Collation of Machine Learning Algorithms for Product Title Generation in E-Commerce

Akshata Katyayana[1], Hemavathy R.[2]

[1]Student, Department of Computer Science and Engineering, R.V. College of Engineering, Bangalore, India
[2]Associate Professor, Department of Computer Science and Engineering, R.V. College of Engineering, Bangalore, India

---***---

Abstract - *E-commerce catalogues can get cluttered with repetitions if each product uploaded by a merchant is uploaded as it is. This necessitates product matching so that the catalogue displays a single result for each unique product. Instead of current systems which search the full catalogue for matches, we can implement a model to generate product titles from the attributes that merchants upload. This saves resources. Since it is a novel proposal, it has to be modelled by taking cues from other similar problems. Title generation is a natural language processing problem which can be solved using a sequence-to-sequence problem. Other problem domains which provide similar solutions are text summarization, neural machine translation, sequence tagging and key phrase extraction. We review several approaches to each of these problems in detail to collate a comprehensive approach to title generation likely to produce the most promising results. Common issues that are addressed by the study are size of merchant attributes, repetition in the generated title and handling out-of-vocabulary words.*

Key Words*: *Product Matching, Title Generation, Sequence-to-Sequence Model, Recurrent Neural Networks, Machine Learning

1. INTRODUCTION

E-commerce websites or applications carry hundreds of millions of items in their catalogue. Several thousand new items are also uploaded by merchants and third-party sellers every day. And the customer traffic per month is in the order of billions. For instance, online retail giant Amazon records more than 2 billion customers per month. With this magnitude of data and traffic, efficient structuring of e-commerce catalogues becomes crucial.

Every time a seller, or merchant wishes to upload their item to the catalogue, it is required to perform a preliminary check to determine if an identical product is already carried by the catalogue. If one proceeds to ingest catalogue items without performing this check, it will result in millions of duplicates in the catalogue. This, firstly, leads to a wastage of memory. Secondly, it contributes to an unsatisfactory customer experience as a customer will have to browse through thousands of nearly identical catalogue options to purchase a single item. Thirdly, merchant business is likely to get affected as products receive reduced visibility in a catalogue cluttered with redundancies. It also introduces a disparity in visibilities among different merchants – products in initial search results are likely to sell more.

Thus, product matching as a preceding step to catalogue ingestion of products is essential. If a match is found, the new product to be uploaded is merged into the product page of the already existing identical item in the catalogue. When a customer clicks on one such product, he/she can then see the prices and offers by each individual seller. This ensures that a consumer is assuredly aware all offers by all sellers for a product before making an informed choice.

The basic idea behind how this product matching is carried out is implementing a search algorithm throughout the catalogue for matches. The search algorithm may employ similarities in keywords or extracted attributes from product descriptions to determine if products are identical. Less common criteria include similarities in prices and images between products. Deep matching algorithms are applied to boost results and make the process more reliable.

There is a scope for converting this procedure from a search problem to a sequence generation problem by applying machine learning. Doing this will eliminate the need to parse the entire catalogue for matches. For a catalogue comprising hundreds of millions of products, this translates into saving several minutes of computation, and in turn, thousands of dollars in revenue.

Given a merchant's description for a product they wish to upload, a title generation system strives to generate the catalogue title of the identical product already present. A merchant's description will be uploaded as a collection of several attributes, for example, product name, type, category, dimensions, color, model number, specifications and so on. Extracting and collating the title from all these fields is the task of the title generator.

The concept of title generation is an uncharted territory; we have to start with a clean slate. The best approach would be to model the problem on an existing similar system and then make customizations. At its crux, title generation is a sequence-to-sequence (generation or conversion) problem. Hence, other sequence-to-sequence problem solution is where our solutions will lie as well. A system that closely resembles our need is text summarization, its slight variations included. Sequence tagging and machine translation are also sufficiently close concepts to our which can be explored in detail.

In this paper, we present an extensive literature review of the above outlined approaches in section 1 and summarize the major findings in section 2.

2. LITERATURE SURVEY

Text Summarization usually coverts a long sequence into a shorter version that encompassed the important points. It is most similar to title generation which aims to covert long merchant descriptions into a crisp title and do so by learning the importance of words in these descriptions. Text Summarization is broadly of two types – extractive and abstractive. Extractive Summarization has in its summary only those words which are already present in the original text. The ordering of words is also the same as the original. In abstractive text summarization, the summary includes novel words, that is, words that were not part of the original text. It also has word re-ordering capabilities.

Whether our title generation system is extractive or abstractive can be decided after conducting a study on the average percentage of catalogue titles being covered by merchant descriptions. If the percentage is high, an extractive approach will suffice, else, an abstractive approach will be needed.

The solution in [1] takes an extractive approach to document summarization. It uses surface, event, relevance and content features of each sentence. A combination of these are used to extract sentences for a summary. For our application, we can reduce sentence level features to word level features. Thus, relevance features which capture the relatedness of a sentence with others can be coerced to indicate the relatedness between tokens. Content features of sentences can be adapted to capture meanings of tokens instead. The paper performs experiments with both supervised and semi-supervised approaches, because labelled data for supervised approaches is often scarce. They show very good results with semi-supervised learning.

A model devised in [2] solves the sequence-to-sequence problem by employing an encoder-decoder architecture. A basic encoder-decoder architecture works by splitting the corpus into tokens, and these tokens are encoded into vectors by the encoder so they can be processed by a neural network. The output tokens from the neural network are then decoded back into words by a decoder and the final results is a sequence of tokens. The encoder units are bidirectional GRU-RNN (Gated Recurrent Unit-Recurrent Neural Network). [3].

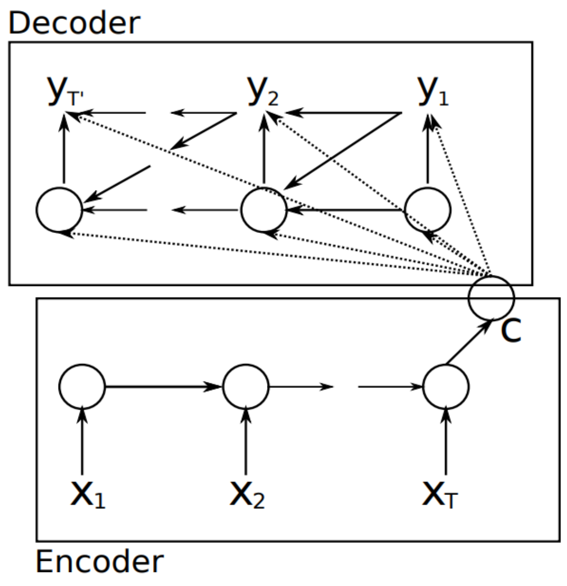

Figure 1: Encoder-Decoder Recurrent Neural Network Model

RNNs are types of neural network where the output at each cell depends not only on the input to that cell, but also on the learnings from the previous cells. This is ideal for our application since the context of a word is as important as the word itself. Bidirectionality means that this context for a word is captured not only for the words that appear before the word in the sequence but also for those that appear after it. The decoder consists of unidirectional GRU-RNN since the decoder does not have the task of encompassing context. The model also introduces attention, which means that it has the capability to give more importance to certain tokens both temporally and spatially. In other words, it mimics the human trait of focus or attention on a certain part of the sequence.

A property of sequence-to-sequence problems is that the lengths of the input and output do not have to be the same and do not have any constraints either. Shifting from fixed to variable length vectors is the main problem that this model solves in comparison to previous work. This model is considered a standard for neural machine translation.

With respect to abstractive text summarization, the model in [4] is a state-of-the-art, and forms the starting point of many other advanced systems. It picks up from the sequence-to-sequence encoder-decoder RNN described in the previous paper and offers advancements. The paper conducts experiments on two corpora – the DUC corpus comprising documents and their corresponding summaries, and the CNN/Daily Mail corpus, comprising news article documents and their summaries. A glaring point of difference between their work and ours is that their corpus consists of documents and ours consists of attributes, typically stored in tables. This means that we will not be utilizing the part of the system that models sentence-to-word hierarchy. Additionally, their system also compresses documents in a *lossy manner,* but we will not need to since title generation does not require extreme summarization. The first way in which it augments the first model is by applying a large vocabulary trick (LVT). This is implemented by deciding on a fixed size for the vocabulary. Then add words in decreasing order of their frequencies to the vocabulary until this fixed size is reached. A vocabulary is central to the idea of abstractive text summarization since producing novel words needs a vocabulary from where such words can be emitted. The next property added to the model is capturing keywords, which again corresponds to the importance of context. An embedding matrix of features for each word which can looked up at the time of generation is how the paper achieves this. Variations of this paper's work are seen in [5] which uses convolutional neural networks for the encoder instead of RNNs.

So this far, the model has taken care of words that are shown in the input during training time as well as words in the curated vocabulary. But what if the title to be generated (in a testing or production scenario) needs a word that is present in neither, but present in the input that the system is currently seeing? Such words are called out-of-vocabulary (OOV) words. The paper [6] is a deep-dive into the concept of switching pointer-generator mechanism which is capable of handling OOV words. In case, the output needs a word from the new input, a pointer to the input word is included as a placeholder in the output sequence, instead of an actual word token. A word's hidden state representation that encompasses its context is used to decide which input word the pointer points to. At the time of sequence generation, a generator generates corresponding output tokens from the pointers. Since the experiments here too are conducted on the CNN/Daily Mail dataset, the paper's work can be considered a feature that augments the work in [4].

Besides pointer generator, the paper introduces the feature of coverage. A model with coverage keeps track of words that have already been emitted into the output, so they are not emitted again. It is a mechanism to remove redundancy. It is particularly useful in title generation because we require titles to be crisp and short. Implementing the coverage mechanism makes use of the already existing attention mechanism. Coverage vectors are used to keep track of the attention given to parts of the sequence. Repeated attention over the same token attracts a penalty and hence is not considered to be outputted again. Their work also includes gradient clipping and early stopping as mechanisms to avoid over-fitting.

The paper draws quantitative comparisons between previous baseline models and its own work. The performance metric is used is a standard for comparing sequence similarities, called the ROUGE metric. ROUGE metric measures the degree of overlap between expected and generated output sequences. The basic unit considered for overlap depends on which ROUGE [7] metric is employed – ROUGE-1 evaluates overlaps in unigrams, ROUGE-2 for bigrams, ROUGE-n for n-grams and ROUGE-L measures the longest sequence overlap between the two sequences. The ROUGE scores for the baseline sequence-to-sequence model successively increase upon adding attention, pointer-generator and coverage. However, the model recorded a low rate of novel word generation, leaving more abstraction to be desired.

So we next explore [8] which proposes a couple of ways to improve abstraction in summarization. The model has two parts to the decoder – one that attentionally attends over the source text to be summarized and one that harbors pre-existing knowledge of language-generation. This enables the decoder to produce outputs by combining knowledge from an external language model as well as from the context of the current text. The results on the CNN/Daily Mail Dataset show increased level of abstraction but as a trade-off, ROUGE scores show a slight decrease.

A popular version of abstractive text summarization models are those which make use of pre-trained encoders. Instead of their own vocabulary, these models turn to already existing, standardized and exhaustive vocabularies that have been curated to be cover a wide range of problem scenarios. The model inherits this vocabulary in the form of an encoder that has been pre-trained on such a standard vocabulary and can be used as a pre-modelled component to replace the encoder of any custom model. One such popular pre-trained encoder that is creating waves in the natural language processing domain is BERT [9] by Google. Another state-of-the-art and popular pre-trained encoder is ELMo.

The paper [10] stacks transformer layers on top of BERT to build an extractive text summarization model. This results in a system where the encoder is pre-trained while the decoder is not. Bridging this gap results in an abstractive model built on top of the extractive one. Thus, the encoder undergoes two levels of fine-tuning, one for the extractive version and another for abstractive. In the extractive version, each token is tagged as either included or not included in the output, and this reduces it to a binary classification problem. The abstractive approach is more complex. It assigns to each token three kinds of embeddings – token embeddings that encompass the meaning of the token, segmentation embedding to indicate delimiters between sentences and positional embedding to indicate the position of the token in the sequence. For the purpose of title generation, the segmentation embedding can be discarded by us. These embeddings are then normalized and the decoder generates the summary. Upon testing with several number of transformer layers with BERT, the extractive model performed best with 2 layers and the abstractive one with 6 layers. The project is implemented in PyTorch and OpenNMT, and uses the BERT tokenizer to tokenize sentences. Final results were obtained for the CNN/Daily Mail and New York Times dataset which are more extractive, and for the XSum dataset which is quite abstractive. All three cases show improved results over state-of-the-art models.

Implementing a very similar model is the paper [11]. Its encoder encodes with the help of BERT not just tokens but also their contexts. The decoder is a two-stage model, the first one a transformer and the second uses BERT. Stage 1 produces a preliminary output sequence which is masked by stage 2 and combined with the input sequence itself to produce a refined output. Essentially, the decoder achieves the same task as in [10], that of combining knowledge from BERT as well as the input sequence.

Few other papers implement pre-training based language models with similar results but slightly different approaches. The system in [12] uses copy mechanism which works very similar to the pointer generator mechanism. Instead of placing pointers to the source text in the output, copy mechanism copies the word itself, thus doing away with the need for a generator.

Carrying forward the idea of transformers for natural language processing, [14] introduces an attentional transformer model that completely does away with the use of recurrent and convolutional networks for machine translation.

Transitioning from RNNs, [15] introduces the concept of using Convolutional Neural Networks (CNNs) for text summarization. This is an innovative approach since CNNs are typically used for computer vision and image processing applications where pixels are encoded into matrices of vectors. The model obviously had to achieve a similar matrix representation with text. Pre-processing steps are typical – tokenization, word stemming and removal of stop words, followed by creating word embeddings of tokens using word2vec. A very important step is feature extraction from the text matrix. The summarization step has three levels, a surface level extracts important sentences from the document, and is irrelevant for our title generation application. The next level captures relationships between entities, or in other words context. This is achieved via a vector space model with a continuous embedding space in which words which are semantically close are mapped or embedded in the space close together. The last level of summarization encompasses the global context of the word in the document, and can be ignored by us. The code is implemented in Theano, a deep learning library based on Keras. Extensive results on the DUC dataset were plotted as boxplots after studying 26 different configurations of CNNs and the best one was chosen.

The system in [16] uses a BiLSTM-CRF model to perform sequence tagging, that is, it replaces the CNN part of [15] with a BiLSTM network. Sequence tagging or labelling is a process in which each token in a sequence is given a tag or label indicating some kind of feature classification for the token. In the most typical form of sequence tagging, called named entity recognition (NER), there are 5 classes such as proper noun, organization and so on, which a token can be tagged as. For our purpose, we can create two classes – token is to be included in the title or not included. LSTMs (Long Short Term Memory networks) are a class of RNNs that can capture long range dependencies in longer sequences. This helps us correlate a larger number of merchant attributes for title generation. A BiLSTM, as the name suggests, captures these dependencies on both sides of the token, that is, it can read context of tokens encountered in the past as well as in the future. This model does away with the need for creating word embeddings and reduces text summarization to a much simpler problem of sequence tagging.

The model in [17] takes a semi-supervised approach to sequence tagging using pre-trained word embeddings and bidirectional language models in addition to an RNN network. The bidirectionality gives a performance boost. Although they have presented results on standard NER and chunking datasets, they also state that the model provides satisfactory results on datasets of other domains, which is a positive from our point of view.

Paper [18] introduces a new architecture to deal with the exposure bias [19] problem exhibited by the standard attentional encoder-decoder model. It proposes an attention mechanism which parses over the input and output separately and continuously. This is unlike the standard where only the input is read in a non-continuous manner. The model also introduces reinforcement learning in the architecture to enable continuous-in-time attention over the input and output. The result is generation of more coherent summaries as demonstrated by them on the CNN/Daily Mail dataset by achieving a ROUGE-1 score of 41.16, higher than the usual scores which lie in the late thirties.

Despite the fact that they can be modelled very similar to text summarization, Product titles are not sentences at the end of the day, they are more accurately a collection of the most important attributes or phrases. Key phrase extraction is a problem domain analogous to this. While most key phrase generation mechanisms for news articles and blogs are extractive, this may not suffice for title generation. This is because titles, unlike key phrases are not just a list of phrases or words. It is necessary for these phrases to have coherence between them to form a single title. This necessitates abstractive approaches. The model devised in [20] uses the RNN-based attentional encoder-decoder, combined with copy and coverage mechanisms discussed earlier and applies these to achieve abstractive key phrase generation. Results are collated for the Inspec, Krapivin, NUS, SemEval-2010, KP20k datasets which are standards in key phrase extraction. Studying results on new datasets helps us to broaden our search which so far had been confined to only news article based datasets. Since there are no results recorded on e-commerce datasets, it is imperative we ensure that the models we are considering have a wide range of applications and are not functional for only a specific dataset.

A third kind of dataset was explored in [21], a set of twitter posts. It uses again, the same RNN-based model, but it addresses an important problem that was constantly encountered by us in all previous models – that of length. The paper states that the performance of these document-based models shows a sharp decline when applied as it is to datasets of much smaller lengths like Twitter articles (Twitter imposes a word limit on posts). Our collection of merchant attributes for each item are much closer in length to twitter posts than to news articles or documents. Hence this version of the model proposed by them offers added advantages for our application of key phrase generation.

So far we have covered approaches to text summarization, sequence tagging and key phrase extraction. Another sequence-to-sequence problem that provides similar solutions is neural machine translation. As the name suggests it is the process of translating sequences from one natural language to another, while conveying the same meaning. In addition to the RNN encoder-decoder model, [22] introduces a recursive convolutional network model that shows success in learning grammatical structures of sentences. Furthermore, the model shows best behavior for smaller sentences. Both these make it advantageous to our application. As discussed, we require the model to learn structure titles and be able to obtain generate these titles from a relatively smaller number of tokens. The model of [23] is also a deep neural network to machine translation. It combines with the already studied RNN, a recursive neural network resulting in a recursive recurrent neural network. It uses word2vec to encode words and employs a tree structure to capture the translation from source to target language.

Probably the sole paper which directly tackles the problem statement of product title generation in e-commerce is [24]. The difference in their problem statement is that they utilize multiple merchant titles to generate a single product title, whereas we utilize all attributes, not just the title, from a single merchant. Hence, their approach is based on a frequency scoring of tokens – tokens that are more common across several titles are important and should be retained. This is done by assigning each token as slot-value pairs, and n-grams from the most important pairs are combined to give the title. Another approach in the same paper uses a stack decoder to get and string together the most important n-grams.

3. CONCLUSIONS

We established that natural language processing solutions applying sequence-to-sequence models are to be explored for a solution to title generation. We started by studying extractive and abstractive text summarization. Abstractive summarization models were all based on an RNN encoder-decoder architecture. This, in different approaches was augmented by additional features like pointer-generator networks, attention, copy mechanism and coverage mechanism to boost results. We also studied key phrase generation, both extractive and abstractive and an approach that adds structure to key phrases generated. Next we had a look at neural machine translation which takes approaches analogous to summarization. Sequence tagging was also studied in detail as an extractive approach to title generation. The results generated from these will have to be verified on our e-commerce dataset applying ROUGE metrics for evaluation.

ACKNOWLEDGEMENT

We would like to thank the department of computer science and engineering at R.V. College of Engineering Bangalore for giving us an opportunity to write this research paper as part of our project.

REFERENCES

[1] Kam-Fai Wong, Mingli Wu, and Wenjie Li., ‘Extractive summarization using supervised and semi-supervised learning’, 22nd International Conference on Computational Linguistics, Vol. 1, Manchester, Aug. 2008, pp 985– 992.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, ’Neural machine translation by jointly learning to align and translate’, CoRR, abs/1409.0473, 2014.
[3] Junyoung Chung, Çaglar Gül.ehre, KyungHyun Cho, Yoshua Bengio, ‘Empirical evaluation of gated recurrent neural networks on sequence modeling’, CoRR, abs/1412.3555, 2014.
[4] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, C¸ aglar G ˘ ulc¸ehre, and Bing Xiang, ‘Abstractive text summarization using sequence-to sequence RNNs and beyond’, The 20th SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, Aug. 2016, pp. 280–290.
[5] Sumit Chopra, Michael Auli, Alexander M. Rush. 2016, ‘Abstractive sentence summarization with attentive recurrent neural networks’, In HLT-NAACL, 2016.
[6] Abigail See, Peter J. Liu and Christopher D. Manning, ‘Get to The Point: Summarization with Pointer-Generator Networks’, Cornell Uviversity Library, arXiv:1704.04368v2 [cs.CL], Apr. 2017.
[7] Chin-Yew Lin, ‘Rouge: A package for automatic evaluation of summaries’, In Text summarization branches out: ACL workshop, 2004.
[8] Wojciech Kryściński, Romain Paulus, Caiming Xiong, Richard Socher. ‘Improving Abstraction in Text Summarization’, Cornell Uviversity Library, arXiv:1808.07913v1 [cs.CL], Aug. 2018.
[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, ‘BERT: Pre-training of deep bidirectional transformers for language understanding’, Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers), Minneapolis, Minnesota, May 2019, pp. 4171–4186.
[10] Yang Liu and Mirella Lapata, ‘Text Summarization with Pretrained Encoders’, Cornell Uviversity Library, arXiv:1908.08345v2[cs.CL], Sep. 2019.

[11] Haoyu Zhang, Jianjun Xu, Ji Wang, 'Pretraining-Based Natural Language Generation for Text Summarization', 23rd Conference on Computational Natural Language Learning, Apr. 2019.

[12] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li, 'Incorporating copying mechanism in sequence-to-sequence learning', 54th Annual Meeting of the Association for Computational Linguistics, Vol.1, Berlin, Germany, Mar. 2018, pp. 1631–1640.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gome, Łukasz Kaiser, 'Attention Is All You Need', Cornell University Library, arXiv:1706.03762v5 [cs.CL], Dec 2017.

[14] Wajdi Homaid Alquliti , 'Convolutional Neural Network based Automatic Text Summarization', International Journal of Advanced Computer Science and Applications, Vol. 10, No. 4, pp. 200-210, Jan. 2019.

[15] Zhiheng Huang, Wei Xu, Kai Yu, 'Bidirectional LSTM-CRF Models for Sequence Tagging', Cornell University Library, arXiv:1508.01991v1 [cs.CL], Aug. 2015.

[16] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, Russell Power, 'Semi-supervised sequence tagging with bidirectional language models', Cornell University Library, arXiv:1508.01991v1 [cs.CL], Apr. 2017.

[17] Romain Paulus, Caiming Xiong, and Richard Socher, 'A deep reinforced model for abstractive summarization', Cornell University Library, arXiv:1705.04304v3 [cs.CL], Nov. 2017.

[18] Florian Schmidt, 'Generalization in Generation: A closer look at Exposure Bias', Cornell University Library, arXiv:1910.00292v2 [cs.LG], Nov 2019.

[19] Yong Zhang and Weidong Xiao, 'Keyphrase Generation Based on Deep Seq2seq Model', IEEE, Vol. 6, No. 3, pp. 3-9, Aug. 2018.

[20] Qi Zhang, Yang Wang, Yeyun Gong, Xuanjing Huang, 'Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter', Conference on Empirical Methods in Natural Language Processing, Austin, Texas, Nov. 2016, pp. 836–845.

[21] Kyunghyun Cho Bart van Merrienboer, Dzmitry Bahdanau, Yoshua Bengio, 'On the Properties of Neural Machine Translation: Encoder–Decoder Approaches', Cornell University Library, arXiv:1409.1259v2 [cs.CL, Oct 2014.

[22] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi and S. Jain, "Machine translation using deep learning: An overview," 2017 International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, 2017, pp. 162-167.

[23] Jos´e G. C. de Souza, Michael Kozielski, Prashant Mathur, Ernie Chang, Marco Guerini, Matteo Negri, Marco Turchi,, Evgeny Matusov, 'Generating E-Commerce Product Titles and Predicting their Quality', The 11th International Natural Language Generation Conference, Tilburg, The Netherlands, November 2018, pp. 233-243.