Text Generation From Keywords via Cumulative Attention

Yuan Shen(yshen47)
Chen Chen(chen228)
Tianyu Chen(tchen72)

Instructor: Alexander Schwing

1. Introduction

For people who keep close pace with the fashion trend, magazines and news feeds are essential sources of information and daily entertainment. However, to produce the articles, editors spend considerable efforts recognize trending fashion trending topics, collect related fashion products, coming up with descriptive language and layout the page. We are thinking whether we can auto-generate fashion magazine contents. Our previous research work has introduced a model that can generate related fashion keywords from a fashion product image. For this course project, we focus on auto-generating fashion product descriptions via keywords

People might be wondering why auto-generating fashion magazine contents. There are two main reasons to support the necessity. First, fashion magazine editing process is highly repetitive. In the fashion world, new trending style rises constantly and fashion media strive to keep the information flow up to date. To achieve that, editors spend great effort looking for emerging topics, discuss and decide upon the themes to present, locate products that serve well as the content and produce the final articles. However, this process includes repetitive and labor-heavy work and many parts may be accelerated through data- and AI- driven assistance. Second, we found most of fashion magazine contents are theme-based magazine posts which has the potential to fully auto-generate. In specific, for product descriptions, we found they are usually short and highly-structured in popular fashion e-commerce web-sites, like Farfetch, net-a-porter (Figure 2). Theme-based magazines refer to the posts that mainly consist of images about fashion product, outfit or capsule closely related a fashion-theme. We manually count magazine pages from Vogue Magazine(Nov. 2018). It turns out 56.2 percent content is theme-based photo series.

Keywords	['dress', 'corset', 'hem']	['cotton', 't-shirt', 'diesel', 'straight', 'hem']
generated	this UNK dress features a notch lapel, a corset design, and a UNK hem EOS	handmade from 18-karat gold, this bracelet has a block pendant that's set with 0 EOS
original	this UNK strapless column dress features a sweetheart neckline, a corset bodice, and pleating at the back hem EOS	this handmade 18-karat gold bracelet features a delicate pendant that UNK out UNK in block lettering EOS

Figure 1. Fashion product descriptions that generated from our model. UNK refers to word with less than 10 occurrence in our training data-set.

Previous research work related to text generation has largely focused on sequence2sequence models or generating text from images (image captioning), while few touches on generating texts from keywords with no order. At first, we try to adjust attention-based machine translation models[8] to fit our goal. However, machine translation models require the input to encoder network to have some orders (namely, some grammar structure) so that the decoder network is able to learn where to look at the encoding sequence. However, in our case, keywords have no order between each others, there is no point to say that style related keywords should always be at the beginning of the encoding sentence. Therefore, our attention shifts to literature that can encode inputs (in our case, keywords) with no sequence order requirement. Later, We considered to use image encoder network (like Res-net) to substitute

the encoder part for our encoder-decoder network, similar to image captioning tasks. However, later, we found this encoding technique problematic as well, because image embedding cannot capture all the information in a product description. For example, information related to the brand or year of a product is hard to learn from a product image even for human.

Our final solution is to modify a cumulative-attention network [3] to fit our purpose. The network is originally designed for Question-Answer System. Originally, using a key-value pair memory, the model tries to output a human-readable sentence which answers an input question that encodes in the encoder network. We modified the structure to input our fashion keyword pairs as memory, where we use fashion categories (style, color, material, shape, pattern, brand, type, trim) as keys, and fashion keywords as values. Moreover, we removed the encoder network in the system because we don't require a sentence about a question in our encoder network. The dictionary format of the input memory structure fits our tasks perfectly, which doesn't require any order in the input. Figure 1 offers a good visualization of our input and output. We also implemented with different attention mechanisms, [1], and consult to different variations, like pointer network [5] and end-to-end memory network [7]

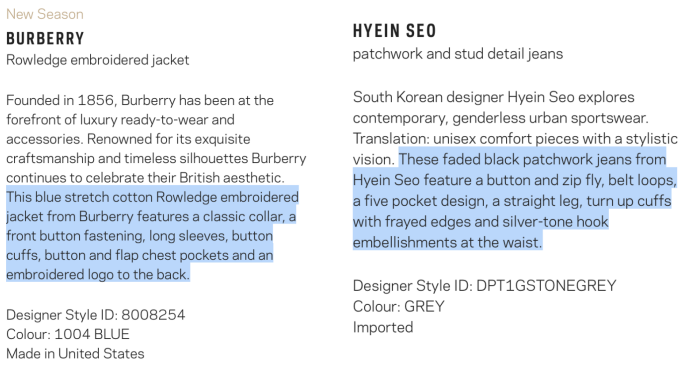

Figure 2. Product description has some similar sentence structures and is always made up by fashion keywords. The fashion descriptions in this figure is from fashion e-commerce website, Farfetch.com

2. Dataset

We crawled our product sentences from five different fashion e-commerce websites, including Farfetch, net-a-porter, zara, www.modaoperandi.com and nordstrom (Figure 3). We did standard text pre-processing procedures, including stop-word removal, lowercase, etc. To make our life easier, we only selected sentences including "this and these" and max length less than 20. As a result, we have 14677 product sentences for training, and 773 product sentences for testing.

Fashion E-commerce Website	Product Count
Farfetch	277,809 (89,062 in stock)
Nordstrom	93,984 (72,671 in stock)
Net-A-Porter	80,146 (28,005 in stock)
Moda Operandi	23,085 (10,303 in stock)
Zara	8,779 (6,649 in stock)

Figure 3. Details about Our fashion product dataset

3. Method

Our model is derived from the traditional seq2seq encode decoder system. In fact previously we have tried a Machine Translation System [8] with encoder decoder and used attention inside the decode. However, it did not work too well. Below is a step by step of our current system.[3]. Since our model does not contain the question part and there exist several dimension mismatch, we did some modifications to the system.

1. Prepossessing: Firstly we created a model called Lang, from [6]. The model handles the transforms of word to integer or integer to word. The dataset contains a large number of sentence descriptions. Then we preprocess our dataset, and extract the keywords and categories out of the sentences and the rest become normal words. Thus we split the sentences into three parts: keywords, category and normal words. Before going deeper, let me introduce what keywords, category and normal words are.

   (a) **category**: We have 8 categories including "styles, types, shapes, materials, trims, colors, patterns and brands". These are predefined categories and we have a set of keywords that belongs to each category.

   (b) **keywords**: Keywords are the list of words that belongs to a category. Our final lexicon contains 3,418 209 unique terms, with 445 styles, 421 types, 207 shapes, 150 210 materials, 97 trims, 107 colors, 42 patterns and 1553 brands.

   (c) **normal words**: The rest of the words inside the sentence description without keywords and category.

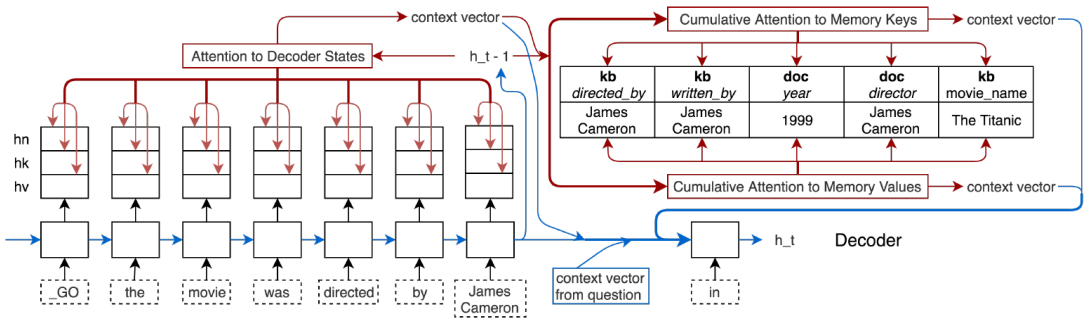

Figure 4. The Cumulative Attention Mechanism

2. **Model Initialization**: we initialize our model with the following parameter:

   (a) **normal-vocab-size**: total number of normal vocabulary

   (b) **keyword-vocab-size**: total number of keyword vocabulary

   (c) **max-len=30**: maximum number of memory pairs

   (d) **max-mem-size=10**: maximum length of input sentences

   (e) **num-layers=1**: number of LSTM/GRU layers

   (f) **embedding-dim=50**: word embedding size

   (g) **batch-size=5**

   (h) **hidden size = embedding-dim**

   Our initial hidden state is the average of average of all the memory key and all the memory value:

$$h_0 = avg(avg(m^{\langle K \rangle}), avg(m^{\langle V \rangle})) \quad (1)$$

   memory construction: For every input sentence, we build a memory, i.e. a knowledge base in the following form: where key is the fashion category and value is the keyword. $((m_0^{\langle K \rangle}, m_0^{\langle V \rangle}), (m_1^{\langle K \rangle}, m_1^{\langle V \rangle})...(m_2^{\langle K \rangle}, m_2^{\langle V \rangle}))$.

   It contains all the keywords and the category. For example: ((”color”:”red”, ”style”,”summer”)...)

3. Forward Pass:

   (a) **Build current context** c_t: Firstly we need to build the current context using cumulative attention. We build the context for memory key $c_t^{\langle HK \rangle}$, memory value $c_t^{\langle HV \rangle}$ and normal words $c_t^{\langle HN \rangle}$ using attention mechanism. Attention mechanism comes from the multi-hop attention (Sukhbaatar et al., 2015). Inputs are the hidden layer and the history states.

$$\begin{aligned} c_t^{\langle HN \rangle} &= attn(h_{t-1}, hist(h_t^{\langle N \rangle})) \\ c_t^{\langle HK \rangle} &= attn(h_{t-1}, hist(h_t^{\langle K \rangle}) \\ c_t^{\langle HV \rangle} &= attn(h_{t-1}, hist(h_t^{\langle V \rangle})) \end{aligned}$$

   History states for keywords, memory and normal words are stored separately.

$$\begin{aligned} hist(h_t^{\langle N \rangle}) &= (h_0^{\langle N \rangle}, h_1^{\langle N \rangle}, ...h_{t-1}^{\langle N \rangle}) \\ hist(h_t^{\langle K \rangle}) &= (h_0^{\langle K \rangle}, h_1^{\langle K \rangle}, ...h_{t-1}^{\langle K \rangle}) \\ hist(h_t^{\langle V \rangle}) &= (h_0^{\langle V \rangle}, h_1^{\langle V \rangle}, ...h_{t-1}^{\langle V \rangle}) \end{aligned}$$

   Then we compute the overall context vector by concatenating the top three.

$$c_t^{\langle H \rangle} = [c_t^{\langle HN \rangle}, c_t^{\langle HK \rangle}, c_t^{\langle HV \rangle}]$$

   After we have this context, we could use it to project to memory key and memory value.

$$\begin{aligned} c_t^{\langle MK \rangle} &= attn([h_{t-1}^{\langle K \rangle}, c_t^{\langle H \rangle}], m^{\langle K \rangle}) \\ c_t^{\langle MV \rangle} &= attn([h_{t-1}^{\langle V \rangle}, c_t^{\langle H \rangle}], m^{\langle V \rangle}) \end{aligned}$$

   The current context vector is the concatenation of all the contexts.

$$c_t = [c_t^{\langle H \rangle}, c_t^{\langle MK \rangle}, c_t^{\langle MV \rangle}] \quad (2)$$

   (b) **Build current hidden state** H_t:

   i. Reshape the context vector c_t into the same size as hidden layer.

ii. Perform LSTM/GRU on previous word embedding, current context and last hidden state.

$$h_t = LSTM_{dec}(x_t, c_t, h_{t-1})$$

(c) Transform this hidden layer into different weighted hidden layers that represents the hidden layer for keywords, category and normal words.

$$\begin{aligned} h_t^{\langle N \rangle} &= W_n h_t \\ h_t^{\langle K \rangle} &= W_k h_t \\ h_t^{\langle V \rangle} &= W_v h_t \end{aligned}$$

(d) Project the three hidden layers separately to keywords, category and normal words to get three distributions $p_t^{\langle N \rangle}, p_t^{\langle MK \rangle}, p_t^{\langle MV \rangle}$. Then we compute $p_t^{\langle M \rangle} = (p_t^{\langle MK \rangle} + p_t^{\langle MV \rangle})/2$

(e) Determine whether next word should be in memory or normal words

$$\begin{aligned} &P(x_t|q, M, x_0, x_1, ...x_{t-1}) = \\ &\quad g \times P(X_t = w_k|q, M, x_0, x_1...x_{t-1}) + \\ &\quad (1-g) \times P(X_t = m_k|q, M, x_0, x_1...x_{t-1}) \end{aligned} \tag{1}$$

where

$$\begin{aligned} P(X_t = w_k|q, M, x_0, x_1...x_{t-1}) &= p_t^{\langle N \rangle} \\ P(X_t = m_k|q, M, x_0, x_1...x_{t-1}) &= p_t^{\langle M \rangle} \end{aligned}$$

(f) Return the word with the highest probability in the chosen distribution.

4. Loss:

We used cross entropy loss H for the gating and NLL loss for the word. Then we optimize L with gradient descent based optimizers.

$$\mathcal{L} = -\sum_{t=1|}^{N} log(P(x_t|q, M, x_0...x_{t-1})) + \mathcal{H}(g, \hat{g})$$

4. Experimental Results

4.1. Metrics

To better evaluate the quality of the generated sentences, we formulated some metrics for evaluation and comparison. These metrics for the generated sentences include keywords coverage, keyword repetition rate and enrichment rate:

$$\textbf{keywords coverage} = \frac{|\text{keywords}|}{|\text{input keywords}|} \tag{3}$$

$$\textbf{keywords repetition rate} = \frac{|\text{repeated keywords}|}{\text{sentence length}} \tag{4}$$

$$\textbf{enrichment rate} = \frac{|\text{words from normal vocab}|}{\text{sentence length}} \tag{5}$$

The reason that we only measured repetition rate for keywords was that by observing the results, we found that the repetition issue seldom occurs with normal words. Table 1 shows the result of these metrics for both models using LSTM and GRU units running with batch size of 5. Table 2 (on Page 6) shows metrics result for different batch sizes with model using LSTM unit.

4.2. Details

We ran our model with two types of RNN units, Long Short-Term Memory [4] and Gated Recurrent Unit [2] for performance comparison. We also explored the effect of batch size on the final result and found that setting batch size to 5 are a good option, so it was used for the final results. The training was ran for 64 epochs on 14677 product sentences with a batch size of 5, which means in each iteration, 5 sentences were randomly sampled for one step of optimization. We used 5% of randomly selected training data for validation after each epoch. We were able to observe that the validation loss dropped significantly at the beginning and started to converge after 30 epochs. Both LSTM and GRU behaved similarly in the change of loss.

After the model was trained, we ran the model with never seen test data, which was 773 product sentences. The input was the list of keywords and the output was a list of generated words which we reassembled into sentences. The normal vocabulary (non-keywords) used to enrich the contents was from the training model. Figure 5 and Figure 6 show some examples of generated sentences from LSTM and GRU unit respectively. Both good cases and bad cases were presented to show the effectiveness of the trained models.

	LSTM	GRU
Keywords Coverage	0.73	0.72
Keywords Repetition Rate	0.13	0.16
Enrichment Rate	0.64	0.62

Table 1. Comparison of Metrics for generated sentences from models with LSTM and GRU units

Good Cases:			
Keywords	['japanese', 'designer', 'kenzo', 'world', 'scarf']	['cotton', 't-shirt', 'diesel', 'straight', 'hem']	['silk', 'everyday', 'top', 'short', 'straight', 'hem']
generated	renowned for its offbeat accessories, japanese fashion kenzo presents this kenzo scarf from the UNK to the world EOS	this grey cotton UNK t-shirt from diesel features a straight fit, a crew neck, a sleeveless design EOS	this UNK silk short hem short features a crew neck, top sleeves, and a straight hem EOS
original	renowned for its offbeat accessories, japanese designer kenzo presents this grey world scarf to its aw17 collection EOS	this grey cotton washed t-shirt from diesel features a crew neck, a UNK effect and a straight hem EOS	fashioned from colourful silk blend, this everyday top features a boat neck, short sleeves and a straight hem EOS
Bad Cases:			
Keywords	['wear', 'mini', 'dress', 'metallic', 'clutch', 'striking', 'cocktail']	['crafted', 'italy', 'soft', 'cotton', 'striking', 'shirt', 'jacket', 'ideal', 'spring', 'wear']	['crafted', 'blue', 'sweater', 'long', 'ribbed', 'hem']
generated	wear this striking mini dress with a striking dress and striking heels for a striking cocktail look EOS	soft and wear this striking pair is ideal for spring wear and wear with a spring wear EOS	crafted from a blue wool-cashmere blend, this sweater features a ribbed hem and a ribbed hem and cuffs EOS
original	wear this runway style with a UNK mini dress and metallic clutch for a striking cocktail look EOS	crafted in italy from a soft cotton blend, this striking shirt jacket is ideal for spring wear EOS	crafted from blue cashmere, this sweater features a crew neck, long sleeves and a ribbed hem and cuffs EOS

Figure 5. Generated sentences from LSTM model

Good Cases:			
Keywords	['italy', 'heel', 'heel', 'ankle', 'strap']	['cropped', 'printed', 'leather', 'jacket', 'cool', 'chic']	['crafted', 'silk', 'beautiful', 'blazer', 'givenchy', 'essential']
generated	this pair has been made in italy from supple ankle for a sturdy strap and has secure heel EOS	team this chic style with a printed jacket and leather cropped jacket for a cool chic look EOS	givenchy by the UNK aesthetic, this blazer is a beautiful silk crafted essential from givenchy EOS
original	this pair has been expertly constructed in italy with a pin-thin heel and supportive ankle strap EOS	team this cropped style with a printed blouse and leather jacket to put a cool slant on UNK chic EOS	crafted from a blend of silk and wool, this beautiful blazer from givenchy is a wardrobe essential EOS
Bad Cases:			
Keywords	['multicolored', 'wool', 'striped', 'oversized', 'coat', 'msgm', 'long', 'button', 'fastening']	['versatile', 'comfortable', 'block', 'heel', 'flattering', 'point', 'calf']	['handcrafted', '18-karat', 'delicate', 'necklace', 'enameled', 'pendant', 'sparkling', 'sparkling', 'diamond']
generated	this long long long long long long long coat from msgm features a long long sleeves and long sleeves EOS	this calf pair has a flattering point point and calf calf calf heel that's UNK in calf EOS	sparkling diamond UNK this necklace is strung with a sparkling diamond diamond and sparkling diamond diamonds EOS
original	this multicolored wool blend striped oversized coat from msgm featuring long sleeves and a button fastening EOS	this versatile pair has a comfortable block heel and hits at a flattering point on your calf EOS	handcrafted from 18-karat gold, this delicate necklace holds an enameled UNK pendant set with a sparkling diamond EOS

Figure 6. Generated sentences from GRU model

Batch Size	2	5	10	128
Keywords Coverage	0.74	0.73	0.70	0.07
Keywords Repetition Rate	0.26	0.13	0.13	0.02
Enrichment Rate	0.50	0.64	0.65	0.96

Table 2. Comparison of Metrics from different batch sizes using LSTM model

5. Discussion and Conclusion

5.1. Discussion

As shown in results, our models were able to learn from the training data and generate sentences that look like valid product descriptions. We also found that in some cases, the generated sentence and the original sentence were very similar. For example, the first good case in Figure 5 had a part 'renowned for its offbeat accessories', which did not appear in the keywords at all. But these words were perfectly put together in the generated sentence. After looking into the our dataset, we found that there were many product descriptions related to the designer 'Kenzo', and these words were usually used for the designer's products. Some of these product descriptions were split into the training set, resulting in the model memorizing the pattern. Although having similar data might not be ideal for some machine learning tasks, we think it makes sense in out task. Our goal was to train a model to generate descriptions after looking at a lot of existing fashion magazines and websites. The model should be able to capture the way certain brands or designers linked to their products. This is exactly what happened in this case. There were also many cases where the generated sentence look valid, but have some different information from the original products. This is usually caused by insufficient information in keywords, or randomly selecting reasonable words from normal vocabulary. The second good case in Figure 5 serves as an example. The generated sentence added 'sleeveless design' while such information was never presented in the keywords. To resolve these issues, we are planning to include image embedding in memory to see if it would help improve the accuracy of descriptions.

Although the models were able to generate some nice valid sentences, there were still cases where the models failed. The main issue we observed was repetition in keywords. For instance, the first bad case in Figure 6 showed that the word 'long' was used over and over again while some other keywords were ignored. The repetition issue was not as bad in LSTM model, but as shown in the bad cases in Figure 5, similar problems do appear from time to time. One possible explanation was that some of these adjectives such as 'long' and 'striking' were much more often used than others such as 'metallic' and 'multicolored'. Therefore, the models tend to put more weights on these words when they appear. Adjusting weights based on keywords occurrences in training set might help solve this problem.

Comparing the metrics of the two models with LSTM and GRU units, we observed that both models had similar keywords coverage. But the keywords repetition rate was higher in the case of GRU. This align with our observation in the generated sentences for the two models. LSTM model had more sentences with better quality. As for batch size, we found that having a small batch size would result in a lot more repetition, but having a really large batch size would result in extremely low keyword coverage. For a batch size of 5, 5 sentences with 20 words each were trained in one step, which was a good balance between the amount of keywords and the ratio of keywords and normal vocabulary.

5.2. Future Work

To improve the quality of the text generation, we find there are three main aspects of future works that we can experiment with:

- Increase size and improve quality of our dataset

  Compared with the dataset constructed by the original paper, we find the key value in the memory not helpful for clarifying attention focus. We used eight different fashion categories as the placeholder for memory key. Even though the categorization is helpful in tagging fashion terms, most of the categories (like shape, pattern) actually share the same grammar component in sentence. In specific, most of the categories consists of words related to adjectives. Only categories like brand or type has a different grammar components. I think that is why keywords related to type and brand appear correctly more often than other categories.

  For future work, we hope to retag the fashion terms with tag that is more helpful in deciding sentence structures. On the other hand, we find our model suffering from overfitting problem. Considering that we only have 15450, we are going to increase the dataset size to address overfitting issues.

- Multi-modal Memory

  Since the memory slots support key-value pairs with different format, we are wondering if we can include image embedding in one memory slot so

that we can support multi-type memory for attention purpose.

- Force Grammar using POS tagging

  Looking at our test set output, we find some sentences still have incorrect grammar structure. We are thinking to add POS tag into the decoder network, and train the network with loss related to POS tag first in order to ensure correct grammar. And then train with actual words in the ground truth sentences. In this way, the grammar quality might be improved.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. 2

[2] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. 4

[3] Y. Fu and Y. Feng. Natural answer generation with heterogeneous memory. pages 185–195, 01 2018. 2

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. 4

[5] M. F. Oriol Vinyals and N. Jaitly. 2017. 2

[6] S. Robertson. Translation with a sequence to sequence network and attention. 2017. 2

[7] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015. 2

[8] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. 1, 2