Question Answering

Supervised and Unsupervised Approach

As expected, **Amazon also has higher net income at $300 million** for the trailing twelve months versus -$147 million for eBay. Both companies have seen some challenges in direct and indirect costs effecting gross margin and operating margin metrics. Through August 31, 2018, Amazon is reporting a five-year gross margin average of 20% compared to 74% for eBay.

How does a company earn $2.7 billion in annual profit — down from last year's $5.4 billion — with such an intangible service? We won't explain the basics of how credit cards work right here, but the late payers make it easy for the rest of us to ride free. With 107 million American Express cardholders charging almost $1.085 trillion in 2017, that's a lot of revolving credit.

What is net income for Amazon? → $300 million

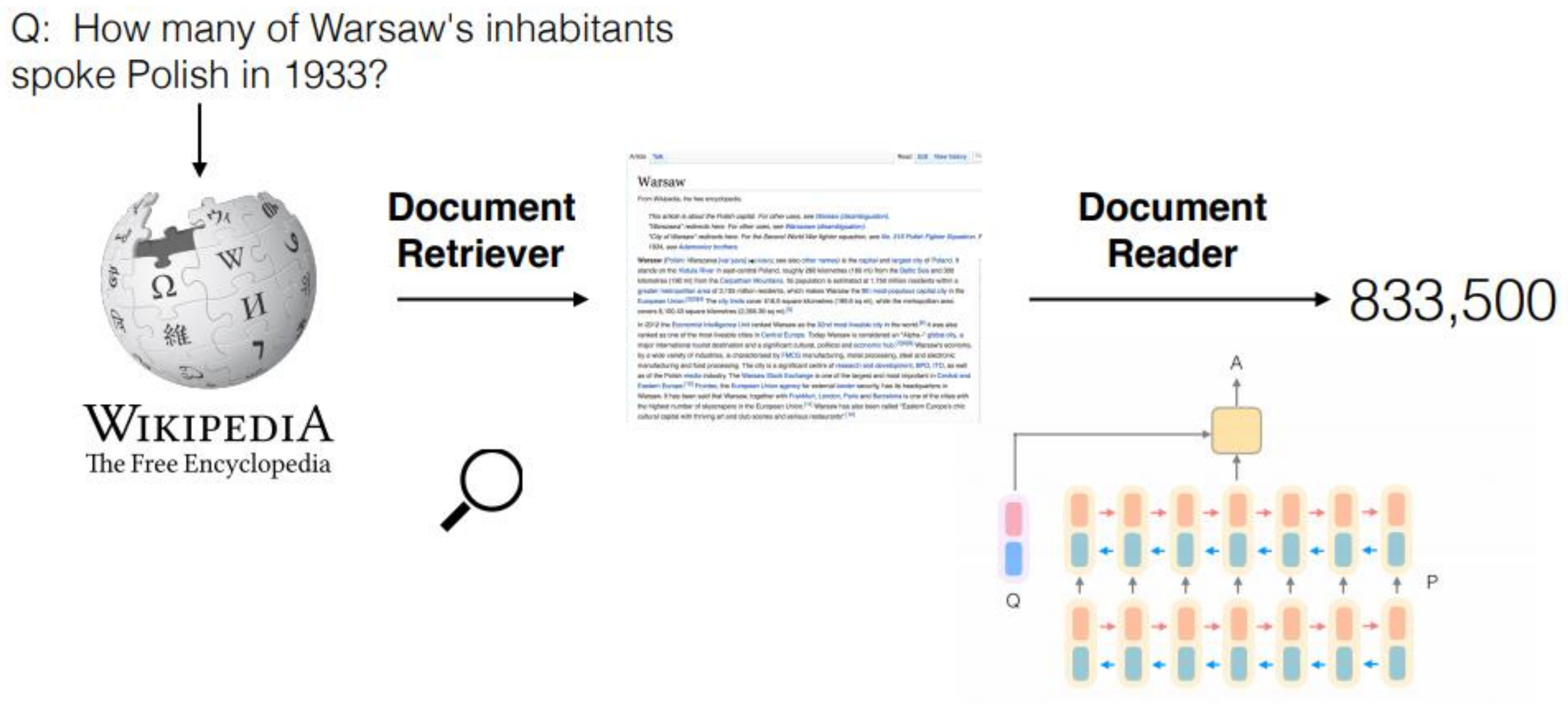
Reading Wikipedia to Answer Open-Domain Questions

- Open-domain question answering using Wikipedia as the unique source.
- The answer to any factoid question is a text span in a Wikipedia article.
- It combines the challenges of document retrieval (finding the relevant articles) with that of machine comprehension of text (identifying the answer spans from those articles)
- It combines a search component based on bigram hashing and TF-IDF matching with a multi-layer recurrent neural network model trained to detect answers in Wikipedia paragraphs.

arXiv:1704.00051 **[cs.CL]** - Danqi Chen

Open-domain QA

SQuAD, TREC, WebQuestions, WikiMovies



Source: https://arxiv.org/pdf/1704.00051.pdf

Document Retriever

- Build a TF-IDF weighted word-doc sparse matrix from the documents.
- Similarly, a query is converted in TF-IDF vector.
- Get the closest docs by dot product between query and documents in tfidf weighted word vector space.
- Uses unigram and bigram features.
- This simple technique outperforms built-in ElasticSearch based Wikipedia Search API.
- Hashing to speed up the process.
- Returns 5 top documents on which document reader algorithm runs.

Document Reader

- Given a question *q* consisting of *l* tokens {*q1, . . . , ql*} and a document or a small set of documents of n paragraphs where a single paragraph p consists of *m* tokens {*p1, . . . , pm*}, an RNN model is applied to each paragraph in turn and then finally aggregate the predicted answers.
- Three main steps:
    1. Paragraph Encoding
    2. Question Encoding
    3. Prediction

Document Reader – Paragraph Encoding

- Represent all tokens *pi* in a paragraph *p* as a sequence of *d*-dimensional feature vectors **pi** and pass them as the input to a recurrent neural network.

$$\{\mathbf{p}_1, \ldots, \mathbf{p}_m\} = \text{RNN}(\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_m\})$$

- Each feature vector **pi** is comprised of:
    - Word embeddings – 300 dimensional GloVe embeddings.
    - Exact match – Indicator variable for each paragraph token if it can be matched with any of the question token.
    - Token properties – For each token, we keep POS tags, NER tags and TF.
    - Aligned question embedding – alignment score is calculated for each paragraph and question token pair and an aligned question embedding is calculated as $\sum_j a_{i,j}\mathbf{E}(q_j)$, where E(qj) is the word embedding of question token qj.
    - The aligned question embedding features add soft alignments between similar but non-identical words (e.g., car and vehicle).

Document Reader – Question Encoding

- For each token, its embedding is passed through a RNN model to get the final encoding of question.

$$\{\mathbf{q}_1, \ldots, \mathbf{q}_l\} \rightarrow \mathbf{q}.$$

- Each token is also weighted by its importance calculated as

$$\mathbf{q} = \sum_j b_j \mathbf{q}_j$$

$$b_j = \frac{\exp(\mathbf{w} \cdot \mathbf{q}_j)}{\sum_{j'} \exp(\mathbf{w} \cdot \mathbf{q}_{j'})}$$

Document Reader – Prediction

- At the paragraph level, the goal is to predict the span of tokens that is most likely the correct answer.
- Paragraph vectors {p1, . . . , pm} and the question vector q are taken as input, and simply two classifiers are trained independently for predicting the two ends of the span.

$$
\begin{aligned}
P_{start}(i) &\propto \exp\left(\mathbf{p}_i \mathbf{W}_s \mathbf{q}\right) \\
P_{end}(i) &\propto \exp\left(\mathbf{p}_i \mathbf{W}_e \mathbf{q}\right)
\end{aligned}
$$

- We rank the answers from different paragraphs based on the highest probabilities.

Document Reader – Results

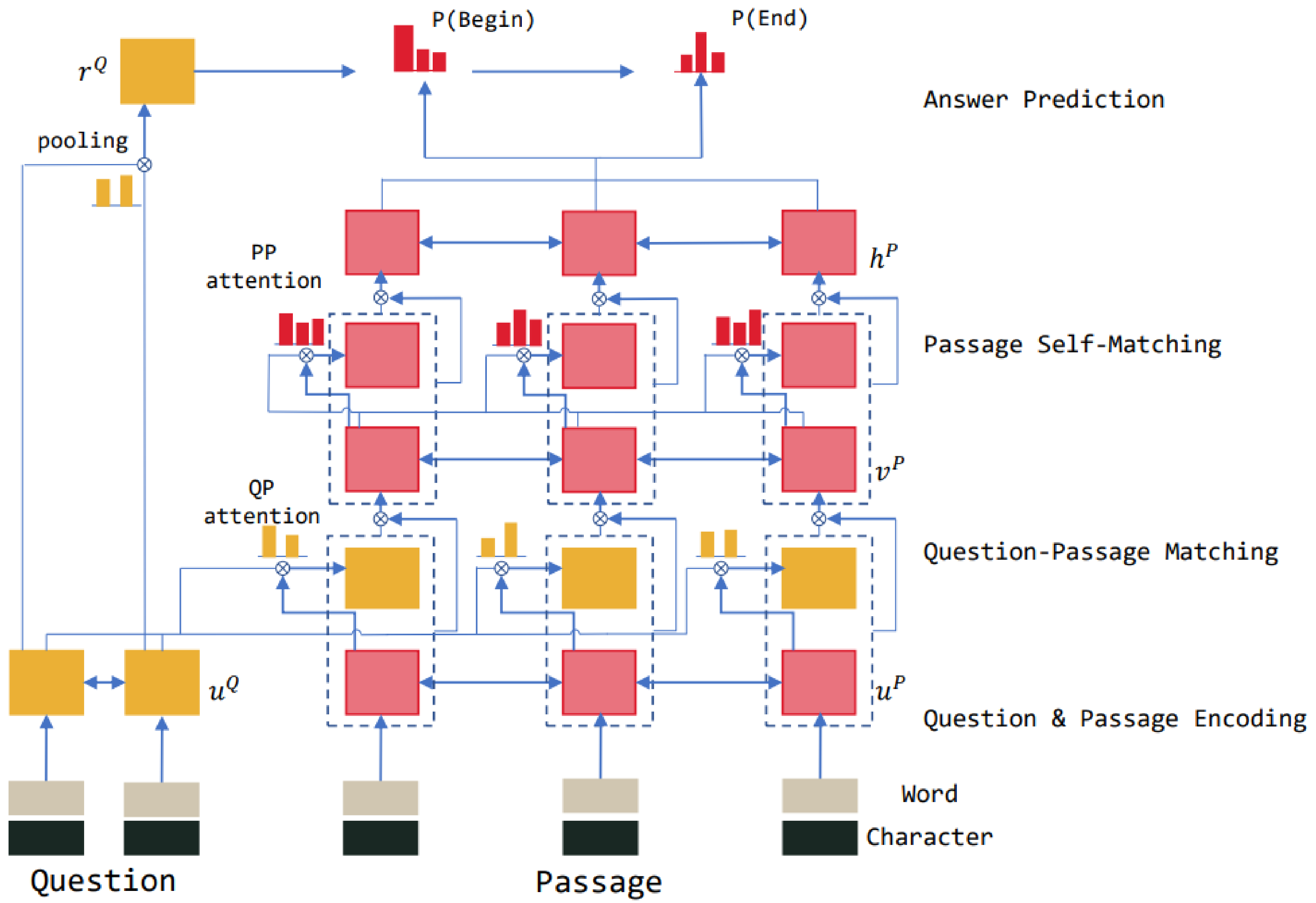
- Three different versions of the model are trained:
  - SQuAD – Document Reader model trained on SQuAD data only.
  - Fine Tune – Model trained on SQuAD and fine tuned on Wiki Data.
  - Multi tasks – Model trained jointly on SQuAD and Wiki Data.

- On SQuAD, the multitask model achieves **69.5 and 78.8** scores for **EM and F1.**
- Initially this model ranked in top three on SQuAD but now there are other models with improvements.

R-NET

- Published by Microsoft Research.
- Currently 3rd best performance on SQuAD dataset.
- Build representation for the passage and the question separately.
- Incorporate the question information into the passage.
- Get final representation of the passage by directly matching it against itself.
- Predict the start and end position of the answer.

R-NET: Machine Reading Comprehension with Self-matching Networks - Microsoft Research Asia



Source: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf

R-NET - Text Representation

- Use **word-level embeddings** and **character-level embeddings** for question and passage separately.
- Use Bidirectional RNN to learn the final representation for question and passage.

$$u_t^Q = \mathrm{BiRNN}_Q(u_{t-1}^Q, [e_t^Q, c_t^Q])$$
$$u_t^P = \mathrm{BiRNN}_P(u_{t-1}^P, [e_t^P, c_t^P])$$

R-NET - Question and Passage

- Incorporates the question information into passage.
- Learns a new passage representation which has aggregated matching information from the whole question using Gated Attention-Based Recurrent Networks.

$$v_t^P = BiRNN(v_{t-1}^P, k_t)$$

- This vector shows that only some parts of passage are relevant for the question. So different value in the vectors would show how important that word in passage is according to the given question.

R-NET - Passage Self-Matching

- Although the new representation can pinpoint important parts in the passage but it has limited knowledge about the context.
- For example:
  - P - Tesla’s mother, Đuka Tesla (née Mandić), whose father was also an Orthodox priest,:10 had a talent for making home craft tools, mechanical appliances, and the **ability** to memorize Serbian epic poems. Đuka had never received a formal education. Nikola credited his eidetic memory and creative **abilities** to his mother’s genetics and influence.
  - Q - “*What were Tesla’s mother’s special abilities?*”
  - First ‘ability’ talks about Tesla’s mother’s abilities while second one about Tesla’s abilities.
- Self-Attention use each passage term to weigh tokens from the passage itself.
- This helps in differentiating the current term from similar-meaning terms in the rest of the passage.

R-NET - Prediction

- Use a pointer network to get the start index and the end index.
- A set of weights is computed for each term in the passage for the starting index and the term with the highest word is selected as the start index.
- Similarly and also using the start index we predict the end index.
- Exact Match and F1-Score - 84.003 and 90.147

Unsupervised Question Answering

A Strong Alignment and Information Retrieval Baseline for Question Answering

- By University of Arizona
- Uses one-to-many alignments to better control for context, as well as negative alignments for discriminative information.
- BM25 is used to get the candidate answers from the questions. LCLR(Learning Latent Representations) for synonyms, antonyms, etc. IDF weighted word cont.

arXiv:1807.01836 [cs.IR]

Preprocessing

- Removes stopwords
- Perform Lemmatization
- Calculate the inverse document frequency (idf) for each term.

Alignment

- Uses candidate answers for a question.
- For each token in question, and for each token in a candidate answer, it calculates the cosine similarity between the tokens.
- For each token in question, we find the k most similar tokens in candidate answer and and k least similar tokens.

Candidate Answer Scoring

- Uses idf of each question term to calculate the score of each question-candidate answer pair based on the weighted sums of the individual term alignment scores.

$$s(Q, A) = \sum_{i=1}^{N} idf(q_i) \cdot align(q_i, A)$$

$$align(q_i, A) = pos(q_i, A) + \lambda \cdot neg(q_i, A)$$

$$pos(q_i, A) = \sum_{k=1}^{K^+} \frac{1}{k} \cdot a^+_{q_i,k}$$

$$neg(q_i, A) = \sum_{k=1}^{K^-} \frac{1}{k} \cdot a^-_{q_i,k}$$

- The intuition behind this formula is that by aggregating several alignments (i.e., through summing), the model can approximate context.

Results

- Simple approach but according to authors, gives better performance. 72.19% on wikiQA, 38% o YA, 53.30 on ScienceQA.

Can all these techniques be combined?

- The first approach for open-domain question answering seems reasonable which consists of two parts – Document Retrieval and Document Reader.
- Document Reader can be improved using latest advancements like Self-Attention, Fully-aware Fusion Networks etc. Also sentence structure knowledge like modifiers can also be feeded in to look for the related results.
- Document Retrieval can be used with unsupervised learning technique.
- Also in the first approach, they tried multiple models where they fine tune model trained on SQuAD. In our real world, same can be done for specific domains. But Data?

Summary

- Most of the time we don't have a labelled dataset and creating one is a huge effort.
- Transfer learning is one approach that could be used where you learn the model on the open domain data and update it using few labelled data points related to our domain.
- Other challenge is ranking the answers based on time. We can't simply sort. This is one problem area which is not given much importance in the open source dataset and many papers.
- Knowledge graphs have been used but that again needs huge effort to construct one. And also they are most effective when answers are one word.
- The main thing is to match the context of question and passages. If this can be done intelligently using the dependency parsing structure, using synonyms would be more effective.

Thank you