Applications in NLP

Marco Kuhlmann
Department of Computer and Information Science
This lecture

- Introduction to natural language processing
- Application 1: Dependency parsing
- Neural dependency parsing
- Application 2: Machine translation
- Neural machine translation
- A central concept – Attention
Introduction to natural language processing
What is natural language processing?

- **Natural language processing** develops methods for making human language accessible to computers.

- Some well-known example applications are smart search engines, machine translation, and dialogue systems.

- These diverse applications are based on a common set of ideas from algorithms, machine learning, and other disciplines.
This Stanford University alumna co-founded educational technology company Coursera.

SPARQL query against DBPedia

```
SELECT DISTINCT ?x WHERE {
  dbr:Coursera dbo:foundedBy ?x.
}
```
General-purpose linguistic representations

dbr:Coursera dbo:foundedBy dbr:Daphne_Koller
Natural language processing from scratch

Figure 1: The Transformer - model architecture.

We employ a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is LayerNorm(x + Sublayer(x)), where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{model} = 512$.

Decoder:
The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position $i$ can depend only on the known outputs at positions less than $i$.

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention “Scaled Dot-Product Attention” (Figure 2). The input consists of queries and keys of dimension $d_k$, and values of dimension $d_v$. We compute the dot products of the values, and then apply a scaling factor $\sqrt{d_k}$ to normalize the attention weights. The scaled dot products are then used to compute the attention output as a weighted sum of the values.

Vaswani et al. (2017)  Tenney et al. (2019)
Two paradigms

- **Linguistic knowledge**
  Build pipelines of modular components that produce general-purpose representations grounded in linguistic knowledge. morphemes, parts-of-speech, dependency trees, meaning representations

- **Deep learning**
  Train end-to-end neural networks that directly transmute raw text into whatever structure the desired application requires.
Search and learning

\[ \hat{y} = \text{argmax} \text{ score}(x, y; \theta) \]
Search and learning

- **Search module**
  
The search module is responsible for finding the candidate output \( y \) with the highest score relative to the input \( x \).

- **Learning module**
  
The learning module is responsible for finding the model parameters \( \theta \) that maximize the predictive performance.

  For example, using supervised machine learning.
Language is special

- Unlike images or audio, text data is fundamentally discrete, with meaning created by combinatorial arrangement.

- Even though text appears as a sequence, machine learning methods must account for its implicit recursive structure.

- The distribution of linguistic elements resembles that of a power law – algorithms must be robust to unobserved events.
Heaps’ law

\[ y = Kx^\beta, \ K = 10, \ \beta = 0.5 \]
This lecture

- Introduction to natural language processing
- Application 1: Dependency parsing
- Neural dependency parsing
- Application 2: Machine translation
- Neural machine translation
- A central concept – Attention
Application 1: Dependency parsing
Dependency parsing

- **Syntactic parsing** is the task of mapping a sentence to a formal representation of its syntactic structure.

- We focus on representations in the form of **dependency trees**.

- A syntactic dependency is an asymmetric relation between a **head** and a **dependent**.

```
subject
Koller
```

```
object
co-founded
Coursera
```
# Current UD Languages

Information about language families (and genera for families with multiple branches) is mostly taken from [WALS Online](https://wals.info) (IE = Indo-European).

<table>
<thead>
<tr>
<th>Language</th>
<th>Subdivision</th>
<th>Count</th>
<th>Genus</th>
<th>Language Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans</td>
<td></td>
<td>49K</td>
<td>IE, Germanic</td>
<td></td>
</tr>
<tr>
<td>Akkadian</td>
<td></td>
<td>1K</td>
<td>Afro-Asiatic, Semitic</td>
<td></td>
</tr>
<tr>
<td>Albanian</td>
<td></td>
<td>&lt;1K</td>
<td>IE, Albanian</td>
<td></td>
</tr>
<tr>
<td>Amharic</td>
<td></td>
<td>10K</td>
<td>Afro-Asiatic, Semitic</td>
<td></td>
</tr>
<tr>
<td>Ancient Greek</td>
<td></td>
<td>416K</td>
<td>IE, Greek</td>
<td></td>
</tr>
<tr>
<td>Arabic</td>
<td></td>
<td>1,042K</td>
<td>IE, Greek</td>
<td></td>
</tr>
<tr>
<td>Armenian</td>
<td></td>
<td>52K</td>
<td>IE, Armenian</td>
<td></td>
</tr>
<tr>
<td>Assyrian</td>
<td></td>
<td>&lt;1K</td>
<td>Afro-Asiatic, Semitic</td>
<td></td>
</tr>
<tr>
<td>Bambara</td>
<td></td>
<td>13K</td>
<td>Mande</td>
<td></td>
</tr>
<tr>
<td>Basque</td>
<td></td>
<td>121K</td>
<td>Basque</td>
<td></td>
</tr>
<tr>
<td>Belarusian</td>
<td></td>
<td>13K</td>
<td>IE, Slavic</td>
<td></td>
</tr>
<tr>
<td>Bhojpuri</td>
<td></td>
<td>4K</td>
<td>IE, Indic</td>
<td></td>
</tr>
<tr>
<td>Breton</td>
<td></td>
<td>10K</td>
<td>IE, Celtic</td>
<td></td>
</tr>
<tr>
<td>Bulgarian</td>
<td></td>
<td>156K</td>
<td>IE, Slavic</td>
<td></td>
</tr>
<tr>
<td>Buryat</td>
<td></td>
<td>10K</td>
<td>Mongolic</td>
<td></td>
</tr>
<tr>
<td>Cantonese</td>
<td></td>
<td>13K</td>
<td>Sino-Tibetan</td>
<td></td>
</tr>
<tr>
<td>Catalan</td>
<td></td>
<td>531K</td>
<td>IE, Romance</td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td></td>
<td>285K</td>
<td>Sino-Tibetan</td>
<td></td>
</tr>
<tr>
<td>Classical Chinese</td>
<td></td>
<td>74K</td>
<td>Sino-Tibetan</td>
<td></td>
</tr>
<tr>
<td>Coptic</td>
<td></td>
<td>40K</td>
<td>Afro-Asiatic, Egyptian</td>
<td></td>
</tr>
<tr>
<td>Croatian</td>
<td></td>
<td>199K</td>
<td>IE, Slavic</td>
<td></td>
</tr>
<tr>
<td>Czech</td>
<td></td>
<td>2,222K</td>
<td>IE, Slavic</td>
<td></td>
</tr>
<tr>
<td>Danish</td>
<td></td>
<td>100K</td>
<td>IE, Germanic</td>
<td></td>
</tr>
<tr>
<td>Dutch</td>
<td></td>
<td>306K</td>
<td>IE, Germanic</td>
<td></td>
</tr>
<tr>
<td>English</td>
<td></td>
<td>620K</td>
<td>IE, Germanic</td>
<td></td>
</tr>
<tr>
<td>Erzya</td>
<td></td>
<td>15K</td>
<td>Uralic, Mordvin</td>
<td></td>
</tr>
<tr>
<td>Estonian</td>
<td></td>
<td>465K</td>
<td>Uralic, Finnic</td>
<td></td>
</tr>
<tr>
<td>Faroese</td>
<td></td>
<td>10K</td>
<td>IE, Germanic</td>
<td></td>
</tr>
<tr>
<td>Finnish</td>
<td></td>
<td>377K</td>
<td>Uralic, Finnic</td>
<td></td>
</tr>
<tr>
<td>French</td>
<td></td>
<td>1,157K</td>
<td>IE, Romance</td>
<td></td>
</tr>
<tr>
<td>Galician</td>
<td></td>
<td>164K</td>
<td>IE, Romance</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td></td>
<td>3,753K</td>
<td>IE, Germanic</td>
<td></td>
</tr>
<tr>
<td>Gothic</td>
<td></td>
<td>55K</td>
<td>IE, Germanic</td>
<td></td>
</tr>
<tr>
<td>Greek</td>
<td></td>
<td>63K</td>
<td>IE, Greek</td>
<td></td>
</tr>
<tr>
<td>Hebrew</td>
<td></td>
<td>161K</td>
<td>Afro-Asiatic, Semitic</td>
<td></td>
</tr>
</tbody>
</table>
Dependency trees

- A **dependency tree** for a sentence $x$ is a digraph $G = (V, A)$ where $V = \{1, \ldots, |x|\}$ and where there exists a $r \in V$ such that every $v \in V$ is reachable from $r$ via exactly one directed path.

- The vertex $r$ is called the **root** of $G$.

- The arcs of a dependency tree may be labelled to indicate the type of the syntactic relation that holds between the two elements.

Universal Dependencies v2 uses 37 universal syntactic relations (list).
Two parsing paradigms

• **Graph-based dependency parsing**
  Cast parsing as a combinatorial optimisation problem over a (possibly restricted) set of dependency trees.

• **Transition-based dependency parsing**
  Cast parsing as a sequence of local classification problems: at each point in time, predict one of several parser actions.
Graph-based dependency parsing

- Given a sentence $x$ and a set $Y(x)$ of candidate dependency trees for $x$, we want to find a highest-scoring tree $\hat{y} \in Y(x)$:

$$\hat{y} = \text{arg max}_{y \in Y(x)} \text{score}(x, y)$$

- The computational complexity of this problem depends on the choice of the set $Y(x)$ and the scoring function.
The arc-factored model

- Under the **arc-factored model**, the score of a dependency tree is expressed as the sum of the scores of its arcs:

\[
\hat{y} = \arg\max_{y \in Y(x)} \sum_{a \in y} \text{score}(x, a)
\]

- The score of a single arc can be computed by means of a neural network that receives the head and the dependent as input. For example, a simple linear layer:  

\[
\text{score}(x, h \rightarrow d) = [h ; d] \cdot w + b
\]
Computational complexity

- Under the arc-factored model, the highest-scoring dependency tree can be found in $O(n^3)$ time ($n = \text{sentence length}$).
  
  Chu–Liu/Edmonds algorithm; McDonald et al. (2005)

- Even seemingly minor extensions of the arc-factored model entail intractable parsing.
  
  McDonald and Satta (2007)

- For some of these extensions, polynomial-time parsing is possible for restricted classes of dependency trees.
Transition-based dependency parsing

- We cast parsing as a sequence of local classification problems such that solving these problems builds a dependency tree.

- In most approaches, the number of classifications required for this is linear in the length of the sentence.
Transition-based dependency parsing

- The parser starts in the **initial configuration**.
  
  empty dependency tree

- It then calls a classifier, which predicts the **transition** that the parser should make to move to a next configuration.

  extend the partial dependency tree

- This process is repeated until the parser reaches a **terminal configuration**.

  complete dependency tree
Training transition-based dependency parsers

- To train a transition-based dependency parser, we need a treebank with gold-standard dependency trees.
- In addition to that, we need an algorithm that tells us the gold-standard transition sequence for a tree in that treebank.
- Such an algorithm is conventionally called an oracle.
Comparison of the two parsing paradigms

<table>
<thead>
<tr>
<th>Graph-based parsing</th>
<th>Transition-based parsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>slow (in practice, cubic in the length of the sentence)</td>
<td>fast (quasi-linear in the length of the sentence)</td>
</tr>
<tr>
<td>restricted feature models (in practice, arc-factored)</td>
<td>rich feature models defined on configurations</td>
</tr>
<tr>
<td>features and weights directly defined on target structures</td>
<td>indirection – features and weights defined on transitions</td>
</tr>
</tbody>
</table>
This lecture

- Introduction to natural language processing
- Application 1: Dependency parsing
- Neural dependency parsing
- Application 2: Machine translation
- Neural machine translation
- A central concept – Attention
Neural dependency parsing
Learning problems in dependency parsing

- Learning a greedy transition-based dependency parser amounts to learning the transition classifier.
  Chen and Manning (2014), Kiperwasser and Goldberg (2016)

- Learning an arc-factored graph-based dependency parser amounts to learning the arc scores.
  Kiperwasser and Goldberg (2016), Dozat and Manning (2017)
Chen and Manning (2014)

- Pre-neural transition classifiers relied on linear models with hand-crafted combination features.
- C&M propose to replace the linear model with a two-layer feedforward network (FNN).
- The standard choice for the transfer function is the rectified linear unit (ReLU).

C&M use the cube function, $f(x) = x^3$. 
I wanted to try someplace new

```
[wanted to try] [someplace new]
```

```
stack buffer
```

```
softmax
```

```
FNN
```

```
concat
```

```
Embed Embed Embed
```

```
to try someplace
```

```
stack 2 stack 1 buffer 1
```

```

scores for the transitions
```
I wanted to try someplace new.

I wanted to try someplace new.

stack 2

stack 1

buffer 1

softmax

scores for the transitions

FNN

concat

Embed

Embed

Embed

wanted

try

someplace
Features

- C&M embed the top 3 words on the stack and buffer, as well as certain descendants of the top words on the stack.
  
  word embedding dimension = 50

- In addition to word embeddings, they also use embeddings for part-of-speech tags and dependency labels.
  
  tag embedding dimension = label embedding dimension = 50

- The resulting input dimension of the FNN is 2400.
Training

- To train their parser, C&M minimise the standard cross-entropy loss, plus an L2 regularisation term.

- To generate training examples for the transition classifier, they use the static oracle for the arc-standard algorithm.
  
can be generated off-line
### Parsing accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline, transition-based</td>
<td>89.4</td>
<td>87.3</td>
</tr>
<tr>
<td>Baseline, graph-based</td>
<td>90.7</td>
<td>87.6</td>
</tr>
<tr>
<td>Chen and Manning (2014)</td>
<td>91.8</td>
<td>89.6</td>
</tr>
<tr>
<td>Weiss et al. (2015)</td>
<td>93.2</td>
<td>91.2</td>
</tr>
</tbody>
</table>

Parsing accuracy on the test set of the Penn Treebank + conversion to Stanford dependencies
Kiperwasser and Goldberg (2016)

- The neural parser of C&M learns useful feature combinations, but the need to carefully design the core features remains.
- K&G propose to use a minimal set of core features based on contextualised embeddings obtained from a Bi-LSTM. Bi-LSTM is trained with the rest of the parser.
- They show that this approach gives state-of-the-art accuracy both for transition-based and for graph-based parsing.
I wanted to try someplace new.
I wanted to try someplace new

```
[I]  |  [wanted]  |  [try]  |  [someplace]  |  [new]
```

`stack 2` `stack 1` `buffer 1`

FNN scores for the transitions

\[ \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_5, \mathbf{v}_6 \]

`Bi-LSTM` `Embed` `Bi-LSTM` `Embed` `Bi-LSTM` `Embed` `Bi-LSTM` `Embed`
Features and training (transition-based parser)

- For their transition-based parser, K&G embed the top 3 words on the stack, as well as the first word in the buffer.
  
  word embedding dimension = 100, tag embedding dimension = 25

- In contrast to C&M, they use a dynamic oracle, so they cannot generate training examples in an off-line fashion.
I wanted to try someplace new.
I wanted to try someplace new.
Features and training (graph-based parser)

- For their graph-based parser, K&G embed the head and dependent of each arc.
  
  word embedding dimension = 100, tag embedding dimension = 25

- The training objective is to maximise the margin between the score of the gold tree $y^*$ and the highest-scoring incorrect tree $y$:

  $$L(\theta) = \max(0, 1 + \max_{y \neq y^*} \text{score}(x, y) - \text{score}(x, y^*))$$
## Parsing accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen and Manning (2014)</td>
<td>91.8</td>
<td>89.6</td>
</tr>
<tr>
<td>Weiss et al. (2015)</td>
<td>93.2</td>
<td>91.2</td>
</tr>
<tr>
<td>K &amp; G (2016), graph-based</td>
<td>93.0</td>
<td>90.9</td>
</tr>
<tr>
<td>K &amp; G (2016), transition-based</td>
<td>93.6</td>
<td>91.5</td>
</tr>
</tbody>
</table>

Parsing accuracy on the test set of the Penn Treebank + conversion to Stanford dependencies.
Based on the context-dependent embeddings, two FNNs create specialised representations of each word as a head/dependent:

\[ h_i = \text{FNN}_h(v_i) \quad d_i = \text{FNN}_d(v_i) \]

These specialised representations are then scored via a bilinear layer with weight tensor \( U \) and bias vector \( b \):

\[ \text{score}(x, i \rightarrow j) = h_i U d_j^T + (h_i b)^T \]
I wanted to try someplace new

\[
\begin{align*}
\mathbf{v}_1 & \quad \mathbf{v}_2 & \quad \mathbf{v}_3 & \quad \mathbf{v}_4 & \quad \mathbf{v}_5 & \quad \mathbf{v}_6 \\
\text{head} & \quad \text{dependent} & \quad \text{head} & \quad \text{dependent}
\end{align*}
\]
### Parsing accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS</th>
<th>LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen and Manning (2014)</td>
<td>91.8</td>
<td>89.6</td>
</tr>
<tr>
<td>K &amp; G (2016), graph-based</td>
<td>93.0</td>
<td>90.9</td>
</tr>
<tr>
<td>K &amp; G (2016), transition-based</td>
<td>93.6</td>
<td>91.5</td>
</tr>
<tr>
<td>Dozat and Manning (2017)</td>
<td>95.7</td>
<td>94.1</td>
</tr>
</tbody>
</table>

Parsing accuracy on the test set of the Penn Treebank + conversion to Stanford dependencies
This lecture

- Introduction to natural language processing
- Application 1: Dependency parsing
- Neural dependency parsing
- Application 2: Machine translation
- Neural machine translation
- A central concept – Attention
Application 2: Machine translation
Machine translation is the task of automatically translating text in one language (the source) into another language (the target).

Maskinöversättning är uppgiften att automatiskt översätta text på ett språk (källan) till ett annat språk (målet).
A timeline of machine translation

1950: rule-based machine translation
1980: example-based machine translation
1990: statistical machine translation
2015-2020: neural machine translation
Interlingual machine translation

Abstract representation:
- **Intentionally_create**
  - Creator: dbr:Daphne_Koller
  - Created_entity: dbr:Coursera

Linguistic structure:
- **Koller co-founded Coursera**
- **Bhunaigh Koller Coursera**

Surface sentence:
- Koller co-founded Coursera
- Bhunaigh Koller Coursera
When I look at an article in Russian, I say: ‘This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.’

Warren Weaver (1894–1978)
Statistical machine translation (SMT)

- Formulate machine translation as an optimisation task: Given a source sentence $x$, find the most probable target sentence $y$:

$$\arg \max_y P(y \mid x)$$

- Use Bayes’ rule to decompose the probability model into two components that can be learned separately:

$$\arg \max_y P(x \mid y) P(y)$$
Parallel corpora

- Canadian Hansard (English–French); extracted from the proceedings of the Canadian Parliament.

- Europarl (21 languages); 30.32 M parallel sentences extracted from the proceedings of the European parliament.
  
  [Link to the Europarl website](#)

- OPUS (several languages); growing collection of translated texts, automatically preprocessed and aligned.

  [Link to the OPUS website](#)
Word-to-word alignments

das
haus
ist
klein

the
house
is
small

klein
ist
das
haus

das
haus
ist
klitzeklein

huset
är
litet

das
haus
ist
klein

the
house
is
just
small

he
kicked
the
bucket
IBM Model 1

- First in a series of increasingly complex statistical translation models; deals only with lexical (word-to-word) translation.

- Central component: The lexical translation probability $t$ of observing a source word $x$, given the aligned target word $y$.

\[
P(x, a \mid y) = \frac{\varepsilon}{(|y| + 1)|x|} \prod_{j=1}^{\mid x \mid} t(x_j \mid y_{a(j)})
\]

Brown et al. (1993)
Training statistical machine translation models

- We would like to estimate the lexical translation probabilities from a parallel corpus – but we do not have the alignments.

- We can bootstrap the translation probabilities and alignments in parallel using the Expectation Maximization (EM) algorithm:
  1. initialise the model parameters randomly
  2. calculate alignments based on the current model parameters
  3. estimate new model parameters from the new alignments
  4. repeat steps 2–3 until convergence

Brown et al. (1993)
Research on statistical machine translation led to significant improvements in the availability and quality of translation.

The first version of Google Translate (2006–2016) was an SMT system.

However, the best systems were extremely complex and required large amounts of external resources and feature engineering.

Open-source example: **Moses**
Evaluation: BLEU (Bilingual Evaluation Understudy)

- BLEU compares the automatic translation of a source sentence to one or several human-created translations.

- BLEU combines $n$-gram precision (for $n$ up to 4) with a brevity penalty for too-short translations.

- BLEU has been criticised for not correlating well with human judgement, and several other evaluation measures exist.

Papineni et al. (2002); Callison-Burch et al. (2006)
This lecture

• Introduction to natural language processing
• Application 1: Dependency parsing
• Neural dependency parsing
• Application 2: Machine translation
• Neural machine translation
• A central concept – Attention
Neural machine translation
Neural Machine Translation (NMT)

- **Neural machine translation (NMT)** models the translation task through a single artificial neural network.

- The first systems for NMT were based on recurrent neural networks; more recent systems typically use Transformers.

- Many practical implementations are based on the OpenNMT ecosystem for neural machine translation.

[Link to OpenNMT]
The sequence-to-sequence model (seq2seq)

The sequence-to-sequence model consists of two components:

- **The encoder** is a neural network that produces a representation of the source sentence.
  
  typically implemented as a recurrent neural network with attention

- **The decoder** is an autoregressive language model that generates the target sentence, conditioned on the output of the encoder.
  
  autoregressive = takes its own outputs as new inputs
Standard seq2seq architecture

Sutskever et al. (2014)
Standard seq2seq architecture

encoder

decoder

Sutskever et al. (2014)
Standard seq2seq architecture

encoder

decoder

Sutskever et al. (2014)
Standard seq2seq architecture

encoder

decoder

Sutskever et al. (2014)
Standard seq2seq architecture

Sutskever et al. (2014)
Properties of the seq2seq model

- The seq2seq model directly learns and uses $P(y|x)$, rather than decomposing it into $P(x|y)$ and $P(y)$ as in SMT.
- The model can be trained end-to-end using backpropagation, without alignments or auxiliary models.
- Only needs parallel data
- The seq2seq model is useful for a range of other tasks, including text summarisation, dialogue, and code generation.
Training an encoder–decoder model
Training an encoder–decoder model

\[ L = \frac{1}{T} \sum_{t=1}^{T} L_t \]
Decoding algorithms

- **Greedy decoding**
  At each step, predict the highest-probability word. Stop when the end-of-sentence marker is predicted.

- **Beam search**
  Keep a limited number of highest-scoring partial translations. Expand the items on the beam, score the new items, and prune.
  
  Typical beam widths are between 2 and 16.
Beam search example
Beam search example

\[
\begin{align*}
\text{<BOS>} & \quad -0.5 \\
\text{drink} & \quad -0.3 \\
\text{just} & \quad -0.5
\end{align*}
\]
Beam search example

-0.5

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.5

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3

-0.1

-0.7

-0.6

-0.4

-0.3
Beam search example
Beam search example

-0.5
<BOSS>
-0.3
drink
  -0.4
  just
    -0.7
    simple
-0.6
just
  -0.2
  more
-0.2
simply
  -0.9
  more
  -0.5
  greater
-0.1
drink
  -0.1
  coffee
  -0.9
-0.2
more
  -0.7
  tea
  -1.5
-0.4
greater
-0.7
-0.6
-0.2
-0.5
-0.9

-1.8
-1.5
-1.1
-1.8
-1.5
-1.1
Beam search example

<BOS> -0.5
  <BOS> -0.3
    drink -0.4
      just -0.1
    simple -0.7
      more -0.2
      greater -0.1
    coffee -0.9
      tea -1.5
    greater -0.5
    tea -1.8
    coffee -1.1
Termination criteria

- When the expansion of a partial translation generates the ⟨EOS⟩ marker, store the result as a complete translation.
- End the search after a fixed number of steps, or when enough complete translations have been generated.
- Evaluate the translations found during search based on their length-normalised scores and return the highest-scoring one.

different from standard beam search
Information bottleneck

Sutskever et al. (2014)
This lecture

- Introduction to natural language processing
- Application 1: Dependency parsing
- Neural dependency parsing
- Application 2: Machine translation
- Neural machine translation
- A central concept – Attention
A central concept – Attention
Recency bias in recurrent neural networks

LSTM → LSTM → LSTM → v

Embed

great

Embed

monster

Embed

movie
Recency bias in recurrent neural networks

The last hidden state is prone to bias towards the recent past.

Chen et al. (2016); Werlen et al. (2018)
Attention

- In the context of text classification, attention enables the model to learn which words are the most important ones.
- Essentially, we compute a set of weights that allow us to score words based on how much the model should ‘attend to them’.
- Attention was first proposed in the context of neural machine translation, but is now used in many models.

Bahdanau et al. (2015)
Attention as word alignments

In the context of the encoder–decoder architecture for neural machine translation, attention can be interpreted as word alignments.

Image source: Bahdanau et al. (2015)
Attention for classification

Cheng et al. (2016)
Attention for classification

Cheng et al. (2016)
Attention for classification

Cheng et al. (2016)
Attention for classification

Cheng et al. (2016)
Attention for classification

Cheng et al. (2016)
A more general characterisation of attention

- In general, attention can be described as a mapping from a query $q$ and a set of key–value pairs $k_i, v_i$ to an output.

- The output is the weighted sum of the $v_i$, where the weight of each $v_i$ is given by the compatibility between $q$ and $k_i$.
  
  The dot product provides a measure of compatibility.

- In the classification architecture, keys and values are the same; they correspond to the hidden states $h_i$.

Vaswani et al. (2017)
Scaled dot-product attention

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V
\]

- Used in the Transformer architecture. [Vaswani et al. (2017)]
- The input consists of queries and keys of dimension \(d_k\), and values of dimension \(d_v\).
- Scaling prevents the softmax from being pushed into regions with small gradients.
Interpretation of attention

- In addition to improved performance, attention is attractive because it allows us to inspect what a network attends to. Visualise weights; correlate weights to external data or human rationales.

- The discussion of the possibilities and limitations of using attention to interpret neural models is ongoing.

  Is Attention Interpretable? (Serrano and Smith, 2019)
  Attention is not Explanation (Jain and Wallace, 2019)
  Attention is not not Explanation (Wiegrefe and Pinter, 2019)
This lecture

- Introduction to natural language processing
- Application 1: Dependency parsing
- Neural dependency parsing
- Application 2: Machine translation
- Neural machine translation
- A central concept – Attention