CNRS - INPT - UPS - UT1 - UTM

Recurrent Neural Networks for Genre-specific Language Generation in Dutch

Tim Van de Cruys
CNRS & IRIT, France

18 December 2015

Introduction

- Neural networks have shown impressive performance on a broad range of natural language processing tasks
  - Semantics
  - Language modeling
  - Machine translation
- A number of advances in neural network research
  - Word embeddings
  - 'deep' architectures
  - recurrent architectures
  - Encoder-decoder framework

Introduction

- This research: modeling of sentences using recurrent encoder-decoder framework
  - source sentence is read in using recurrent encoder framework
  - construction of a single vector representing the sentence
  - output of target sentence using recurrent decoder framework
- Genre-specific language generation: work in progress

Standard recurrent neural network

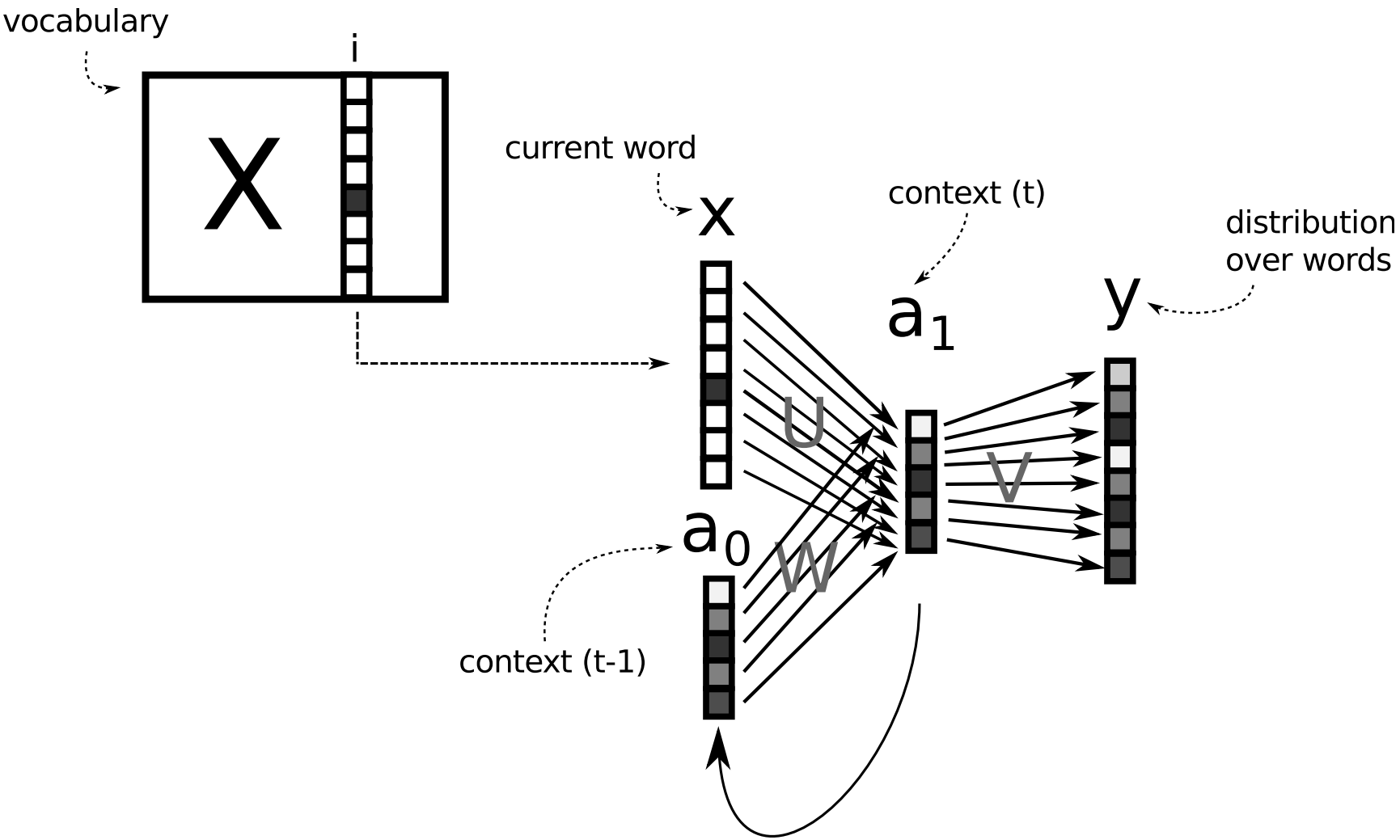
- Architecture for language modeling [Mikolov et al. 2010]
  - input layer: current word at time t
  - hidden layer: layer with recurrent connection
  - output layer: probability distribution over vocabulary words
- Hidden layer maintains representation of sentence history

Recurrent neural network

- $\mathbf{x}_1$: input layer (current word)
- $\mathbf{a}_1$: hidden layer of current timestep
- $\mathbf{a}_0$: hidden layer of previous timestep
- $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{V}$: weights matrices
- $f(\cdot)$: element-wise activation function (sigmoid)
- $g(\cdot)$: softmax function to ensure probability distribution

$$\begin{aligned} \mathbf{a}_1 &= f(\mathbf{U}\mathbf{x}_1 + \mathbf{W}\mathbf{a}_0) \\ \mathbf{y}_1 &= g(\mathbf{V}\mathbf{a}_1) \end{aligned}$$

Graphical representation



Recurrent neural network: problems

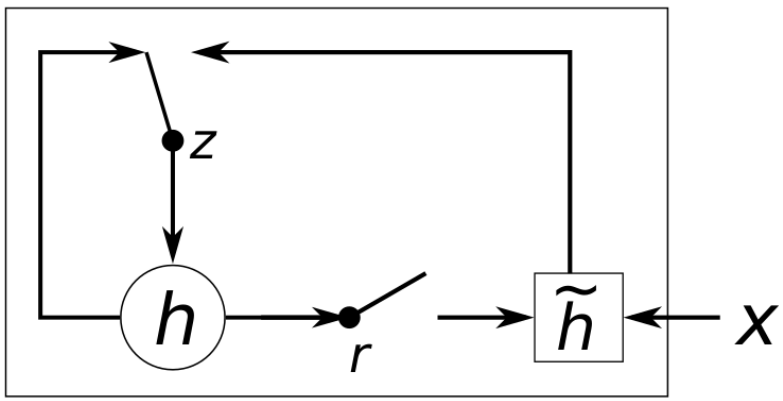
- Standard recurrent neural network architecture is somewhat problematic
  - Problem of 'vanishing gradients': gradients vanish exponentially quickly over time
  - Standard architecture is not able to capture long-range dependencies
  - A number of architectures have been proposed specifically designed to capture long-range dependencies
    - Long Short Term Memory networks (LSTM)
    - Gated Recurrent Units (GRU)

Recurrent neural network: problems

- Standard recurrent neural network architecture is somewhat problematic
  - Problem of 'vanishing gradients': gradients vanish exponentially quickly over time
  - Standard architecture is not able to capture long-range dependencies
  - A number of architectures have been proposed specifically designed to capture long-range dependencies
    - Long Short Term Memory networks (LSTM)
    - **Gated Recurrent Units (GRU)**

Gated Recurrent Unit

- Learn when and how to update the hidden state
- Controlled by two **gates**
  - reset gate (r): how much information from previous hidden state needs to be included (reset with current information?)
  - upgate gate (z): controls updates to hidden state (how much does hidden state need to be updated with current information?)

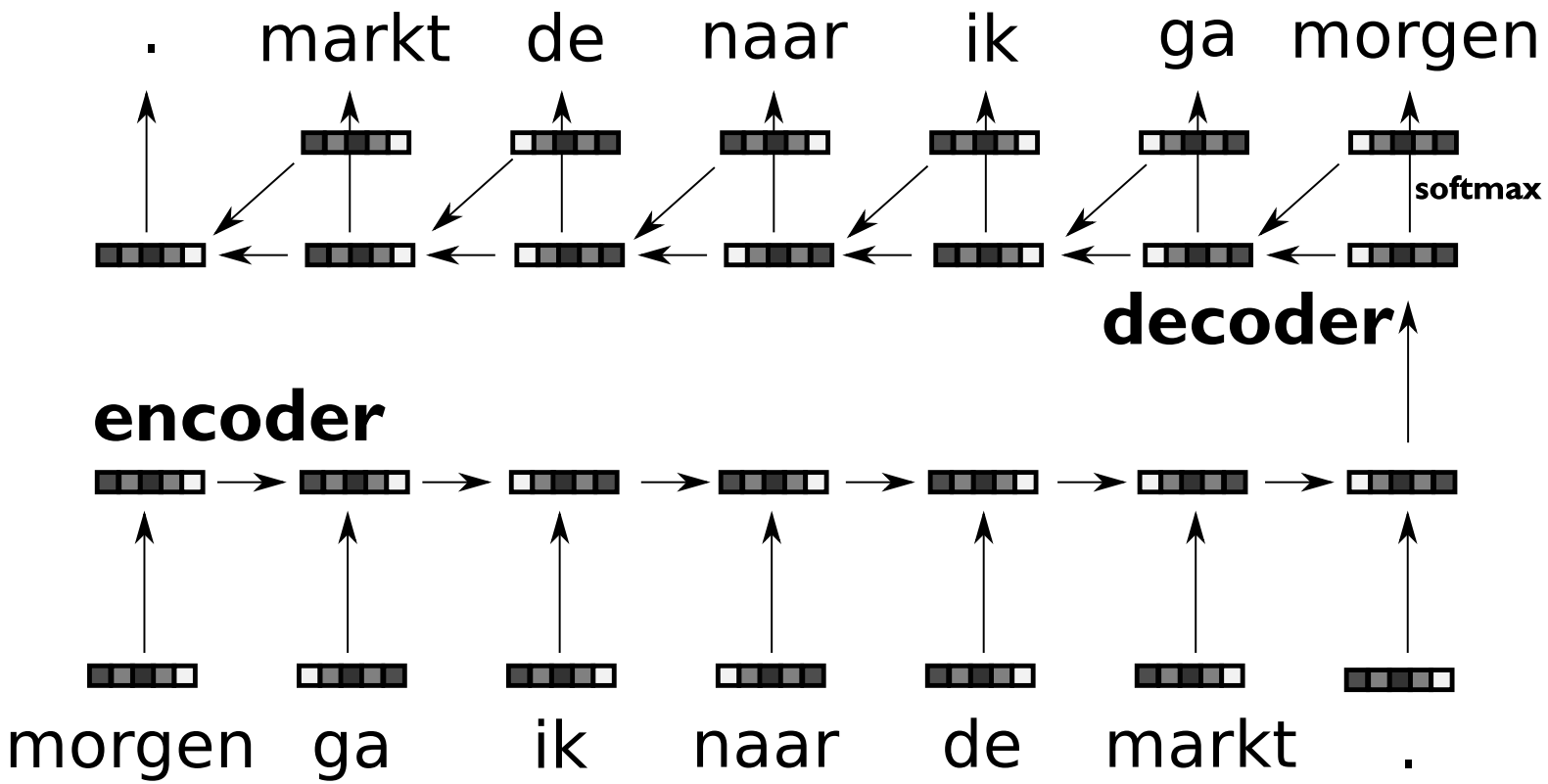


- Allows recurrent architecture to capture both short-term and long-term depedencies

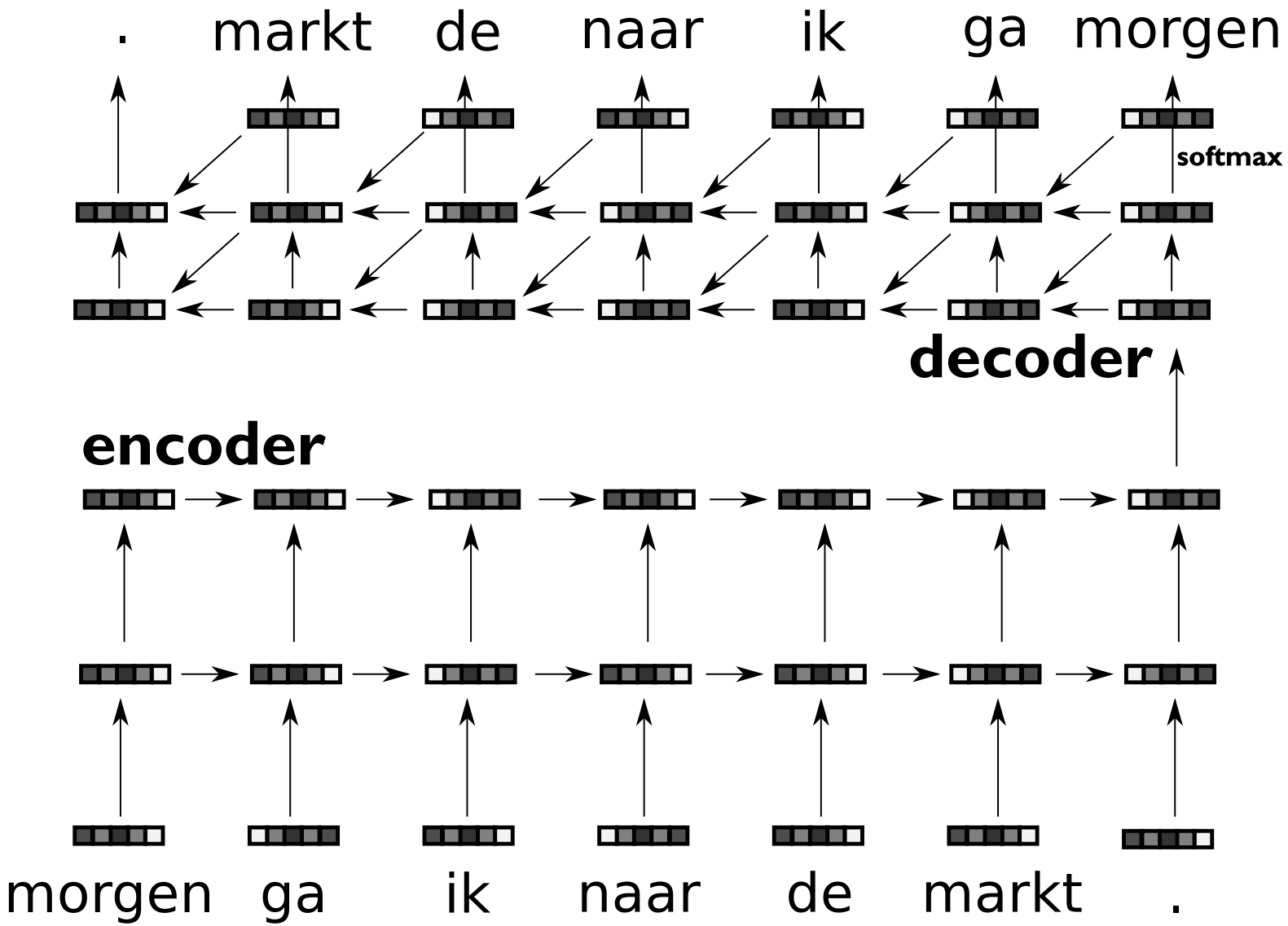
Encoder-decoder framework

- Encoder
  - One recurrent neural network acts as encoder
  - At each timestep, network iteratively reads in word from sentence, updating the hidden state
  - After encoding, hidden state (single vector) represents the entire sentence
- Decoder
  - Second recurrent neural network acts as decoder
  - At each timestep, recurrent neural network outputs word probability distribution (softmax)
  - Output of next timestep depends on hidden state and previous word

Encoder-decoder framework



Deep architecture



Implementational Details

- Model for Dutch, using NLCOW corpus
- Vocabulary of 10K words
- Word embeddings (256 dimensions) constructed using 1 billion words (python gensim module)
- Trained on 100 million sentences, containing up to 20 words
- Network architecture
  - Encoding: Embedding layer + 2 $\times$ GRU layer (256 units)
  - Decoding: Embedding layer + 2 $\times$ GRU layer (256 units)
  - Final softmax layer
- Stochastic gradient descent
- Dropout regularization: 0.2
- Algorithm implemented in Google's Tensorflow framework
- Run on GPU (NVIDIA Titan X)

Reconstructing sentences: examples (1)

```
wie belt om een zaal te reserveren krijgt iemand aan de telefoon
die nergens van weet .
-> wie belt om een zaal te reserveren krijgt iemand aan de
telefoon die nergens van weet .

voor het vormen van een betrouwbare theorie zijn meestal meerdere
waarnemingen nodig .
-> voor het vormen van een realistische theorie zijn meestal
meerdere waarnemingen nodig .

wat goed dat je meteen actie hebt ondernomen , laten we hopen dat
de medicijnen aan gaan slaan .
-> wat goed dat je meteen plannen hebt ondernomen , laten we
hopen dat de medicijnen aan gaan slaan .

wij zijn blij dat we van onze rotzooi af zijn .
-> wij zijn blij dat we van onze troep af zijn .
```

Reconstructing sentences: examples (2)

```
ze komen in amersfoort terecht , de meest gemiddelde stad van
nederland .
-> ze komen in apeldoorn terecht , de meest gemiddelde stad van
nederland .

vaak stonden wij als headliner heel laat op het podium .
-> vaak stonden wij als _UNK heel laat op het vuur .

want als je twijfelt moet je niet denken dat je iets van de here
zult krijgen .
-> want als je bid moet je niet denken dat je iets van de here
regelen krijgen .
```

Conclusion

- Recurrent neural network encoder-decoder architecture works remarkably well for reconstructing sentences
  - Encoder constructs hidden representation for complete sentence
  - GRU is able to capture long range dependencies
  - Decoder outputs word probabilities based on hidden state and previously chosen word
- Neural network is able to capture latent representation of sentences in one concise but sufficiently rich vector

Future work

- Genre-specific language generation
  - Neural network is able to capture latent representation of sentences in one concise but sufficiently rich vector
  - Train a decoder to output latent representation according to genre constraints
  - $\rightarrow$ Train on correlations with regard to a specific corpus
- Quantitative evaluation


Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. 2014.


Jiwei Li, Thang Luong, and Dan Jurafsky. A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1106–1115. 2015.


Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pp. 1045–1048. 2010.


Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531. 2011.


Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 2014.