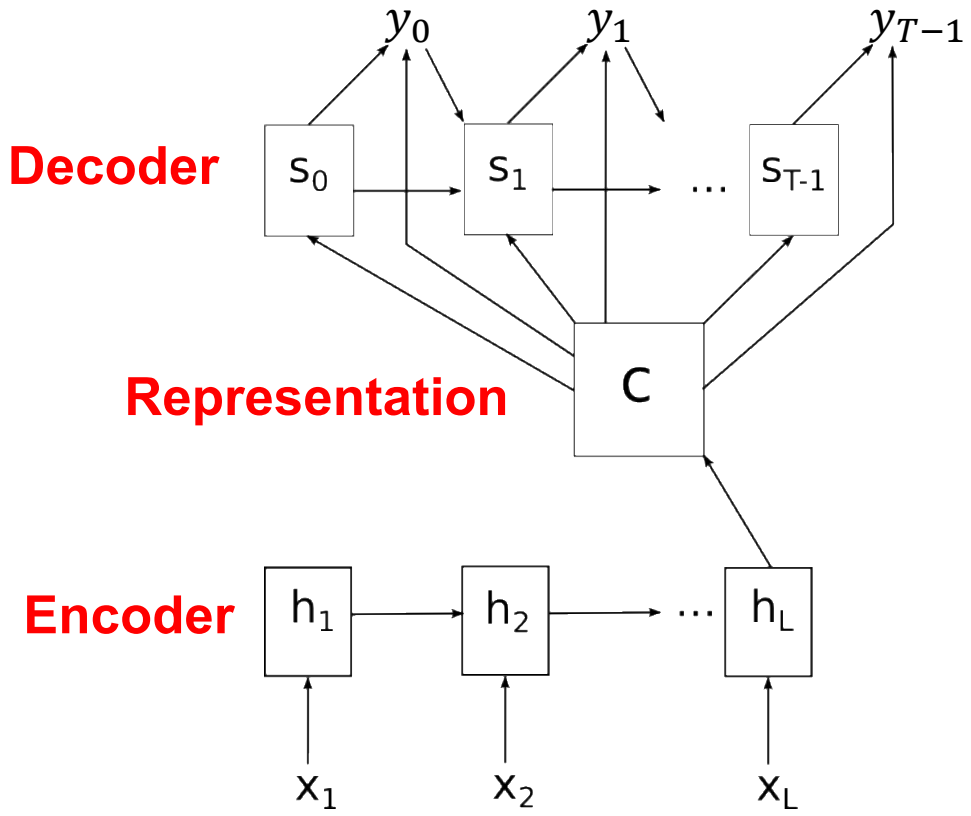
Decoder: another Recurrent Neural Network

- Decode the output sequence **y** based on
- the input sequence **x**

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1, y_2, \ldots, y_T|x_1, x_2, \ldots, x_L)$$

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1, y_2, \ldots, y_T|\boldsymbol{c})$$

RNN Encoder-Decoder (Cho et al. 2014):



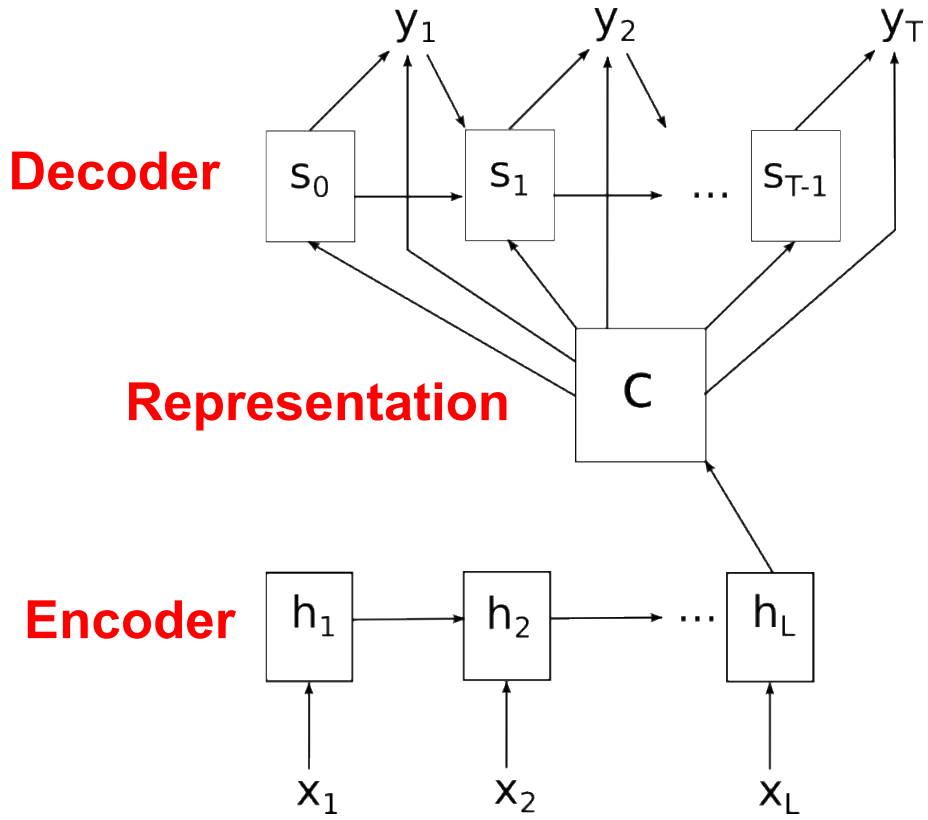
Decoder

- The hidden representation:

$$s_t = f(s_{t-1}, y_t, \boldsymbol{c})$$

- The conditional distribution of next symbol

$$p(y_t | y_{t-1}, y_{t-2}, \dots, y_1, c) = g(s_t, y_{t-1}, \boldsymbol{c})$$



Optimization with Maximum Likelihood

- Given a set of training data $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$, the encoder-decoder components are jointly optimized by maximizing the conditional log-likelihood:

$$\max_{\theta} \frac{1}{N} \sum \log p_\theta(\boldsymbol{y}_n | \boldsymbol{x}_n)$$

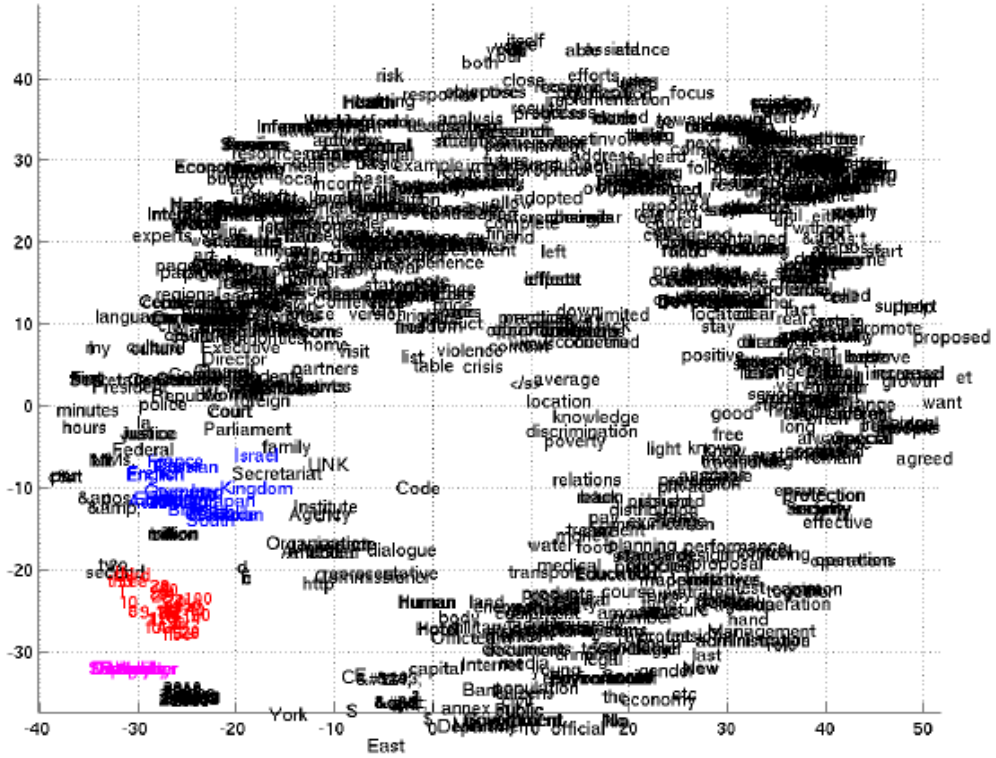
Results on Machine Translation

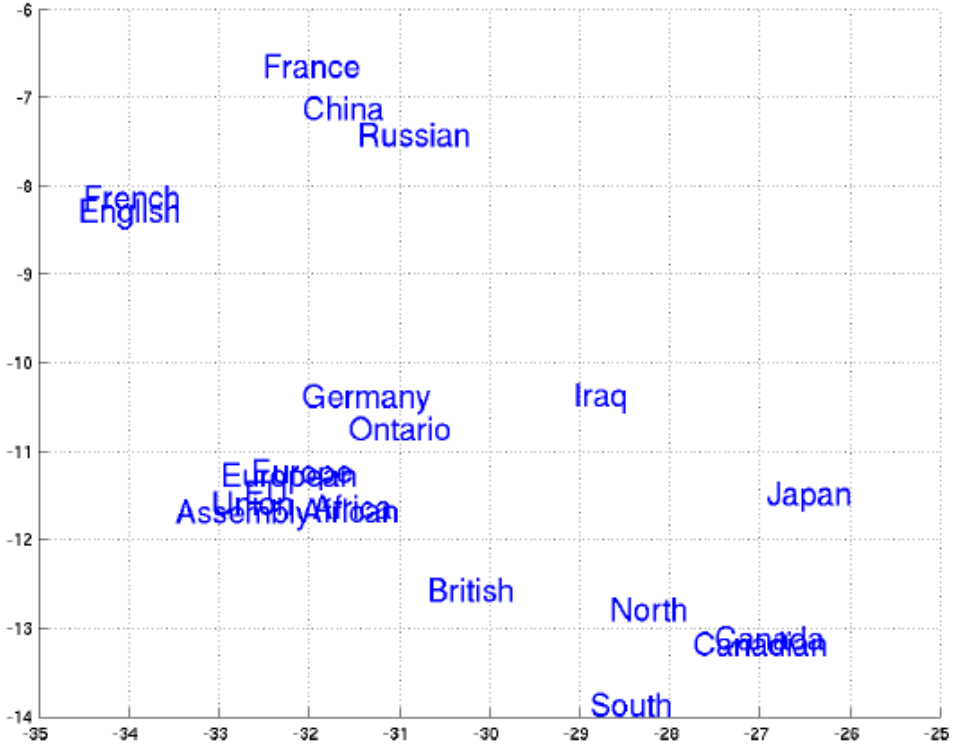
- Both encoder and decoder are RNNs

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

Table: Results from *English* to *French*

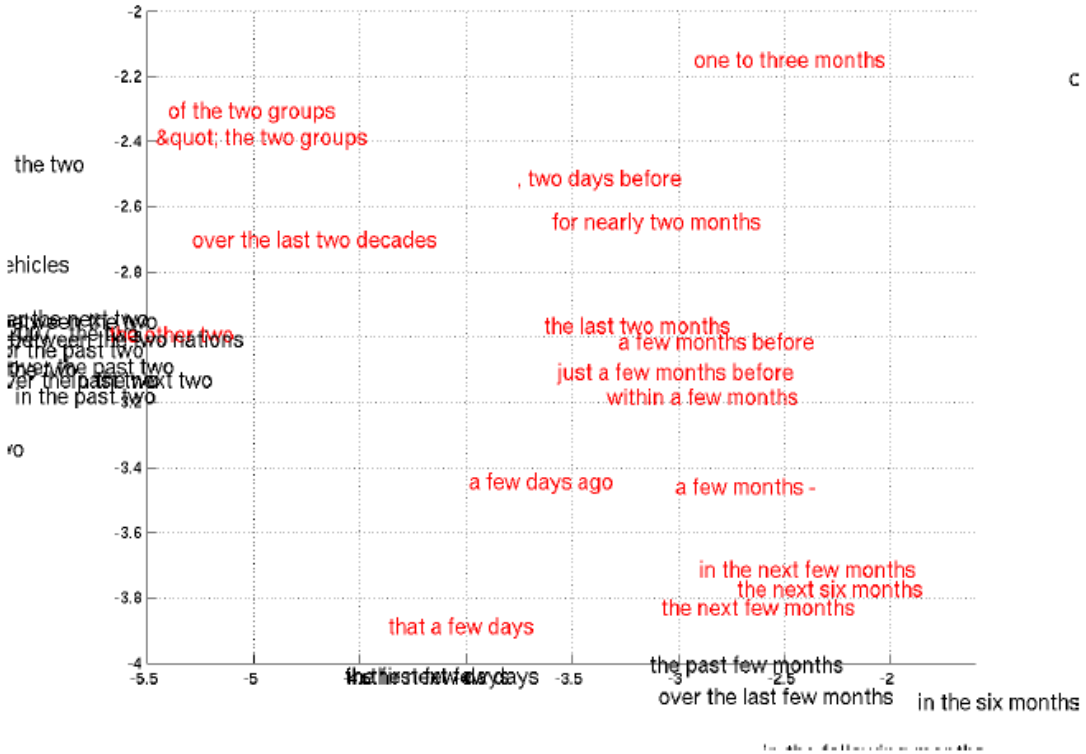
Worde Embeddings Learned by Seq2Seq

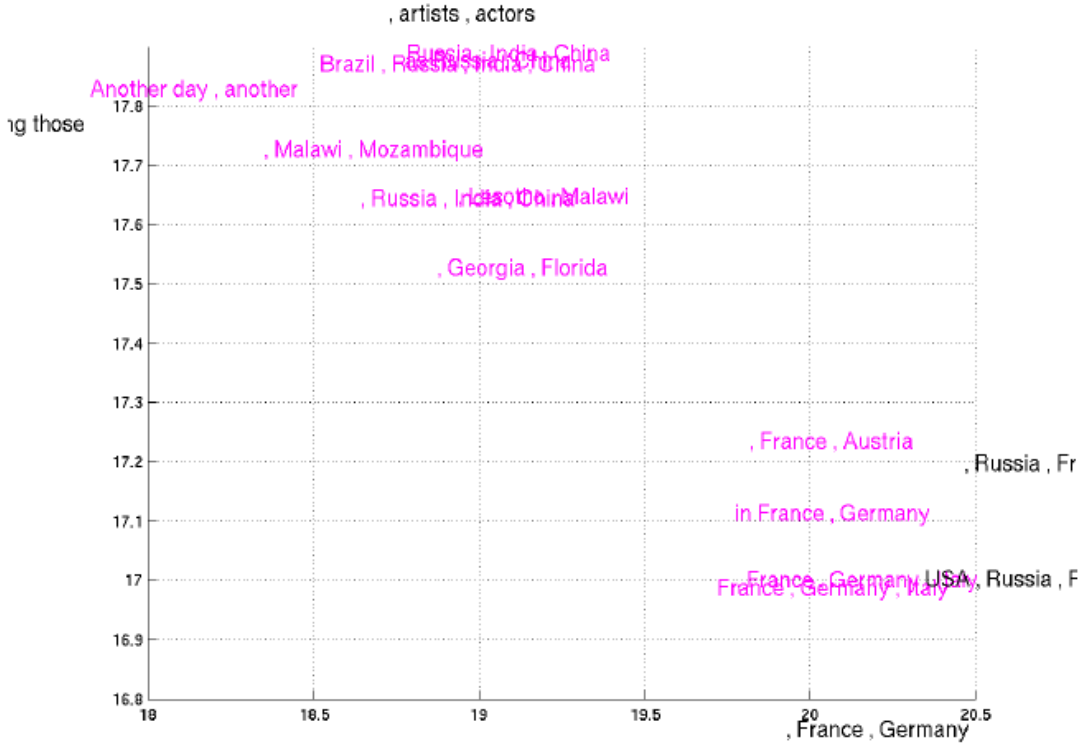




Phrase Embeddings Learned by Seq2Seq

- Phrase embeddings are encoded with the summary vector c





Seq2Seq in Pytorch

- https://github.com/bentrevett/pytorch-seq2seq/blob/master/2%20-%20Learning%20Phrase%20Representations%20using%20RNN%20Encoder-Decoder%20for%20Statistical%20Machine%20Translation.ipynb

Thanks!