Generating E-Commerce Product Titles and Predicting their Quality

José G. C. de Souza[1], Michael Kozielski[1], Prashant Mathur[1]*, Ernie Chang[2], Marco Guerini[3], Matteo Negri[3], Marco Turchi[3], and Evgeny Matusov[1]†

[1]eBay Inc., Core AI, Germany
{jgcdesouza,mkozielski}@ebay.com
[2]University of Washington, Seattle
cyc025@uw.edu
[3]Fondazione Bruno Kessler, Trento, Italy
{guerini,negri,turchi}@fbk.eu

Abstract

E-commerce platforms present products using titles that summarize product information. These titles cannot be created by hand, therefore an algorithmic solution is required. The task of automatically generating these titles given noisy user provided titles is one way to achieve the goal. The setting requires the generation process to be fast and the generated title to be both human-readable and concise. Furthermore, we need to understand if such generated titles are usable. As such, we propose approaches that (i) automatically generate product titles, (ii) predict their quality. Our approach scales to millions of products and both automatic and human evaluations performed on real-world data indicate our approaches are effective and applicable to existing e-commerce scenarios.

1 Introduction

E-Commerce websites are now an established way to buy and sell products using online platforms that have a vast and diverse catalog of products. A catalog is composed of a series of products that are unique and can broadly be identified by their brand, model and main features that vary according to the type of product (clothes, electronics, books). A product title is the realization of this information in a human-readable way so that users can understand the main features of the product.

Online platforms expose the products via product pages that condense the information for a product and can use the title as the product's main summary. A product page for "*ACME Model Smartphone 64GB Black Unlocked*" is shown in Figure 1. The product page also aggregates all the listings of the product being sold (bottom of the figure). A listing (or item) is an instance of a product sold in the platform by a seller. Its title might contain information such as condition of the item (used, new, among others), price, shipping and quantity tags, and other information specific to a particular item. Product titles cannot contain such information because they describe the product and not item-specific details like its price.

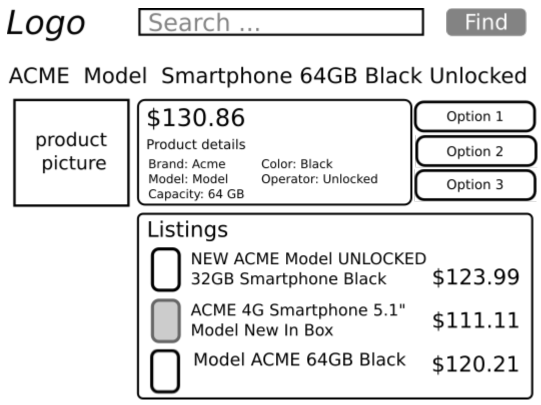

Figure 1: Example of product page.

Large e-commerce platforms have millions of products and manually creating titles for such products is not feasible. In order to scale the process of creating product titles, such platforms need to employ algorithms that automatically generate titles and are fast enough to scale. One possible way to generate text is to build a rule-based system that uses slot-value pairs to generate text, selecting the most important pairs in the output (Dale et al., 1998). Another possible way, that we propose in this study, is to leverage large amounts of seller-provided listing titles and recombine their n-grams to form hypotheses of product titles.

Observing the listing titles created by sellers, it

*Now at AWS Inc.
†Now at Apptek Inc.

is noticeable that many of them contain a mix of irrelevant information (shipping, condition, price, information, among others) and relevant information (brand, model and features). The occurrence of ready-to-use product titles among listing titles is possible but not guaranteed. Based on this observation, one hypothesis that emerges is that a good product title could be formed by tokens that occur most frequently across listings aggregated for the product, and that combining these frequent expressions could yield a good product title.

The title generation approach presented here builds on top of this hypothesis. It is based on a statistical approach that first counts n-grams occurring in the titles aggregated into a product and then recombines them to form product title hypotheses. Furthermore, the available slot-value pairs can be used to enforce tokens that are important, ensuring that relevant product information is present in the generated output.

Though algorithms can scale the title generation process, often the quality of the output generated is not good, and some titles might not be adequate. This can happen due to several reasons, ranging from noisy input data (e.g. noisy aggregation of titles into a product) or bad hypothesis generation. Therefore, in addition to automatically generating titles, there should be a way to assess whether the generated titles are good enough for publishing, in order to avoid bad user experience. For example, in Figure 1, if a generated title has a brand other than "ACME", it would not be an appropriate title for that specific product.

In this paper we present approaches to both problems: (i) generating e-commerce product titles and (ii) predicting their quality. Our main contributions are:

- An approach that generates product titles taking seller-provided listing titles as input and that scales to millions of products. The method is based on a stack decoder search algorithm that recombines frequent n-grams observed in the listing titles to form a product title hypothesis.
- An approach to estimating the quality of titles based on supervised machine learning methods, in particular neural networks trained on human-annotated data.
- A thorough evaluation of the approaches on in-house data and a qualitative analysis of the system's outputs using human evaluation.

The remainder of the paper is organized as follows: Section 2 describes work in text generation related to the approaches described here; Section 3 describe the title generation approach; Section 4 presents the title quality prediction approach; Section 5 lays out the experimental settings used for evaluating the approaches described in the paper; Section 6 presents and discusses the results obtained and Section 7 summarizes the conclusions and lays out future work directions.

2 Related Work

Prior work on title generation for e-commerce focused on browse pages and has only explored a hybrid approach combining rule-based and statistical machine translation models (Mathur et al., 2017). The input to this approach consists of structured information about products in terms of slot/value pairs (e.g. `Watch_Type: wrist watch`). Although the task is similar to ours, hand-crafting and encoding product-specific rules is a time-consuming endeavour which does not scale to the hundreds of slot-value pairs and millions of products in the catalog. Below, we discuss three approaches that can either be directly applied or adapted to product title generation. The first approach is selection-based, while the last two are generation-based.

Hypothesis Selection. The most intuitive approach is to select, among the listing titles, the one that most "appropriately" describes the product. This can be achieved by applying diversity-based ranking techniques used in extractive summarization, such as Maximal Marginal Relevance (MMR), to prune and select from the set of titles (Carbonell and Goldstein, 1998; Gillick, 2011). Alternatively, systems can also learn how to pick the candidate title that is closest to the reference title. This approach can rely on ranking scores produced by models trained on listing titles and the corresponding human-curated reference product titles, with automatic metrics such as BLEU (Papineni et al., 2002) or TER (Snover et al., 2006) as labels. Some of these techniques are employed in system combination (Rosti et al., 2007; Barrault, 2010; Devlin and Matsoukas, 2012; Suzuki, 2011). However, this approach limits the number of possible generated titles and can potentially introduce seller-biases when a single seller's title is selected as the product title.

Re-decoding Approaches. Re-decoding is a generative process that learns to predict the posterior probability $p(y|x)$ (in our case, the posterior probability of a generated title y given the initial list of user-created titles x), which can also be viewed as a sequence quality score. Since quality scores are used to rank the partial sequences, an accurate scoring function would yield the highest quality outputs. Decoding can be seen in Minimum Bayes' Risk Combination (González-Rubio et al., 2011; González-Rubio and Casacuberta, 2013), abstractive summarization (Rush et al., 2015; Chopra et al., 2016), and Neural Machine Translation (NMT) models (Bahdanau et al., 2014; Chen et al., 2016; Vaswani et al., 2017). State-of-the-art approaches utilize encoder-decoder models that extract a feature representation of a variable-length input sentence before generating an output. However, the bottleneck of this approach is its dependence on the size of quality data; it often performs poorly when annotated data is noisy and/or insufficient (Koehn, 2017).

Hypothesis Fusion. An alternative approach is neither to select nor to generate, but to 'fuse' already generated hypotheses. This, for instance, can be done using Confusion Network (CN) decoding (Ma, 2014). In this approach, a confusion network is generated by first selecting a listing title as backbone, and then by aligning it to all the other listings. The network is then traversed to obtain the product title with the highest consensus among the input hypotheses. This title can be decoded with decoding units that include either phrase-level (Feng et al., 2009; Du and Way, 2010) or word-level (Barrault, 2010; Rosti et al., 2007; Fiscus, 1997). Systems can choose between 1-to-1 mappings (Barrault, 2010; Rosti et al., 2007; Du and Way, 2010) or many-to-many mappings (lattice) (Feng et al., 2009; Ma and McKeown, 2015; Matusov et al., 2006) in hypothesis alignment. The main drawbacks of these solutions are: (1) final output quality is highly dependent on the quality of the selected backbone (aligning hypotheses to a poor-quality listing can result in outputs that are far from being usable in real industrial settings), and (2) lattice creation becomes computationally expensive as the number of initial hypotheses grows, potentially $\mathcal{O}(n^2)$. This makes approaches based on CN unsuitable for our working scenario where there is the need of generating titles for millions of products where each product consists of a potentially large number of listing titles. Our approach must generate product titles in linear time and must be robust to noises present in seller-created titles.

3 Title Generation

This system's purpose is to provide hypotheses of product titles. It receives as input a list of item titles for listings previously aggregated into one product (like the titles at the bottom of Figure 1) and product-related data in the form of slot-value pairs (as the name-value pairs shown under "Product Details" in Figure 1). In addition to these, a human-curated reference product title is required during training time.

The process of generating titles can be roughly summarized into two steps. The first is computing different statistics about the item titles: n-gram counts (in this implementation fixed to bi-grams), inverse document frequency (IDF) of each unigram, listing titles length, counts of tokens given the position of each unigram in the listings, and filtering of slot-value pairs. The slot-value pairs are defined a priori and they are based on the aggregation of titles into products, which means some of the pairs can present noise. In order to filter out noisy slot-value pairs and to understand which pairs are important, we derive an importance score for each pair. This score is computed by dividing the number of times a value appears at least once in the listing titles by the number of listing titles aggregated to the product. The top-k pairs according to this score are kept. The second step consists of performing the recombination of n-grams found in the titles using an heuristic stack-based search algorithm, also known as stack decoding (Wang and Waibel, 1997) using all the information computed in the first step.

3.1 Stack Decoding for Title Generation

The idea of stack decoding is to keep a list of multiple stacks, in which each stack represents a position of the title being generated. The search algorithm initiates with a start symbol (`<s>`) and expands the title hypotheses position-by-position (given the pre-computed bi-gram counts). The process is summarized in Algorithm 1.

The hypotheses are expanded by the `get_transitions` function. It consists of retrieving all the possible transitions from the current token (the last token of the hypothesis in hyp, that is the hypothesis being generated).

Algorithm 1: Stack decoder for title generation

```
Data: titles and SV_pairs preprocessed and
      tokenized
Result: product_titles sorted by score
Initialize stacks with first stack with single
  hypothesis <s> and max_stack_number- 1
  empty stacks;
for current stack in stacks do
    while previous stack in stacks is not
      empty do
        Get the top hypothesis hyp in the
          previous stack;
        candidates ← get_transitions
          (hyp, titles, SV_pairs);
        for each candidate in candidates do
            if candidate token is not EOS
              symbol then
                create new_hyp out of
                  candidate;
                adds new_hyp to current stack
                  in stacks;
            end
        end
        compute_scores (candidates);
        order candidates by score;
        add prune (candidates) to current
          stack in stacks;
    end
end
```

This is performed by looking up the most likely words to follow the current word as given by the bi-gram and token position counts (transformed into probabilities and represented by titles in Algorithm 1). For each hypothesis candidate token a new hypothesis is created and placed in the current position stack in stacks. If the candidate token of the new hypothesis is the end-of-sentence (EOS – </s>) symbol, the new hypothesis is not created. This whole process is repeated until the current stack is not empty, i.e., there are no hypotheses to expand.

The next step is to score all the hypotheses in candidates. Here, the approach taken is to build a regressor that predicts a score used to rank the hypotheses. This is implemented using an algorithm that induces a model that predicts BLEU scores (Papineni et al., 2002). The approach is simple: at training time, the scores are derived by computing the sentence-level BLEU score between each title hypothesis in candidates and the human-curated reference provided. At inference time, the score is the one predicted by the regressor.

For training the regressor we explore information computed during the search process. For each hypothesis (which can be a partial, not complete title), 13 features are extracted. The feature set contain features that are **global** and applied to every title under a product, such as: the number of listing titles of the product and average title length of the listings of the product. The other features are **local** to the hypotheses, such as: *1*) the cumulative bi-gram probability of the hypothesis; *2*) the cumulative position probability over all tokens in the hypothesis; *3*) the IDF score of the last token of the hypothesis; *4*) a ratio between the last token position and the average title length among all listings of the product; *5*) the hypothesis length; *6*) the number of irrelevant information matches in the hypothesis (computed based on lists of irrelevant condition-, shipping- or quantity-related tokens); *7*) coverage penalty: a slot-value pair coverage penalty that given the list of important slot-value pairs, computes a score that is a ratio of the importance score and the number of uncovered slots; *8*) language model (LM) score for the whole hypothesis string (4-gram LM trained with Kneser-Ney smoothing (Kneser and Ney, 1995) on a set of human-curated titles); *9*) number of values of slot-value pairs present in the hypothesis; *10*) length penalty: the absolute difference between the average title length and the current position of the candidate token divided by the average title length and *11*) gain function score: a log-linear combination of 1, 2, 6, 8 and 10. These features are descriptors that try to capture content and structure of the titles using information about what is important in a product title and what is not. They are language agnostic and can be applied to any language.

For training the regressor we use a least squares linear regression algorithm which is fast both during training and inference time. Before training, the feature matrix columns are normalized by removing the mean and scaling to unit variance.

After obtaining a score for each candidate in the current stack, the candidates are sorted in descending order and a pruning strategy is used to filter the hypothesis. The pruning approach that yielded best results during development was keeping the top-k candidates of the ordered list. This pruning is implemented in the `prune` function. After this step, the current stack is updated with the kept candidates and the process moves to the stack representing the next position in the generated title.

In the next section, we describe an approach that performs title quality prediction, similar to what the regressor used for ranking the hypotheses does. The main difference is that the quality prediction model is trained on tagged data with quality-oriented tags instead of BLEU scores. The quality tags are based on a pre-defined set of quality requirements which are a better proxy of quality than edit-distance referece-based metrics such as BLEU. Furthermore, the system can use a set of more diverse global information to incorporate during modeling as well as more complex learning algorithms, as there are no speed performance limiting issues. The regression model used in search, instead, needs to be fast enough to produce predictions at inference time during search without making the process slow.

4 Title Quality Prediction

The purpose of a title quality prediction system is to assess at real-time whether a title can be used or not, without relying on humans to perform the decision and independently of the system used to provide the title. Therefore, the system must be system/source-agnostic and work in an absolute notion of quality.

An important step of building a system that predicts the quality of automatically-generated output is the definition of quality itself. Here, we define what is a good product title and what are the main dimensions of this definition. Overall, a product title should provide a concise but accurate description of what the product is about. What is important to be in the title depends on different types of products (or categories). Cell phones for example require the brand, model, color and carrier, but not a shoe size.

The main dimensions used to determine whether a title is good or not are: absence of both important information issues and irrelevant information issues. The former refers to relevant information that is missing or incorrect in a title; for instance, the brand, model, and product type specification (what is the product) should be appropriate. The latter refers to information that is not required and should be omitted such as condition (e.g. "*new*", "*used*", "*in a box*", etc), shipping (e.g. "*free shipping*", "*U.S shipping*"), marketing (e.g. "*amazing*", "*best offer*"), quantity, and price expressions or any other kind of expressions that are not related to the product itself but to the listing. Furthermore, the latter also includes any kind of repetition (same surface word or related words). Next, we describe the approach to modeling title quality prediction using classification algorithms.

4.1 Learning Algorithms

We cast the title quality prediction problem as a classification problem in which the labels indicate whether the product title is good for usage. More details about the data used to train the classifier are given in Section 5.1. We have explored two different learning algorithms to induce classifiers for this task: random forests (Breiman, 2001) and Bidirectional Long Short-Term Memory models (LSTMs, Hochreiter and Schmidhuber (1997); Schuster and Paliwal (1997)). Random forests (RF) are ensemble classifiers that induce several decision trees using some source of randomness to form a diverse set of estimators (Breiman, 2001).

Recurrent neural networks (RNNs) are models well-suited to deal with variable-length input like natural language sentences. Though RNNs can cope with variable-length sequences, the optimization of the weight matrices in RNNs is hard: when the gradients are back-propagated, they decrease to the point of becoming so small that the weights cannot be updated, specially over long input sequences. Hochreiter and Schmidhuber (1997) proposed LSTMs, which are able to overcome the vanishing gradients problem by capturing long-range dependencies through the use of gated memory cell units that can sustain information across long input sequences. In this work, we use bidirectional LSTMs, which have an additional layer that receives the reversed sequence as input, thus keeping track of past and future states. For more details on RNNs, LSTMs and their bidirectional counterparts (biRNN and biLSTM) we refer the interested reader to Goldberg (2016).

4.2 Features and Architecture

For modelling the problem using the learning algorithms described in Section 4.1, we resort to several kinds of information. The RF models use as a basis a bag-of-words (BoW) representation of the titles whereas the biLSTM-based models use the embedded representation of the words. In addition to these, several features are extracted (total of 80 features) and they can be roughly grouped into: length features (e.g. length of the titles in tokens and chars, ratios of the title length and the aggregated average, max or min title length under a product), counts of repeated tokens (excluding punctuation and numbers), counts of encoding errors, and slot-value pairs coverage (counts and ratios of values matching the tokens of the title). All of these features are independent from the generation process and were designed to be agnostic with respect to the way the title has been obtained.

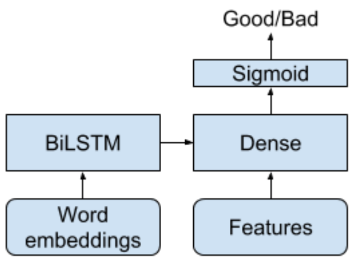

Figure 2: BiLSTM architecture with features concatenated in the hidden layer.

The feature extraction step for the RF models includes concatenating the BoW representation to the features extracted. The biLSTM-based models can also be used with features and the architecture is represented in Figure 4.2. The network has two inputs: the embedded representation of the words in the title and the features computed for the title. The word embeddings are used as input for the biLSTM network and the features are concatenated to the output representation of the biLSTM in a hidden layer. The output layer predicts the class using a sigmoid activation. When the features are not used, the only difference in Figure 4.2 is the concatenation of features.

5 Experimental Settings

In this section we describe relevant settings used in our experiments with the title generation and quality prediction systems.

5.1 Data

The input data for the systems consists of a series of listing titles aggregated into a product. In addition, we have access to slot-value pairs at the product-level. All the data used in our experiments is proprietary user-generated data containing different kinds of peculiarities and noise, such as spelling errors, emoticons and punctuation marks. In addition, the aggregation of the listing titles into a specific product is not perfect, featuring noise due to the presence of items of other products. As an example, in Figure 1, one of the listing titles lists a smartphone with 32GB memory capacity while the product is of 64GB. Therefore, the challenges presented by the data are the same challenges posed by real-world contexts, in which noise is a constant. In order to alleviate some of these problems, we preprocess the data using an in-house modified version of the Stanford tokenizer that does not break certain tokens (e.g. model identifiers – "DM-234/5").

Generation Data. For training the generation model, we used about 1.4 million listing titles which are part of 16,733 products. The number of listings per product ranges from 1 to 500 with an average of 87.6 and a median of 6. About 62% of the products have 10 or less listings. There are about 1200 popular products which have 500 listings. In Addition we have on average 31.3 slot-values per product (median 27). The average length of a tile is 64.4 characters or 10.4 tokens. For each product we have one human-curated title as reference. As development data we use a set of 749 products with 51,441 listing titles which are sampled out of the same distribution of the training data. An important detail about the data splits is that they are performed at the product-level, i.e., listing titles of a product do not appear both in training and test splits. This is true for all the splits described in this section.

Quality Prediction Data. For the scoring model, a sub-sample of the generation data had each listing title annotated by humans, checking for quality problems related to irrelevant information or important missing information in the titles. The problems are organized in a hierarchy in which the main groups are: copyright issues (e.g. brand, model spelling problems); encoding issues; offensive wording issues; required data not present in the title (e.g. brand, or important features for some products such as color, capacity, among others); irrelevant data present in the title (e.g. condition, shipping, or marketing information, price tags, duplicated words and synonyms, among others); syntax/grammar problems (e.g. title not comprehensible because of word ordering or lack of words). This hierarchy has a total of 22 issues. By the end of this process we have a set of products with all the titles aggregated into it annotated with issues (there can be more than one issue per title or none if they are good titles).

The same annotation process and guidelines are used for evaluation. Since annotating each title requires effort, we used only 9,823 products with 52,050 titles, which are completely annotated. This sub-sample was created with a limited amount of listings per product to ensure a higher diversity in the data set. The evaluation process is carried out by several annotators (with no title overlap) and the quality of a sub-sample of their annotation has been assessed by a separate group of annotators.

We have on average 5.3 (median 5) titles and 37.2 (median 34) slot-values per product. The average title length is 63.7 characters and 10.6 tokens. The balance of positive and negative labels is 42.9/57.1. The development data consists of 1,000 products with 5,174 listing titles, which are sampled out of the same distribution of the scorer training data. The balance of positive and negative labels is 44.3/55.7.

Evaluation Data. In order to evaluate the generated output, human evaluation on an additional set of 2,000 products has been carried out. The evaluation is performed by analyzing the output of each system. The analysis process follows the same guidelines applied for annotating the title quality predictor training data described above, analyzing the different dimensions of a title that can lead to poor product titles (laid out in Section 4 and fine-grained in the quality prediction data description, above).

5.2 Baselines

For the **generation** task, a reasonable baseline is to have a rule-based system that combines the slot-value pairs while performing some content selection similar to the rule-based approach proposed by Mathur et al. (2017). For that, we built a simple system that concatenates the most important values of the product's slot-value pairs to form the product title. We derive the importance of a slot-value pair in the same way we obtain the score derived for filtering them, which is described in Section 3. After deriving the score, we order pairs by this value and select the top-10 to form a title. Another baseline is the selection of the most frequent listing title under a product. The fact that different sellers independently used it can in fact indicate that it represents a good title hypothesis.

For the **quality prediction** task the baseline is the majority class of the training set.

5.3 Parameter Settings

The most important parameter of the **title generation** system is the beam size, which was set to 3 (the one giving the best performance in terms of BLEU score during the development of the model). We used sentence-level BLEU to compute the labels for training the regression model, set to 4-grams over cased titles and the smoothing mechanism described in (Chen and Cherry, 2014).

The hyper-parameters of the LSTM-based and RF-based **title quality prediction** models were respectively optimized with 300 and 600 iterations of random search with an inner 3-fold cross-validation over the training data. With RFs, we were able to explore the hyper-parameter search space more than with the neural-network-based models due to its faster training time.

6 Results and Discussion

In this Section, we report and discuss the results obtained for each task. We start with the quality prediction problem. It can work as a method for selecting good candidate product titles that complements the generation approach described in Section 3, working as a re-scorer. Next, we discuss the results of the generation task, from a quantitative and qualitative point of view.

6.1 Title Quality Prediction

The intrinsic evaluation of the quality prediction models is carried out on the development set described in Section 5.1. We use classification evaluation metrics to assess the performance of the models. One important remark about this task is that the most important class to predict correctly is the good class. In this problem, it is a bigger issue to have a false positive than a false negative. The metrics we use are the F1-score (harmonic mean of precision and recall for each class averaged), the F1-score for the positive (good) class and the Matthew's correlation coefficient (MCC). The latter is the main metric used in this evaluation because it takes into consideration the class imbalance of the data set.

The results of the experiments are summarized in Table 1. The simplest models trained were RF BoW and biLSTM which are both showing big improvements over the simple Majority baseline when looking at F1 only. When no features are involved, the best choice is to use biLSTM, which reaches a MCC of 34.6. An important trend observed in the results is the strength of the features developed for this problem. The RF trained with the features alone reaches the same performance of the biLSTM. Furthermore, both RF BoW and biLSTM models show large improvements when using the features (around 8 and 10 MCC absolute points, respectively). The best performance is achieved when concatenating the features to the biLSTM representation of the titles, yielding the best results in all metrics (in bold in Table 1). The quality prediction score could be used as a system that selects the best title out of the original listing titles or as a re-scorer mechanism for the stack decoder. We evaluate these in the next section.

6.2 Title Generation

In this section we report results of the title generation task and the human evaluation results. The numbers reported here and in Table 2 are sentence-level BLEU (sBLEU) scores on the development set (described in Section 5.1). The best performance of the generation approach was obtained with beam size 3 (60.1) after trying different values (a beam size of 5 yields 58.5 and performance is not improved with larger values). This is a large improvement over the slot-value pair concatenation baseline, which achieves only 17.1 sBLEU. The assumption that seller-provided titles could provide good hypothesis of titles is supported by the high sBLEU score achieved by the most frequent title baseline (58.8, 1.3 absolute points below generation). Leveraging the quality prediction to select the best title among the seller-provided titles also proves a very strong approach achieving the highest score (68.9). Using the quality prediction system as a re-scorer of the seller-provided and generated titles improves the generation approach by 6.3 absolute points but it does not match the performance of performing selection over seller titles only.

System / Metric	MCC	F1	F1 good
Majority	0.0	35.8	0.0
RF BoW	31.5	61.1	66.8
RF feats	34.6	67	65.5
RF BoW + feats	40.8	69.9	69
biLSTM	34.6	64.1	67.8
biLSTM + feats	**44.7**	**71.7**	**71.2**

Table 1: Results for the title quality prediction models. MCC is Matthews correlation coefficient.

System	sBLEU
Slot-value pairs baseline	17.1
Most frequent title	58.8
(1) SD, beam = 3	60.1
(2) biLSTM + features	68.9
(1) + (2)	66.4

Table 2: Intrinsic evaluation of outputs of different approaches on development sent. Scores are sentence-level BLEU (sBLEU).

In addition, we performed a qualitative evaluation involving humans that inspected the outputs of the systems. The evaluation was performed to identify problems in the outputs that render them not useful, the same way the data for the quality prediction task is obtained (Section 5.1). We summarize the evaluation by reporting the number of outputs with no issues, represented by the number of good titles provided by each approach. The results of the human evaluation are summarized in Table 3, which shows that the approach with the highest number and proportion of produced outputs is the generation one (SD, beam = 3). It is followed by the combination of generation and quality prediction as re-scorer and last the quality prediction system over seller-provided titles only.

System	# good	% good
(1) SD, beam = 3	**754**	**37.7**
(2) biLSTM + feats	660	33
(1) + (2)	700	35

Table 3: Human evaluation results.

The human evaluation results contrasts with those obtained in the intrinsic evaluation using the BLEU metric over references. The main reason for this contrast is due to the fact that metrics based on string distances between outputs and references penalize very lightly crucial tokens that might render the output useless. For example, in our case, having an expression like "*new in a box*" makes the title not a good product title candidate anymore. Likewise, having a wrong brand or model, renders the title useless.

A few examples can be seen in Table 4, in which the third column lists issues found by the annotators during the qualitative evaluation of the sys-

System	Output	Comments
(1)	Edifier Studio R1280T 2.0 Channel Speaker	No issues
(2)	Edifier R1280T Wired Active	Missing product type
(1) + (2)	Edifier R1280T Wired Active	Missing product type
(1)	PLAYSTATION4 Bundle Sony Console-Uncharted 4 Slim 500GB	Casing of model, order of tokens
(2)	PLAYSTATION4 Slim 500GB Console-Uncharted 4 Bundle Sony	Casing of model, order of tokens
(1) + (2)	PLAYSTATION4 Slim 500GB Console-Uncharted 4 Bundle Sony	Casing of model, order of tokens
(1)	Quell Carbon Monoxide Detector Digital Display Alarm (No Wiring (Model PD04) Operated 130415	Unnecessary tokens and segmentation
(2)	Quell Carbon Monoxide Detector & Alarm	No issues
(1) + (2)	Quell Carbon Monoxide Detector & Alarm	No issues
(1)	Pioneer N-P01-K Compact Network Audio Player-Black but 2 Lines on Display	Unwanted tokens (but 2 Lines on Display)
(2)	Pioneer N-P01-K Network Audio Player-Black	No issues
(1) + (2)	Pioneer N-P01-K Compact Network Audio Player-Black Bluetooth Lines on Display	Unwanted tokens (Bluetooth Lines on Display)

Table 4: Output examples generated by the systems evaluated in the human evaluation. Third column lists the issues in each output.

tems outputs. In the first block of outputs, for example, some titles do not present the specification of the type of the product (what is the product). The observation that BLEU alone is not appropriate for evaluating natural language generation systems is not new and corroborates previous work on the field, most notably the recent work by Reiter (2018).

Another important trend observed in Table 3 is that using the quality prediction system as a rescorer of the generated and seller titles does not improve over generation alone. We hypothesize this is due to the fact that both systems are trained separately and therefore do not leverage from the signals and features both systems explore. As future work we would like to experiment with joint training of the generation and quality prediction systems, in order to cope with this gap.

7 Conclusion

We present an approach that automatically generates e-commerce product titles out of seller-provided titles aggregated into a product. Furthermore, we devise an approach that automatically assesses the quality of a candidate product title without resorting to human references. We evaluate both approaches on a challenging real-world setting and perform quantitative and qualitative evaluation of the systems. Results show that the best generation approach is based on the stack decoder search algorithm followed by the combination of the search with the quality predictor as a re-scorer. Furthermore, both approaches presented in this work are robust enough to deal with real world user-generated data, i.e. they can produce good quality outputs even when the input data is noisy. Finally, this work sets a few interesting directions such as exploring ways of jointly training both the generation and quality prediction approach in order to improve the overall generation and quality prediction accuracy.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Loïc Barrault. 2010. Many: Open source machine translation system combination. *The Prague Bulletin of Mathematical Linguistics*, 93:147–155.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA. Association for Computational Linguistics.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *arXiv preprint arXiv:1607.01628*.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16*, pages 93–98.

Robert Dale, Stephen J Green, Maria Milosavljevic, C Ecile Paris, Cornelia Verspoor, and Sandra Williams. 1998. The Realities of Generating Natural Language from Databases. In *Proceedings of the Applications Track of the 11th Australian Joint Conference on Artificial Intelligence*, pages 62–74.

Jacob Devlin and Spyros Matsoukas. 2012. Trait-based hypothesis selection for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 528–532, Montréal, Canada. Association for Computational Linguistics.

Jinhua Du and Andy Way. 2010. Using TERp to augment the system combination for SMT. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.

Yang Feng, Yang Liu, Haitao Mi, Qun Liu, and Yajuan Lü. 2009. Lattice-based system combination for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1105–1113, Singapore. Association for Computational Linguistics.

Jonathan G Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 347–354. IEEE.

Daniel Jacob Gillick. 2011. *The elements of automatic summarization*. Ph.D. thesis, UNIVERSITY OF CALIFORNIA, BERKELEY.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *J. Artif. Int. Res.*, 57(1):345–420.

Jesús González-Rubio and Francisco Casacuberta. 2013. Improving the minimum Bayes' risk combination of machine translation systems. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*.

Jesús González-Rubio, Alfons Juan, and Francisco Casacuberta. 2011. Minimum bayes-risk system combination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1268–1277. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995*, pages 181–184.

Philipp Koehn. 2017. Neural machine translation. *arXiv preprint arXiv:1709.07809*.

Wei-Yun Ma. 2014. *Hybrid System Combination for Machine Translation: An Integration of Phrase-level and Sentence-level Combination Approaches*. Columbia University.

Wei-Yun Ma and Kathleen McKeown. 2015. System combination for machine translation through paraphrasing. In *EMNLP*, pages 1053–1058.

Prashant Mathur, Nicola Ueffing, and Gregor Leusch. 2017. Generating titles for millions of browse pages on an e-commerce site. In *The 10th International Conference on Natural Language Generation*, pages 158–167. Association for Computational Linguistics.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation for multiple machine translation systems using enhanced hypothesis alignment. In *EACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ehud Reiter. 2018. BLEU Structured Review A Structured Review of the Validity of BLEU. *Computational Linguistics*.

Antti-Veikko I Rosti, Spyridon Matsoukas, and Richard Schwartz. 2007. Improved word-level system combination for machine translation. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 312.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389. The Association for Computational Linguistics.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Hirokazu Suzuki. 2011. Automatic post-editing based on SMT and its selective application by sentence-level automatic quality evaluation. In *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, pages 156–163. International Association for Machine Translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Ye-Yi Wang and Alex Waibel. 1997. Decoding algorithm in statistical machine translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for*

Computational Linguistics, ACL '98, pages 366–372, Stroudsburg, PA, USA. Association for Computational Linguistics.