IN4080 – 2022 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

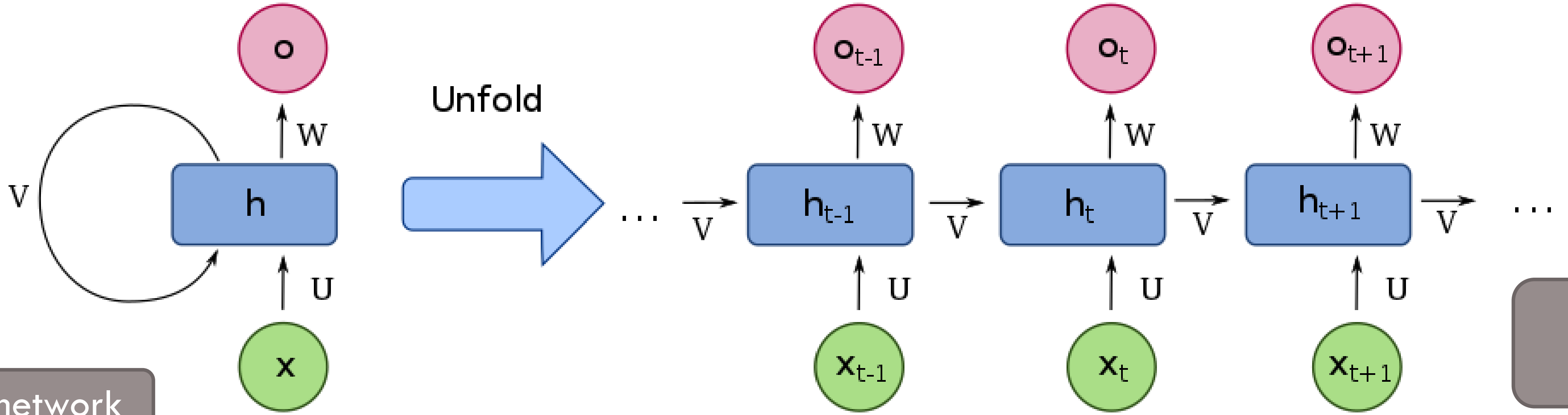
Lecture 13, Nov. 17 (&24)

This lecture

- Recurrent Neural Networks
  - RNN Language Models
  - Other applications of RNNs
  - Extended architectures and challenges
- Encoder-Decoder models and Machine Translation
- Transformers as Language Models
- Information extraction – some loose ends

Recurrent neural nets

- Model sequences/temporal phenomena
- A cell may send a signal back to itself – at the next moment in time

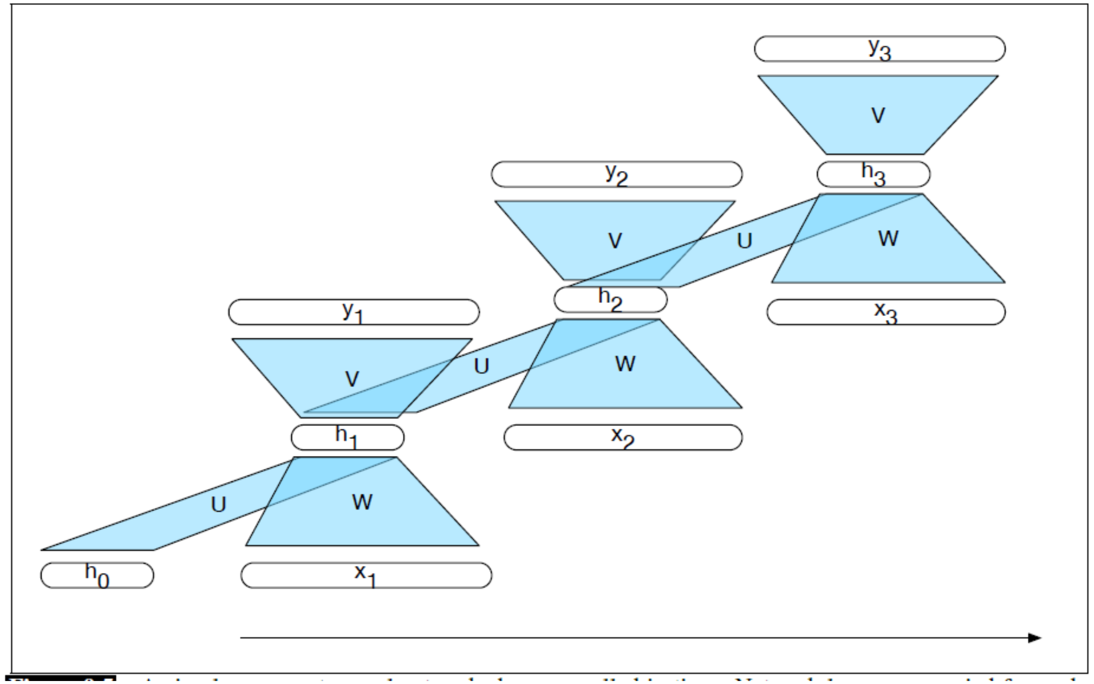


The network

The processing during time

https://en.wikipedia.org/wiki/Recurrent_neural_network

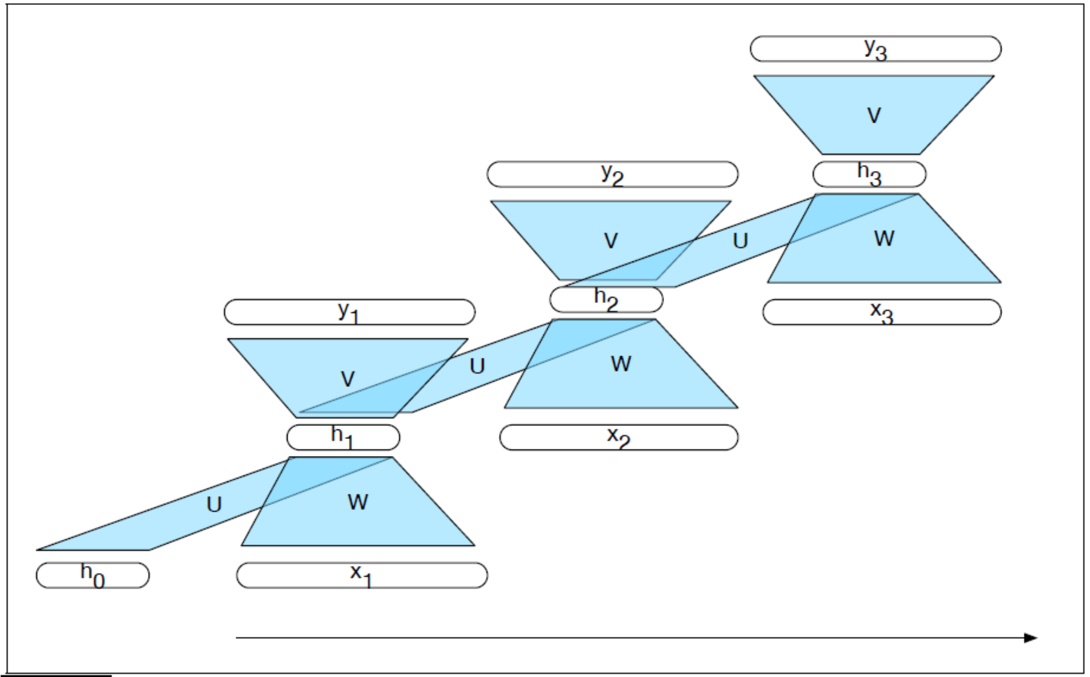
Forward



From J&M, 3.ed., 2019

- Each U, V and W are edges with weights (matrices)
- $x_1, x_2, \ldots, x_n$ is the input sequence
- $y_1, \ldots, y_n$ is the output sequence
- Forward:
  - (Initialize h_0)
  - For $i = 1$ to n:
    - Calculate h_i from h_{i-1} and x_i,
    - calculate y_i from h_i

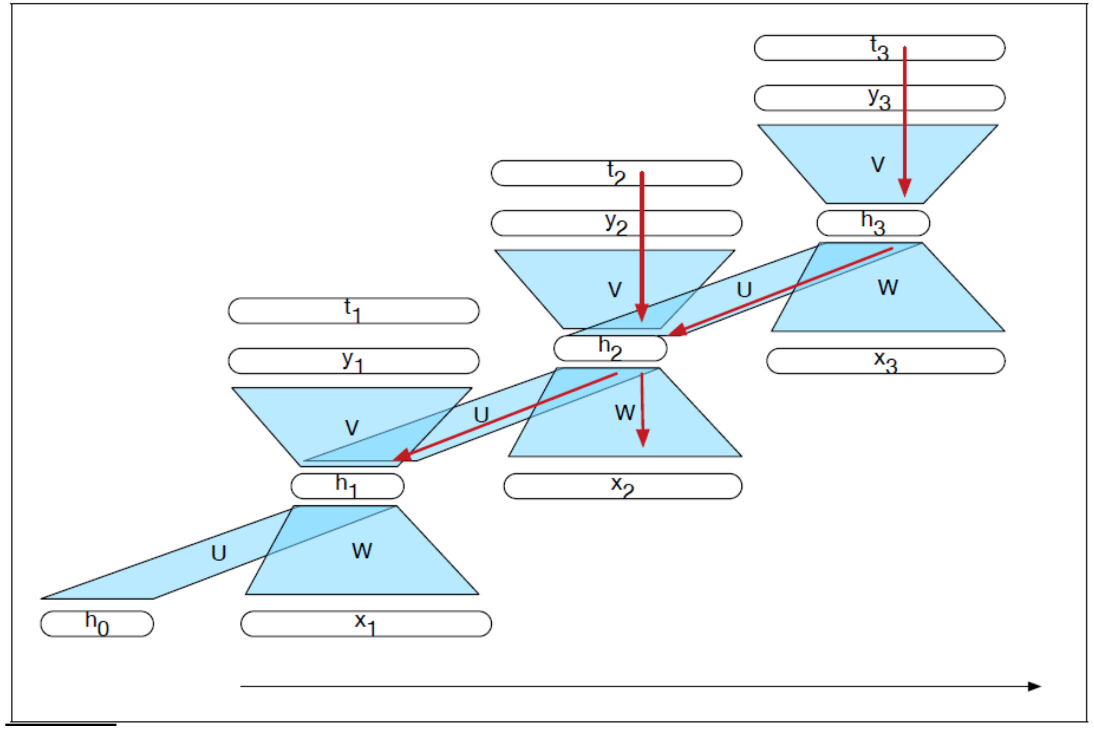
Forward



From J&M, 3.ed., 2019

- $\boldsymbol{h}_t = g(U\boldsymbol{h}_{t-1} + W\boldsymbol{x}_t)$
- $\boldsymbol{y}_t = f(V\boldsymbol{h}_t)$
- g and f are activation functions
- (There are also bias which we didn't include in the formulas)

Training



From J&M, 3.ed., 2019

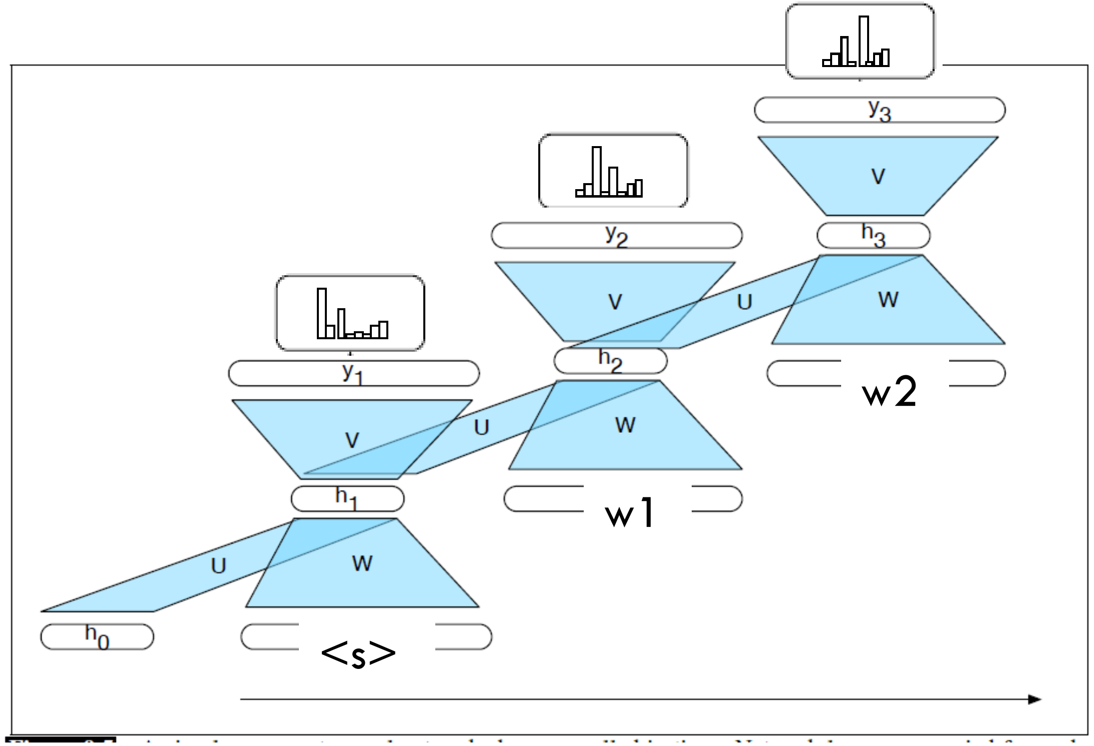
Process one sequence:

- At each output node:
  - Calculate the loss and the
  - δ-term
- Back-propagate the error, e.g.
  - the δ-term at $h_2$is calculated
    - from the δ-term at h_3 by U and
    - the δ-term at y_2 by V
- Update
  - V from the δ-terms at the y_i-s and
  - U and W from the δ-terms at the h_i-s

This lecture

- Recurrent Neural Networks
  - RNN Language Models
  - Other applications of RNNs
  - Extended architectures and challenges
- Encoder-Decoder models and Machine Translation
- Transformers as Language Models
- Information extraction – some loose ends

RNN Language model



From J&M, 3.ed., 2019

- $\hat{y} = P(w_n | w_1^{n-1}) = softmax(V\boldsymbol{h}_n)$
- In principle:
  - unlimited history
  - a word depends on all preceding words
- The word w_i is represented by an embedding
  - or a one-hot and the embedding is made by the LM

Training

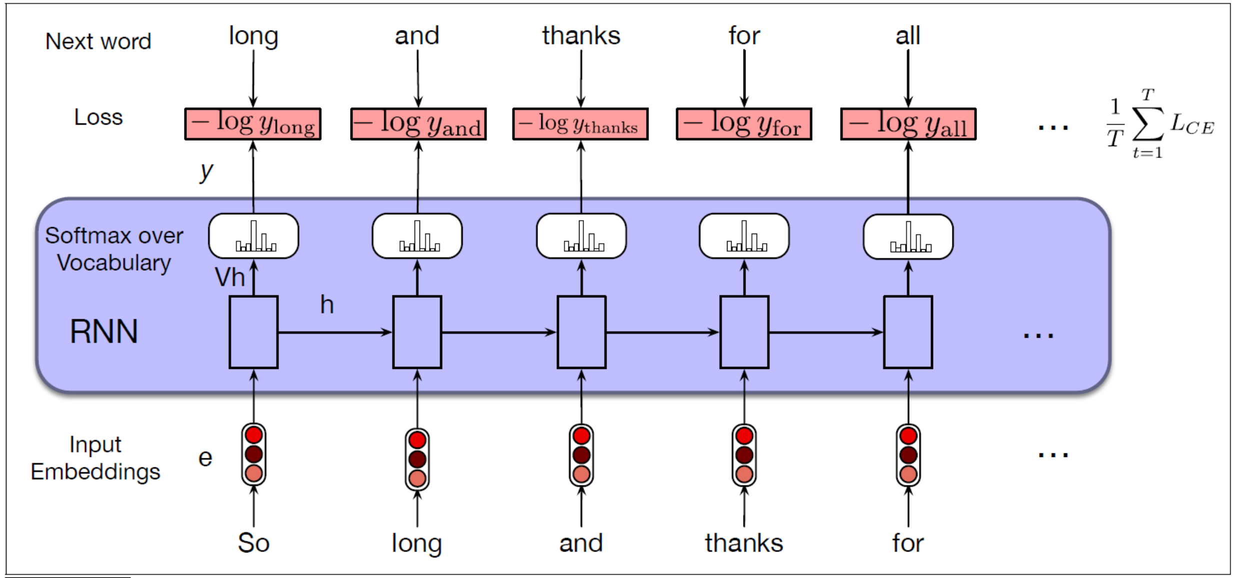


Figure 9.6 Training RNNs as language models.

From J&M, 3.ed., 2022

- The predicted output is a softmax prob.distribution over the vocabulary
- The target is the next word
- Cross-entropy loss compare the two
- The errors are back-propagated through the network
- The weights get updated.

This lecture

- Recurrent Neural Networks
  - RNN Language Models
  - Other applications of RNNs
  - Extended architectures and challenges
- Encoder-Decoder models and Machine Translation
- Transformers as Language Models
- Information extraction – some loose ends

Neural sequence labeling: e.g., tagging

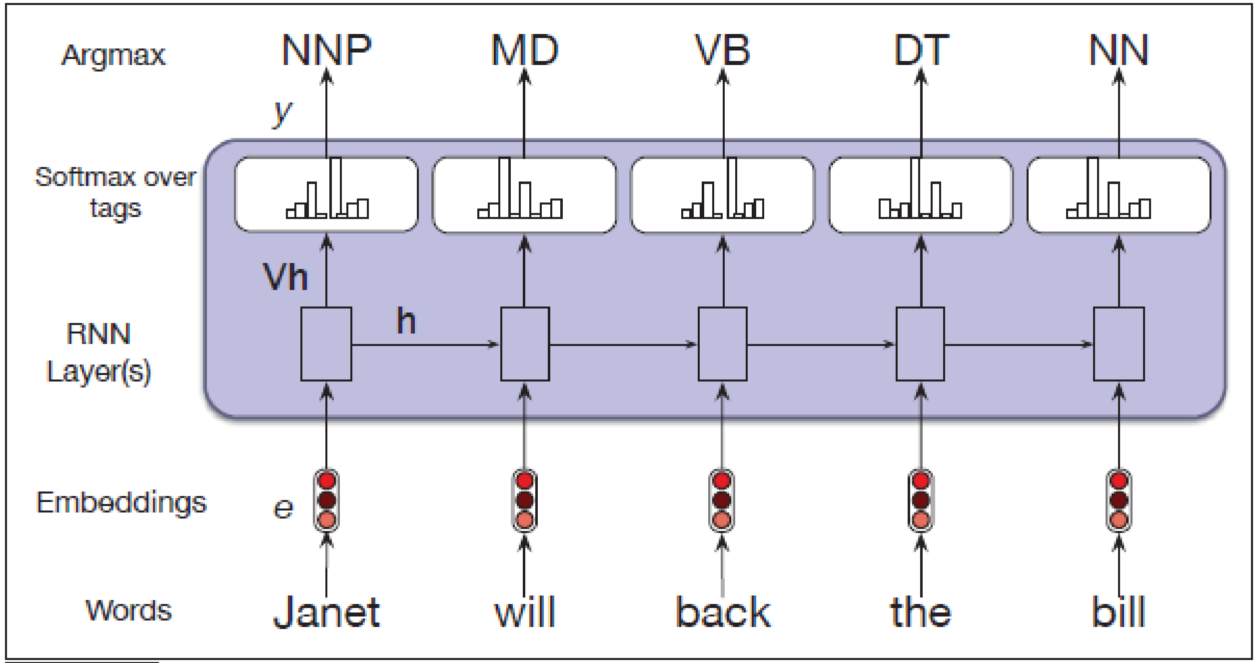


Figure 9.7 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

- $\hat{y} = P(t_n | w_1^n) = softmax(V\boldsymbol{h}_n)$

From J&M, 3.ed., 2021

Neural sequence classification

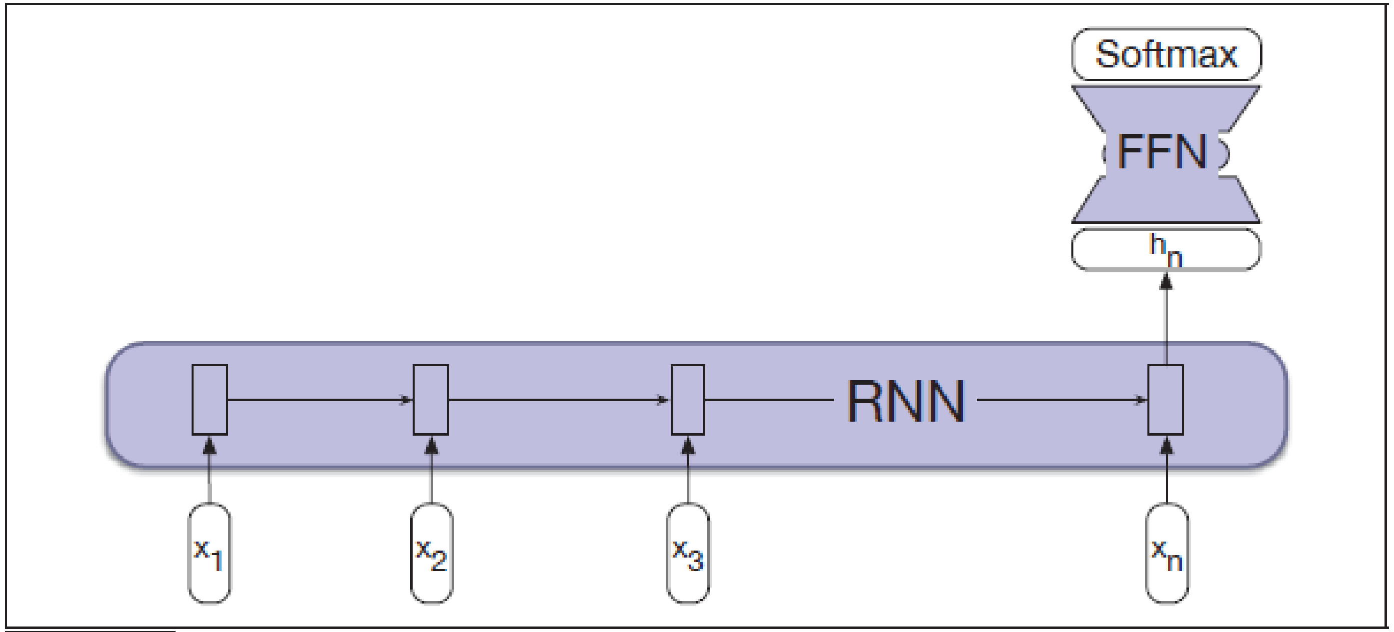


Figure 9.8 Sequence classification using a simple RNN combined with a feedforward network. The final hidden state from the RNN is used as the input to a feedforward network that performs the classification.

From J&M, 3.ed., 2021

- E.g., sentiment
- Consider only the final state
- Might have a feed-forward network before the softmax.

Autoregressive generation

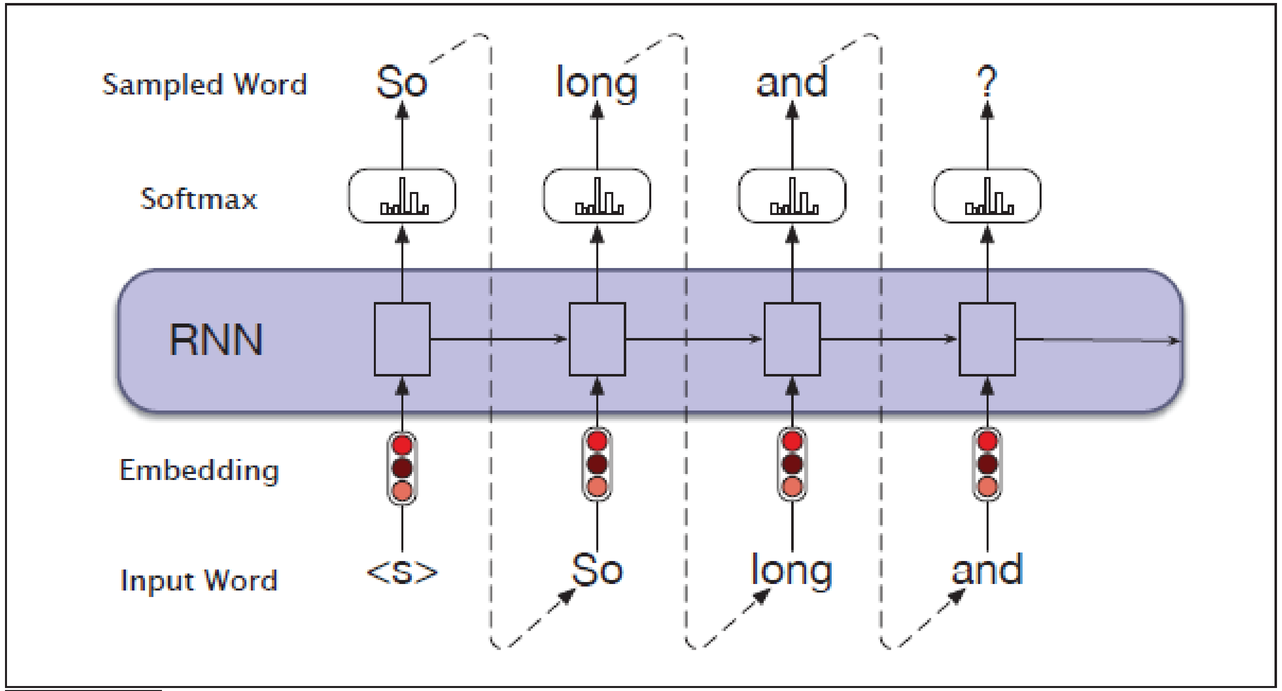


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

From J&M, 3.ed., 2021

- From a trained neural LM, generate text
- Initialize with some (or no) words
- The network will at each stage:
  - Alt. 1: pick the most likely word (argmax)
  - Alt. 2: sample a word randomly according to the softmax probability

This lecture

- Recurrent Neural Networks
  - RNN Language Models
  - Other applications of RNNs
  - Extended architectures and challenges
- Encoder-Decoder models and Machine Translation
- Transformers as Language Models
- Information extraction – some loose ends

Sequence labeling

- Actual models for sequence labeling, e.g. tagging, are more complex
- For example, it may take words after the tag into consideration.

Stacked RNN

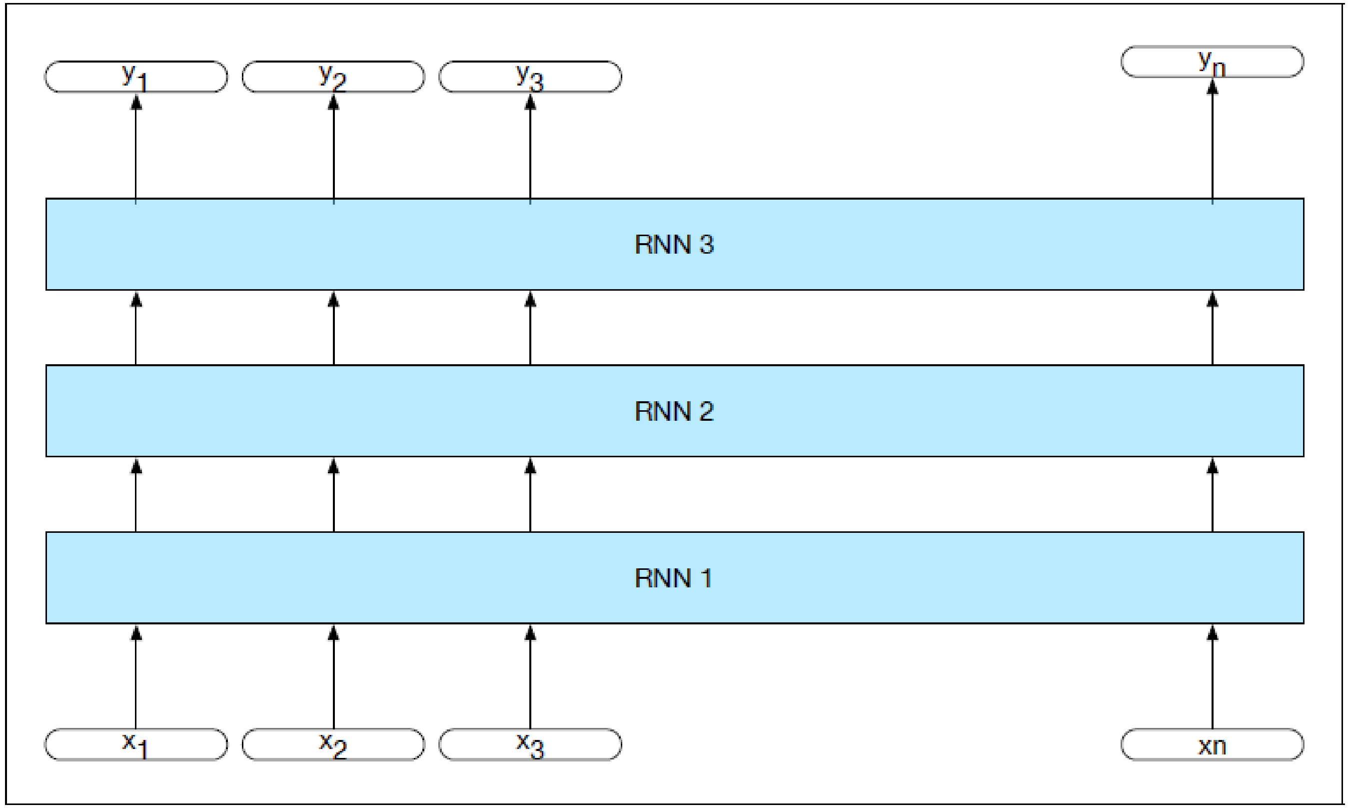


Figure 9.10 Stacked recurrent networks. The output of a lower level serves as the input to higher levels with the output of the last network serving as the final output.

From J&M, 3.ed., 2019

- Can yield better results than single-layers
- Reason?
  - Higher-layers of abstraction
    - similar to image processing (convolutional nets)

Bi-directional RNN

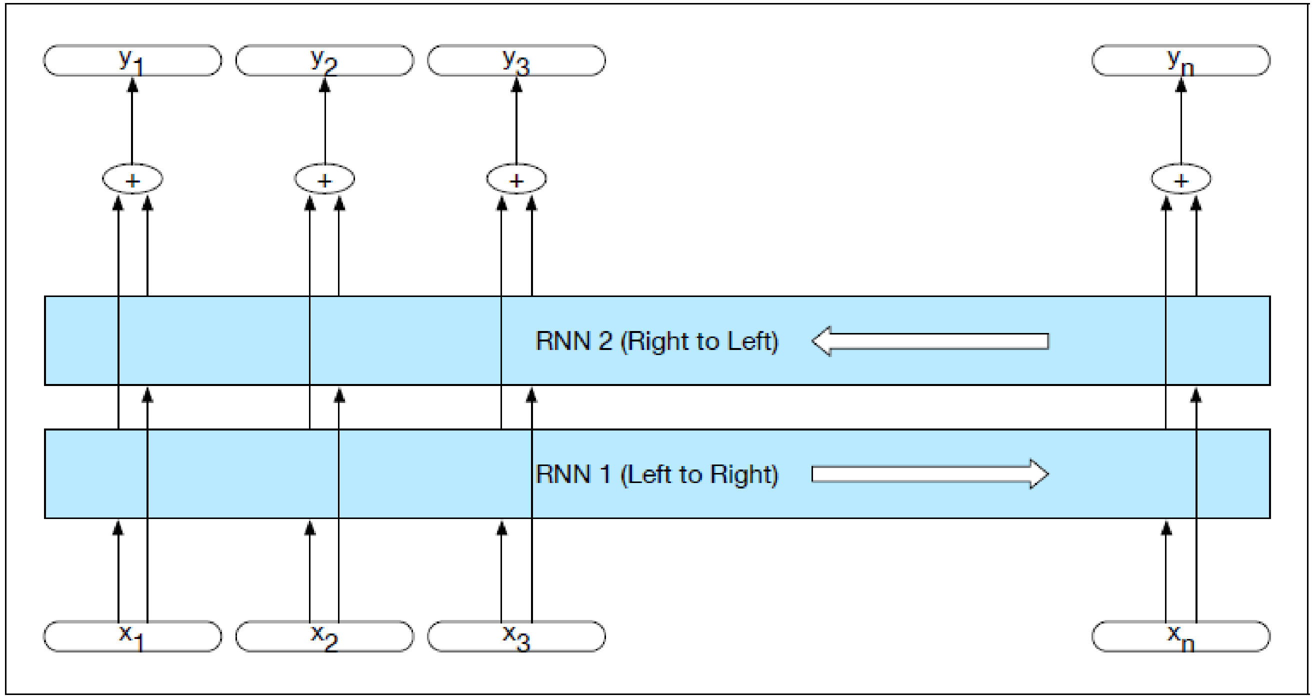


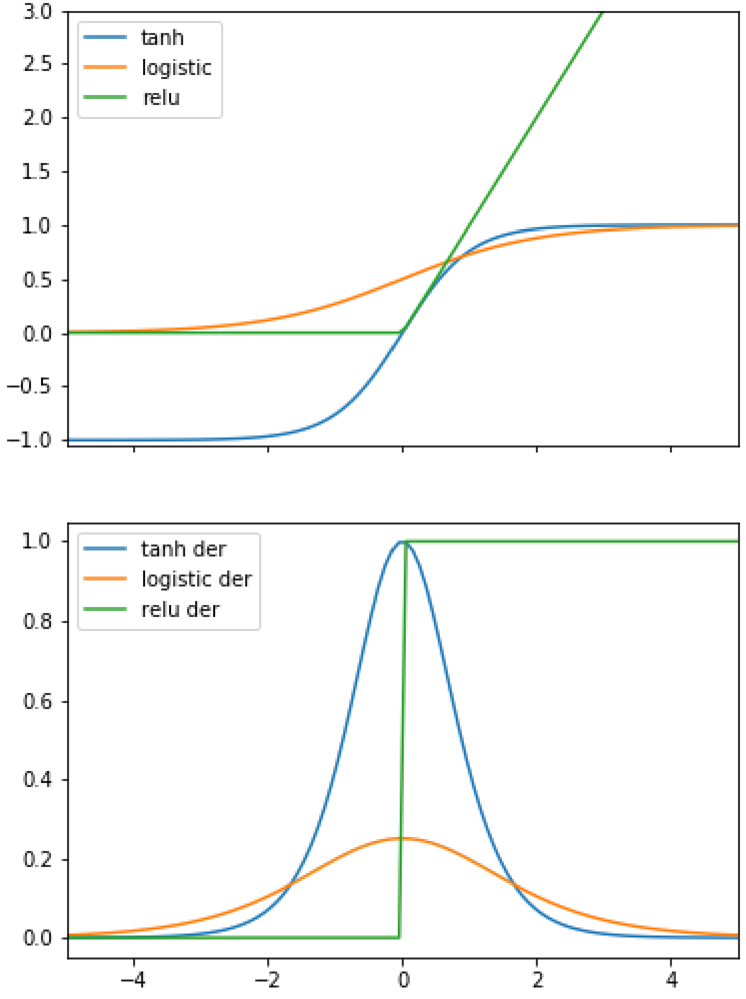
Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions with the output of each model at each time point concatenated to represent the state of affairs at that point in time. The box wrapped around the forward and backward network emphasizes the modular nature of this architecture.

From J&M, 3.ed., 2019

- Example: Tagger
- Considers both preceding and following words

Challenges: Vanishing gradient

- A problem for all deep neural networks
- When back-propagating through many layers
  - the gradient may approach 0
  - little update
- Partial help:
  - Other activation functions, e.g. RelU
  - Various forms of data normalization
  - Adjusted architecture



LSTM

- Problems for RNN
  - Keep track of distant information

- Long Short-Term Memory
  - An advanced architecture with additional layers and weights
    - Not consider the details here
- Bi-LSTM (Binary LSTM)
  - Popular standard architecture in NLP

This lecture

- Recurrent Neural Networks
- Encoder-Decoder models and Machine Translation
- Transformers as Language Models
- Information extraction – some loose ends

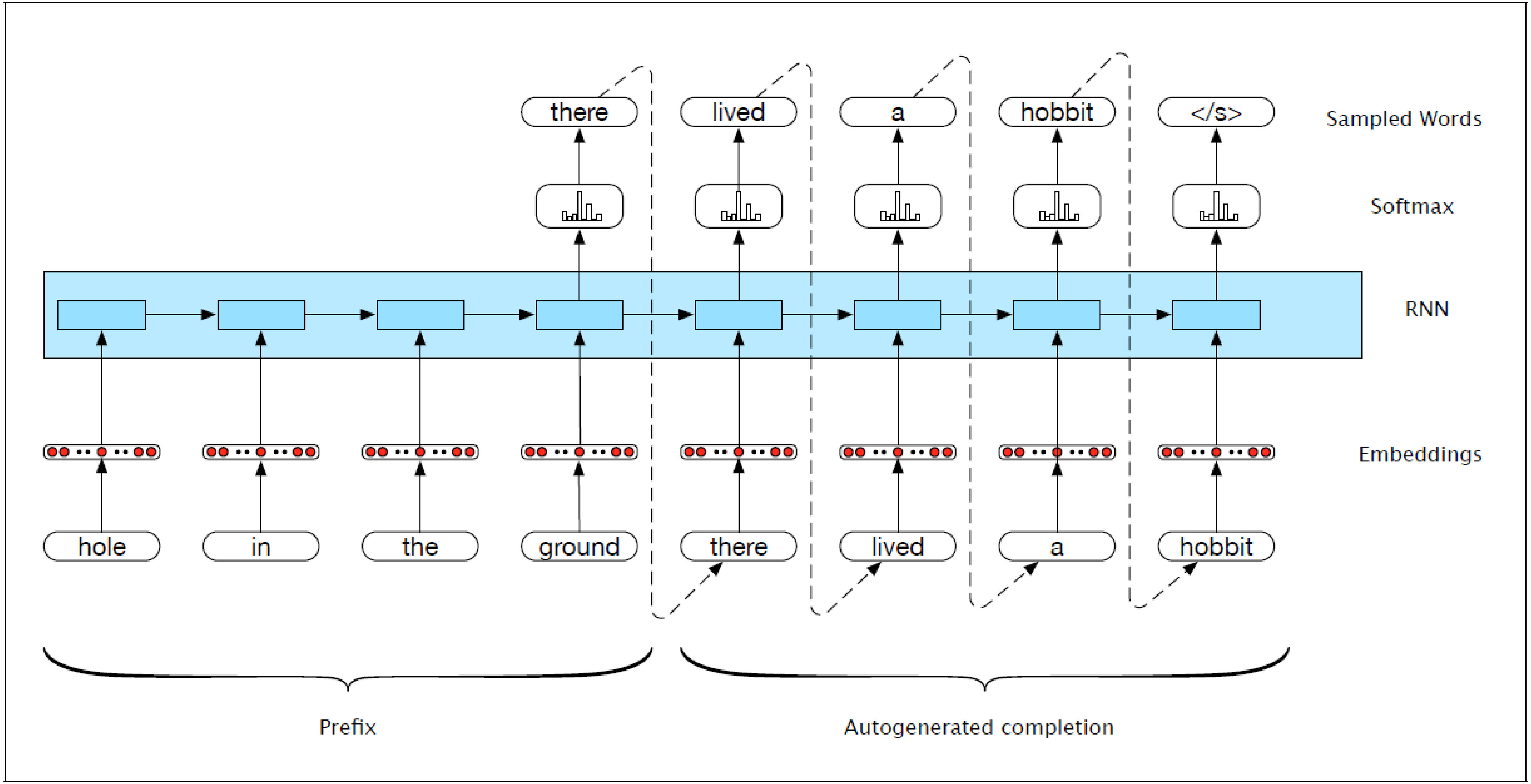


Figure 10.1 Using an RNN to generate the completion of an input phrase.

Idea

- C.f., the autoregressive generation:
  - Read-in the first part of the sentence, and
  - then predict the rest of the sentence
  - using an RNN trained on sentences

- Generalize to other tasks
  - e.g., Machine Translation

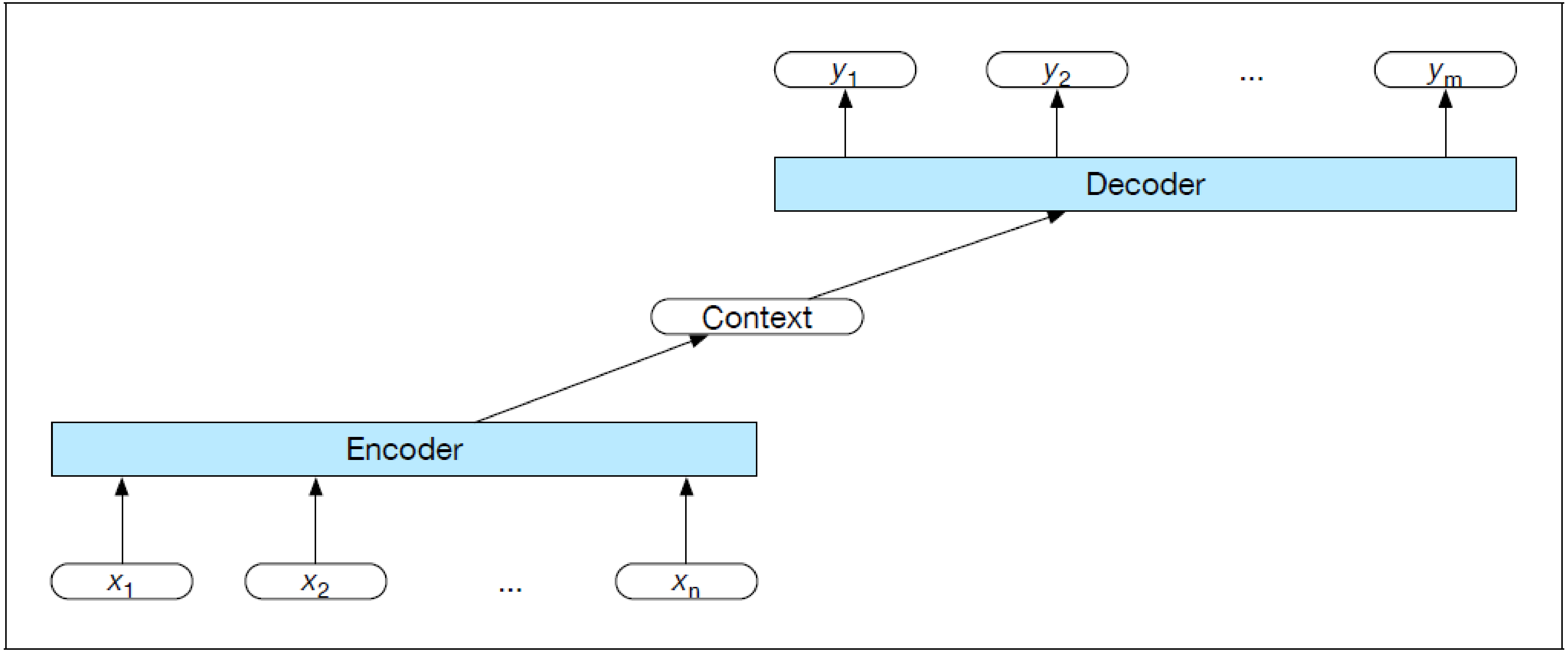


Figure 10.4 Basic architecture for an abstract encoder-decoder network. The context is a function of the vector of contextualized input representations and may be used by the decoder in a variety of ways.

Machine Learning-based Machine Translation

- Bi-text
  - Text translated between two languages
  - The translated sentences are aligned into sentence pairs
- Machine learning based translation systems are trained on large amounts of bi-text

Encoder-decoder based translation

- Concatenate the two sentences in a pair:
  - source sentence_<\s>_target sentence
- Train an RNN on these concatenated pairs
- Apply by reading a source sentence and from there predict a target sentence

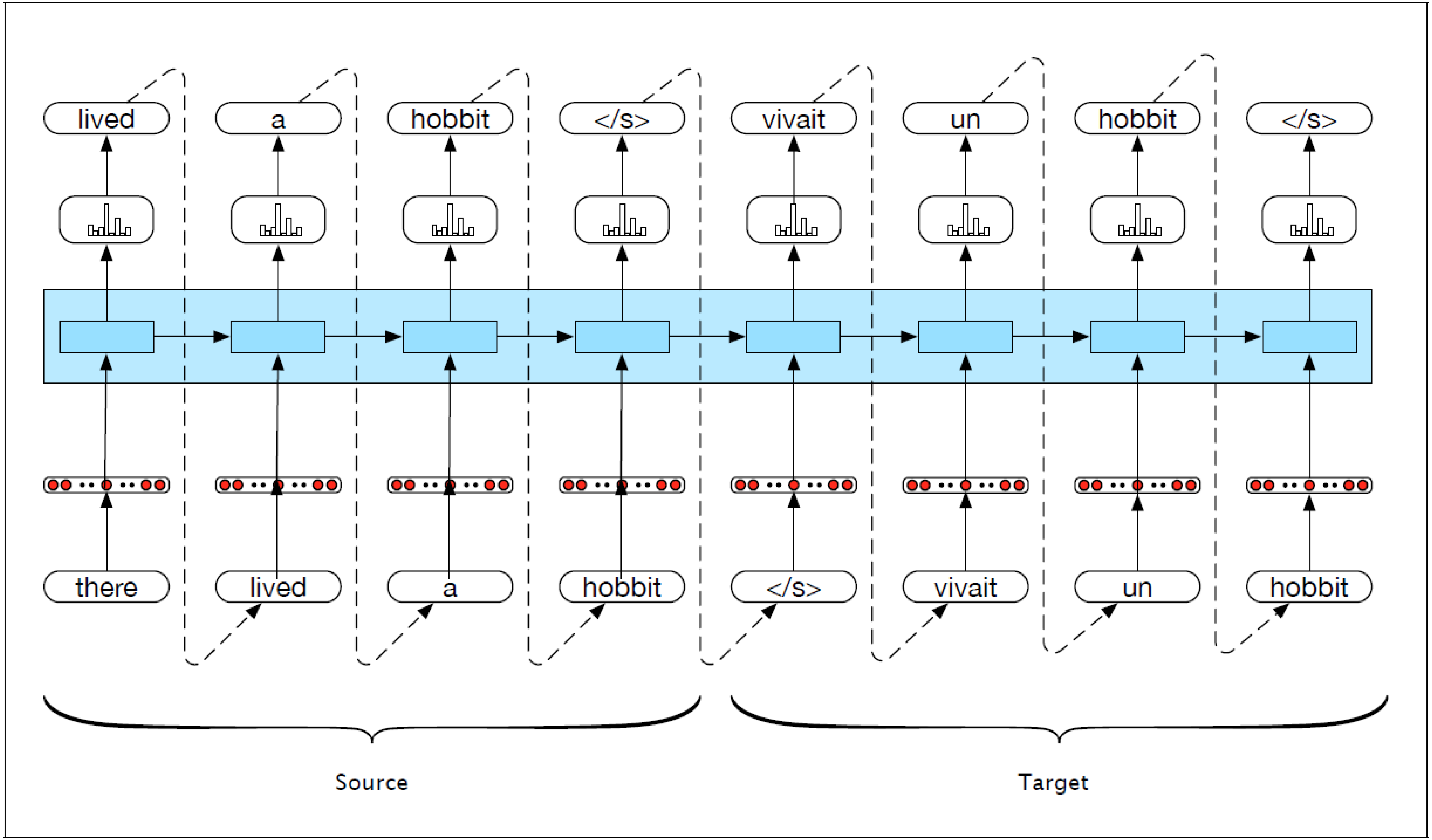


Figure 10.2 Training setup for a neural language model approach to machine translation. Source-target bitexts are concatenated and used to train a language model.

Application

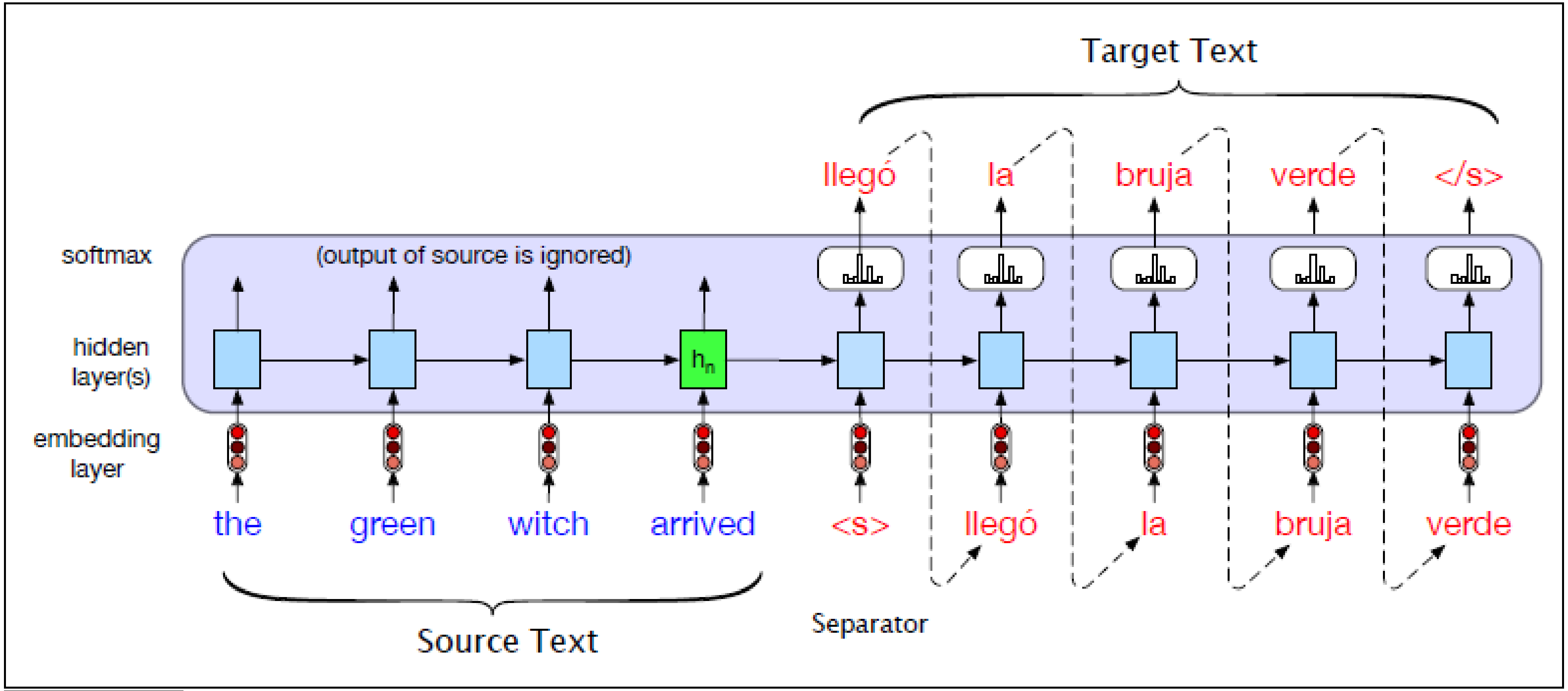
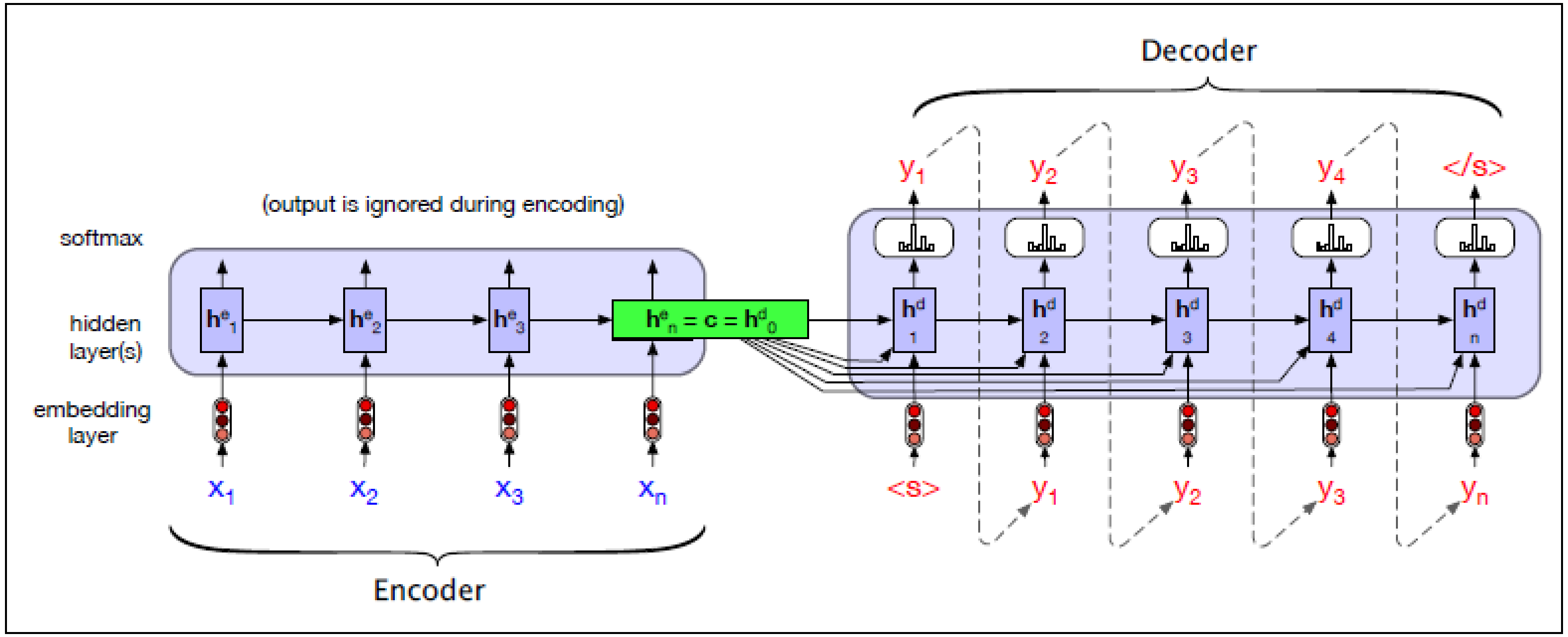


Figure 10.4 Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between, and the decoder uses context information from the encoder's last hidden state.

Refinements

- The encoder can be more refined than a simple RNN,
  - e.g. bi-LSTM
- The decoder may take more information into consideration:
- Each output state has access to c



$$h_t^d = g(\hat{y}_{t-1}, h_{t-1}^d, \mathrm{c})$$

Figure 10.5 A more formal version of translating a sentence at inference time in the basic RNN-based encoder-decoder architecture. The final hidden state of the encoder RNN, h_n^e, serves as the context for the decoder in its role as h_0^d in the decoder RNN.

This lecture

- Recurrent Neural Networks
- Encoder-Decoder models and Machine Translation
  - Attention
- Transformers as Language Models
- Information extraction – some loose ends

Further refinement

- Challenge for one context vector to code a whole sentence.
  - In particular if the sentence is long

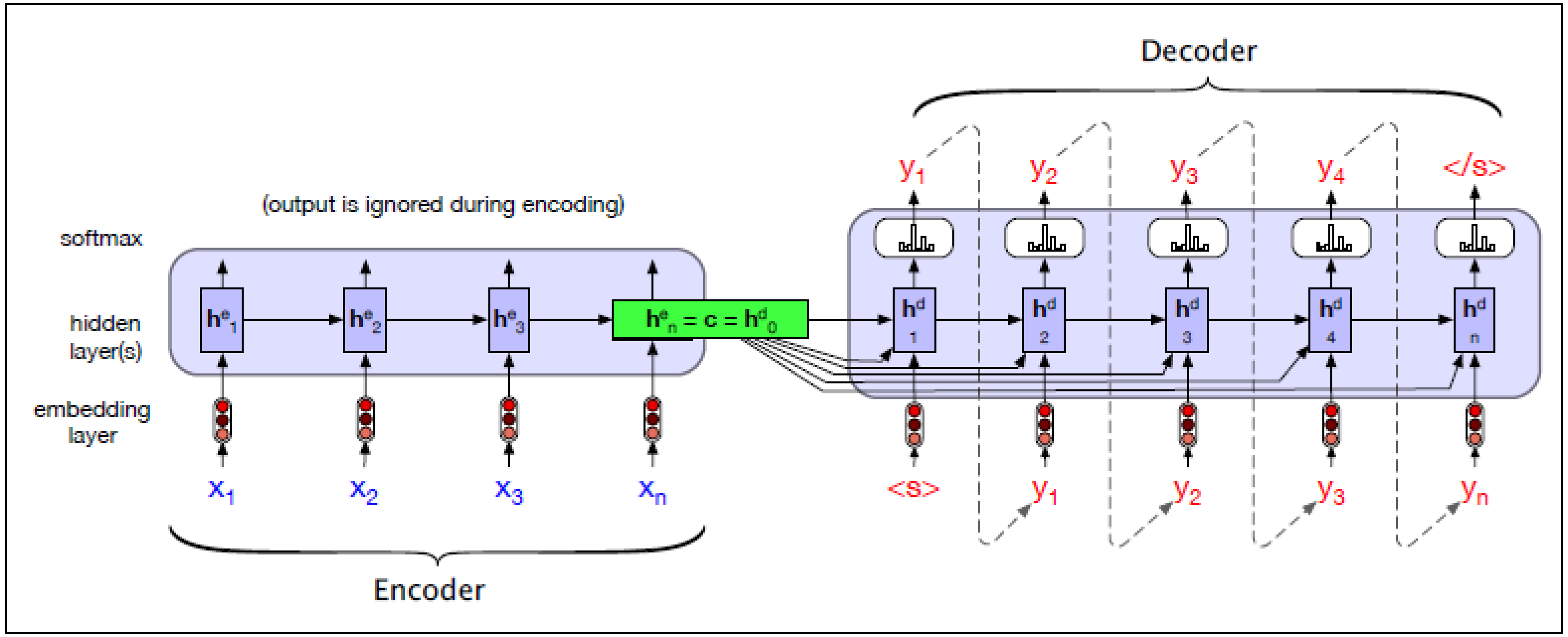


Figure 10.5 A more formal version of translating a sentence at inference time in the basic RNN-based encoder-decoder architecture. The final hidden state of the encoder RNN, h^e_n, serves as the context for the decoder in its role as h^d_0 in the decoder RNN.

$$h^d_t = g(\hat{y}_{t-1}, h^d_{t-1}, \mathrm{c})$$

Attention - sketch

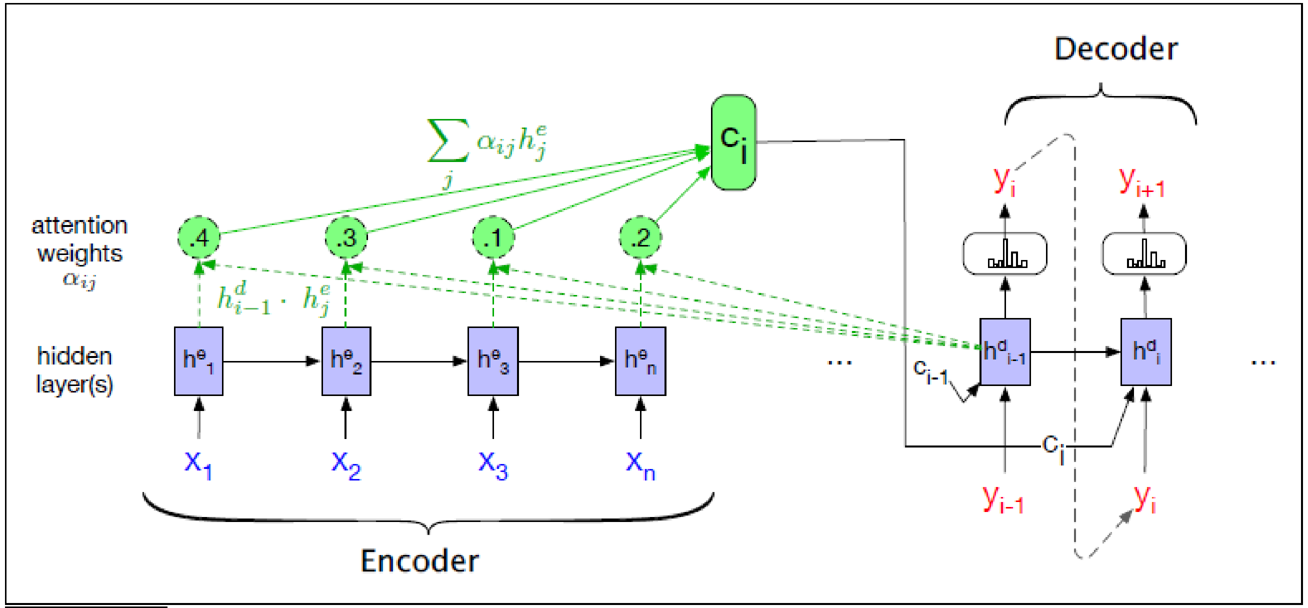
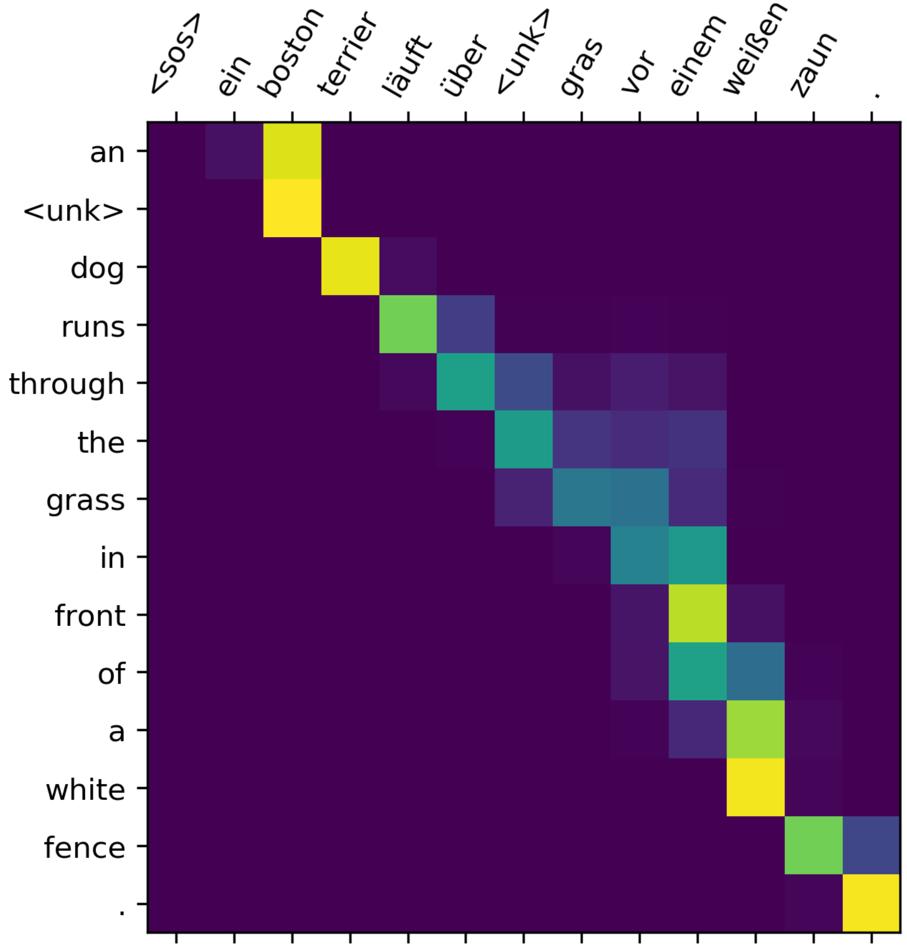


Figure 10.10 A sketch of the encoder-decoder network with attention, focusing on the computation of c_i. The context value c_i is one of the inputs to the computation of h_i^d. It is computed by taking the weighted sum of all the encoder hidden states, each weighted by their dot product with the prior decoder hidden state h_{i-1}^d.

- The context vector is:
  - different for each step:
    - $\boldsymbol{h}_t^d = g(\hat{y}_{t-1}, \boldsymbol{h}_{t-1}^d, \boldsymbol{c}_t)$
  - a weighted sum of input states:
    - $\boldsymbol{c}_t = \sum_{j=1}^{n} \alpha_{i,j} \boldsymbol{h}_j^e$
  - where the weight $\alpha_{i,j}$ is determined by $\boldsymbol{h}_j^e$ and $\boldsymbol{h}_{t-1}^d$
- "How much attention shall $\boldsymbol{h}_t^d$ pay to each of the input words?"

Attention for translation - sketch

- $\boldsymbol{h}_t^d = g(\hat{y}_{t-1}, \boldsymbol{h}_{t-1}^d, \boldsymbol{c}_t)$
- $\boldsymbol{c}_t = \sum_{j=1}^{n} \alpha_{i,j} \boldsymbol{h}_j^e$
- "How much attention shall $\boldsymbol{h}_t^d$ pay to each of the input words?"
- Which words in the source sentence determine which words in the target sentence?
- $\alpha_{i,j}$ is learned from the examples sentence pairs (as the rest)



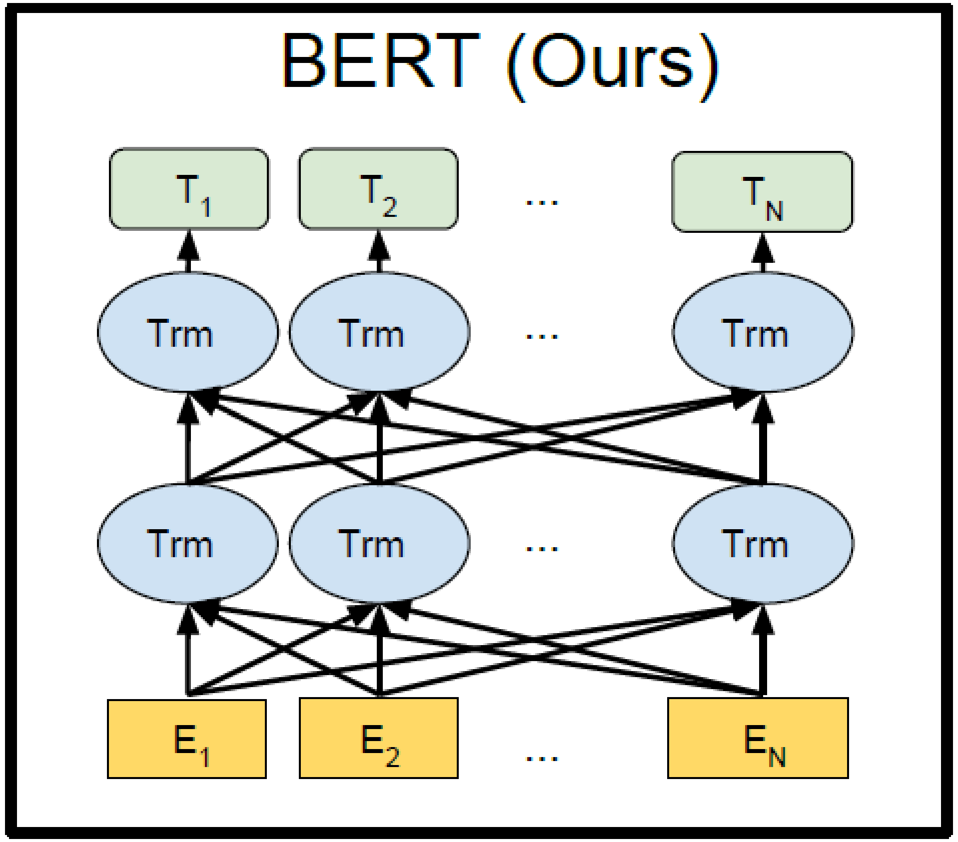
https://teddykoker.com/2020/02/nlp-from-scratch-annotated-attention/

This lecture

- Recurrent Neural Networks
- Encoder-Decoder models and Machine Translation
- Transformers as Language Models
- Information extraction – some loose ends

Language transformers – self-attention (simplified)

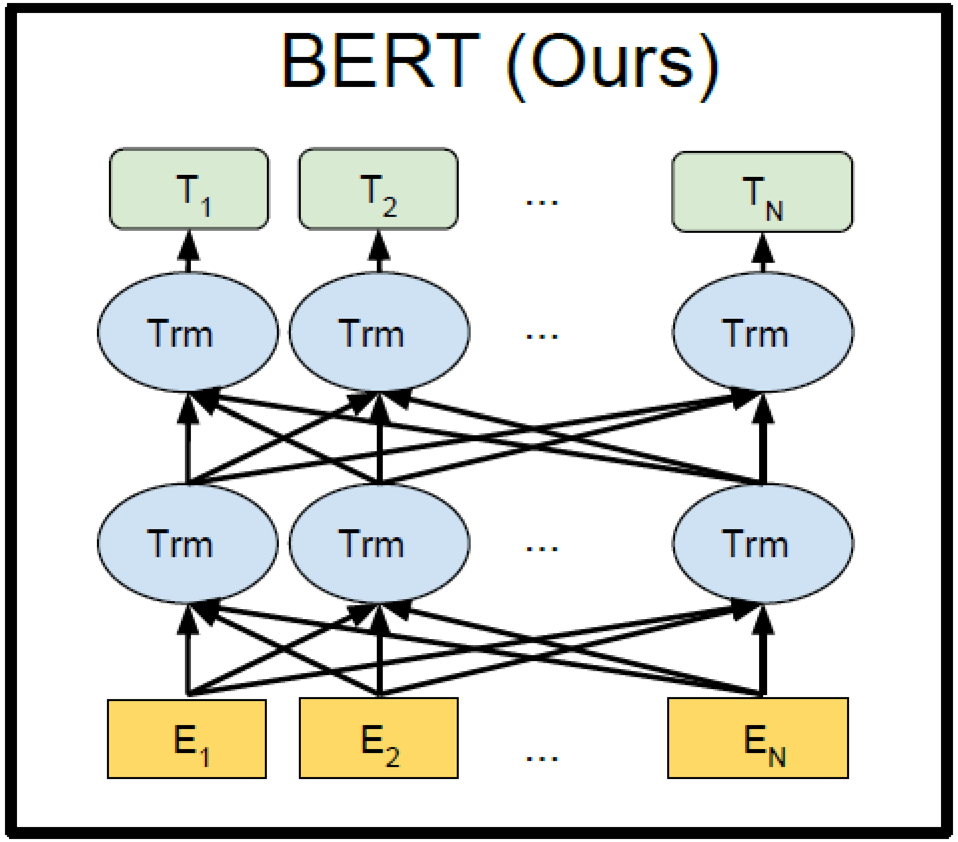
- **Self-attention**: Attention between one word and the other words in the sentence.
- **Transformer**:
  - A number of layers (e.g., 8)
  - Each layer:
    - A vector representation for each word in the input
    - Predicted from the representation of the same word in the preceding layer + the attention it receives from the other words



J. Devlin et. al. BERT, https://arxiv.org/pdf/1810.04805.pdf

Language transformers – self-attention (simplified)

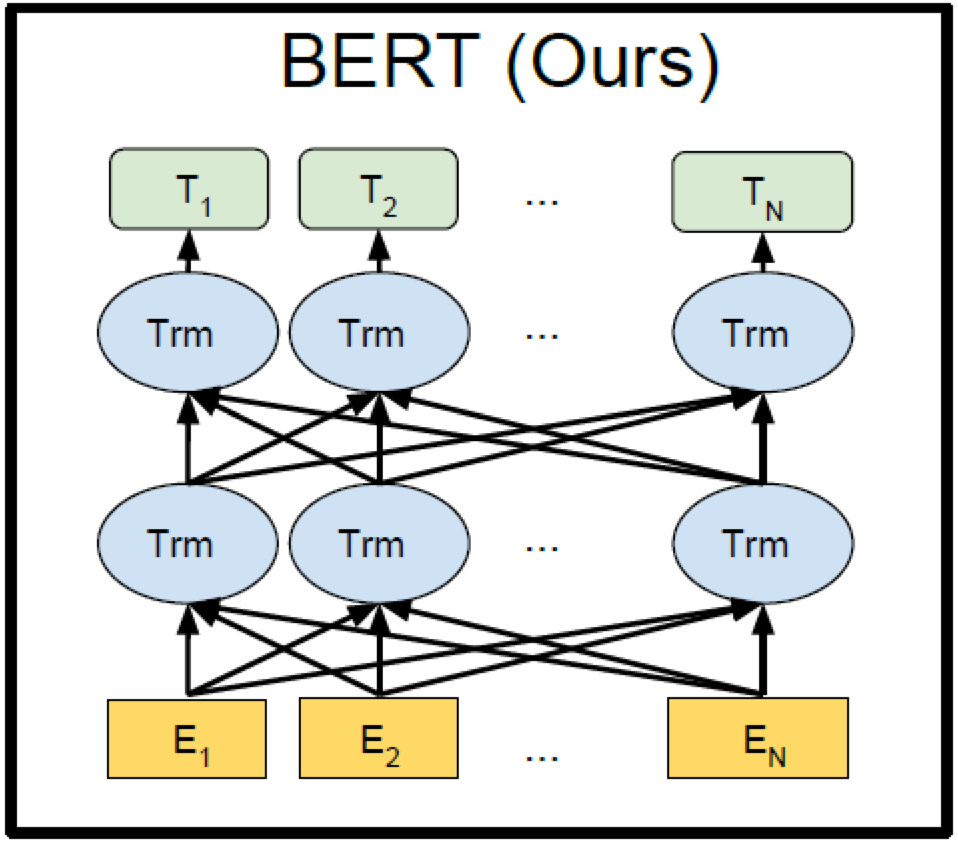
- This results in a vector (embedding) for each word which is context-dependent
- Can be applied to various down-street tasks
- In general, better results than with static embeddings (Word2Vec)



J. Devlin et. al. BERT, https://arxiv.org/pdf/1810.04805.pdf

Language transformers – self-attention (simplified)

- Pre-trained language models, e.g., BERT
- Trained on large amounts on data
- Trained on some prediction task, e.g., next word, masked word, next sentence, etc.
- Can then be applied to all kinds of jobs.
- The transformer can be tuned when training the downstream task



J. Devlin et. al. BERT, https://arxiv.org/pdf/1810.04805.pdf

Neural Methods in Natural Language Processing

- This completes the introduction to neural NLP
  - We did not go very far
- You will learn more in
  - IN5550 – Neural Methods in Natural Language Processing in January
  - In particular, implementations using HPCs

This lecture

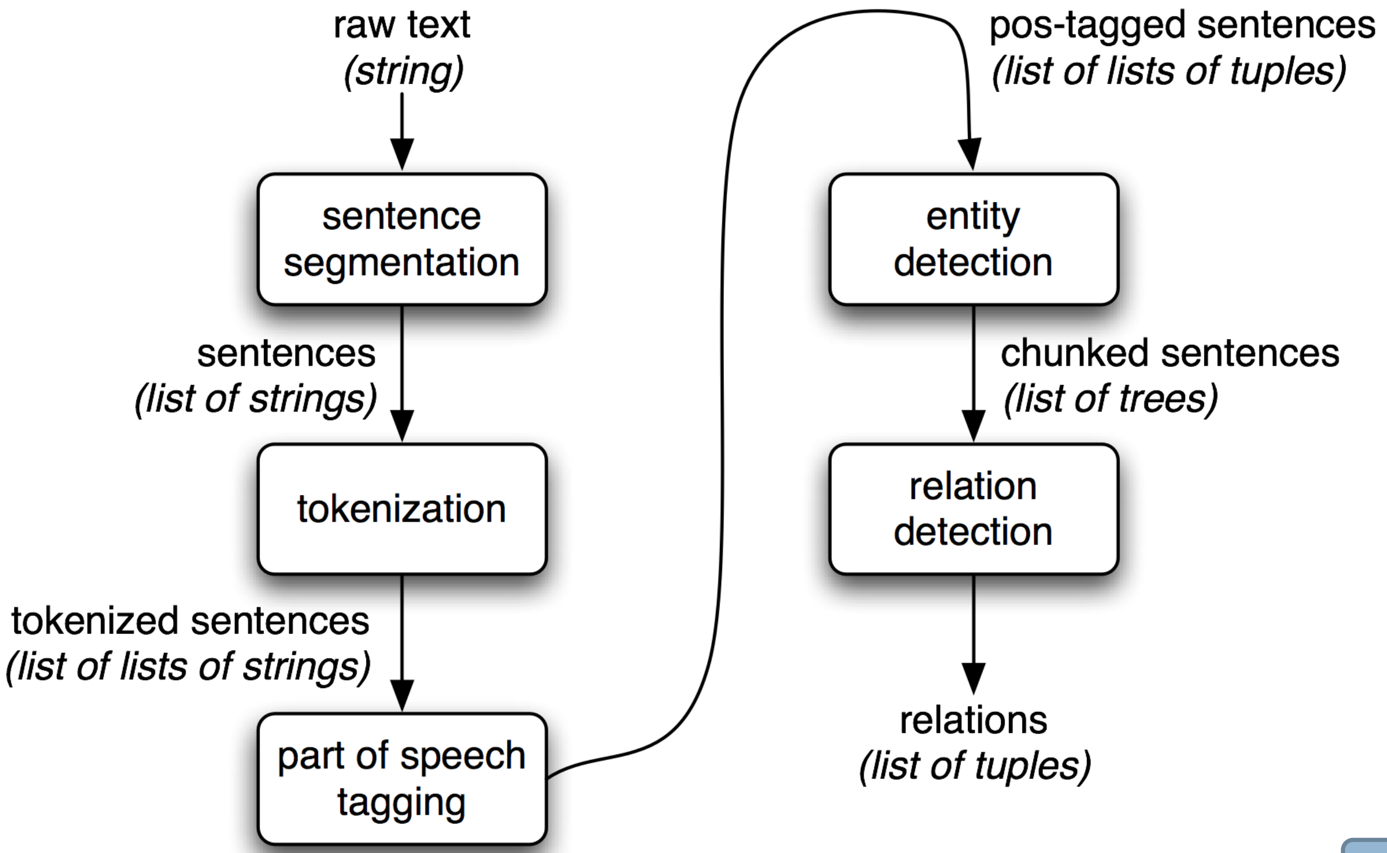
- Recurrent Neural Networks
- Encoder-Decoder models and Machine Translation
- Transformers as Language Models
- Information extraction – some loose ends
  - Repeat the basics of IE and NER
  - Evaluation of NER
  - Relation (extraction)
  - Comparison with the transformer approach

IE basics

Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. (Wikipedia)

- Bottom-Up approach
- Start with unrestricted texts, and do the best you can
- The approach was in particular developed by the Message Understanding Conferences (MUC) in the 1990s
- Select a particular domain and task

A typical pipeline



From NLTK

Named entities

Citing high fuel prices, [ORG United Airlines] said [TIME Friday] it has increased fares by [MONEY $6] per round trip on flights to some cities also served by lower-cost carriers. [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said. [ORG United], a unit of [ORG UAL Corp.], said the increase took effect [TIME Thursday] and applies to most routes where it competes against discount carriers, such as [LOC Chicago] to [LOC Dallas] and [LOC Denver] to [LOC San Francisco].

- Named entity:
  - Anything you can refer to by a proper name
- NE Recognition
  - Find the phrases
  - Classify them

BIO Labels (IOB)

- B-PER:
  - First word in this PER-NE
- I-NP:
  - Part of PER-NE
- O:
  - Not part of any NE

Words	BIO Label
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

- Can code where something begins and ends without altering the word sequence
- Applying "CONNL-format"
  - one word per line
  - info in columns
  - we may add more columns, e.g. for POS-tag

Tag accuracy

In	IN	O	O
addition	NN	B-NP	B-NP
to	TO	O	O
his	PRP$	B-NP	B-NP
previous	JJ	I-NP	I-NP
real-estate	NN	I-NP	I-NP
investment	NN	I-NP	I-NP
and	CC	I-NP	I-NP
asset-management	NN	I-NP	I-NP
duties	NNS	I-NP	I-NP
,	,	O	O
Mr.	NNP	B-NP	B-NP
Meador	NNP	I-NP	I-NP
takes	VBZ	O	O
responsibility	NN	B-NP	B-NP
for	IN	O	O
development	NN	B-NP	B-NP
and	CC	O	I-NP
property	NN	B-NP	I-NP
management	NN	I-NP	I-NP
.	.	O	O

- 2 out of 21 tags are incorrect
- Tag-accuracy: 19/21

Counting chunks

- Left column: Gold
- Right column: Predicted

tp: 4 fp: 1 fn: 2

In	IN	O	O
addition	NN	B-NP	B-NP
to	TO	O	O
his	PRP$	B-NP	B-NP
previous	JJ	I-NP	I-NP
real-estate	NN	I-NP	I-NP
investment	NN	I-NP	I-NP
and	CC	I-NP	I-NP
asset-management	NN	I-NP	I-NP
duties	NNS	I-NP	I-NP
,	,	O	O
Mr.	NNP	B-NP	B-NP
Meador	NNP	I-NP	I-NP
takes	VBZ	O	O
responsibility	NN	B-NP	B-NP
for	IN	O	O
development	NN	B-NP	B-NP
and	CC	O	I-NP
property	NN	B-NP	I-NP
management	NN	I-NP	I-NP
.	.	O	O

Precision: ?

Recall: ?

Relation extraction

- Extract the relations that exist between the (named) entities in the text
- A fixed set of relations (normally)
  - Determined by application:
    - Jeopardy
    - Preventing terrorist attacks
    - Detecting illness from medical record
    - …

- Born_in
- Date_of_birth
- Parent_of

- Author_of
- Winner_of

- Part_of
- Located_in

- Acquire
- Threaten

- Has_symptom
- Has_illness

Examples

Relations		Examples	Types
Affiliations			
	Personal	*married to, mother of*	PER $\longrightarrow$ PER
	Organizational	*spokesman for, president of*	PER $\longrightarrow$ ORG
	Artifactual	*owns, invented, produces*	(PER $\mid$ ORG) $\longrightarrow$ ART
Geospatial			
	Proximity	*near, on outskirts*	LOC $\longrightarrow$ LOC
	Directional	*southeast of*	LOC $\longrightarrow$ LOC
Part-Of			
	Organizational	*a unit of, parent of*	ORG $\longrightarrow$ ORG
	Political	*annexed, acquired*	GPE $\longrightarrow$ GPE

Methods for relation extraction

1. Hand-written patterns
2. Machine Learning (Supervised classifiers)
3. Semi-supervised classifiers via bootstrapping
4. Semi-supervised classifiers via distant supervision
5. Unsupervised

Different approaches

Tradition pipeline

One step after another

1. Tokenize
2. Tag
3. Maybe some form of parsing
4. NER
5. Relations
6. Application

Large pre-trained models

1. Tokenize
2. Pre-trained LM
3. Application:
   - Machine-learning
   - Fine-tune pre-trained model

Some example systems

- Stanford core nlp (Java): http://corenlp.run/
- Stanza (Python): https://stanfordnlp.github.io/stanza/
  - with a wrapper for Stanford Core NLP
- SpaCy (Python): https://spacy.io/docs/api/