ML for Human Vision and Language

MSc Artificial Intelligence

Lecturer: Tejaswini Deoskar
t.deoskar@uu.nl

UiL-OTS, Utrecht University

Block 1, 2021-22

DS Summary

- Count-based models:
  - a word is represented as a long and sparse vector
  - values are a transformation of the count of the word co-occurring with a context word; information-theoretic transformations (e.g. PMI)
  - dimensions correspond to the words in the vocabulary; are interpretable
  - dense vectors can be obtained via dimensionality reduction techniques

DS Summary

- Count-based models:
  - a word is represented as a long and sparse vector
  - values are a transformation of the count of the word co-occurring with a context word; information-theoretic transformations (e.g. PMI)
  - dimensions correspond to the words in the vocabulary; are interpretable
  - dense vectors can be obtained via dimensionality reduction techniques

DS Summary

- Count-based models:
  - a word is represented as a long and sparse vector
  - values are a transformation of the count of the word co-occurring with a context word; information-theoretic transformations (e.g. PMI)
  - dimensions correspond to the words in the vocabulary; are interpretable
  - dense vectors can be obtained via dimensionality reduction techniques
- Predict-based models:
  - Train a (neural network) model to predict plausible contexts for a word
  - learn word representations in the process
  - **short and dense** vectors with **latent** dimensions
  - very useful as word representations for other NLP tasks

DS Summary

- GloVe
  - Glove is efficient, and gives good performance for small datasets as well
  - GloVe has access to global statistics, similar to count-based models
- Evaluations of the semantic space/model: comparison to human judgements, accuracy on other NLP tasks (pre-trained embeddings)

DS Summary

- GloVe
  - Glove is efficient, and gives good performance for small datasets as well
  - GloVe has access to global statistics, similar to count-based models
- Evaluations of the semantic space/model: comparison to human judgements, accuracy on other NLP tasks (pre-trained embeddings)
- Prediction of brain-activations

Today

- Parts-of-speech tags: Grammatical word classes
- Recurrent Neural Networks for sequence tagging

- Encoder-decoder networks
- Encoder-decoders for morphological learning (English past tense)

Parts-of-speech

grammatical classes, word classes, lexical classes, lexical tags, ...

- *Dionysius Thrax of Alexandria* (c. 100 BC) wrote a grammatical sketch of Greek involving **8 traditional parts-of-speech**

noun	verb	pronoun	preposition
conjunction	adverb	particle	article/determiner

Parts-of-speech

grammatical classes, word classes, lexical classes, lexical tags, ...

- *Dionysius Thrax of Alexandria* (c. 100 BC) wrote a grammatical sketch of Greek involving **8 traditional parts-of-speech**

noun	verb	pronoun	preposition
conjunction	adverb	particle	article/determiner

- Various granularities of POS tags: fine-grained tags/coarse-grained tags
- Penn Treebank Tag set ($\sim$ 45 tags):

EX	VBD	JJ	NN	IN	DT	NNS
There	was	still	lemonade	in	the	bottles

Parts-of-speech

grammatical classes, word classes, lexical classes, lexical tags, ...

- *Dionysius Thrax of Alexandria* (c. 100 BC) wrote a grammatical sketch of Greek involving **8 traditional parts-of-speech**

noun	verb	pronoun	preposition
conjunction	adverb	particle	article/determiner

- Various granularities of POS tags: fine-grained tags/coarse-grained tags
- Penn Treebank Tag set ($\sim$ 45 tags):

EX	VBD	JJ	NN	IN	DT	NNS
There	was	still	lemonade	in	the	bottles

- POS tagging (inference): process of assigning a part-of-speech or lexical class marker/tag to each word (token) in a collection (usually a sentence, or a piece of text).

POS tagging: Why?

- first step towards grammatical analysis; simpler and faster than full syntactic parsing or analysis
- POS tags can be useful features in e.g. text classification, authorship identification, text-to-speech, etc.
- A variety of NLP and other (AI-related) applications are sequence modelling tasks
  - Named Entity Recognition (tagging as Person/ Organisation/ Location etc.)
  - Music generation/cognition
  - Robot actions

Criteria for classifying words

When should words be put into the same class?

Criteria for classifying words

When should words be put into the same class?

- Semantic criteria : What does the word refer to? (Nouns often refer to 'people', 'places' or 'things')

Criteria for classifying words

When should words be put into the same class?

- Semantic criteria : What does the word refer to? (Nouns often refer to 'people', 'places' or 'things')
- Formal criteria : What form does the word have?
  - e.g. -tion, -ize. What affixes can it take, e.g. -s, -ing, -est?
  - *slice*, *donate*, *believe* don't have much in common semantically, but can all combine with suffix *-s* or *-ed*

Criteria for classifying words

When should words be put into the same class?

- Semantic criteria : What does the word refer to? (Nouns often refer to 'people', 'places' or 'things')
- Formal criteria : What form does the word have?
  - e.g. -tion, -ize. What affixes can it take, e.g. -s, -ing, -est?
  - *slice*, *donate*, *believe* don't have much in common semantically, but can all combine with suffix *-s* or *-ed*
- Distributional criteria : In what contexts can the word occur ?
  - *crocodile*, *pencil*, *mistake* have different meanings, but can occur in the same contexts (for e.g. after *the*).
  - *slice*, *donate*, *believe* can all occur after *to*

Criteria for classifying words

	Semantically	Formally	Distributionally
Nouns	refer to things, concepts	-ness, -tion, -ity, -ance	After determiners, possessives
Verbs	refer to actions, states	-ed, -ate, -ize	infinitives: to jump, to learn
Adjectives	properties of nouns	-al, -ble	appear before nouns
Adverbs	properties of actions	-ly	next to verbs, beginning of sentence

Importance of formal and distributional criteria

Formal and distributional criteria help people (and machines) recognize which (open) class a word belongs to, even if we don't know its meaning.

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

Importance of formal and distributional criteria

Formal and distributional criteria help people (and machines) recognize which (open) class a word belongs to, even if we don't know its meaning.

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

Often in text, we come across unknown words. Formal and distributional criteria are also useful when we (or our system) comes across unknown words

Why is POS tagging hard?

Why is POS tagging hard?

The usual reasons:

- Ambiguity: Words often can belong to more than one category.

  back

  - The *back* door = JJ (**bijvoeglijk naamwoord**)
  - On my *back* = NN (**zelfstandig naamwoord**)
  - Win the voters *back* = RB (**bijwoord**)
  - Promised to *back* the bill = VB (**werkwoord**)

Why is POS tagging hard?

The usual reasons:

- Ambiguity: Words often can belong to more than one category.

  back
  - The *back* door = JJ (**bijvoeglijk naamwoord**)
  - On my *back* = NN (**zelfstandig naamwoord**)
  - Win the voters *back* = RB (**bijwoord**)
  - Promised to *back* the bill = VB (**werkwoord**)

- Sparse data
  - Words we haven't seen before
  - Word-tag pairs that we haven't seen before

Extent of POS Ambiguity

The Brown corpus (1M word tokens) is a commonly used English corpus

- 39,440 different words (types), and a tag-set of 80.
- 89.6% word types (35340) have only **1** POS tag anywhere in corpus

Extent of POS Ambiguity

The Brown corpus (1M word tokens) is a commonly used English corpus

- 39,440 different words (types), and a tag-set of 80.
- 89.6% word types (35340) have only **1** POS tag anywhere in corpus
- 10.4% word types (4100) have **2 to 7** POS tags

Extent of POS Ambiguity

The Brown corpus (1M word tokens) is a commonly used English corpus

- 39,440 different words (types), and a tag-set of 80.
- 89.6% word types (35340) have only **1** POS tag anywhere in corpus
- 10.4% word types (4100) have **2 to 7** POS tags
- Ambiguous word-types tend to be high-frequency: more than 50% of the word *tokens* are ambiguous.

He wants to/TO go to/IN the store.

The teacher asked that/DT student to eat asparagus
The teacher asked that/CS students eat asparagus
He asked for asparagus that/WPS are fresh

Three sources of difficulty

- A large percentage of word *tokens* are ambiguous.
- Many words (or word-tag combinations) might be unseen in training corpus.
- POS tag for a particular instance (token) of a word should be determined in context

Three approaches for POS tagging

- Rule-based tagging using dictionary (traditional approach, but still used for many languages)

Three approaches for POS tagging

- Rule-based tagging using dictionary (traditional approach, but still used for many languages)
- Stochastic/ Probabilistic models for sequence tagging (not covered here)
  - HMMs (Hidden Markov Model)
  - MEMMs (Maximum Entropy Markov Model)

Three approaches for POS tagging

- Rule-based tagging using dictionary (traditional approach, but still used for many languages)
- Stochastic/ Probabilistic models for sequence tagging (not covered here)
  - HMMs (Hidden Markov Model)
  - MEMMs (Maximum Entropy Markov Model)
- Recurrent neural networks

Some tagging strategies

- One simple strategy: just assign to each word its most common tag. (Call this Uni-gram tagging)
- Surprisingly, even this crude approach typically gives around 90% accuracy. (State-of-the-art (English) on PTB is about 98%).

Some tagging strategies

- One simple strategy: just assign to each word its most common tag. (Call this Uni-gram tagging)
- Surprisingly, even this crude approach typically gives around 90% accuracy. (State-of-the-art (English) on PTB is about 98%).
- Can we do better?

Bi-gram tagging

Bi-gram tagging

- Look at pairs of adjacent POS tokens.
- For each word (e.g. still), tabulate the frequencies of each possible POS given the POS of the preceding word.

still	DT (det.)	MD (modal)	JJ (adj.)
NN (noun)	8	0	6
JJ (adjec.)	23	0	12
VB (verb)	1	12	2
RB (adverb)	6	45	5

Bi-gram tagging

- Look at pairs of adjacent POS tokens.
- For each word (e.g. still), tabulate the frequencies of each possible POS given the POS of the preceding word.

still	DT (det.)	MD (modal)	JJ (adj.)
NN (noun)	8	0	6
JJ (adjec.)	23	0	12
VB (verb)	1	12	2
RB (adverb)	6	45	5

- Given a new text, tag the words from left to right, assigning each word the most likely tag given the preceding one.

  DT JJ NN ...

  The still lemonade ...

Bi-gram tagging

- Look at pairs of adjacent POS tokens.
- For each word (e.g. still), tabulate the frequencies of each possible POS given the POS of the preceding word.

still	DT (det.)	MD (modal)	JJ (adj.)
NN (noun)	8	0	6
JJ (adjec.)	23	0	12
VB (verb)	1	12	2
RB (adverb)	6	45	5

- Given a new text, tag the words from left to right, assigning each word the most likely tag given the preceding one.

  DT JJ NN ...

  The still lemonade ...

- Could also do trigram, 4-gram etc., but frequencies might be too sparse to be useful...

Problems with bi-gram tagging

- One incorrect tagging choice might have unintended effects:

	The	still	smoking	remains	of	the	campfire
Intended:	DT	RB	VBG	NNS	IN	DT	NN
Bigram:	DT	JJ	NN	VBZ	...		

Problems with bi-gram tagging

- One incorrect tagging choice might have unintended effects:

	The	still	smoking	remains	of	the	campfire
Intended:	DT	RB	VBG	NNS	IN	DT	NN
Bigram:	DT	JJ	NN	VBZ	...		

- No lookahead: choosing the "most probable" tag at one stage might lead to highly improbable choice later.
- We want to find the **overall most likely** tagging sequence given the n-gram frequencies of tags.

The most likely tag sequence

- Stochastic approach
  - Formalise model in terms of Hidden Markov Model (HMM) or MEMM

$$\hat{t_1^n} = \arg\max_{t_1^n} P(t_1^n \mid w_1^n)$$

$$\hat{t_1^n} \approx \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)\ P(t_i|t_{i-1})$$

  - Viterbi algorithm to find the best (tag) sequence
- RNN family can model sequences (usually can capture more context than N-grams)
  - Use Viterbi algorithm on the output to find the best (tag) sequence

Recurrent Neural Networks (RNNs)

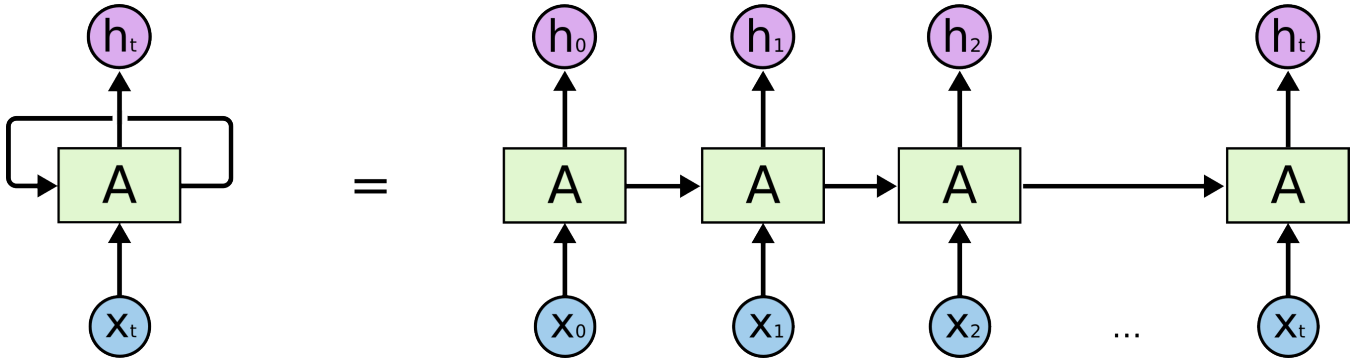
- RNNs for modelling sequences (e.g. language modelling, and sequence labelling)

Recurrent Neural Networks (RNNs)

- RNNs for modelling sequences (e.g. language modelling, and sequence labelling)
- FF neural networks: do not have *persistence*; presented with a new input, they forget the old one

Recurrent Neural Networks (RNNs)

- RNNs for modelling sequences (e.g. language modelling, and sequence labelling)
- FF neural networks: do not have *persistence*; presented with a new input, they forget the old one
- RNNs solve this problem by having a loop within its network connections



- Input to the hidden layer is augmented with the activation value of the hidden layer from a previous point in time

Computation at the hidden layer

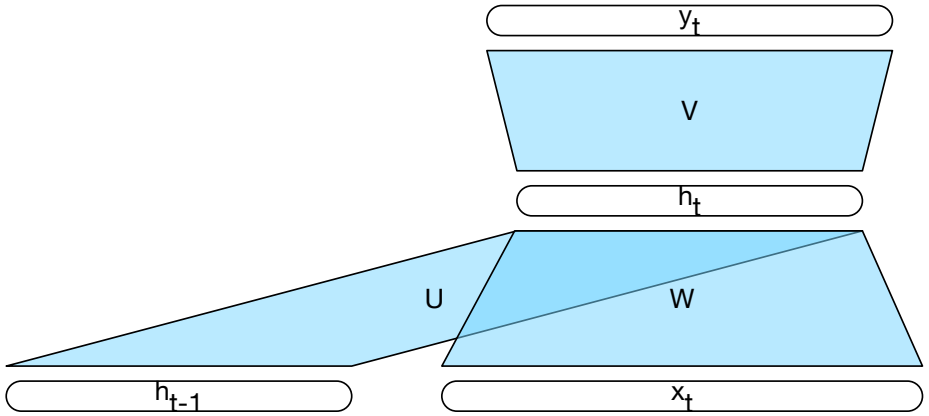


Figure 9.3 Simple recurrent neural network illustrated as a feed-forward network.

- A new set of weights U connect the hidden layer from the previous timestep to the current hidden layer
- U determines how the network makes use of past context in calculating the current output
- Just like other weights, these are also trained with backpropagation

$$h_t = g(Uh_{t-1} + Wx_t)$$

RNNs for Language Modelling

- Two previous methods : n-gram models, FF networks (with sliding window)
- Both compute

$$P(w_n \mid w_1^{n-1}) = P(w_n \mid w_{n-N+1}^{n-1})$$

RNNs for Language Modelling

- Two previous methods : n-gram models, FF networks (with sliding window)
- Both compute

$$P(w_n \mid w_1^{n-1}) = P(w_n \mid w_{n-N+1}^{n-1})$$

- RNNs predict the next word by using the current word and the previous hidden state as input
- The hidden state encodes information about all preceeding words, up to the beginning of the sentence

RNNs for Language Modelling

- Output at each step is a prob. distribution over the vocabulary

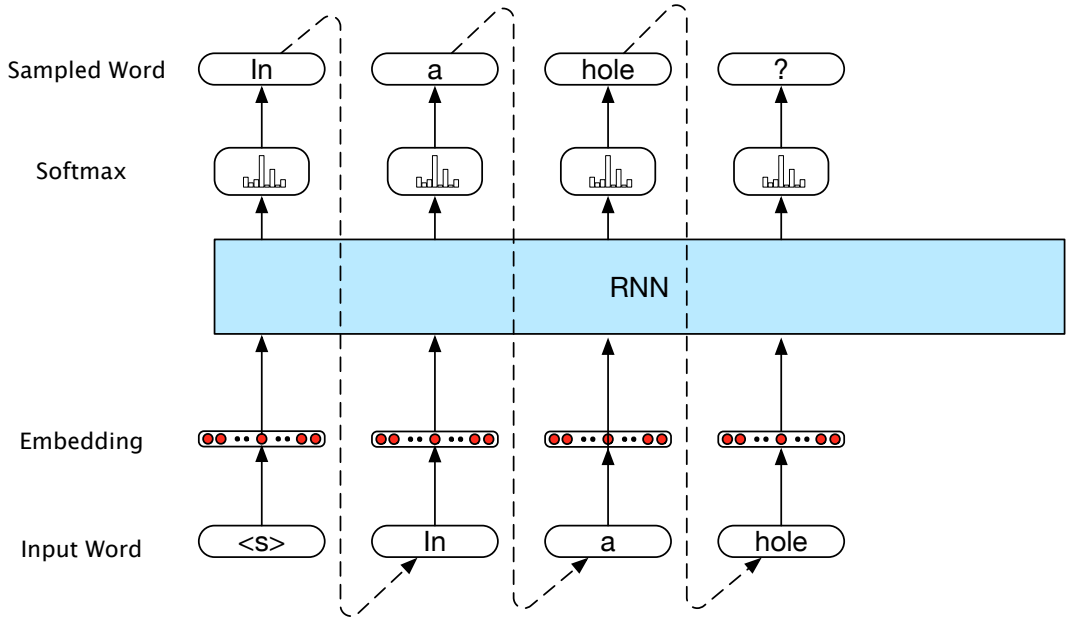
$$y_n = P(w_n \mid w_1^{n-1}) = softmax(Vh_n)$$

- Probability of a sequence w_1^n:

$$P(w_1^n) = \prod_{k=1}^{n} y_k$$

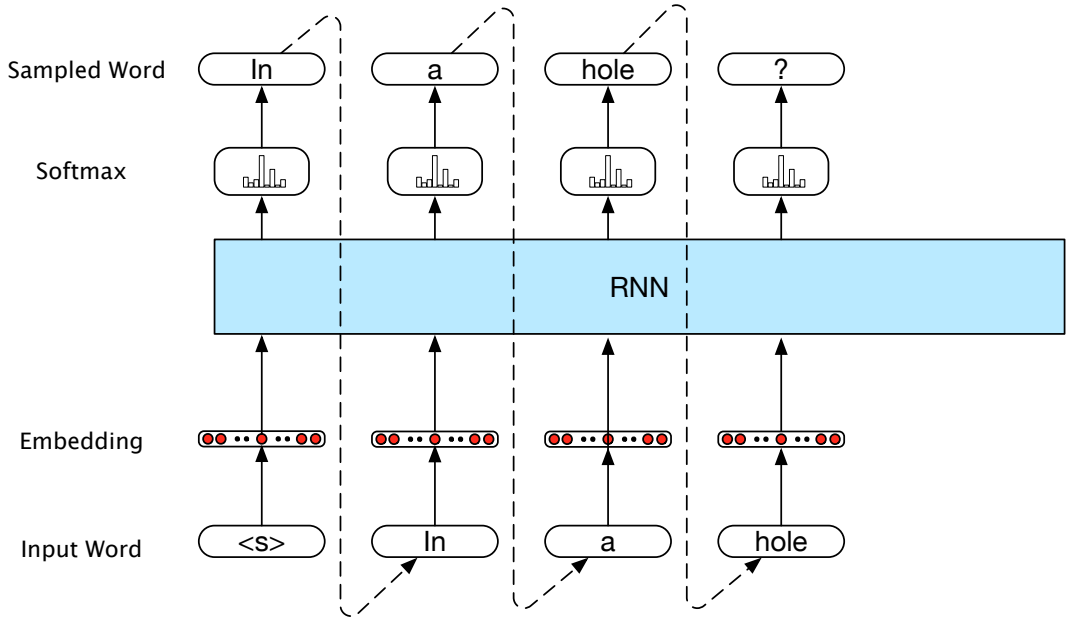
Autoregressive generation with Neural LMs

- Sample first word in the output, at time-step $t = 0$ (Caveat: first word is usually a beginning of sentence marker <s>)
- Use word embedding of first word as input for the next time step



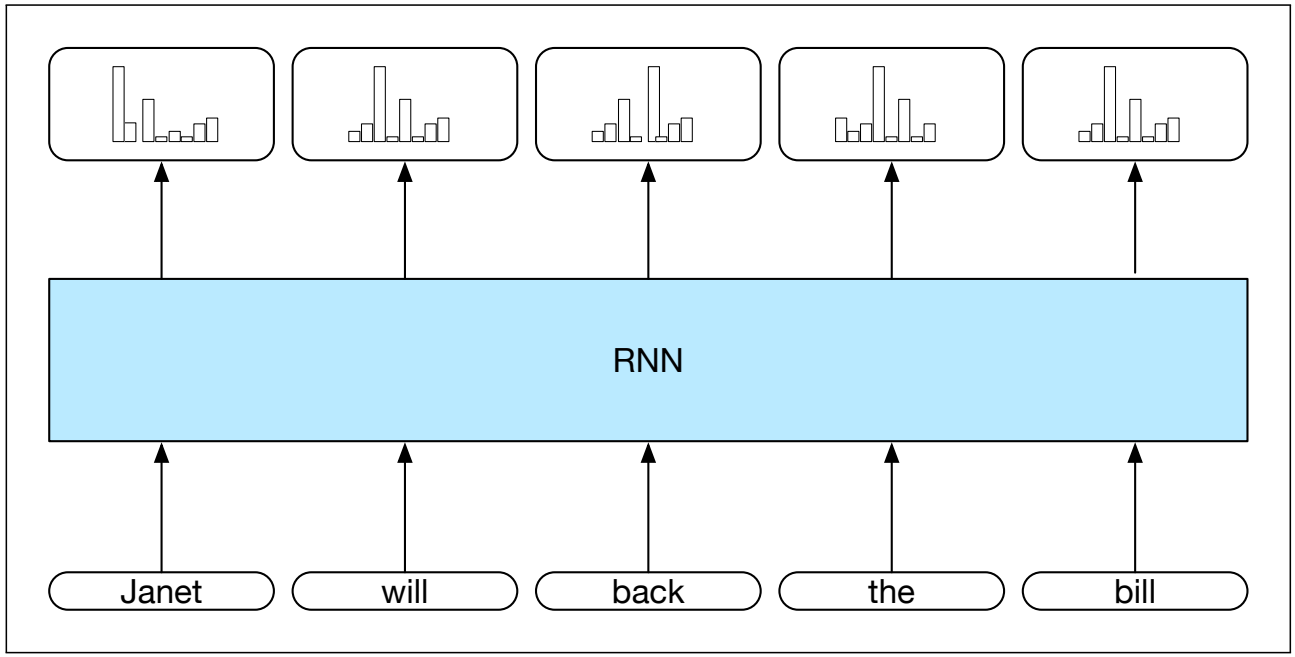
Autoregressive generation with Neural LMs

- Sample first word in the output, at time-step $t = 0$ (Caveat: first word is usually a beginning of sentence marker <s>)
- Use word embedding of first word as input for the next time step



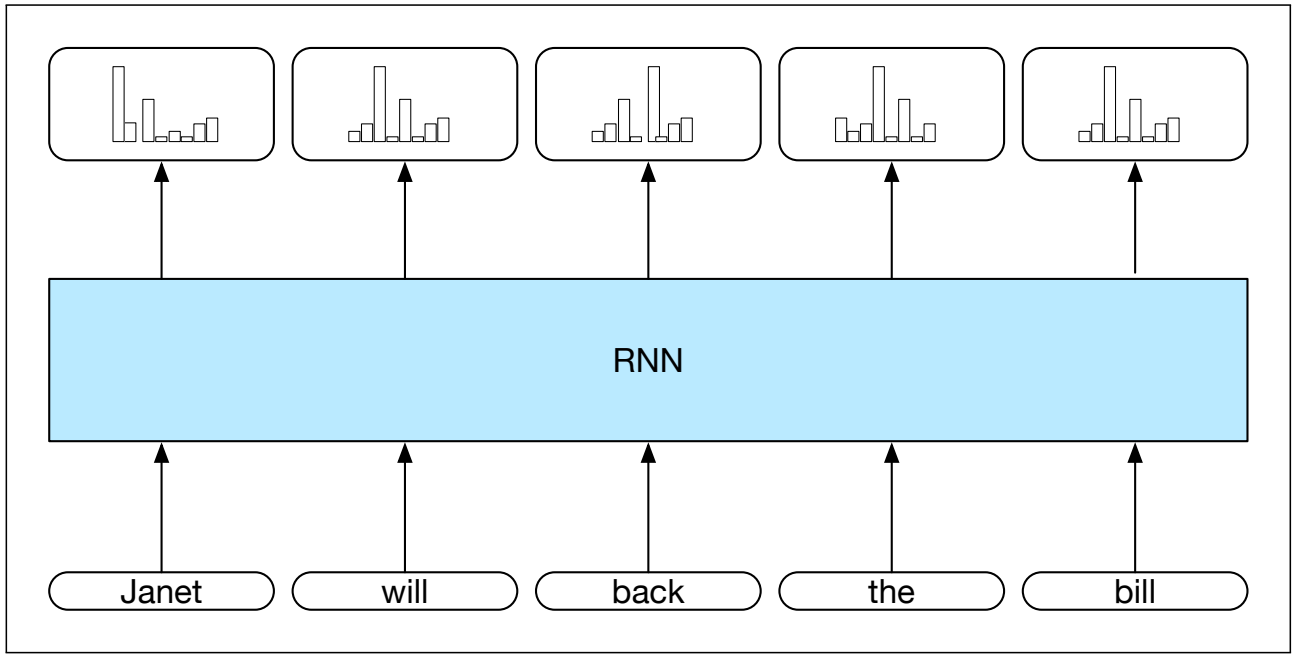
- Used in all "generation" tasks like machine translation, summarisation, generating image descriptions.
- Also in encoder-decoder networks (coming up later...)

Sequence labelling with RNNs (POS tagging)



- RNN block: input, hidden, output layer + U, W, V weight matrices
- inputs are pre-trained word embeddings

Sequence labelling with RNNs (POS tagging)



- RNN block: input, hidden, output layer + U, W, V weight matrices
- inputs are pre-trained word embeddings
- output at each time step is a distribution over POS tagset generated by softmax layer

Sequence labelling with RNNs (POS tagging)

- To get POS tags, select the most likely tag from the softmax at each step.

Sequence labelling with RNNs (POS tagging)

- To get POS tags, select the most likely tag from the softmax at each step.
- However,
  - choosing best (max. prob.) label for each word in a sequence does not yield the best sequence
  - sequence of probability distributions (from softmax) can be combined with a tag-level language model, and selecting the most likely tag sequence
  - Viterbi (or variants) to choose the best tag sequence.

Sequence labelling with RNNs and variants

- Deep networks: stacked RNNs
- Bi-directional RNNs
- Managing context (what to remember and what to forget) with LSTMs or GRUs.
- Subword and character level models.

Encoder-decoders

Recall Auto-regressive language generation

- Train an RNN network on the task of predicting the next word in a sequence

$$h_t = g(h_{t-1}, x_t)$$

$$y_t = f(h_t)$$

- Generate a sequence by first sampling a first word from output, and using it to as input at the next time step.
- Variation: Instead of generating from scratch:
  - pass initial n words (prefix) through the LM to generate a sequence of hidden states
  - Use the last hidden state of the prefix to start generating
  - The result is a reasonable completion of the prefix

Encoder-decoders

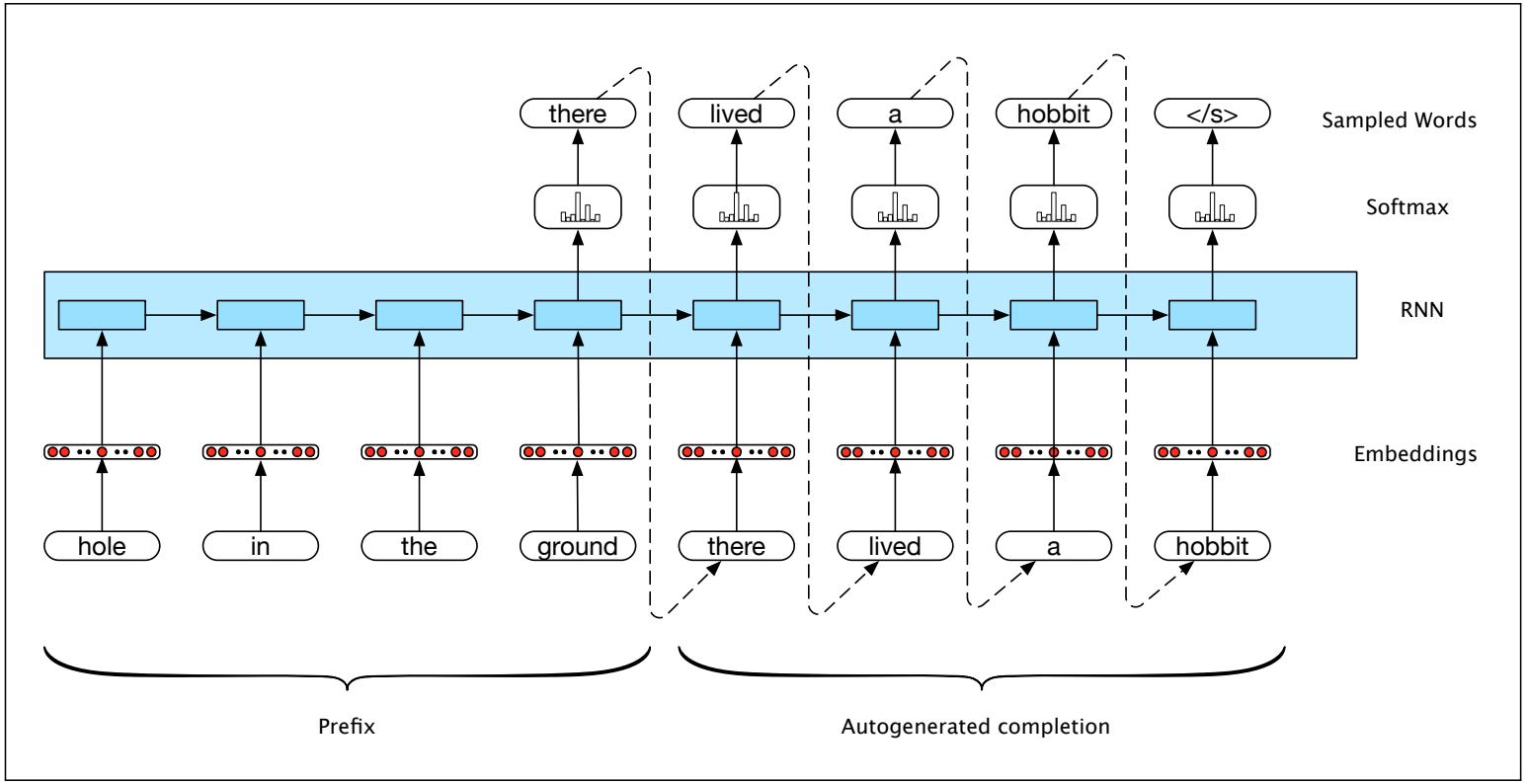


Figure 10.1 Using an RNN to generate the completion of an input phrase.

- Initially used for machine translation: given a sentence in source language, generate translation in target language
- Now generalised to translate (almost) any sequence to another sequence

Encoder-decoder networks for English Past Tense

- Encoder decoder based architectures are used very widely for NLP tasks
  - MT, POS-tagging, super-tagging (very fine-grained tagging), summarisation, parsing, etc. etc.
- An unusual but more cognitively grounded use: learning verb conjugations

A little history: the. English past tense debate

- Rumelhart and McClelland (1986): describe a neural network capable of transducing English verb stems to their past tense
- Pinker and Prince (1988)'s rebuttal highlighted the very poor empirical performance of the model, along with many theoretical flaws
  - propose separate modules for regular and irregular verbs
  - regular verbs go through a rule-governed transduction mechanism
  - irregular verbs are produced via memory-lookup (or a context-specific rule)

A little history: the. English past tense debate

- Rumelhart and McClelland (1986): describe a neural network capable of transducing English verb stems to their past tense
- Pinker and Prince (1988)'s rebuttal highlighted the very poor empirical performance of the model, along with many theoretical flaws
  - propose separate modules for regular and irregular verbs
  - regular verbs go through a rule-governed transduction mechanism
  - irregular verbs are produced via memory-lookup (or a context-specific rule)
- A lot of word devoted to cognitively-plausible connectionist (neural) models of inflection since
  - Cotterel and Plunket (1994) presented an early use of a simple recurrent network (Elman, 1990).

English past tense morphology

orthographic			IPA		
stem	past	part.	stem	past	part.
go	went	gone	goʊ	wɛnt	gon
sing	sang	sung	sɪŋ	sæŋ	sʊŋ
swim	swam	swum	swɪm	swæm	swʊm
sack	sacked	sacked	sæk	sækt	sækt
sag	sagged	sagged	sæg	sægd	sægd
pat	patted	patted	pæt	pætɪd	pætɪd
pad	padded	padded	pæd	pædɪd	pædɪd

- Relatively impoverished, with 5 forms of *to be* and only three forms for most verbs
- Both regular and irregular patterning exits in English
- Canonical regular suffix is -ed (phonologically rendered as `[-d]` `[-t]` `[-Id]`, depending on the previous phonological segment)
- English irregulars are either unrelated e.g. go -> went, or sub-regular patterns like e.g. sing -> sang with several verbs in each sub-pattern

Acquisition of the past tense

- Both children and adults can generalise patterns to novel verbs (Wug-test, Berko(1958)) wug -> wugged

Acquisition of the past tense

- Both children and adults can generalise patterns to novel verbs (Wug-test, Berko(1958)) wug -> wugged
- Acquisition patterns for irregular verbs
  - Children exhibit a "U-shaped" learning curve
  - They first conjugate irregular forms correctly (come -> came)
  - Then then regress and go through a period of overgeneralisation (come -> comed), as they acquire the general past tense formation.
  - Finally they learn both irregular and regular conjugations correctly

Acquisition of the past tense

- Both children and adults can generalise patterns to novel verbs (Wug-test, Berko(1958)) wug -> wugged
- Acquisition patterns for irregular verbs
  - Children exhibit a "U-shaped" learning curve
  - They first conjugate irregular forms correctly (come -> came)
  - Then then regress and go through a period of overgeneralisation (come -> comed), as they acquire the general past tense formation.
  - Finally they learn both irregular and regular conjugations correctly
- They rarely over-irregularise, i.e. misconjugate a regular verb as if it were irregular (ping -> pang)
- rarely blend regular and irregular form, e.g. past tense of *come* is either produced as *comed* or *came*, but not *camed*.

Modern encoder-decoder architecture:

- Encoder RNN reads each symbol of the input string one at a time
- Decoder RNN produces a sequence of output phonemes one at a time
- Attention mechanism to look back at encoder states as needed
- Decoding ends when a stop symbol is produced

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N} p(y_i|y_1, \ldots, y_{i-1}, c_i)$$

where c_i is the (attention-weighted) sum of the encoder RNN hidden states h_i

- The log-likelihood of the training data is optimised, i.e. $\sum_{i=1}^{M} log\ p(\mathbf{y}^i|\mathbf{x}^{(i)})$

Experiments

- Dataset1 : CELEX Data set (4039 verb types), with present tense and past tense forms in IPA
  - Each verb is marked as regular or irregular
  - All verb types weighted equally for training (uniform distribution)
  - In real human communication, Zipfian distribution, but there is some evidence that human learners generalise phonological patterns based on the word types they appear in, ignoring the token frequency of those types
- Dataset 2: "Wug-test" : 74 nonce verb stems (Albright and Hayes, 2003)

Model

Kirov and Cotterel(2018)

- Encoder is a bi-directional LSTM with 2 layers and 100 hidden units
- Decoder is a uni-directional LSTM with 1 layer and 100 hidden units
- Each character has embedding size of 100, and training is done for 100 epochs

Results

- Kirov and Cotterell (2018):
  - learns to conjugate all verbs seen in the training set, including irregulars.
  - There are no blend errors of the sort eat -> ated
  - Accuracy is lower for irregular verbs, due to overregularisation. throw-> throwed
  - Macro-U-shaped curve is not observed.

Results

- Kirov and Cotterell (2018):
  - learns to conjugate all verbs seen in the training set, including irregulars.
  - There are no blend errors of the sort eat -> ated
  - Accuracy is lower for irregular verbs, due to overregularisation. throw-> throwed
  - Macro-U-shaped curve is not observed.
- Corkery, Matusevych, Goldwater (2019)
  - Reproduce broad results above.
  - Go into depth for "Wug-testing"
    - Behaviour on nonce verbs does not correlate with human-experimental data
    - Earlier rule-based models (Albridge and Hayes, 2003) perform better
    - Overproduces irregulars, while humans prefer the regular form.

Conclusion

- Modern encoder-decoder architectures overcome *some* of the flaws of earlier networks on this task
- Fit to human acquisition data and patterns is weak.

Summary of topics so far

- Distributional Semantics as a theory of word meaning
- Count-based models and neural-network (predict-based) models
- Concepts for language models (n-gram and neural), and their use in predict-based models
- Grammatical word-classes (POS-tags) and RNNs for sequence labelling (POS-tagging)
- Encoder-decoder networks and use for morphological learning
- Next time: encoder-decoders for generating linguistic descriptions of images