Sequence-to-sequence models
used for machine translation and

Murat Apishev
Katya Artemova

Computational Pragmatics Lab, HSE

December 2, 2019
Today

1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models
Sequence-to-sequence

Neural encoder encoder architectures

- Achieves high results on machine translation, spelling correction, summarization and other NLP tasks
- The encoder inputs sequence of tokens $x_{1:n}$ and outputs hidden states $h_n^E$
- The decoder decodes an output sequence of tokens $y_{1:n}$ by decoding last hidden state $h_0^D = h_n^E$
- seq2seq architectures are trained on parallel corpora
Seq-to-seq for MT

- Both encoder and decoder are recurrent networks
- Input words $x_i$ ($i \in [1, n]$) are represented as word embeddings (w2v for example)
- The context vector: $h_n$, last hidden state of RNN encoder, turns out to be a bottleneck
- It is challenging for the models to deal with long sentences as the impact of last words is higher
- **Attention mechanism** is one of the possible solutions
**Seq-to-seq for MT + attention**

**Attention mechanism** allows to align input and output words.

- The encoder passes all the hidden states to the decoder: not $h_n^E$, but rather $h_i^E, i \in [1, n]$
- The hidden states can be treated as context-aware word embeddings
- The hidden states are used to produce a context vector $c$ for the decoder

![Diagram](image-source)
Seq-to-seq for MT + attention

- At the step $j$ the decoder inputs $h_{j-1}^{D}, j \in [n+1,m]$ and a context vector $c_j$ from the encoder.
- The context vector $c_j$ is a linear combination of the encoder hidden states:
  $$c_j = \sum_i \alpha_i h_i^E$$

$\alpha_i$ are attention weights which help the decoder to focus on the relevant part of the encoder input.
Seq-to-seq MT + attention

To generate a new word the decoder at the step $j$:
- inputs $h_{j-1}^D$ and produces $h_j^D$
- concatenates $h_j^D$ to $c_j$
- passes the concatenated vector through linear layer with softmax activation to get a probability distribution over target vocabulary
### Attention weights

- The attention weights $\alpha_{ij}$ measure the similarity of the encoder hidden state $h^E_i$ while generating the word $j$

$$a_{ij} = \frac{\exp(\text{sim}(h^E_i, h^D_j))}{\sum_k \exp(\text{sim}(h^E_k, h^D_j))}$$

- The similarity $\text{sim}$ can be computed by
  - dot product attention:
    $$\text{sim}(h, s) = h^T s$$
  - additive attention:
    $$\text{sim}(h, s) = w^T \tanh(W_h h + W_s s)$$
  - multiplicative attention:
    $$\text{sim}(h, s) = h^T W s$$

- Weights are trained jointly with the whole model
Attention map

Figure: Visualisation of attention weights
**MT metrics**

- BLEU compares system output to reference translation
  
  **Reference translation:** *E-mail was sent on Tuesday*
  
  **System output:** *The letter was sent on Tuesday*

- Given $N$ ($N \in [1, 4]$) compute the number of $N$-grams present both in system output and reference translation:
  
  $N = 1 \Rightarrow \frac{4}{6}$  
  $N = 2 \Rightarrow \frac{3}{5}$  
  $N = 3 \Rightarrow \frac{2}{4}$  
  $N = 4 \Rightarrow \frac{1}{3}$

- Take geometric mean $N$:
  
  $$\text{score} = \sqrt[4]{\frac{4}{6} \cdot \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3}}$$

- Brevity penalty:
  
  $$BP = \min(1, \frac{6}{5})$$
Today

1. Machine translation
2. **Task oriented chat-bots**
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models
Natural language understanding

Two tasks (intent detection and slot filling): identify speaker’s intent and extract semantic constituents from the natural language query

<table>
<thead>
<tr>
<th>Sentence</th>
<th>first</th>
<th>class</th>
<th>fares</th>
<th>from</th>
<th>boston</th>
<th>to</th>
<th>denver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slots</td>
<td>B-class_type</td>
<td>I-class_type</td>
<td>O</td>
<td>O</td>
<td>B-fromloc</td>
<td>O</td>
<td>B-toiloc</td>
</tr>
<tr>
<td>Intent</td>
<td></td>
<td></td>
<td>airfare</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure: ATIS corpus sample with intent and slot annotation

- Intent detection is a classification task
- Slot filling is a sequence labelling task

NLU datasets: ATIS [1], Snips [2]
Joint intent detection and slot filling [3]

1. The encoder models is a biLSTM
2. The decoder is a unidirectional LSTM
3. At each step the decoder state $s_i$ is: $s_i = f(s_{i-1}, y_{i-1}, h_i, c_i)$, where $c_i = \sum_j^T \alpha_{i,j} h_j$.

   $\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_k^T \exp(e_{i,k})}$,

   $e_{i,k} = g(s_{i-1}, h_k)$

The inputs are explicitly aligned. Costs from both decoders are back-propagated to the encoder.

**Figure:** Encoder-decoder models
Joint intent detection and slot filling [3]

- BiLSTM reads the source sequence
- forward RNN models slot label dependencies
- the hidden state $h_i$ at each step is a concatenation of the forward state $fh_i$ and backward state $bh_i$
- the hidden state is $h_i$ combined with the context vector $c_i$
- $c_i$ is calculated as a weighted average of $h = (h_1, ..., h_T)$

**Figure:** RNN-based model

**Figure:** Attention weights
Today

1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models
Grammar as a Foreign Language [4]

![Constituency Parsing Example](attachment:grammar_example.png)

**Figure**: Example parsing task and its linearization

John has a dog. → S

- NP
- VP
  - NNP
  - VBZ
  - NP
    - DT
    - NN

John has a dog. → (S (NP NNP)_{NP} (VP VBZ (NP DT NN)_{NP} )_{VP} . )_{S}
Grammar as a Foreign Language [4]

**Figure:** LSTM + attention encoder-decoder model for parsing
Grammar as a Foreign Language [4]

- The encoder LSTM is used to encode the sequence of input words $A_i, |A| = T_A$
- The decoder LSTM is used to output symbols $B_i, |B| = T_B$
- The attention vector at each output time $t$ over the input words:

$$
  u_i^t = v^T \tanh(W_1 h_i^E + W_2 h_t^D)
$$

$$
  a_i^t = \text{softmax}(u_i^T)
$$

$$
  d_t' = \sum_{i=1}^{T_A} a_i^t h_i^E,
$$

where the vector $v$ and matrices $W_1, W_2$ are learnable parameters of the model.
Today

1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models
Neural Language Correction with Character-Based Attention [5]

- Trained on a parallel corpus of “bad” \((x)\) and “good” \((y)\) sentences
- Encoder has a pyramid structure:
  \[
  f_t^{(j)} = \text{GRU}(f_{t-1}^{(j-1)}, c_t^{(j-1)})
  \]
  \[
  b_t^{(j)} = \text{GRU}(b_{t+1}^{(j-1)}, c_t^{(j-1)})
  \]
  \[
  h_t^{(j)} = f_t^{(j)} + b_t^{(j)}
  \]
  \[
  c_t^{(j)} = \tanh(W_{pyr}[h_{2t}^{(j-1)}, h_{2t+1}^{(j-1)}]) + b_{pyr}^{(j)}
  \]

Figure: An encoder-decoder neural network model with two encoder hidden layers and one decoder hidden layer
Neural Language Correction with Character-Based Attention [5]

- **Decoder network:**
  \[ d_t^{(j)} = \text{GRU}(d_{t-1}^{(j-1)}, x_t) \]

- **Attention mechanism:**
  \[ u_{tk} = \phi_1(d^{(M)})^\top \phi_2(c_k), \phi : \text{tanh}(W \times \cdot) \]
  \[ \alpha_{tk} = \frac{u_{tk}}{\sum_j u_{tj}} \]
  \[ a_t = \sum_j \alpha_{tj} c_j \]

- **Loss:**
  \[ L(x, y) = \sum_{t=1}^T \log P(y_t | x, y_{<t}) \]

*Figure:* An encoder-decoder neural network model with two encoder hidden layers and one decoder hidden layer.
Neural Language Correction with Character-Based Attention [5]

- Beam search for decoding:
  \[ s_k(y_{1:k}|x) = \log P_{NN}(y_{1:k}|x) + \lambda \log P_{LM}(y_{1:k}) \]

- Synthesizing errors: article or determiner errors (ArtOrDet) and noun number errors (Nn)

Figure: An encoder-decoder neural network model with two encoder hidden layers and one decoder hidden layer
Today

1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models
Summarization & Simplification

Summarization

**Summarization** is the task of producing a shorter version of one or several documents that preserves most of the input's meaning.

1. **Abstractive** summarization: paraphrase the corpus using novel sentences
2. **Extractive** summarization: concatenate extracts taken from a corpus into a summary

Simplification

**Simplification** consists of modifying the content and structure of a text in order to make it easier to read and understand, while preserving its main idea and approximating its original meaning.

Image source: nlpprogress.com
ROUGE is used to compare a system summary or translation against a set of reference human summaries:

$$\text{ROUGE}_n = \frac{\text{number of overlapping n-grams}}{\text{number of n-grams in reference summary}}$$

$$R_{LCS} = \frac{LCS(X, Y)}{|X|}, P_{LCS} = \frac{LCS(X, Y)}{|Y|}, \text{ROUGE}_L = \frac{(1 + \beta^2)R_{LCS}P_{LCS}}{R_{LCS} + \beta^2P_{LCS}}$$

where $LCS(X, Y)$ is the length of a longest common subsequence of $X$ and $Y$. 
Metrics: METEOR [7]
Metric for Evaluation of Translation with Explicit ORdering

METEOR is used to compare a system summary or translation against a set of reference human summaries:

\[ P = \frac{\text{number of overlapping words}}{\text{number of words in system summary}} \]

\[ R = \frac{\text{number of overlapping words}}{\text{number of words in reference summary}} \]

\[ F_{\text{mean}} = \frac{10PR}{R + 9P}, \text{penalty} = 0.5\left(\frac{\text{number of chunks}}{\text{number of overlapping words}}\right)^3 \]

\[ M = F_{\text{mean}}(1 - p) \]
Datasets: CNN / Daily Mail [8], [9]

The dataset contains online news articles (781 tokens on average) paired with multi-sentence summaries (3.75 sentences or 56 tokens on average). The processed version contains 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs.
Datasets: Webis-TLDR-17 [10]

The dataset contains 4 million content-summary pairs from Reddit.

---

Example Submission

**Title:** Ultimate travel kit

**Body:** Doing some traveling this year and I am looking to build the ultimate travel kit... So far I have a Bonavita 0.5L travel kettle and AeroPress. Looking for a grinder that would maybe fit into the AeroPress. This way I can stack them in each other and have a compact travel kit.

**TL;DR:** What grinder would you recommend that fits in AeroPress?

---

Example Comment (to a different submission)

**Body:** Oh man this brings back memories. When I was little, around five, we were putting in a new shower system in the bathroom and had to open up the wall. The plumber opened up the wall first, then put in the shower system, and then left it there while he took a lunch break. After his break he patched up the wall and left, having completed the job. Then we couldn’t find our cat. But we heard the cat. Before long we realized it was stuck in the wall, and could not get out. We called up the plumber again and he came back the next day and opened the wall. Out came our black cat, Socrates, covered in dust and filth.

**TL;DR:** Plumber opens wall, cat climbs in, plumber closes wall, fucking meows everywhere until plumber returns the next day.
Datasets: headline generation

2. RIA news dataset [12]
Datasets: WikiSmall [13]

Main source for simplified sentences is Simple English Wikipedia. WikiSmall is a parallel corpus with more than 108K sentence pairs from 65,133 Wikipedia articles, allowing 1-to-1 and 1-to-N alignments.
Get to the Point [14]
Sequence-to-sequence attentional model

- Bahdanau attention: \( e_t^i = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \), \( a_t = \text{softmax}(e_t) \)
- Context vector: \( h_t = \sum_i a_t^i h_i \)
- Vocabulary distribution: \( P_{vocab} = \text{softmax}(V'(V[s_t, h_t] + b) + b) \)
- NLL loss: \(-\frac{1}{T} \sum_{t=0}^{T} \log P(w_t^*)\)
Get to the Point [14]

Pointer-generator model
Get to the Point [14]

Pointer-generator model

- Generation probability: \( p_{gen} = \sigma(w_h^T h_t + w_s^T s_t + w_x^T x_t + b_{ptr}) \)
- \( p_{gen} \) is used to switch between sampling from \( P_{vocab} \) or copying by sampling \( a^t \)
- \( P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \)
A Deep Reinforced Model
for Abstractive Summarization [15]
A Deep Reinforced Model for Abstractive Summarization [15]

- **Intra-temporal attention:**
  \[
  e_{ti} = h^d_t W^e_{attn} h^e_i, \\
  \alpha^e_{ti} = \text{softmax}(e_{ti}), \\
  c^e_t = \sum_{i=1}^{n} \alpha^e_{ti} h^e_i
  \]

- **Intra-decoder attention:**
  \[
  e_{tt'} = h^d_t W^d_{attn} h^d_i, \\
  \alpha^d_{tt'} = \text{softmax}(e_{tt'}), \\
  c^d_t = \sum_{i=j}^{t-1} \alpha^d_{tj} h^d_k
  \]
A Deep Reinforced Model
for Abstractive Summarization [15]

- Token generation:

\[ p(y_t|u_t = 0) = \text{softmax}(W_{out}[h^d_t, c^e_t, c^d_t] + b_{out}) \]

- Pointer:

\[ p(y_t = x_i|u = 1) = \alpha^e_{ti} \]

\[ p(u_t = 1) = \sigma(W_u[h^d_t, c^e_t, c^d_t] + b_u) \]

- Probability distribution for the output token:

\[ p(y_t) = p(u_t = 1)p(y_t|u_t = 1) + p(u_t = 0)p(y_t|u)t = 0 \]

- Sharing decoder weights: \( W_{out} = \text{tanh}(W_{emb}W_{proj}) \)
A Deep Reinforced Model for Abstractive Summarization [15]

- Hybrid learning objective:

  $$L_{\text{mixed}} = \gamma L_{rl} + (1 - \gamma) L_{ml}$$

- Teacher forcing:

  $$L_{ml} = \sum_{t=1}^{n'} \log p(y_t | y_1, \ldots, y_{t-1}, x)$$

- Policy learning:

  $$L_{rl} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{n'} \log p(y^s_t | y^s_1, \ldots, y^s_{t-1}, x),$$

where $r$ is a reward function, $\hat{y}$ is the baseline output, obtained by maximizing the output probability distribution at each time step.
Today

1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models
Types of questions

1. Factoid questions:
   - What is the dress code for the Vatican?
   - Who is the President of the United States?
   - What are the dots in Hebrew called?

2. Commonsense questions:
   - What do all humans want to experience in their own home? (a) feel comfortable, (b) work hard, (c) fall in love, (d) lay eggs, (e) live forever

3. Opinion questions:
   - Can anyone recommend a good coffee shop near HSE campus?

4. Cloze-style questions
Types of questions

1. Types of answers
   - binary (yes / now)
   - find a span of text
   - multiple choice
Major paradigms for factoid question answering

1. Information retrieval (IR)-based QA: find a span of text, which answers a question

2. Open-domain Question Answering (ODQA): answer questions about nearly anything

3. Knowledge (KB)-based QA: build a semantic representation of question are used to question knowledge bases

*When Bernardo Bertolucci died?* → death-year(Bernardo Bertolucci, ?x)
Today

1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering

7. IR-based QA
   - Datasets
   - Models
1. Question processing
   ▶ answer type (PER, LOC, TIME)
   ▶ focus
   ▶ question type

2. Query formulation
   ▶ question reformulation: remove *wh*-words, change word order
   ▶ query expansion
3. Document and passage retrieval

4. Answer extraction

*What are the dots in Hebrew called?*

*In Hebrew orthography, niqqud or nikkud, is a system of diacritical signs used to represent vowels or distinguish between alternative pronunciations of letters of the Hebrew alphabet.*
1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models

Apisheva, Artemova (HSE)  Sequence-to-sequence models  December 2, 2019
Datasets for IR-based QA

Passage: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla’s breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.

Question: On what did Tesla blame for the loss of the initial money?

Answer: Panic of 1901

1. Stanford Question Answering Dataset (SQuAD)
2. NewsQA
3. WikiQA
4. WebQuestions
5. WikiMovies
6. Russian: SberQUAD
7. MedQuAD [16]
SQuAD2.0 [17], [18]

100,000 questions in SQuAD1.1 and over 50,000 unanswerable questions in SQuAS2.0

1 Project Nayuki’s Wikipedia’s internal PageRanks to obtain the top 10000 articles of English Wikipedia, from which we sampled 536 articles uniformly at random

2 Articles splitted in individual paragraphs

3 Crowsourcing: ask and answer up to 5 questions on the content of that paragraph

4 Crowdworkers were encouraged to ask questions in their own words, without copying word phrases from the paragraph

5 Analysis: the (i) diversity of answer types, (ii) the difficulty of questions in terms of type of reasoning required to answer them, and (iii) the degree of syntactic divergence between the question and answer sentences.

https://rajpurkar.github.io/SQuAD-explorer/
Figure: An example from RACE dataset

RACE consists of near 28k passages and near 100k questions generated by human experts (English instructors), and covers a variety of topics which are carefully designed for evaluating the students’ ability in understanding and reasoning.
**Figure**: Statistic information about Reasoning type in different datasets

RACE includes five classes of questions: word matching, paraphrasing, single-sentence reasoning, multi-sentence reasoning, insufficient or ambiguous questions.

http://www.cs.cmu.edu/~glai1/data/race/

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RACE-M</th>
<th>RACE-H</th>
<th>RACE</th>
<th>CNN</th>
<th>SQUAD</th>
<th>NEWSQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Matching</td>
<td>29.4%</td>
<td>11.3%</td>
<td>15.8%</td>
<td>13.0%</td>
<td>39.8%*</td>
<td>32.7%*</td>
</tr>
<tr>
<td>Paraphrasing</td>
<td>14.8%</td>
<td>20.6%</td>
<td>19.2%</td>
<td>41.0%</td>
<td>34.3%*</td>
<td>27.0%*</td>
</tr>
<tr>
<td>Single-Sentence Reasoning</td>
<td>31.3%</td>
<td>34.1%</td>
<td>33.4%</td>
<td>19.0%</td>
<td>8.6%*</td>
<td>13.2%*</td>
</tr>
<tr>
<td>Multi-Sentence Reasoning</td>
<td>22.6%</td>
<td>26.9%</td>
<td>25.8%</td>
<td>2.0%*</td>
<td>11.9%*</td>
<td>20.7%*</td>
</tr>
<tr>
<td>Ambiguous/Insufficient</td>
<td>1.8%</td>
<td>7.1%</td>
<td>5.8%</td>
<td>25.0%</td>
<td>5.4%*</td>
<td>6.4%*</td>
</tr>
</tbody>
</table>
MS Marco

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>A question query issued to Bing.</td>
</tr>
<tr>
<td>Passages</td>
<td>Top 10 passages from Web documents as retrieved by Bing. The passages are presented in ranked order to human editors. The passage that the editor uses to compose the answer is annotated as is_selected: 1.</td>
</tr>
<tr>
<td>Document URLs</td>
<td>URLs of the top ranked documents for the question from Bing. The passages are extracted from these documents.</td>
</tr>
<tr>
<td>Answer(s)</td>
<td>Answers composed by human editors for the question, automatically extracted passages and their corresponding documents.</td>
</tr>
<tr>
<td>Well Formed</td>
<td>Well-formed answer rewritten by human editors, and the original answer.</td>
</tr>
<tr>
<td>Answer(s)</td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td>QA classification. E.g., tallest mountain in south america belongs to the ENTITY segment because the answer is an entity (Aconcagua).</td>
</tr>
</tbody>
</table>

Figure: The final dataset format for MS MARCO

Three tasks:
1. first predict whether a question can be answered, if so, generate the correct answer
2. the generated answer should be well-formed
3. the passage re-ranking

http://www.msmarco.org
1. Machine translation
2. Task oriented chat-bots
3. Constituency parsing
4. Spelling correction
5. Summarization
6. Question answering
7. IR-based QA
   - Datasets
   - Models
DrQA [20]

**Document Retriever**: return 5 Wikipedia articles, using simple $tf - idf$-based retrieval

**Document Reader**: we are given a query $q = q_1, \ldots, q_l$ and $n$ paragraphs $p_1, \ldots, p_m$

**Question encoding**: weighted sum of $RNN(q_1, \ldots, q_l)$

**Paragraph encoding**: $RNN(\tilde{p}_1, \ldots, \tilde{p}_m)$, where $\tilde{p}_1$ is comprised of:

- word embedding $f_{emb}$
- exact match $f_{exact \ match}$
- token features (POS, NER, TF), $f_{token \ features}$
- aligned question embedding $f_{align} = \sum_j a_{ij} q_j$

$$
\frac{\exp(\alpha(E(p_i))) \cdot \exp(\alpha(E(q_i)))}{\sum_{j'}(\alpha(E(p_i))) \cdot \exp(\alpha(E(q_{j'}))}
$$
DrQA [20]

**Prediction:** \( P_{\text{start}} \propto \exp(p_i W_s q) \), \( P_{\text{end}} \propto \exp(p_i W_e q) \)

Choose the best span from token \( i \) to token \( i' \) such that \( i \leq i' \leq i + 15 \) and \( P_{\text{start}}(i) \times P_{\text{end}}(i') \) is maximized.
R-NET [21]
R-NET [21]

1. **Question and passage encoder**: BiRNN to convert the words to their respective word-level embeddings and character-level embeddings.

2. **Gated attention-based recurrent networks**: to incorporate question information into passage representation.

3. **Self-matching attention**: passage context is necessary to infer the answer.

4. **Output**: use pointer networks to predict the start and end position of the answer. To generate the initial hidden vector for the pointer network an attention-pooling over the question representation is used.

5. **Training**: minimize the sum of the negative log probabilities of the ground truth start and end position by the predicted distributions.
BiDAF [22]
BiDAF [22]

1. **Character Embedding Layer** maps each word to a vector space using character-level CNNs

2. **Word Embedding Layer** maps each word to a vector space using a pre-trained word embedding model

3. **Contextual Embedding Layer** utilizes contextual cues from surrounding words to refine the embedding of the words. These first three layers are applied to both the query and context

4. **Attention Flow Layer** couples the query and context vectors and produces a set of query-aware feature vectors for each word in the context

5. **Modeling Layer employs** a Recurrent Neural Network to scan the context. 6. Output Layer provides an answer to the query.
Next generation of QA models

1. S-NET [23]: Extraction-then-synthesis framework
2. QANet [24] benefits from data augmentation techniques, such as paraphrasing and back translation
3. V-NET [25]: end-to-end neural model that enables answer candidates from different passages to verify each other based on their content representations
4. Deep Cascade QA [26]: deep cascade model, which consists of the document retrieval, paragraph retrieval and answer extraction modules
Take away messages

1. seq2seq architectures are exploited in a variety of NLP tasks
2. Attention mechanism helps to find soft alignments
3. The metrics are rarely differentiable, hence reinforcement learning


Reference II


Reference III


Reference V


Reference VI


Reference VII


