FNLP - Week 9: Neural Networks for NLP & Discourse Coherence

Antonio León Villares

April 2022

Contents

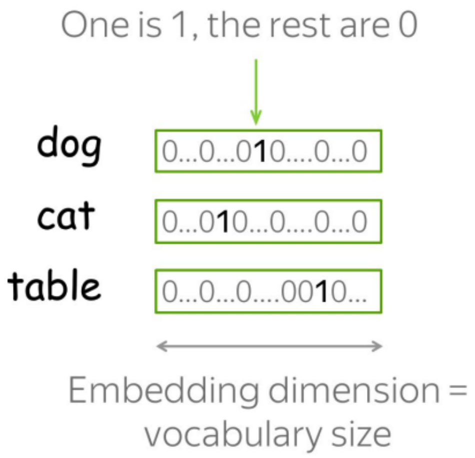
Graphics from: Lena Voita

1 Word2Vec Embeddings

1.1 Issues with Sparse Vector Representations

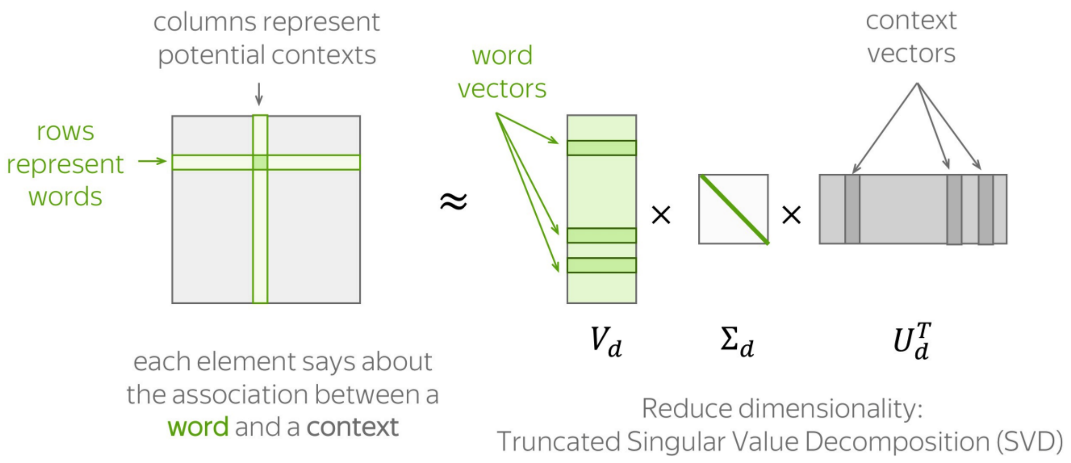
Last week we learnt about ***distributional semantics****, and how word meaning can be derived from* ***context****.*
Crucial to this theory was the ***vector representation*** *of words based on* ***context****, such that* ***similar words*** *would have* ***similar vector representations****.*
Techniques like ***Pointwise Mutual Information*** *could be used to* ***weight*** *the vectors, but the vectors produced where still very* ***large*** *and* ***sparse*** *(corresponding to vocabulary size).*
Latent Semantic Analysis *employs SVD to reduce the dimensionality of the matrices containing the vector representations.*

- **What are vector embeddings?**
  - represent words via vectors which are:
    - **short** (low dimensional)
    - **dense** (each entry in $\mathbb{R}$)
  - last week, we "freely" denominated the sparse vectors as embeddings, when this isn't (technically) true
- **Why are less sparse vectors desirable?**
  - require less **memory/storage**
  - not too good at capturing **semantic similarity** (i.e "car" and "automobile" can even be orthogonal)
  - many dimensions $\implies$ learn more weights $\implies$ can overfit



1.2 Issues with Reducing Dimensionality of Vector Representations

- **How does SVD reduce dimensionality?**
  - with vector representations, create a matrix
  - SVD decomposes matrix into 3 matrices
  - truncate matrices $\implies$ lower dimensional word representation


columns represent potential contexts
rows represent words
word vectors
context vectors
V_d
Σ_d
U_d^T
each element says about the association between a word and a context
Reduce dimensionality: Truncated Singular Value Decomposition (SVD)

- **Why is SVD not a good option?**
  - computationally expensive
  - doesn't **scale** well with matrix size

1.3 The Skip-Gram Method for Vector Embeddings

1.3.1 Aim

- **What is the Skip-Gram Method?**
  - learn **static** embeddings (i.e fixed for a given vocabulary)
  - **self-supervised** procedure (no labels required, only having context words)
- **How does Skip-Gram learn the embeddings?**
  - given a **central word**, seeks to predict its **context words** over a given **window**

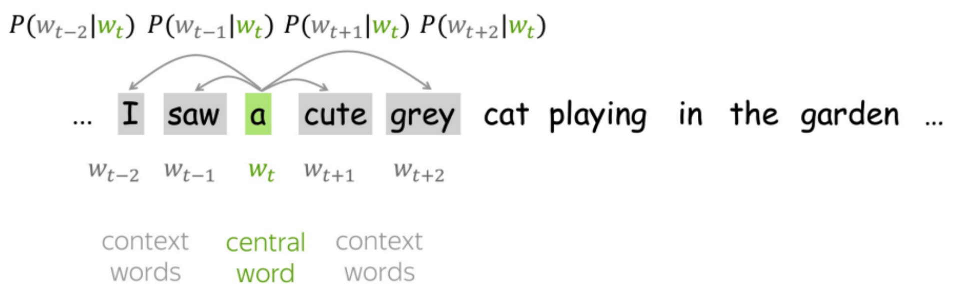
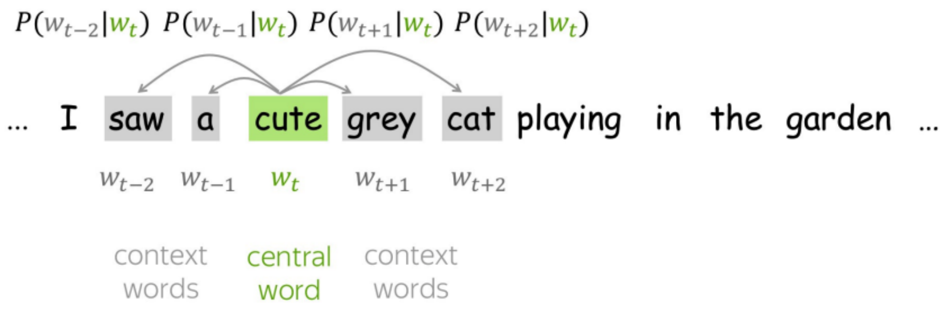
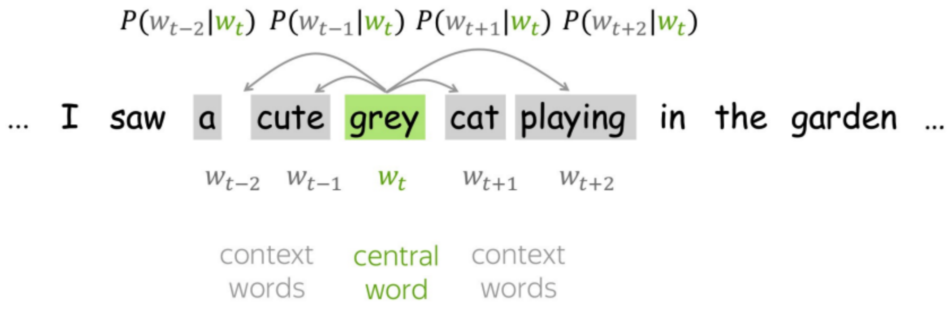

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$
... I saw a cute grey cat playing in the garden ...
w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}
context words
central word
context words

Figure 1: We try to predict the context words of "a" by learning a **probability distribution** for said words.

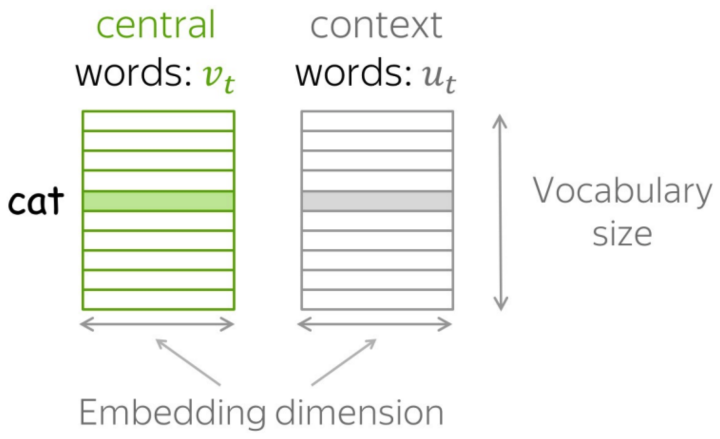

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$
... I saw a cute grey cat playing in the garden ...
w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}
context words
central word
context words


$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$
... I saw a cute grey cat playing in the garden ...
w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}
context words
central word
context words

- ultimately, a **classification problem** (use **logistic regression**) - but we don't care about classification result: the **embeddings** will be the **model weights**

1.3.2 Model Structure

- **What are the model parameters?**
  - 2 matrices:
    * V: rows represent the **central word embeddings**
    * U: rows represent the **context word embeddings**
  - row t in U, V represent the **same word**, but in a **different** function



- **How are the parameters used to perform classification?**
  - **distributive hypothesis**: words in similar context have a similar meaning

- using this, we use **softmax** and the **dot product** to determine the probability of a word o appearing in the context of a central word c:

$$P(o \mid c) = \frac{exp(\underline{u}_o \cdot \underline{v}_c)}{\sum_{\underline{w} \in V} exp(\underline{u}_w \cdot \underline{v}_c)}$$

- 2 vectors are more similar $\iff$ larger dot product $\implies$ higher probability
- this formula ensures learnt embeddings for **central words** are similar to the **context words** with which they tend to appear $\implies$ words appearing in similar contexts will have similar representations

1.3.3 Learning Embeddings

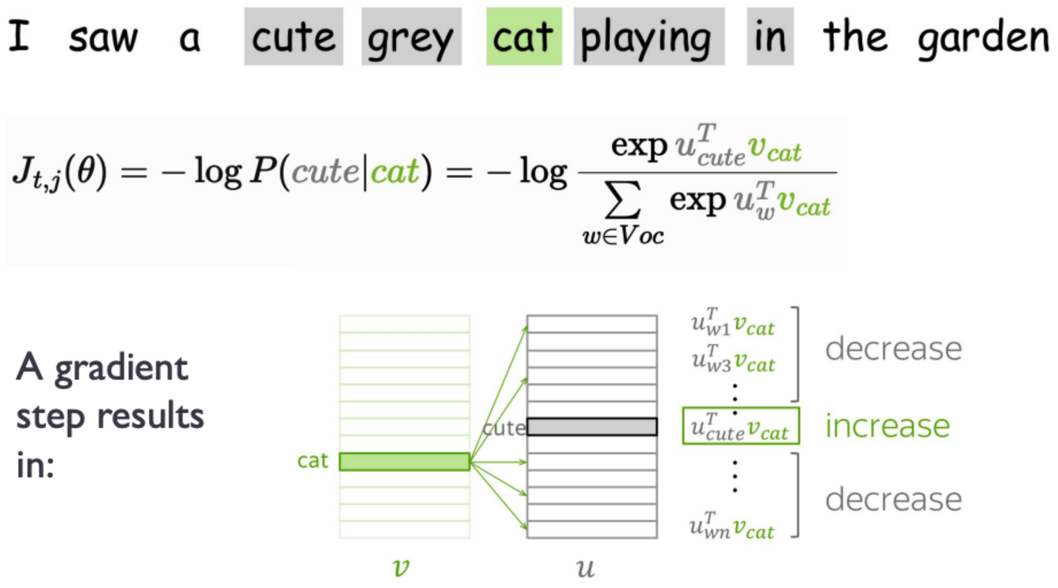
- **How is the model trained?**
  - **Loss Function**: negative log likelihood. If θ is our model parameters (U, V), and each word in a sentence is w_t, then we optimise:

$$J(\theta) = -\frac{1}{T} \times \log(L(\theta)) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log(P(w_{t+j} \mid w_t, \theta))$$

    where:
    * T is the vocabulary size
    * the first sum iterates over each word in the vocabulary
    * the second sum iterates over each context word of the central word w_t
    * we make an independence assumption (i.e w_{t+j} appears in context of w_t independently of other context words)
  - to **optimise**, use **gradient descent**:

$$\theta^{neq} = \theta^{old} - \alpha(\nabla_\theta J(\theta))$$

  - gradient updates are done after a single context word is processed:



I saw a cute grey cat playing in the garden
$J_{t,j}(\theta) = -\log P(cute|cat) = -\log \frac{\exp u_{cute}^T v_{cat}}{\sum_{w \in Voc} \exp u_w^T v_{cat}}$
A gradient step results in:
cat
cute
$u_{w1}^T v_{cat}$
$u_{w3}^T v_{cat}$
decrease
$u_{cute}^T v_{cat}$
increase
$u_{wn}^T v_{cat}$
decrease
v
u

Figure 2: Once the probability of "cute" appearing in the context of "cat" is computed, the gradient is calculated.
The gradient is such that the probability of "cute" appearing with "cat" increases, whilst all other probabilities are decreased.

1.4 Skip-Gram with Negative Sampling

- **What is an issue with the Skip-Gram model above?**
  - **softmax** is expensive: computing the **denominator** involves iterating over the whole vocabulary
- **How does negative sampling reduce the computational cost of Skip-Gram?**
  - **negative sampling** mitigates this by introducing a **hyperparameter** k
  - for each context word, we sample k non-context words
  - probabilities are computed based **only** on the context and non-context words selected (instead of whole vocab)
  - the aim of Skip-Gram with Negative Sampling is to produce θ such that embeddings are **similar** to context words, and **dissimilar** to non-context words
- **How are non-context words sampled?**
  - using **weighted unigram frequency** (with the constraint of never picking a central word):

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w' \in V} count(w')^\alpha}$$

  - typically use $\alpha = 0.75$ so that uncommon words can be sampled
- **How does Skip-Gram look with negative sampling?**
  - **Computing Probability**
    - use logistic regression

    * the probability of o being a context word to c is:

$$P(+ \mid o, c) = \sigma(\underline{u}_0 \cdot \underline{v}_c) = \frac{1}{1 + exp(-\underline{u}_0 \cdot \underline{v}_c)}$$

    * the probability of o being a non-context word to c is:

$$P(- \mid o, c) = 1 - P(+ \mid o, c) = \sigma(-\underline{u}_0 \cdot \underline{v}_c) = \frac{1}{1 + exp(\underline{u}_0 \cdot \underline{v}_c)}$$

  - **Optimising Model**
    * seek to **maximise** similarity of (**center**, **context**) words, whilst **minimising** similarity of (**center**, **non-context**) words
    * use negative log-likelihood. For a given central word c, with context o and non contexts $o_1, \ldots, o_k$ we seek to minimise:

$$J(\theta) = -\left[\log(P(+ \mid o, c)) + \sum_{i=1}^{k} \log(P(- \mid o_i, c))\right] = -\left[\log(\sigma(o \cdot c)) + \sum_{i=1}^{k} \log(-\sigma(o_i \cdot c))\right]$$

- **What word embeddings are used?**
  - can combine both **center** and **context** embeddings:

$$w_t = \underline{u}_t + \underline{v}_t$$

  - can just use center word representation:

$$w_t = \underline{v}_t$$

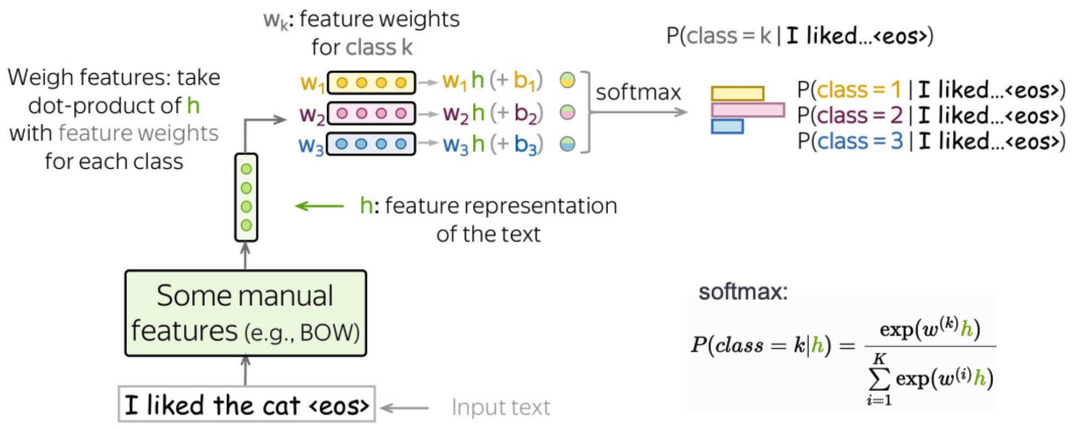
- **How is Skip-Gram related to Latent Semantic Analysis?**
  - optimising Skip-Gram (with negative sampling) is **equivalent** to factorising PMI matrix with SVD

2 Neural Architectures for NLP

2.1 Logistic Regression and NNs: Text Classification

We mean this as an introduction to the versatility of NNs, and to show how they are related to logistic regression.

- **How is logistic regression used to classify text?**
  - text converted to a set of **manual features**
  - sigmoid/softmax is applied to generate a **probability distribution**
  - select most likely class



w_k: feature weights for class k
Weigh features: take dot-product of h with feature weights for each class
w_1 → $w_1 h\ (+b_1)$
w_2 → $w_2 h\ (+b_2)$
w_3 → $w_3 h\ (+b_3)$
softmax
P(class = k | I liked...<eos>)
P(class = 1 | I liked...<eos>)
P(class = 2 | I liked...<eos>)
P(class = 3 | I liked...<eos>)
h: feature representation of the text
Some manual features (e.g., BOW)
I liked the cat <eos>
Input text
softmax:
$P(class = k|h) = \frac{\exp(w^{(k)}h)}{\sum_{i=1}^{K}\exp(w^{(i)}h)}$

- **How do NNs differ from standard logistic regression?**
  - no need to create a **feature representation**: NNs derive it by using the token (word) embeddings of the text
  - weights also learnt with gradient descent + -ve log likelihood

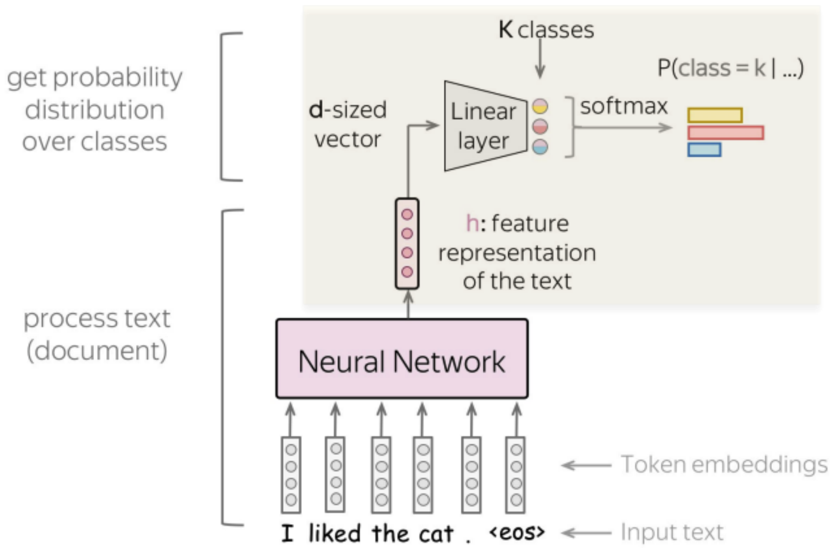


Figure 3: The power of NNs comes from being able to create very good feature representations. Classifying just requires the use of logistic regression (highlighted part).

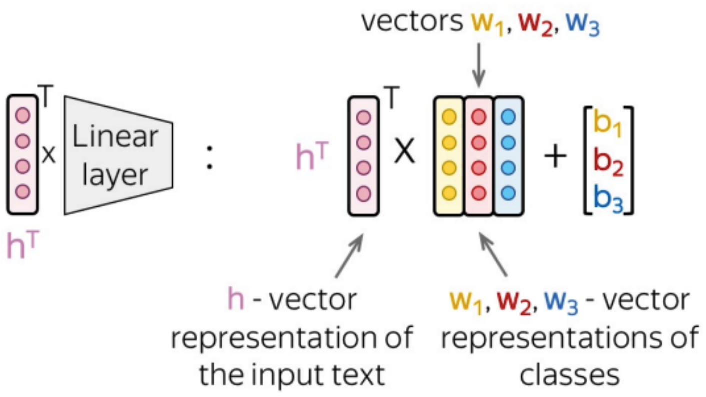

vectors w_1, w_2, w_3
h^T
x
Linear layer
:
h^T
X
+
b_1
b_2
b_3
h - vector representation of the input text
w_1, w_2, w_3 - vector representations of classes

Figure 4: The linear layer is our standard procedure of multiplying the input freatures by the weights.

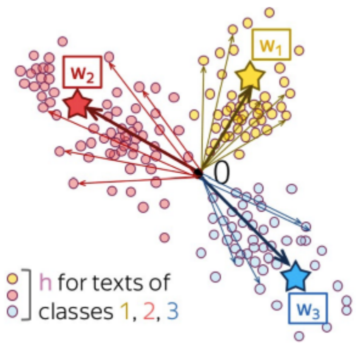


Figure 5: Intuitively, the weights "point" in the direction of a class representation.
The longer a vector h, the more confident a classifier is in the classification.

- **What are the simplest feature representations which can be made by using the word embeddings?**
  - **Sum of Embeddings**: create feature vector by adding the embedding vectors (too simple: representation invariant given word order)
  - **Weighted Sum of Embeddings**: weight each embedding (quite good, simple, still used in practice)
    - can use **tf-idf**
    - upweights words important for classification, downweight words unimportant (i.e unusual across documents)
    - measures how unusual a given word is in a document:

$$tf - idf(w, d, D) = tf(w, d) \times idf(w, D)$$

      where:
      - tf is the **term frequency** - how often w appears in document d:

$$tf(w, d) = count(w, d)$$

        Altenative: apply log

$$tf(w, d) = \log_1 0(count(w, d) + 1)$$

        (add 1 in case count is 0)

· idf is the **inverse document frequency -** the reciprocal of the proportion of documents out of D in which w appears (typically squashed using log, due to large number of documents):

$$idf(w, D) = \log\left(\frac{|D|}{|\{d \in D \mid w \in d\}|}\right)$$

idf increases the weight of words appearing in a few documents (since these words can be indicative of a given class, whilst words which are common across **all** documents are less useful)

2.2 Recurrent Neural Networks

- **What is a recurrent neural network?**
  - network containing a **cycle** in its units
  - good for processing **sequential data**
  - composed of **RNN cells**

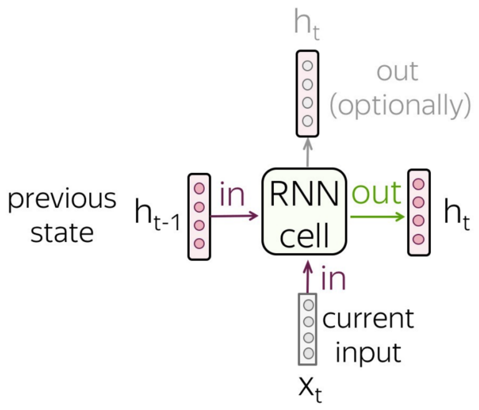


Figure 6: A RNN cell take 2 inputs: h_{t-1} (hidden layer from preceding cell) and x_t (current input).

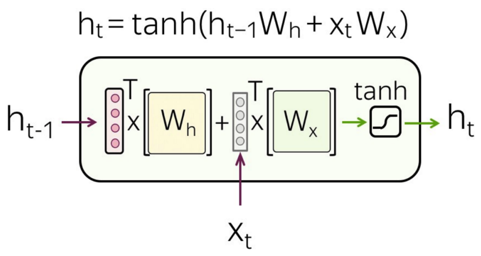


Figure 7: A vanilla RNN cell. The matrices W_h, W_x are the same throughout each RNN cell, and are used to produce the **hidden layer** of the cell (alongside the non-linear function tanh/sigmoid). The RNN cell can also produce an output, by using a third matrix W_k and a non-linear function g. The output will be $g(W_k h_t)$.

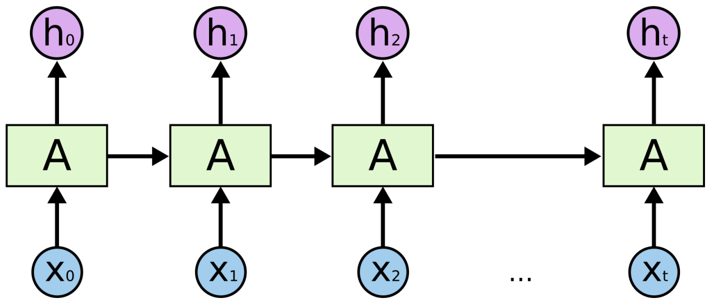
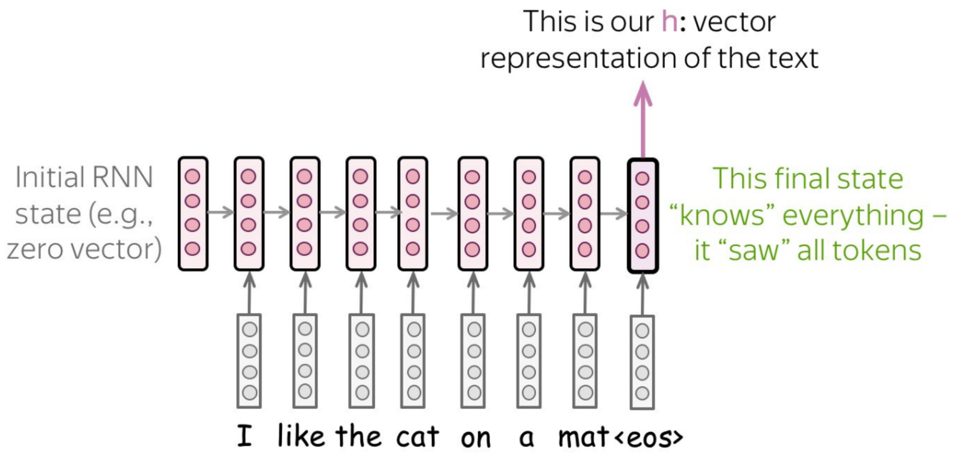


Figure 8: A full RNN is composed by "concatenating" RNN cells. The output of the network is thus the output of the final RNN cell.

- **How are RNNs trained?**
  - loss at time t requires the hidden layer at $t-1$
  - loss at time $t+1$ influenced by hidden layer at t
  - computing the gradient involes a technique known as **Backpropagation Through Time**
- **Do RNNs have a notion of memory?**
  - each cell receives a hidden state, dependent on the history of the input sequence: this can be regarded as a **memory**
  - however, if RNN is too "long", the influence of the starting input elements can become negligible (need more sophisticated architectures to handle this)

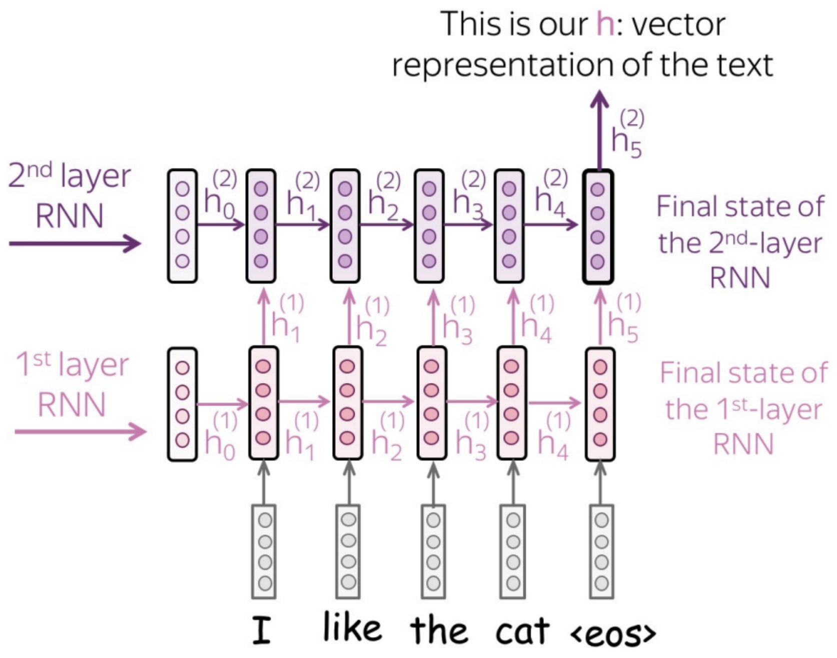
2.3 RNNs for Text Representation

- **How can RNNs be used for text representation?**



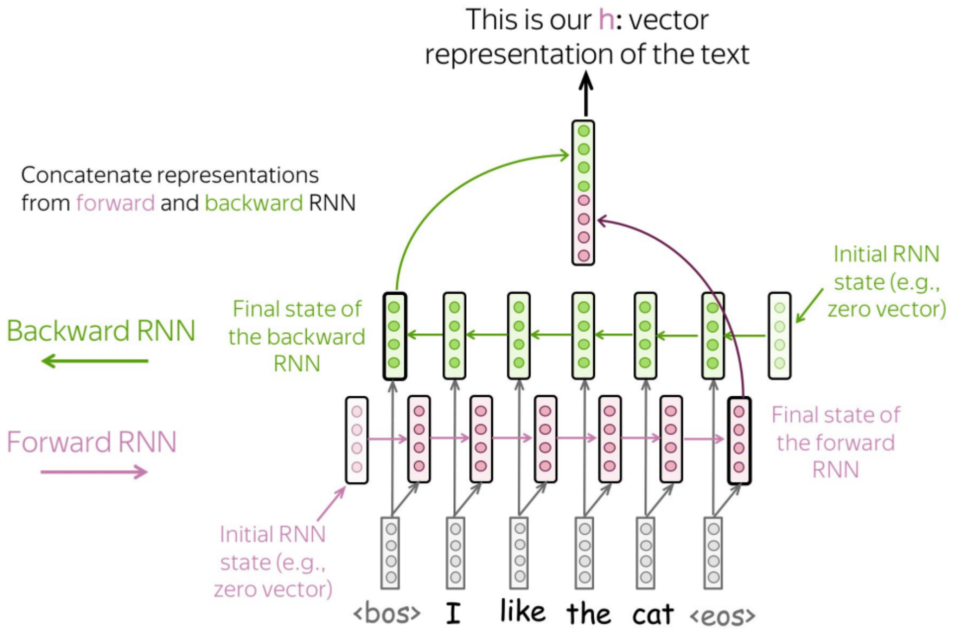
This is our h: vector representation of the text
Initial RNN state (e.g., zero vector)
This final state "knows" everything – it "saw" all tokens
I like the cat on a mat<eos>

Figure 9: Most straightforward: give a sentence to the RNN, and take the output of the last cell as the vector representation of sentence.



This is our h: vector representation of the text
$h_5^{(2)}$
2nd layer RNN
$h_0^{(2)}$ $h_1^{(2)}$ $h_2^{(2)}$ $h_3^{(2)}$ $h_4^{(2)}$
Final state of the 2nd-layer RNN
$h_1^{(1)}$ $h_2^{(1)}$ $h_3^{(1)}$ $h_4^{(1)}$ $h_5^{(1)}$
1st layer RNN
$h_0^{(1)}$ $h_1^{(1)}$ $h_2^{(1)}$ $h_3^{(1)}$ $h_4^{(1)}$
Final state of the 1st-layer RNN
I like the cat <eos>

Figure 10: The output of intermediate cells can be used as input for another RNN; in this way, we can stack RNNs to get more rich, complex text representations.



This is our h: vector representation of the text
Concatenate representations from forward and backward RNN
Initial RNN state (e.g., zero vector)
Backward RNN
Final state of the backward RNN
Forward RNN
Final state of the forward RNN
Initial RNN state (e.g., zero vector)
<bos> I like the cat <eos>

Figure 11: Another alternative is to use 2 RNNs: one processes the input from left to right; the other from right to left. The text representation is obtained by concatenating the produced vector representations.

2.4 RNNs for Language Modelling

A ***language model*** *allows us to compute the* ***probability*** *of a word occurring given its history:*

$$P(y_1, \dots, y_n) = \prod_{t=1}^{n} P(y_t \mid y_{<t})$$

- **Can RNNs be used as language models?**
  - say $\underline{x}_t$ is a word embedding for a token in a sentence
  - the hidden layer of a RNN cell computes:

    $$\underline{h}_t = g(W_h \underline{h}_{t-1} + W_x \underline{x}_t)$$

  - for a **language model**, we can make the output of the cell:

    $$\underline{y}_t = softmax(W_k \underline{h}_t)$$

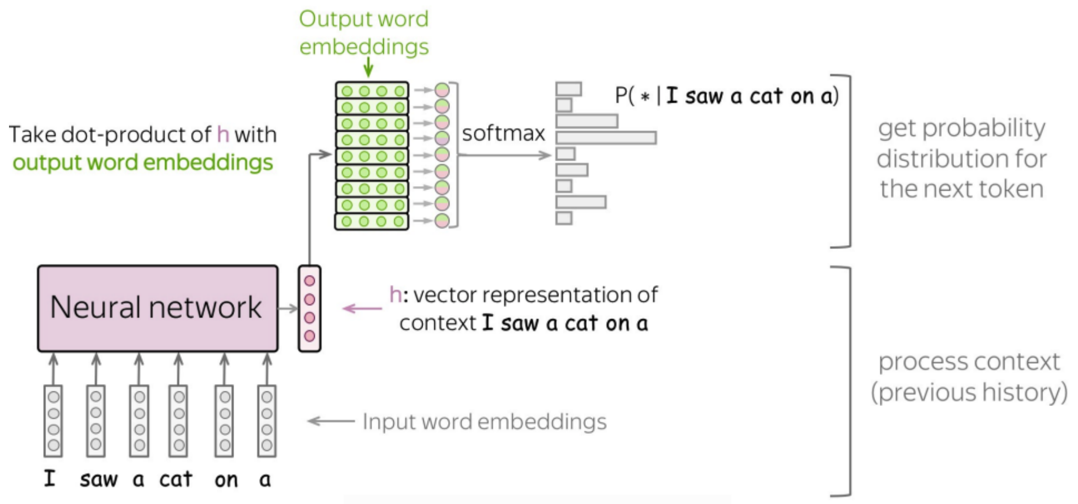
  - $\underline{y}_t$ represents a **probability distribution** for the $t+1$th word in the sequence, such that $\underline{y}_t[i]$ is the probability of the ith word in the vocabulary:

    $$P(w_{t+1} = w_i \mid w_1, \dots, w_t) = \underline{y}_t[i]$$

  - the probability of a whole sequence then becomes:

    $$P(w_{1:n}) = \prod_{i=1}^{n} \underline{y}_i[w_i^{(s)}]$$

    where $\underline{y}_i[w_i^{(s)}]$ is the probability of the expected word in the ith position of the sequence


Output word embeddings
Take dot-product of h with output word embeddings
softmax
P(* | I saw a cat on a)
get probability distribution for the next token
Neural network
h: vector representation of context I saw a cat on a
process context (previous history)
Input word embeddings
I saw a cat on a

$$p(y_t|y_{<t}) = \frac{exp(h_t^T e_{y_t})}{\sum\limits_{w \in V} exp(h_t^T e_w)}$$

Figure 12: Basic structure of an RNN as a LM.

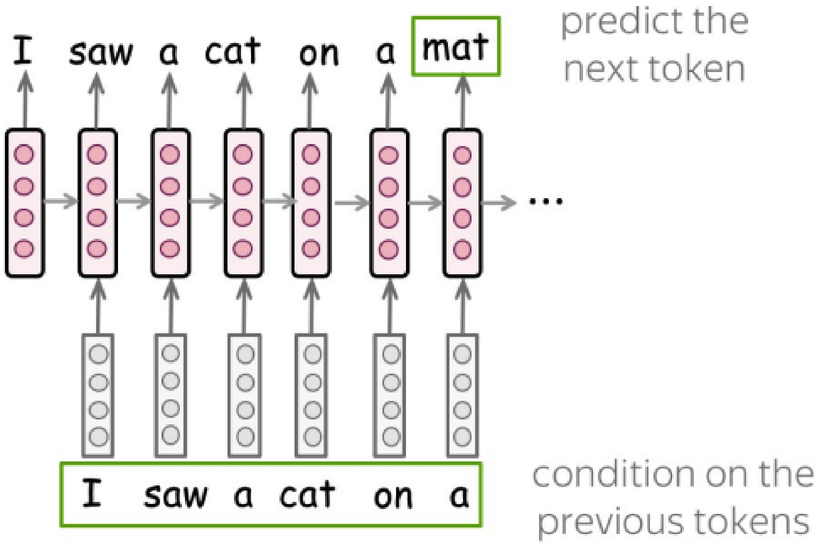


Figure 13: High level RNN LM with one layer.

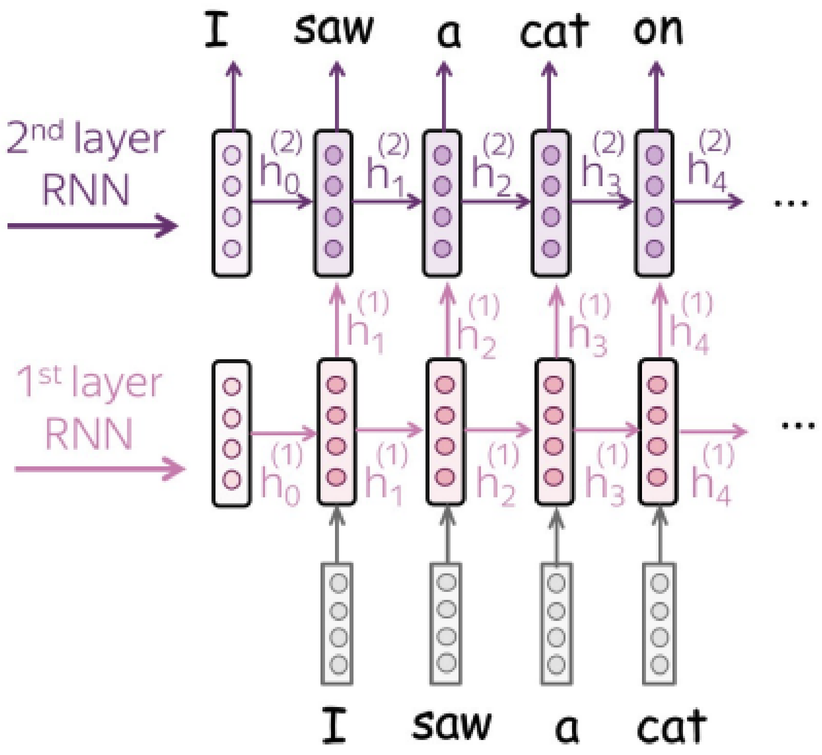


Figure 14: High level RNN LM with two layer.

- **How does the RNN Language Model learn its weights?**

- use **cross-entropy** for loss:

$$L_{CE} = -\sum_{w \in V} \underline{y}_t[w] \log(\underline{\hat{y}}_t[w])$$

- for a language model, the true word $\underline{y}_t[w]$ is a one-hot vector, and the loss is with respect to the word predicted by the model after word t:

$$L_{CE}(\underline{\hat{y}}_t, \underline{y}_t) = -\log(\underline{\hat{y}}_t[w_{t+1}])$$

- this way of learning is **teacher forcing**: label for $t+1$ predicted by using the **true** label (word) at time t (instead of the label predicted by the model in the previous iteration)

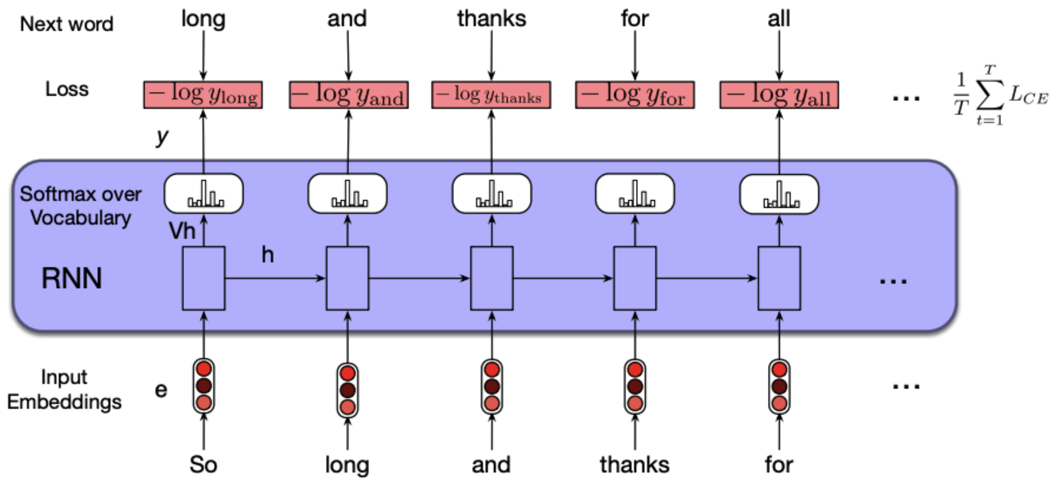
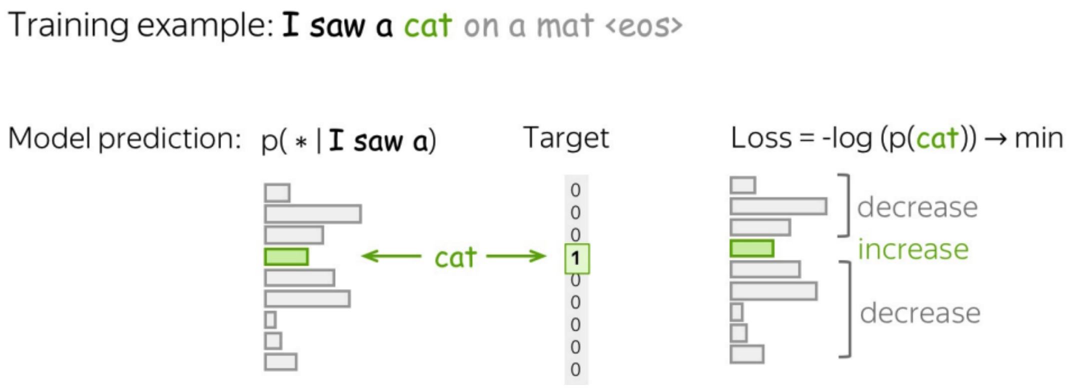


Figure 15: RNN during training.



Training example: I saw a cat on a mat <eos>
Model prediction: p(* | I saw a)
cat
Target
0
0
0
1
0
0
0
0
0
Loss = -log (p(cat)) → min
decrease
increase
decrease

Figure 16: The process of learning for a RNN.

2.5 RNNs for Sequence-to-Sequence Modelling

- **What is the task of machine translation?**
  - given a sequence x, generate a sequence y (translation of x in different language):

$$\hat{y} = \underset{y}{argmax} P(y \mid x, \theta)$$

- need to know:
  1. how to model $P(y \mid x, \theta)$?
  2. how to find θ?
  3. how to find $argmax$?

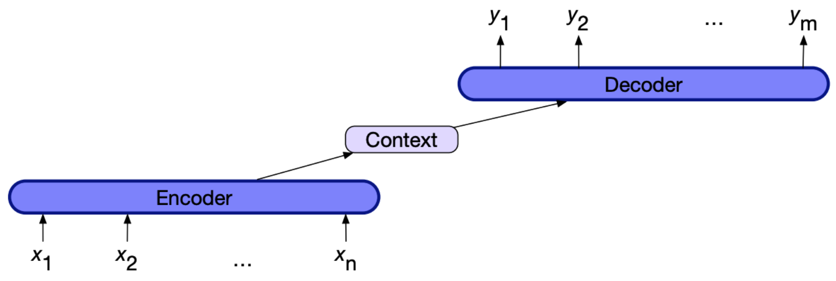
- **How can machine translation be seen as a language modelling problem?**
  - want to translate a sentence x into $y_1, \ldots, y_n$
  - equivalent to finding:

$$P(y_1, \ldots, y_n \mid x)$$

  - as a language model:

$$P(y_1, \ldots, y_n \mid x) = \prod_{t=1}^{n} P(y_t \mid y_{<t}, x)$$

- **What is an encoder-decoder framework?**
  - model used to perform **sequence-to-sequence** conversions via:
    * an **encoder**: takes **input sequence**, converts to **compressed representation**
    * a **decoder**: takes **compressed representation**, converts to **output sequence**



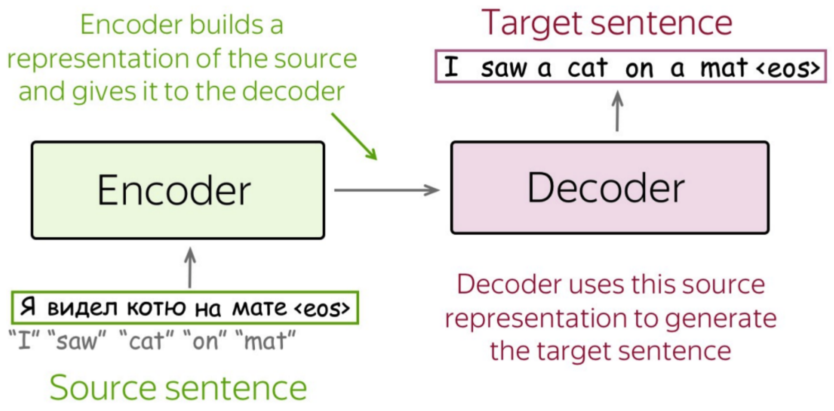

Encoder builds a representation of the source and gives it to the decoder
Target sentence
I saw a cat on a mat <eos>
Encoder
Decoder
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Source sentence
Decoder uses this source representation to generate the target sentence

Figure 17: Encoder-decoder framework for translation (Russian to English).

- **What is a neural framework for an encoder-decoder framework?**

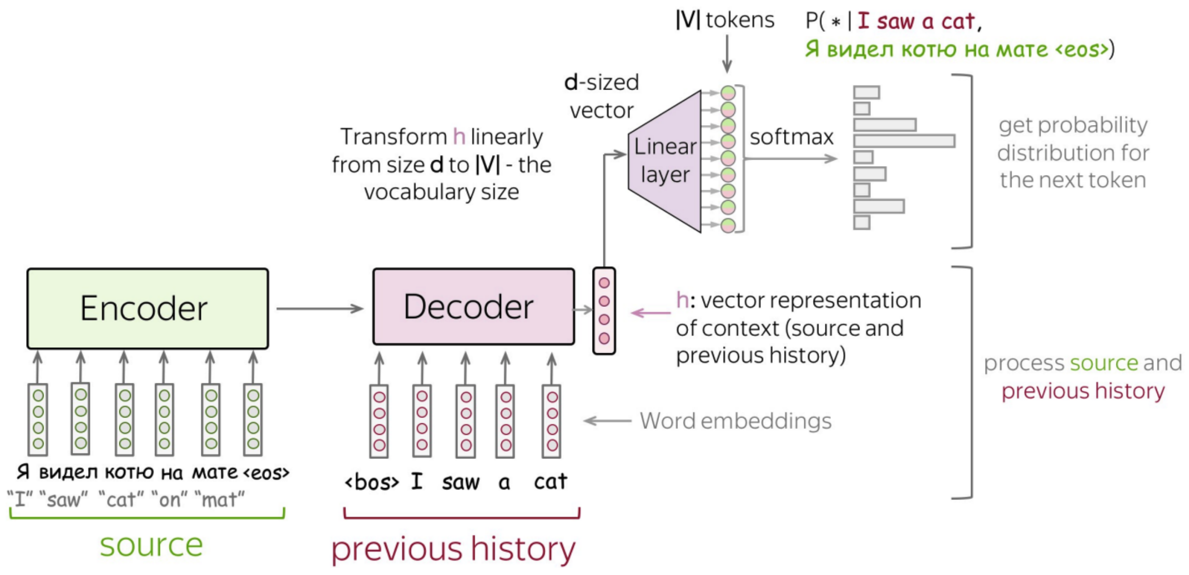
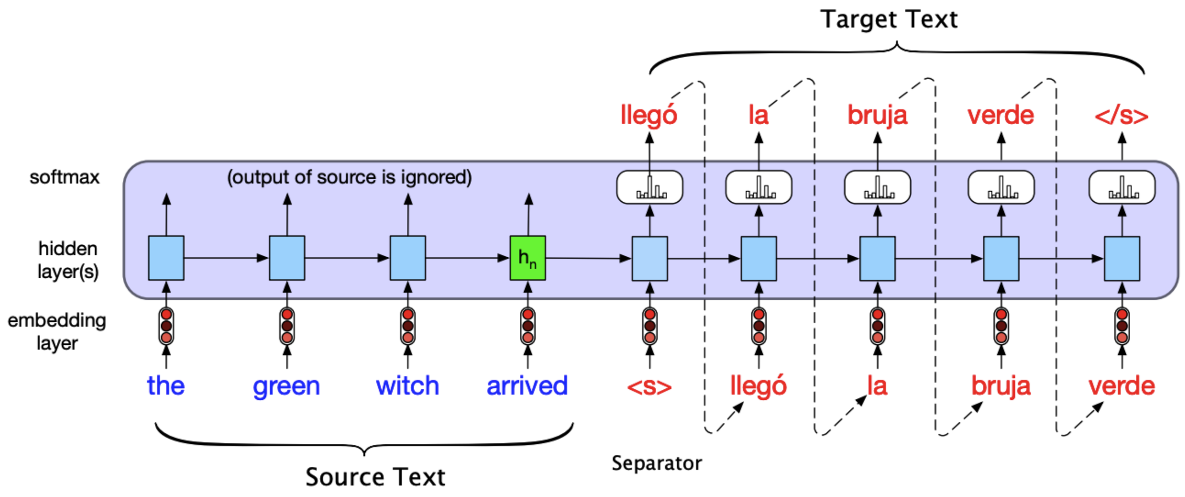
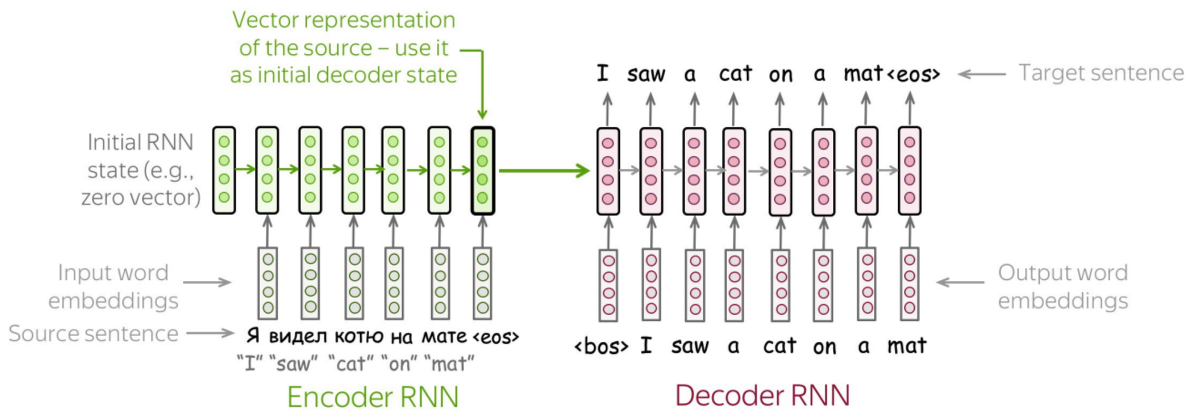

|V| tokens
P(* | I saw a cat, Я видел котю на мате <eos>)
d-sized vector
Transform h linearly from size d to |V| - the vocabulary size
Linear layer
softmax
get probability distribution for the next token
Encoder
Decoder
h: vector representation of context (source and previous history)
process source and previous history
Word embeddings
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
source
<bos> I saw a cat
previous history

Figure 18: As we saw above, machine translation can be seen as language modelling, which is nothing but text classification. A neural approach to this follows the following:
- Convert input sequence into representative vector via encoder
- Using the decoder + representation vector + previous history, generate an output vector
- Convert output vector (via softmax) into a probability distribution for the next word y_t

- **How can RNNs be used for an Encoder-Decoder Framework**
  - use RNNs for encoder and decoder
  - **encoder RNN** generates a **vector representation** of the sentence to be translated
  - **decoder RNN** uses representation vector as an input
  - at step t, generates possible translation, using the translation token from previous step




Vector representation of the source – use it as initial decoder state
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Encoder RNN
I saw a cat on a mat<eos>
Target sentence
Output word embeddings
<bos> I saw a cat on a mat
Decoder RNN

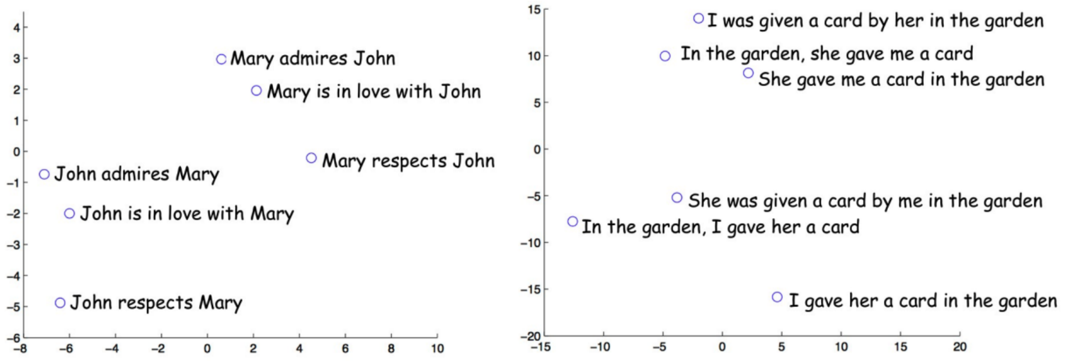


Figure 19: Result of encoded vector. Paraphrases appear close together in representation space.

- **Why use 2 different RNNs?**
  - could be possible to use a single RNN to take y as input, and give output x
  - however, this is messy and harder to train (would be directly translating)
- **How is the Encoder-Decoder Framework trained?**
  - same as all previous models (negative log likelihood)
  - encoder and decoder trained **simultaneously**
  - update weights for each decoded word

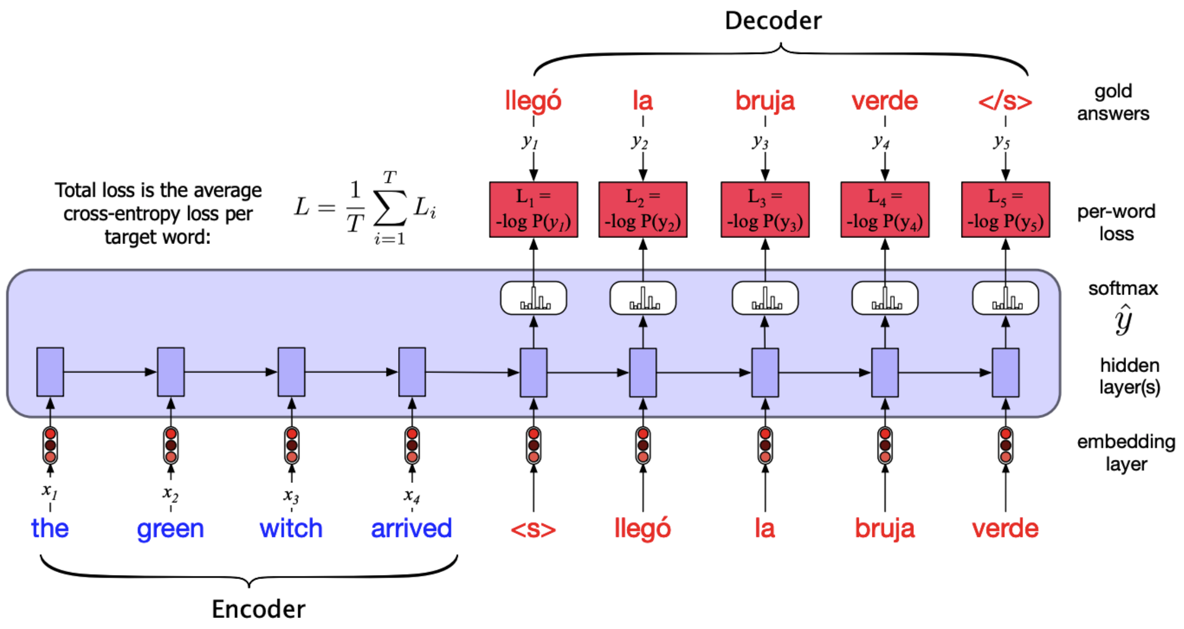
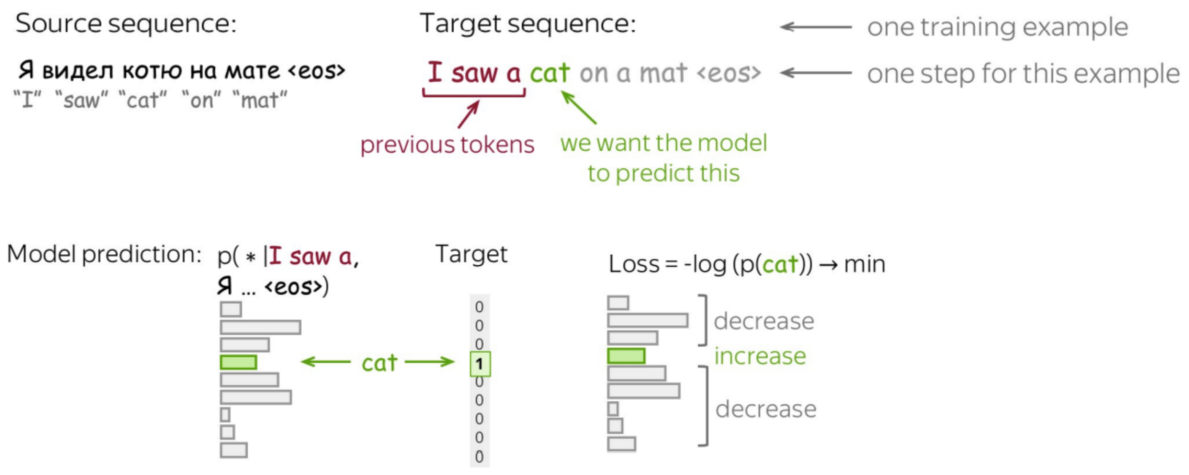


Figure 20: Training data composed oftuples of source and translated text.
For training decoder, use **teacher forcing**: output for $t+1$ based on a gold label input
When the decoder **generates** text, output for $t+1$ depends on its output at t.


Source sequence:
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Target sequence:
I saw a cat on a mat <eos>
one training example
one step for this example
previous tokens
we want the model to predict this
Model prediction: p(* |I saw a, Я ... <eos>)
cat
Target
0 0 0 1 0 0 0 0 0
Loss = -log (p(cat)) → min
decrease
increase
decrease

- **How can decoding be improved?**
  - currently, follow **greedy decoding**: use argmax to pick most likely translated token at each time
  - greedy strategy doesn't guarantee optimal translation
  - to improve: treat translation as a search problem through a tree of possible interpretations
  - use **beam search** to find best translation: expand the k most likely paths (based on $P(y_i \mid x, y_{<i})$)
  - process ends when an end token (`</s>`) is generated

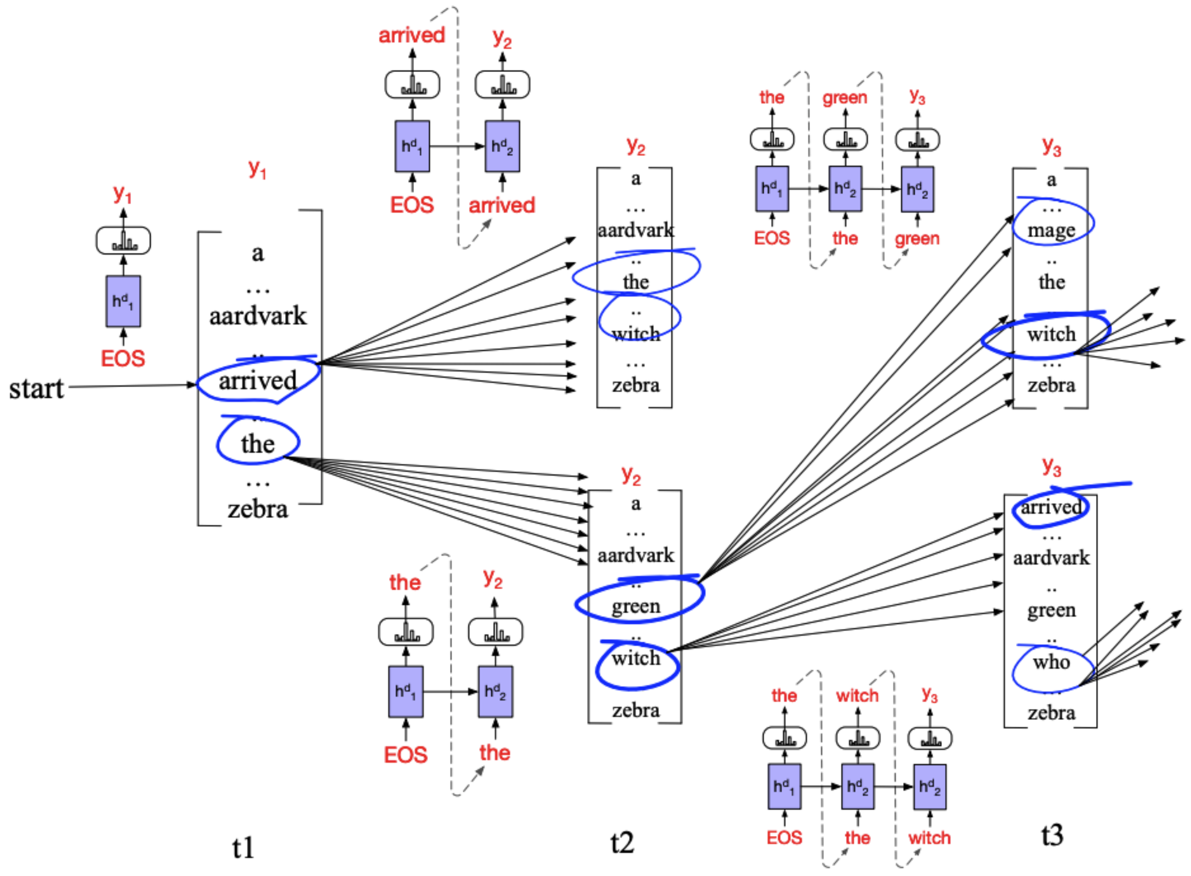


Figure 21: Beam search process, with $k = 2$. At the first step, we expand the search by considering as the first word in the translation either "arrived" or "the". At the next iteration, the best 4 paths are "the" and "witch" (for "arrived") and "green" and "witch" (for "the"). We expand based on the 2 most likely out of "arrived the", "arrived witch", "the green" and "the witch". The latter 2 are most likely, so once again, we expand based on them.
If an animation is clearer, check this out.

- **What are the key issues of the RNN Encoder-Decoder Framework?**
  - the vector representation produced by the encoder is a **bottleneck**:
    1. **Encoder**: needs to represent the full text in a single vector; info from start of sentence might not be represented
    2. **Decoder**: different parts of sentence might be relevant at different points of decoding, but a single representation given
- **How does attention mitigate these problems?**
  - **attention** creates a **dynamic** vector representation by using a **weighted sum** of all the **hidden states** in the encoder
  - these states hold information about specific parts of the input sentence
  - for each token being decoded, the weighting changes, as to shift the **attention** to certain parts of the sentence

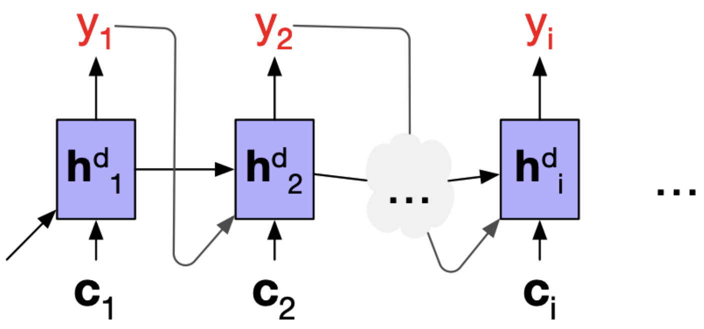


Figure 22: Each hidden state in the decoder h_i^d gets access to a dynamic vector representation c_i, based on a weighted sum of the hidden states in the encoder.

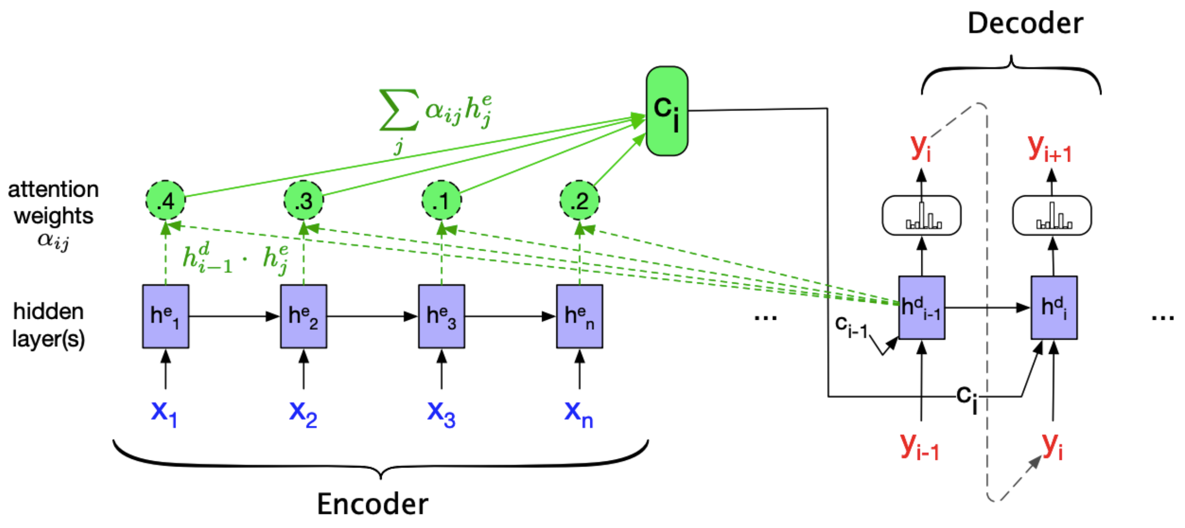
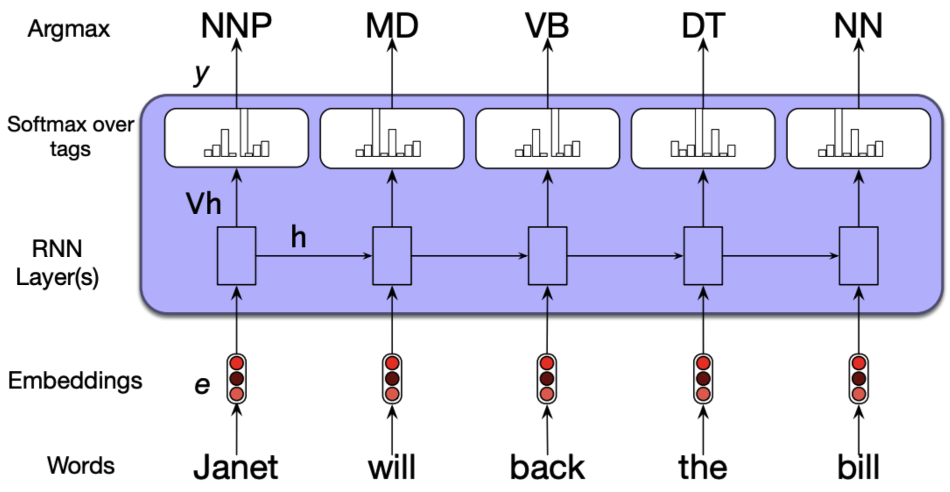


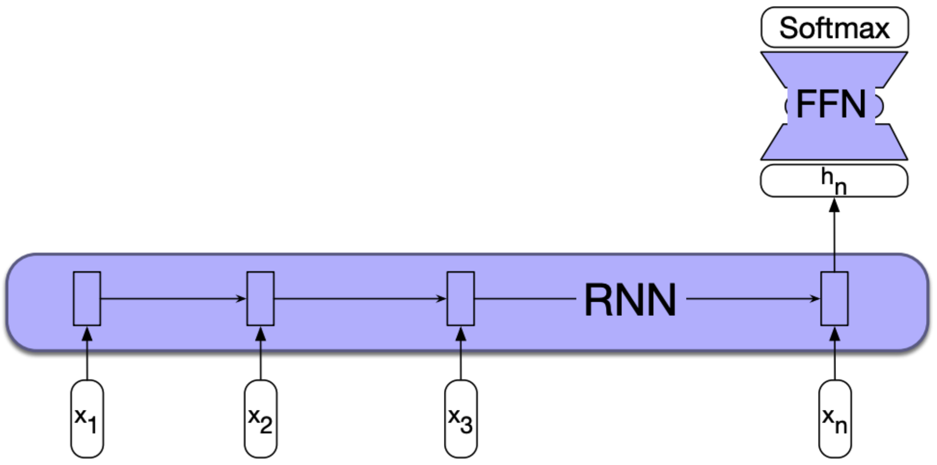
Figure 23: One simple weighting involves taking the dot product of a decoder hidden state with each of the encoder hidden states; this produces a vector of weighting scores, which can be normalised via softmax.

2.6 Further Applications of RNNs for NLP Tasks

- **How can RNNs be used for sequence labelling?**
  - make output of RNN cell a **distribution** over labels (i.e **POS tags**



- **How can RNNs be used for text classification?**
  - already used to create a **text representation**; just use softmax to change this into a distribution
  - loss for network based solely on output from last RNN cell (**end-to-end training**)



3 Discourse Coherence

3.1 Motivating Discourse Coherence

- **What is pragmatics?**
  - study of how meaning is conveyed, beyond what is **linguistically explicit**
- **What is discourse?**
  - a **coherent, structured** group of sentences
  - **coherence** differentiates **discourse** from a random group of sentences
  - **discourse coherence** is a way of studying **pragmatics**: coherence allows us to make inferences, beyond what is written/said

"It's a beautiful night.
We're looking for something dumb to do.
Hey baby, I think I wanna marry you."
We can infer that Bruno Mars thinks that marriage is something dumb to do, even if it is not linguistically explicit.
This inference is based on ***coherence****: it wouldn't make much sense to continue with the notion of marriage, if marriage were unrelated to the act of doing something dumb.*

- **Is coherence a black and white concept?**
  - no: some sentences can be more coherent than others
  - for example, "Jane took a train from Paris to Istanbul. She had to attend a conference." is more coherent than "Jane took a train form Paris to Istanbul. She likes spinach."
  - both are coherent, but the latter one requires that we make up some form of motivation, as to why both sentences are related (i.e there is a famous spinach market in Istanbul, which Jane is interested in)
- **What are coherence relations?**
  - a way of formalising the **semantic relations** existing between parts of text
  - relational types of speech act: success of conveying content depends on the content itself, and the other content to which it is related

3.1.1 Motivation: Pronouns and Coherence

Pronouns are generally meaningless. However, we interpret them such that we maximise the coherence of a sentence.

"John can open Bill's safe.
He knows the combination."

- coherence tells us that "he" refers to "John"
- this is because the second sentence is an **explanation** of the first sentence (*explanation relation*
- if "he" referred to "Bill": still coherent (i.e *continuation relation* - discussing the topic "Bill"), but very vague (so less coherent)

"John can open Bill's safe.
He should change the combination."

- coherence now tells us that "he" refers to "Bill"
- *result relation*: John can open Bill's sage **so/therefore** Bill should change the combination
- if "he" referred to "John": still coherent, but we wouldneed to make up a justification (i.e John wants to change the combination so that Bill can no longer access it)

3.1.2 Motivation: Coherence and Time

We can infer event order based on what is more coherent (i.e haveing cause-effect relations).

> *"Max fell. John helped him up."*

- coherence tells us that "falling" happened before "helping"

> *"Max fell. John pushed him."*

- coherence tells us that "falling" happened after "pushing"

> *"John hit Max on the back of his neck.*
> *Max fell. John pushed him.*
> *Max rolled over the edge of the cliff."*

- changing context changes inferred order of actions
- coherence tells us that "pushing" happened after "falling"
- it would be unorderly to have a structure of the form `cause` ("hit") → `effect` ("fell") → `further cause` ("pushed") (even if "falling" typically occurs after "pushing", coherence tells us to seek an orderly structure of events)

3.1.3 Motivation: Word Meaning and Coherence

> *Q: "Did you buy the apartment?"*
> *A: "Yes, but we rented it."*

- coherence indicates that A is a **landlord**
- has bought and apartment, and is renting it
- wouldn't make sense if tenant did this (i.e would need to make up a backstory, like they bought it, sold it and then rented it)

> *Q: "Did you buy the apartment?"*
> *A: "No, but we rented it."*

- coherence indicates that A is a **tenant**
- hasn't bought and apartment, and is renting it
- wouldn't make sense if landlord did this (i.e would need to make up a backstory)

3.1.4 Motivation: Bridging and Coherence

*"John took an engine from Avon to Dansville.
He picked up a boxcar."*

- coherence tells us that the "boxcar" was in Dansville
- the event of "picking up" occurs once the event of "taking" finishes (coherence relation of *narration*)

*"John took an engine from Avon to Dansville.
He also took a boxcar."*

- coherence tells us that the "boxcar" was in Avon
- *parallel coherence relation*: there is a parallelism between "engine" and "boxcar"

3.1.5 Motivation: Discourse Coherence and Implicit Agreement

*a - M (to K and S): Karen 'n' I're having a fight,
b - M (to K and S): after she went out with Keith and not me.
c - K (to M and S): Wul Mark, you never asked me out.*

- coherence allows us to infer agreement
- K is agreeing with M: she went out with Keith (c is **justifying** b), and further agrees with the existence of the fight (b explains a, so if c justifies b, c agrees with a)
- K commits to a and b
- no explicit agreement - it is often implicated

3.1.6 Motivation: Discourse Coherence and Dishonesty

*a - P: Do you have any bank accounts in Swiss banks, Mr. Bronston?
b - B: No, sir.
c - P: Have you ever?
d - B: The company had an account there for about size months, in Zurich.*

- famous case: Bronston was found to have account in Geneva
- b is true: at the time of the question, he didn't have an account

- d can be taken as an indirect answer to c, implying "no" (for this, the prosecutor charged him with perjury)
- however, also coherent to say that d is a follow up to b

3.1.7 Motivation: Coherence as Driving Understanding of Gestures

Gestures oftentimes provide information which complements and remarks what is said. The interpretation of these gestures will oftentimes depend on the context in which they are used.

*In fact, **coherence relations** can connect speech and gesture and sequences of gestures: speech **so that** gesture; speech **by** gesture; or speech **and moreover** gesture.*

3.2 Representing Discourse Coherence

- **Why are coherence relations key to study pragmatics?**
  - as we have seen, **coherence relations** allows us to infer and understand implicit knowledge
  - for example, pronoun resolution, temporal inference, agreement, etc ...
- **What are examples of coherence relations?**
  - Narration
  - Explanation
  - Background
  - Contrast
  - Parallel
  - Question-Answer
  - Correction
- **What is Segmented Discourse Representation Theory?**
  - theory which Alex (Lascarides) worked on developing
  - used to handle the **representation** of coherence relations
- **What does SDRT consiste of?**
  - represents **logical form** of discourse via:
    1. A **set of labels**
       * $A = \{\pi_1, \pi_2, \ldots, \pi_n\}$
       * each label represents a **segment** of discourse
    2. A **mapping** $\mathcal{F}$
       * $\mathcal{F}$ maps a label π_i of a segment to a formula of the content of the segment
       * these formula can include **coherence relations**. For example, $\mathcal{F}$ can map a label to $Elaboration(\pi_1, \pi_2)$
  - SDRT can thus generate a **hierarchical structure** to the discourse units (which froms subsegments), based on units which cohere with each other

– thus, according to SDRT, a **coherent discourse** can be thought as a tree with a single root node, as this indicates that the discourse is composed of coherent subsegments

π_1: John can open Bill's safe.
π_2: He knows the combination.

$$\begin{aligned}
\pi_0 &: \quad \textit{Explanation}(\pi_1, \pi_2)\\
\pi_1 &: \quad \iota x(\textit{safe}(x)\ \&\ \textit{possess}(x, \textit{bill})\ \&\ \textit{can}(\textit{open}(e_1, \textit{john}, x))\\
\pi_2 &: \quad \iota y(\textit{combination}(y)\ \&\ \textit{of}(y, x)\ \&\ \textit{knows}(\textit{john}, y))
\end{aligned}$$

Figure 24: This is an example for a **monologue**.
Parts in red are **not** linguistically explicit, but constructed using **pragmatics** to provide the maximally coherent interpretation of the sentences.
We can see that for example, coherence tells us that in π_2, the combination y is part of the safe x .- this is inferred, and not linguistically explicit.
π_0 can be thought of as the interpretation of the text. This interpretation is **coherent**, since it is a root node for the whole text.

$$\begin{aligned}
\pi_0 &: \quad \textit{Explanation}(\pi_1, \pi_2)\\
\pi_1 &: \quad \iota x(\textit{safe}(x) \wedge \textit{possess}(x, \textit{bill}) \wedge \textit{can}(\textit{open}(e_1, \textit{john}, x))\\
\pi_2 &: \quad \iota y(\textit{combination}(y) \wedge \textit{of}(y, x) \wedge \textit{knows}(\textit{john}, y))
\end{aligned}$$

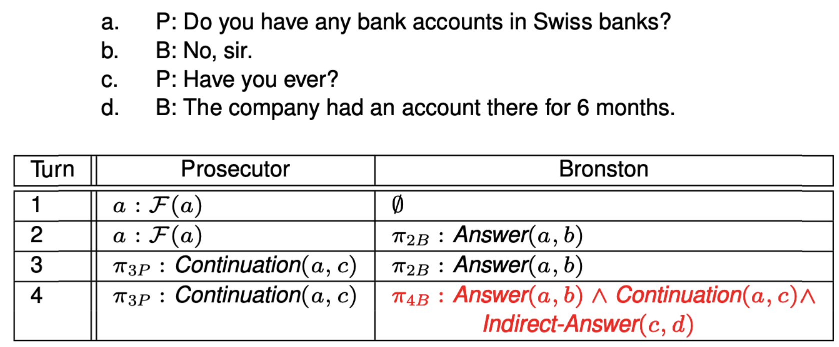
$$\begin{aligned}
[\mathcal{F}(\pi_0)]\text{iff}\quad & [\textit{Explanation}(\pi_1, \pi_2)]\\
\text{iff}\quad & \mathcal{F}(\pi_1) \wedge \mathcal{F}(\pi_2) \wedge \varphi_{\textit{Expl}}(\pi_1, \pi_2)\\
\text{iff}\quad & \iota x(\textit{safe}(x) \wedge \textit{possess}(x, \textit{bill}) \wedge \textit{can}(\textit{open}(e_1, \textit{john}, x))\wedge\\
& \quad \iota y(\textit{combination}(y)\ \&\ \textit{of}(y, x) \wedge \textit{knows}(\textit{john}, y))\wedge\\
& \quad\quad \wedge \textit{cause}(e_{\pi_2}, e_{\pi_1})
\end{aligned}$$

Figure 25: This wasn't part of lectures, but I'm assuming its giving the formula which defines $\mathcal{F}(\pi_0)$.

a. M (to K and S): Karen 'n' I're having a fight,
b. M (to K and S): after she went out with Keith and not me.
c. K (to M and S): Wul Mark, you never asked me out.

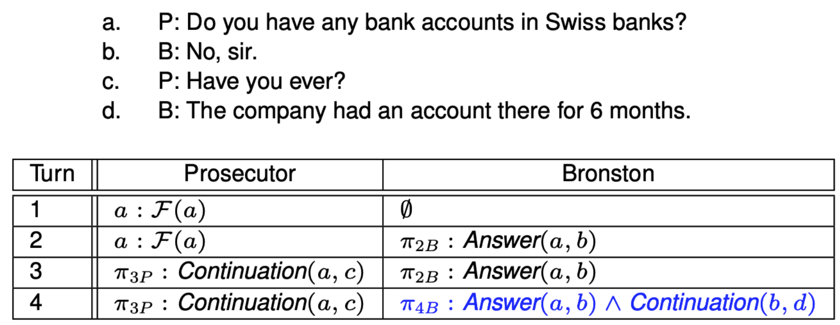
Turn	M	K
1	π_{1M} : *Explanation*(a, b)	$\emptyset$
2	π_{1M} : *Explanation*(a, b)	π_{2K} : *Explanation*$(a, b)\wedge$ *Explanation*(b, c)

Figure 26: In **dialogues**, SDRT keeps track of **public commitments** made by each agent in the dialogue. These commitments might change as the dialogue progresses.
In this example, at the start only M talks, committing b as an explanation for a.
After K speaks, M's commitment doesn't change, whilst K is committed not only to her explanation of b via c, but also agrees with M's explanation (as discussed above, her justification can be thought as agreement with Ms premise).


a. P: Do you have any bank accounts in Swiss banks?
b. B: No, sir.
c. P: Have you ever?
d. B: The company had an account there for 6 months.

Turn	Prosecutor	Bronston
1	$a : \mathcal{F}(a)$	$\emptyset$
2	$a : \mathcal{F}(a)$	π_{2B} : *Answer*(a, b)
3	π_{3P} : *Continuation*(a, c)	π_{2B} : *Answer*(a, b)
4	π_{3P} : *Continuation*(a, c)	π_{4B} : *Answer*$(a, b) \wedge$ *Continuation*$(a, c) \wedge$ *Indirect-Answer*(c, d)

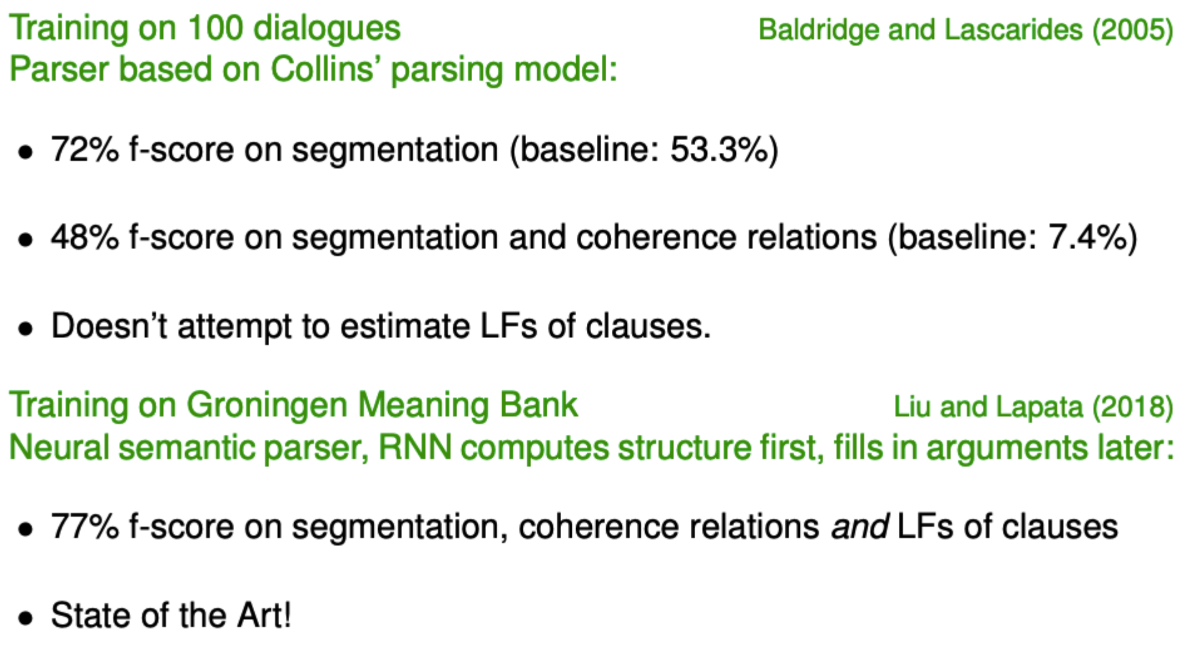
Figure 27: This is the prosecutor's interpretation of Bronston's answer: he assumes that Bronston is still committed to his answer b, and takes d as a response to c under this commitment.


a. P: Do you have any bank accounts in Swiss banks?
b. B: No, sir.
c. P: Have you ever?
d. B: The company had an account there for 6 months.

Turn	Prosecutor	Bronston
1	$a : \mathcal{F}(a)$	$\emptyset$
2	$a : \mathcal{F}(a)$	π_{2B} : *Answer*(a, b)
3	π_{3P} : *Continuation*(a, c)	π_{2B} : *Answer*(a, b)
4	π_{3P} : *Continuation*(a, c)	π_{4B} : *Answer*$(a, b) \wedge$ *Continuation*(b, d)

Figure 28: This is an alternative interpretation, by which Bronston follows his answer to a with b and then d. This is coherent, and does **not** imply that he has a ban k account in Switzerland.

- **Can logical forms be constructed symbolically?**
  - brittle approach
  - need to use **consistency tests** to check validity of implied content (very hard in FOL - beyond recursively enumerable, which is the least expressibility you need)
  - can't construct logical form in same logic as used to evaluate
  - given a logical form, can't know whether you believe/accept it (difference between providing an expression, and providing your interpretation/understanding of said expression)
- **Can machine learning be used to learn a discourse parser?**
  - best approach
  - however, very few corpora have discourse annotations
- **How does supervised learning perform?**



Training on 100 dialogues — Baldridge and Lascarides (2005)
Parser based on Collins' parsing model:

- 72% f-score on segmentation (baseline: 53.3%)
- 48% f-score on segmentation and coherence relations (baseline: 7.4%)
- Doesn't attempt to estimate LFs of clauses.

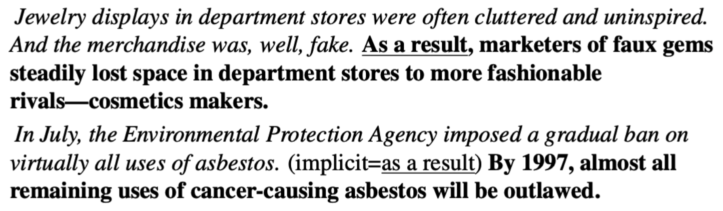
Training on Groningen Meaning Bank — Liu and Lapata (2018)
Neural semantic parser, RNN computes structure first, fills in arguments later:

- 77% f-score on segmentation, coherence relations *and* LFs of clauses
- State of the Art!

- **Can we use automatic annotation to improve performance?**
  - idea: **cue phrases** (i.e "because") might indicate some coherence relation (i.e *explanation)*; whenever cue phrase appears, annotate with corresponding relation

> *"Max fell because John pushed him."*
> $\Longrightarrow$
> *Explanation(Max fell, John pushed him)*

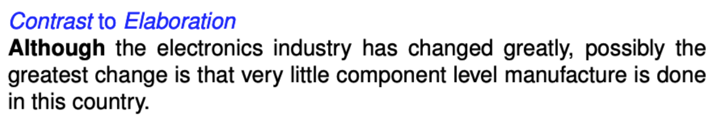
  - doesn't work in practice: many coherence relations lack cue phrases



Jewelry displays in department stores were often cluttered and uninspired. And the merchandise was, well, fake. **As a result, marketers of faux gems steadily lost space in department stores to more fashionable rivals—cosmetics makers.**

In July, the Environmental Protection Agency imposed a gradual ban on virtually all uses of asbestos. (implicit=as a result) **By 1997, almost all remaining uses of cancer-causing asbestos will be outlawed.**

  - moreover, removing a cue phrase from a sentence might change its coherence relation (so model can be prone to mislabel a sentence which is similar to a training sentence, but missing a cue phrase)

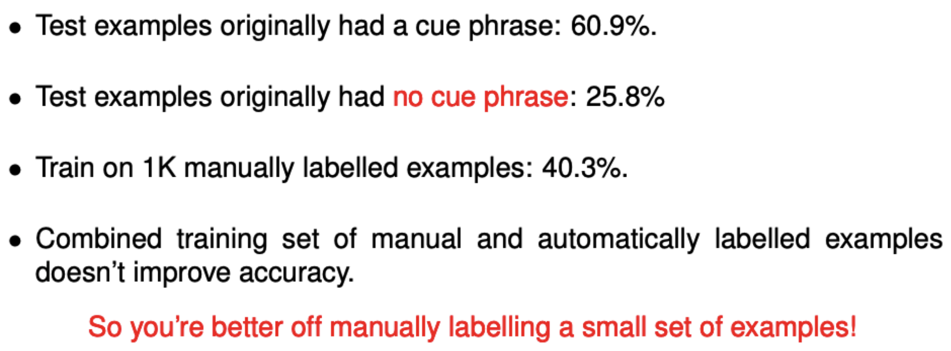


Contrast to *Elaboration*

Although the electronics industry has changed greatly, possibly the greatest change is that very little component level manufacture is done in this country.

Figure 29: With although, we have *contrast.*
Removing it leads to *elaboration.*
However, model will learn to assign *contrast* to such sentences, not realising that the lack of “although” changes the coherence relation.

– if you have a cue phrase, you get good result; otherwise, catastrophic:



- Test examples originally had a cue phrase: 60.9%.
- Test examples originally had no cue phrase: 25.8%
- Train on 1K manually labelled examples: 40.3%.
- Combined training set of manual and automatically labelled examples doesn’t improve accuracy.

So you’re better off manually labelling a small set of examples!