Review-based Search Engine

Jung-a Kim

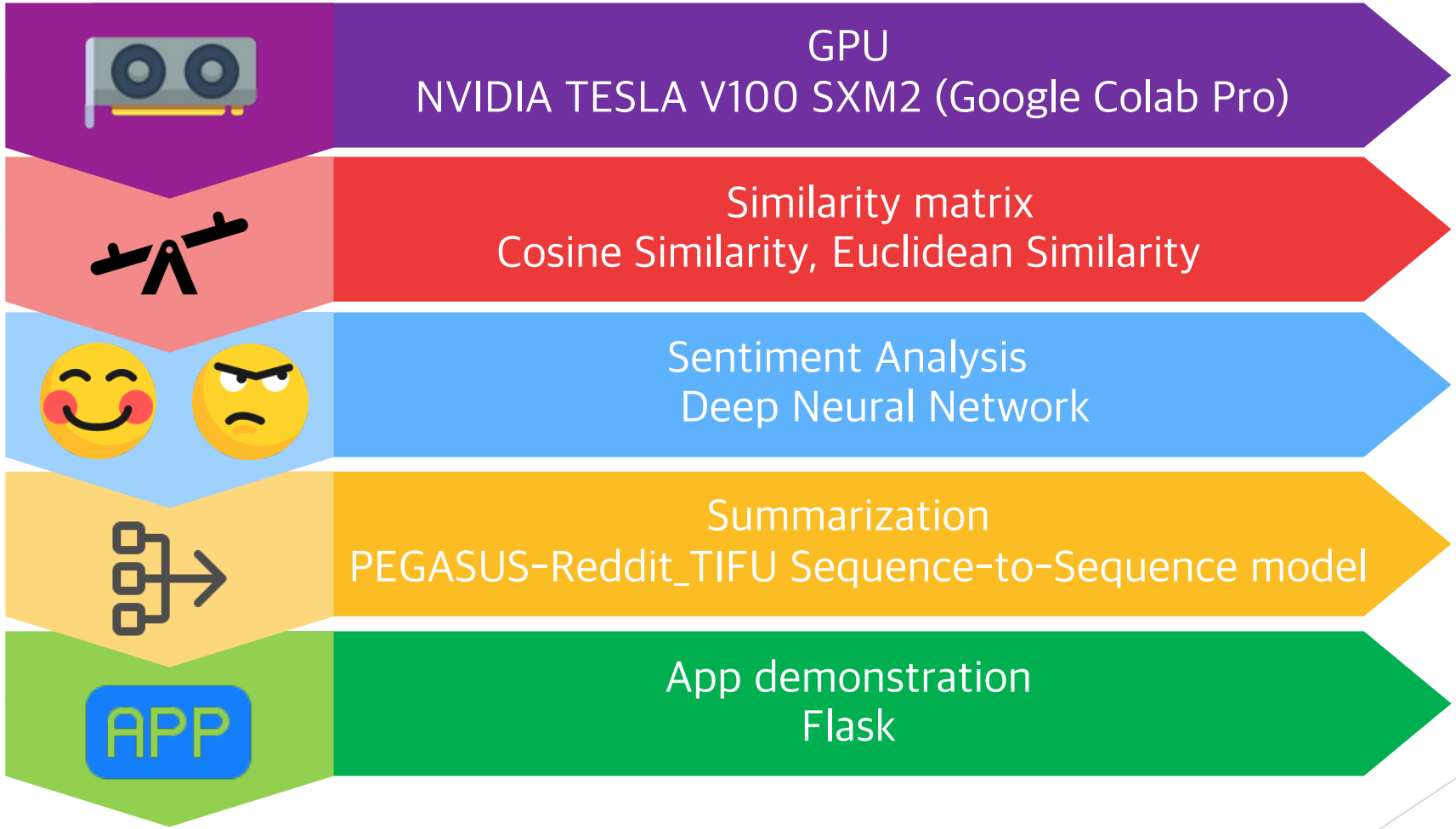
Importance of reviews

- 88% of consumers trust online reviews as much as personal recommendations
- Average consumer reads 10 reviews before purchase

Review-based search engine

- Retrieve the optimal products positively relevant to the query.
- Output the product's most relevant pros and cons that consumers would want to check before purchasing.
- Amazon Reviews for electronics between 1999 and 2015
  - 3,093,869 reviews
  - 166,244 Unique products

Workflow


GPU
NVIDIA TESLA V100 SXM2 (Google Colab Pro)
Similarity matrix
Cosine Similarity, Euclidean Similarity
Sentiment Analysis
Deep Neural Network
Summarization
PEGASUS-Reddit_TIFU Sequence-to-Sequence model
APP
App demonstration
Flask

Positive Similarity score

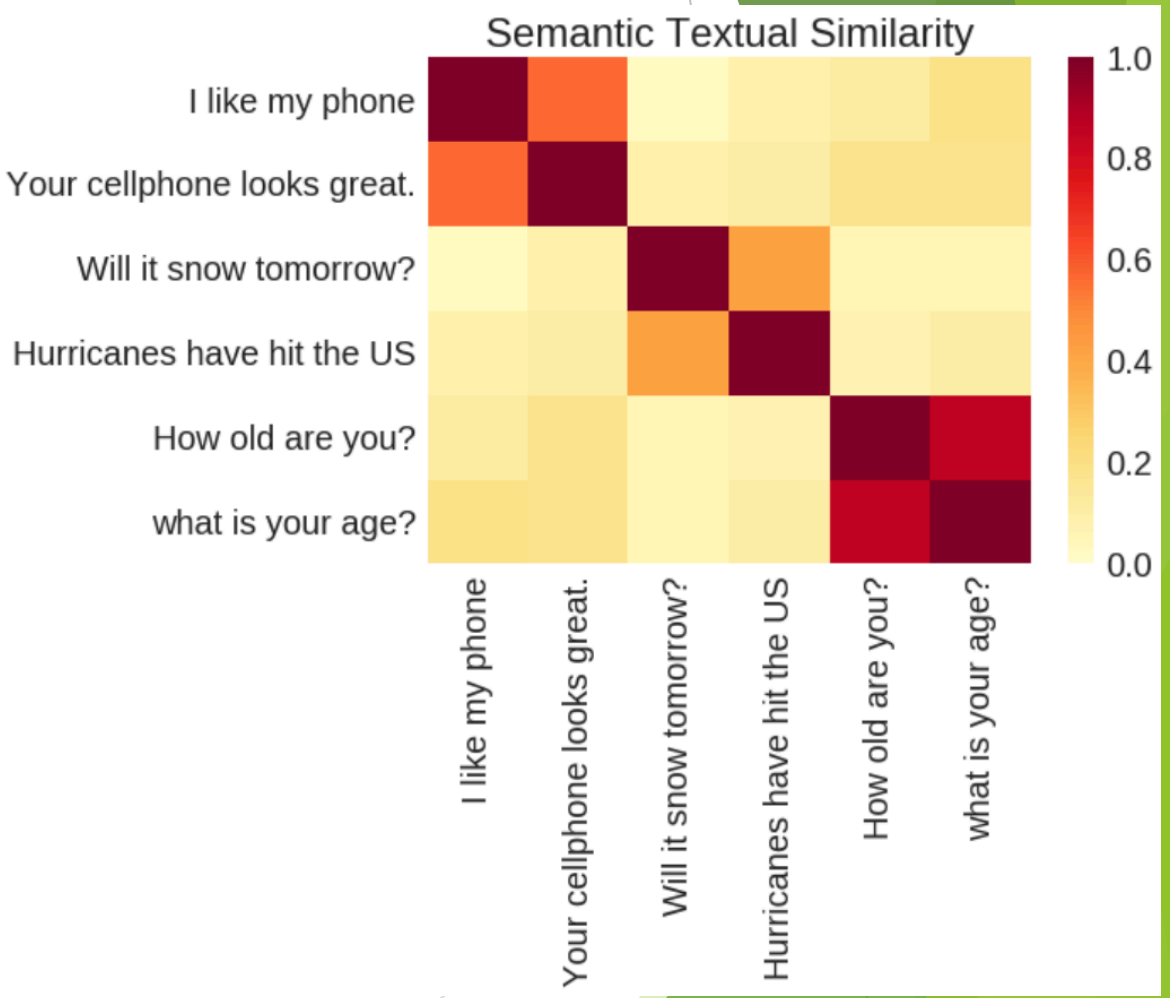
: Measures how relevant a product is to the query based on its reviews

1. Sentiment score for each sentence of a review
   - Star rating 5 as 'Positive', trained with products with fewer than 10 reviews
   - 82% accuracy on test set
     - Each review into 512-dimensional embeddings
     - ReLU -> Dropout(0.5) -> ReLU -> Dropout(0.5) -> Sigmoid
     - 40 epochs, 6 batches
2. Similarity score of a review using Cosine similarity/Euclidean similarity

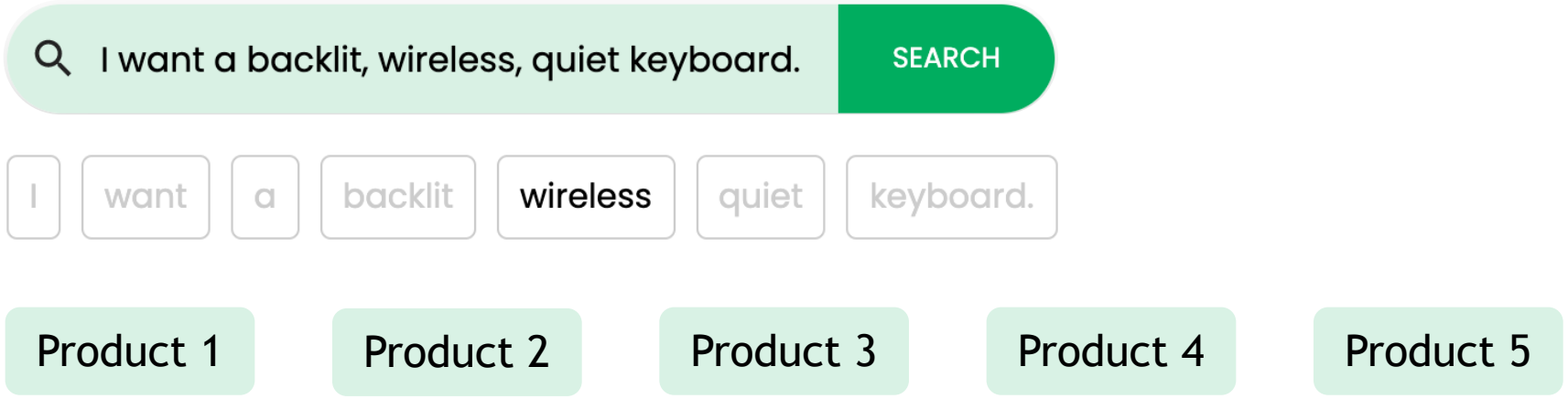
   Tokenize a query in sentences vs. words and return the minimum score
3. (Optional) Weights on keywords selected by user

Universal Sentence Encoder(USE)

- Transformer-based sentence encoder (published in 2018 by Google Research)
  - Element-wise sum of context-aware word representations(e.g. BERT) at each word position divided by $\sqrt{sentence\ length}$
  - BERT cannot detect similar sentences that do not follow each other
    - e.g. “How much does this cost?”, “Is this expensive?”
- Trained to identify the similarity between the pairs of sentences
  - Semantic Textual Similarity scored by Pearson correlation with human judgments



Search engine - Input



- The input would be a few sentences with keywords that the consumer is looking for in a product.
- The engine matches the products with key qualities based on the reviews.

Search engine - Output

< 5. RK728 Wireless Keyboard >

Pros	Stars	Date
If you are looking for the the keyboard to complete your laptop/hdtv combination THIS IS PERFECT. I just got this last week opened the box plugged it in and it started surfing the net rite away never even rebooted the computer. Netflicks has never been better!!!!	5	2008-08-02
I used this with my laptop because I never liked the feel of the crammed laptop layout. It allows me to comfortably type in crammed spaces while in flight. Plus I like to type on my lap and is so much better not having a hot heavy notebook on my lap.	5	2008-07-18
this product is similar to the FK760 I purchased recently and it has an integrated touchpad. I like it. Very good price and the company delivered the product quickly.	5	2008-07-17
Loved It	5	2014-07-22
good staff, we like it very much easy to use and easy to operations, no problems at alll, if you like it go get it.	5	2013-02-13

Cons	Stars	Date
this keyboard has some very nice features, but it is very hard to push the keys down, and the keyboard is cramped. also there is some delay on the touch pad, not much, but fine movements are hard. this is a great keyboard for use with a media center, but not for your primary keyboard.	3	2009-03-18
...i bought a keyboard that won't respond to your touch and signal drop out more than 75% time of useage, so i thought it's defected....	1	2008-08-19
...don't touch type on a wireless keyboard, it's a pain and you'll get a 50% drop in speed and accuracy if you touch....	3	2009-06-10
...wireless keyboard/mouse combo is a good compliment, but sometimes you would prefer changing back to normal keyboard or mouse. ;-(;)...	3	2008-11-12
...i bought a wireless keyboard for my t.v. monitor and am using it with an old laptop connected to it for surfing and watching videos....	4	2008-09-10

PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization

- Transformer Encoder-Decoder model in 2020 by Google
- State-of-the-art performance on 12 downstream datasets
- TIFU sub-reddit stories ≈ Amazon reviews

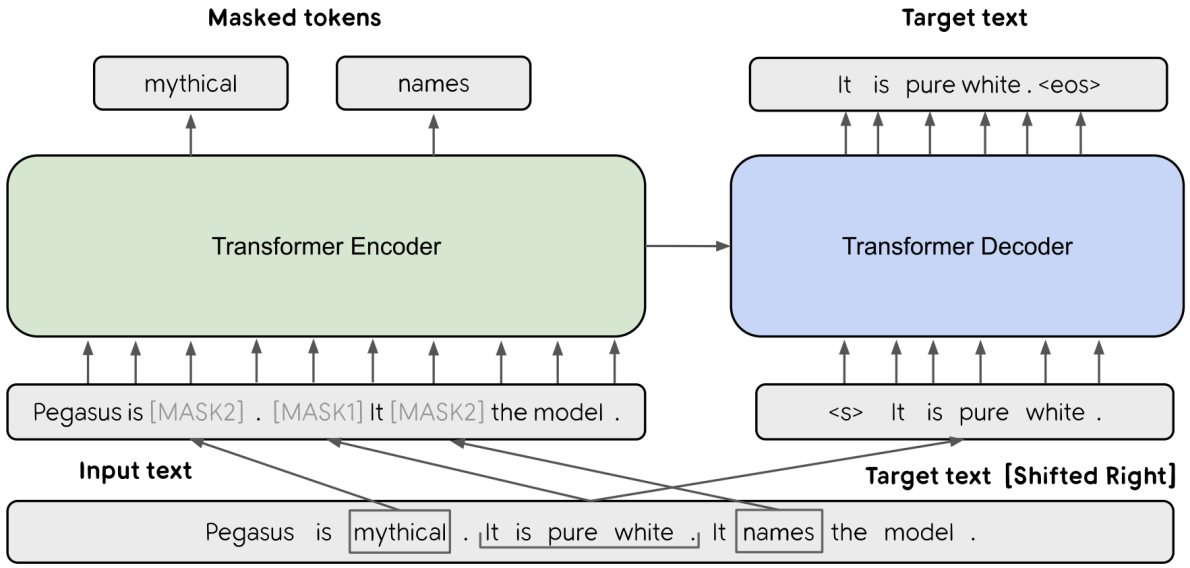


Figure 1: The base architecture of PEGASUS is a standard Transformer encoder-decoder.

Summary of a review using PEGASUS

I've never seen a review of this keyboard and beside I needed one for my HTPC at the time of purchased. What a mistake. Cursor won't respond to your touch and run on its own (WTF!!)and signal drop out more than 75% time of useage, so I thought it's must be defected. RMA the first one (still waiting for my refund..) and order another one. Guess what, hahaa...what a POS! Don't buy it. You're warned....

...i bought a keyboard that won't respond to your touch and signal drop out more than 75% time of useage, so i thought it's defected....

Conclusion

- The engine returns mostly a relevant product.
  - ✓ Maximum similarity with any part of the query.
  - ✓ Be conservative by computing two types of similarity scores
- Time complexity of positive similarity score $\approx O(n^3)$
  - ✓ Vectorized to reduce time
- Mostly cannot detect a category from a product title using only embeddings
  - ✓ Weighting categorical keywords increases accuracy
- PEGASUS summarizes well, but it can take ~2 mins per product.

Future work

- Cannot detect a category from a product title.
  - 👉 Transfer learning sentence embeddings with reviews dataset
- PEGASUS summarizes well, but it takes a long time.
  - 👉 Use a streamlined encoder-decoder model e.g. DistilBART

APPENDIX

Pseudocode for Positive Similarity score of a product

```
For each review in a product:
    For each sentence in a review:
        For each keyword in a query:
            Compute maximum positive * similarity score of (weighted) keywords
        For each sentence in a query:
            Compute maximum positive * similarity score of (weighted) sentences

    Average maximum positive * similarity scores of keywords
    Average maximum positive * similarity scores of sentences

    Save the smaller average score
Return median of the scores
```

Vectorized in the actual implementation

Pseudocode for Similarity score of a review

~~For each review in a product:~~

```
    For each sentence in a review:
        For each keyword in a query:
            Compute maximum ~~positive *~~ similarity score of (weighted) keywords
        For each sentence in a query:
            Compute maximum ~~positive *~~ similarity score of (weighted) sentences

    Average maximum ~~positive *~~ similarity scores of keywords
    Average maximum ~~positive *~~ similarity scores of sentences

    Return the smaller average score
```

~~Return median of the scores~~

Vectorized in the actual implementation

Vectorization of Positive Similarity score (words-to-sentence)

$$\begin{pmatrix} & query's\ Keyword1 & query'sKeyword2 & query'sKeyword3 & \\ Review's\ Sentence1 & \text{Similarity} & \text{Similarity} & \text{Similarity} & \cdots \\ Review's\ Sentence2 & \text{Similarity} & \text{Similarity} & \text{Similarity} & \cdots \\ Review's\ Sentence3 & \text{Similarity} & \text{Similarity} & \text{Similarity} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

$$* \begin{pmatrix} \text{Sentiment score of Review's Sentence1} \\ \text{Sentiment score of Review's Sentence2} \\ \text{Sentiment score of Review's Sentence3} \\ \vdots \end{pmatrix}$$

$$* (\text{Weight of Keyword1} \quad \text{Weight of Keyword2} \quad \text{Weight of Keyword3} \quad \cdots)$$

$$=$$

$$\begin{pmatrix} Review's\ Sentence1 & \text{similarity with Keyword1} * \text{Sentiment} * \text{weight} & \text{similarity with Keyword2} * \text{Sentiment} * \text{weight} & \cdots \\ Review's\ Sentence2 & \text{similarity with Keyword1} * \text{Sentiment} * \text{weight} & \text{similarity with Keyword2} * \text{Sentiment} * \text{weight} & \cdots \\ Review's\ Sentence3 & \text{similarity with Keyword1} * \text{Sentiment} * \text{weight} & \text{similarity with Keyword2} * \text{Sentiment} * \text{weight} & \cdots \\ \vdots & \vdots & \vdots & \end{pmatrix}$$

Take the max (axis = 1) → shape = (number of review's sentences, 1)

→ Take the mean (axis = 0) which is a scalar ⇐ This measures the positivity and (weighted) similarity of a review with respect to the query

PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization

- Published in 2020 by Google
- Abstractive summarization
  - Sequence-to-sequence text generation problem
  - cf) Extractive summarization is a binary classification problem
- Gap-Sentence-Generation(GSG)
  - Follows BERT's masking algorithm
  - Mask important sentences and concatenate them into a pseudo-summary
- Transformer Encoder-Decoder model
- Corpus used for pre-training
  - 120K posts of informal stories from TIFU sub-reddit from 2013 to 2018

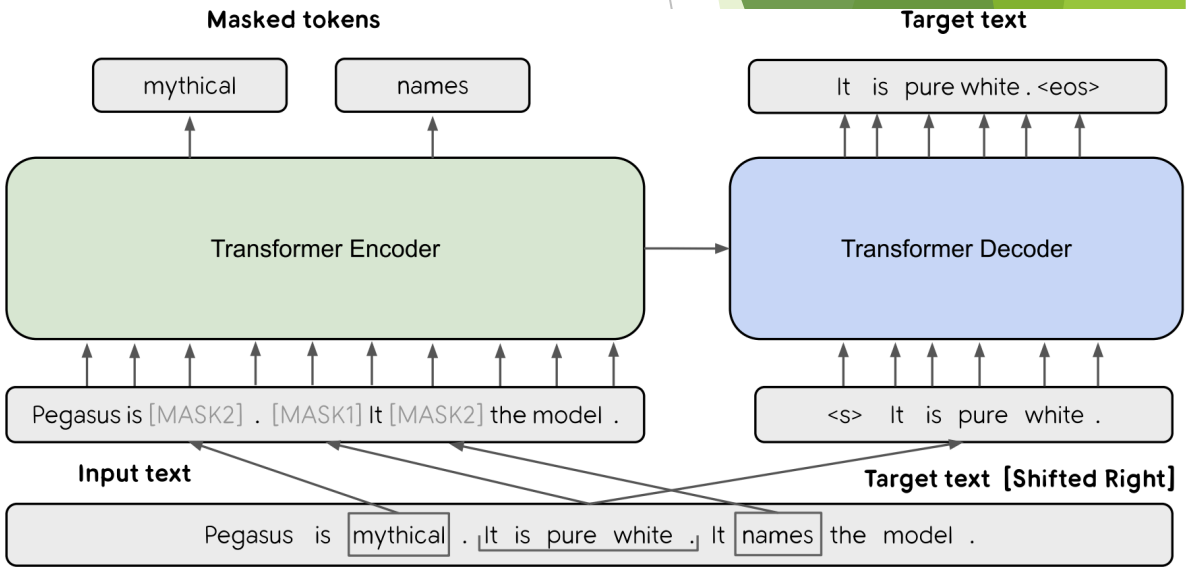


Figure 1: The base architecture of PEGASUS is a standard Transformer encoder-decoder.

What are you looking for?

SEARCH

EMPHASIZE KEYWORDS

References for images

- Photo by Morning Brew on Unsplash
- Photo by KOBU Agency on Unsplash
- Icons: https://www.iconfinder.com/
- Process Template: https://slidesgo.com/theme/process-diagrams
- Templates for Flask: Colorlib (https://colorlib.com)
- Figure1: Universal Sentence Encoder, Daniel Cer and Yinfei Yang and Sheng-yi Kong and Nan Hua and Nicole Limtiaco and Rhomni St. John and Noah Constant and Mario Guajardo-Cespedes and Steve Yuan and Chris Tar and Yun-Hsuan Sung and Brian Strope and Ray Kurzweil, 2018, arXiv:1803.11175
- When Not to Choose the Best NLP Model, DEEP LEARNING, FLOYDHUB, 2019, https://blog.floydhub.com/when-the-best-nlp-model-is-not-the-best-choice/
- Figure2: PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization, Jingqing Zhang and Yao Zhao and Mohammad Saleh and Peter J. Liu, 2019, arXiv:1912.08777