Learning to Rank Generated Portmanteaus

Lara Pollet and **Thomas Winters** and **Pieter Delobelle**
Computer Science Department
KU Leuven, Belgium
lara.pollet@student.kuleuven.be; {thomas.winters, pieter.delobelle}@kuleuven.be

Abstract

Portmanteaus are a type of neologism combining two source words, for example *brunch* (from *breakfast* and *lunch*), and are popular for naming all kinds of phenomena. While coming up with suitable portmanteaus is a difficult creative endeavor, several portmanteau generators already exist for assistance in this process. When using these systems, it is often hard to find out which of the generated portmanteau is likely to be the best, and consequently also hard to automatically compare the quality of different portmanteau generators. In this paper, we create a model that can rank portmanteaus for two given source words, which thus aims to help find the best portmanteau to help further improve portmanteau generators. Our model first uses XGBoost trained on unlabeled generated outputs and existing portmanteaus to learn to rank portmanteaus and shows that this already greatly improves the performance of the initial generator. By ranking outputs of a state-of-the-art generator and a new simple portmanteau generator, we show by validating its quality in a human evaluation that the ranker can help visually identify the better generator, thus providing an alternative to only calculating real portmanteau generation frequency. Additionally, we find that this first model performs almost as well as a model trained on more fine-grained human-labeled portmanteaus. This indicates that just using generated and real portmanteaus is enough to create a ranker that can in turn improve the quality of the initial generator, and could additionally be of use in comparing different portmanteau generators.

Introduction

A portmanteau is a type of neologism mixing two words based on their pronunciation and meaning, where the resulting word is not simply concatenating the two words into a normal compound word, for example *brunch* (which mixes *breakfast* and *lunch*). This wordplay is popular in all kinds of domains, such as naming new objects (e.g. *jeggings* for jeans leggings), pop culture (e.g. for relationship names like *Brangelina* for Brad Pitt and Angelina Jolie), animals (e.g. crossbreeds like *liger* for the child of a tiger and lion) and company names (e.g. *Netflix* for *internet flicks*). There already exist several portmanteau generators that use heuristics or neural networks for estimating the quality of portmanteaus in their generation process (Smith, Hintze, and Ventura 2014; Deri and Knight 2015; Gangal et al. 2017; Simon 2018). In this study, we focus on the task of portmanteau quality estimation, which can help further enhance the generation quality of any other portmanteau generator by sorting the outputs on their perceived quality. To achieve this, we create a portmanteau evaluator for further ranking the outputted portmanteaus. Additionally, this technique further allows comparing different portmanteau generators. More specifically, we implement a model ranking the quality of a merge of two input source words, and show that this model can be effectively trained using only generated and existing real portmanteaus.

Background

Portmanteau Generators

Nehovah is a collaborative, rule-based system for generating portmanteaus that use synonyms and hyponyms of the input source words to enrich the search space (Smith, Hintze, and Ventura 2014). The user can decide which portmanteau factors are most important. It searches for possible letter-level overlap, thus limiting its possibility to generate certain portmanteaus without letter overlap (e.g. *brunch*).

Frenemy is a more data-driven approach for generating portmanteaus (Deri and Knight 2015). It trained a multi-tape finite-state transducer to map the source words to the portmanteau using existing portmanteaus from Wikipedia and Wiktionary. By mixing the two given source words, it generates the real portmanteau 45% of the time.

Charmanteau extends the dataset used in Frenemy with Urban Dictionary and BCU Neologism List dataset (Gangal et al. 2017). This model does not use explicit grapheme to phoneme conversion but uses character embeddings to train a noisy channel model, maximizing the probability of generating the correct portmanteau given the source words. This is implemented as a neural sequence-to-sequence model using LSTMs and attention. When generating portmanteaus for two given source words, its implementation only returns the top 5 portmanteaus.

Entendrepreneur uses FastText word embeddings to find the best related source words for creating a portmanteau (Simon 2018). The focus of this system is more on finding words with good overlap, and thus much less on finding the ideal way of merging two words. As such, many existing

portmanteaus can not be generated with this system.

Learning to Rank

In a learning to rank problem, the task is to predict the order of a given set of elements. A popular system for doing so is XGBoost trees (Chen and Guestrin 2016). This algorithm uses a form of tree boosting, which iteratively extends simple decision trees with other simple decision trees to minimize the quadratic error of the model. There are three types of ranking models, namely those that predict scores for each element, compare elements pairwise, or list-wise. While the last one usually gets the best performance, they are the most difficult to model and train. In this paper, we thus focus on pairwise comparison.

Ranking Portmanteaus

To rank portmanteaus, we employ the following features, inspired by the aforementioned portmanteau generators.

- **Word structure:** proportion the source words present in the portmanteau by calculating the length of overlap between the cut-off parts of the source words and the actual portmanteau (Smith, Hintze, and Ventura 2014).
- **Source word fraction:** fraction of the length of the longest part of the source words present in the portmanteau.
- **Fraction of syllables:** number of syllables of the portmanteau divided by the number of syllables of the source words, as well as for each source word separately.
- **Memorability:** meaningful character ratio (Schiavoni et al. 2014), which is the ratio of meaningful subsequent characters that create a word present in WordNet (Miller 1995).
- **Pronounceability:** weighted frequency of letter n-grams ($n \in [2, 4]$) in the Wordlist Corpus from NLTK (Bird, Klein, and Loper 2009), since popular longer n-grams serve as proxy for pronounceable words.
- **Length of portmanteau** the absolute length of the portmanteau, and also features for the difference in length between the portmanteau and both source words separately.

These features are used to train an XGBoost model that learns to rank real portmanteaus higher than generated ones. This is done by assigning a weight of 1 to the real portmanteau, and a weight of 0 to all portmanteaus generated by the portmanteau generator for the same source words. While this assumes generated portmanteaus are worse than existing portmanteaus, it might still result in a model that can distinguish portmanteaus on their quality, as this assumption is usually true (as confirmed by our evaluation). One downside to using this way of ranking is that it might assign the same rank to multiple elements.

Simple Portmanteau Generator

We want to evaluate whether the ranker can differentiate between different portmanteau generators. However, due to the nature of the ranker and its dataset, we only want to compare their word combiner algorithm, rather than their synonym exploration capabilities. As such, comparing to generators like Entendrepreneur and Nehovah (which focus on finding good related words) would unfairly disadvantage them. While Charmanteau and Frenemy are thus the only documented models we could meaningfully compare, we were only able to gain access to the code of the former. We thus created a simple portmanteau generation algorithm, that while generating decent portmanteaus, should perform measurably worse than Charmanteau due to its lack of portmanteau quality knowledge.

We use three simple mechanisms simultaneously in the simple portmanteau generator, namely splitting on hyphenation, finding mutual letters and random splits. First, the source words are split using Pyphen[1]. If source words have letters in common, the naive generator adds combinations of these subwords split on the common letter, except if the common letter is the first letter of the first source word or the last letter of the last source word. After these generation methods, the algorithm adds five additional portmanteaus by splitting the source words into random possible substrings larger than 1, and merging randomly to create the remaining candidate portmanteaus.

Data

Real Portmanteau Dataset Extension We extended the dataset from Charmanteau with the more recently added portmanteaus from Wikipedia. We then filtered out normal compound words, portmanteaus based on proper names (like *Jedward*), fandom names (like *Cumberfan*) and chemical compounds (like *glyoxime*).

Negatives Generation For the source words of each real portmanteaus from the Wikipedia dataset, we used Charmanteau to generate five portmanteaus, which we use as negatives for training the ranking model.

Human-Annotated Labels We generated a portmanteau dataset and annotated this to evaluate the human-perceived quality of portmanteaus. We did this by first training an XGBoost ranking model on the aforementioned dataset. Then, for 700 real portmanteaus, we generated portmanteaus using Charmanteau and our simple generator, and took the top 4 of each list according to our ranker, and added the first three that were not the real portmanteau, resulting in 700 portmanteau groups of at most 7 possible portmanteaus. Each generator thus provided 2100 portmanteaus, of which there was an overlap of 331 that both generated. The annotators were 10 non-experts, recruited by sending a link to the annotation platform to willing friends of the first author. Each human annotator could annotate as many or as few portmanteau sets as they wanted. Human annotators were allowed to annotate these portmanteaus with labels denoting first, second or third place, and also annotate portmanteaus as "very bad" for portmanteaus that were fundamentally worse (e.g. due to being hard to pronounce) than all others. In case a human perceives two portmanteaus to be of equal quality, the

[1] https://pyphen.org

labels could be used multiple times within the same portmanteau group (e.g. shared first place).

Evaluation

In the evaluation, we aim to answer the following questions about our portmanteau ranker:

Q1 Is the ranker better at identifying existing portmanteaus than previous approaches?

Q2 Can the ranker improve the quality of the outputs of existing portmanteau generators?

Q3 Can the ranker evaluate the quality of two given portmanteau generators, and help identify the better one?

Q4 How well does the ranker trained on generated negative portmanteaus approximate the human-perceived quality?

Q5 Is just knowing the real or best portmanteau enough for improving generation quality? In other words, how fine-grained does the ranker training dataset need to be?

Identifying Real Portmanteaus

Due to multiple elements possibly being assigned the same rank, we break the ties in favor of the real portmanteau and in favor of the generated portmanteau to find the best and worst-case performance. We compare this with Charmanteau in Table 1. Note that Charmanteau is only able to generate 59% of the portmanteaus of our test set as one of the five candidates it generates and that we thus only use this part of the test set to more fairly compare the ranking capabilities of Charmanteau and our ranker. In the worst case, 58.63% of all portmanteau groups the right one was identified in the first position, and in the best case 72.57%. This means that either way, the ranker is better at identifying the best portmanteau given the top 5 possibilities generated by Charmanteau. Note that due to the nature of portmanteau quality in relation to its real daily use, false positives might actually still be good portmanteaus too (e.g. *sendex* instead of *sensex* for *sensitive* + *index*, and *plebevision* instead of *plebvision* for *pleb* + *television*). To answer Q1: our ranking model does indeed perform better than the state-of-the-art in identifying the real portmanteau.

Several portmanteau research projects used the real portmanteau metric to measure a generator's quality. Given that we used only Charmanteau generated portmanteaus to augment the dataset, adding the ranker to filter the generator's output would help it achieve higher scores on this quality metric on average, thus positively answering Q2.

Comparing Portmanteau Generators

We hypothesize that this ranking model could potentially be used as a form of automatic quality evaluation. Portmanteau generators are often evaluated in terms of how many times the real portmanteau can be reconstructed (Gangal et al. 2017; Deri and Knight 2015). A ranking model could automatically rank generated portmanteaus from different sources. Assuming the ranker does a good job estimating relative quality (which it does according to Table 1), the higher a generator's outputs are ranked compared to the other, the better one would expect it to be. This would then create a less accurate metric than full human evaluation for quality, but a much easier metric for comparing portmanteau generators to each other. By visually plotting how high generated portmanteaus are ranked, one can thus see which rankings are more populated by which algorithms, regardless of how many outputs are generated by each generator. This thus provides a more detailed way than only comparing real portmanteau generation frequency.

	1st	2nd	3rd	4th	5th
Charmanteau	45.70%	20.97%	15.01%	9.71%	8.61%
Ranker (worst case)	58.63%	17.26%	10.62%	7.30%	6.19%
Ranker (best case)	72.57%	12.83%	7.74%	4.65%	2.21%

Table 1: Comparing how often the true portmanteau is ranked as the best portmanteau given four other portmanteaus generated by Charmanteau using the 59.38% of the test set where Charmanteau is actually able to generate the real portmanteau. The worst and best case reflect the tie-breaking mechanism in the ranker.

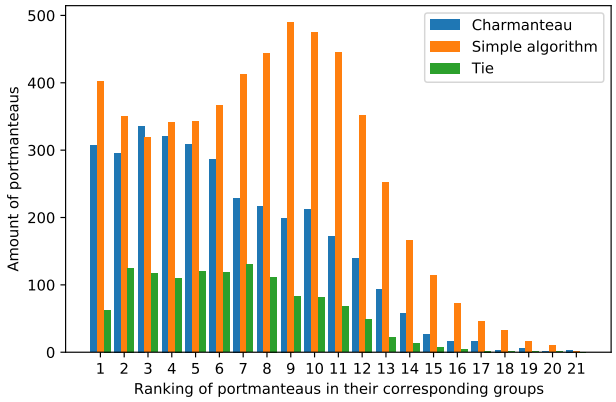

Figure 1: Comparing Charmanteau and the simple portmanteau generator automatically. Using the ranking algorithm, the list of generated portmanteaus by each generator is sorted, and then merged. As the ties can be ignored, the figure shows that Charmanteau's largest peak is earlier than the simple algorithm's, showing that Charmanteau thus indeed generates higher-ranking portmanteaus more consistently than the simple baseline.

We used our simple portmanteau generator and Charmanteau to generate portmanteaus for the source words of the test dataset and made our ranker rank all given portmanteaus. The algorithm ranked both lists separately, then merged them in a similar way as the merge sort step by comparing the first elements of each queue. If there was a tie between the highest of the two lists, then it counted as a tie, which can be ignored in the visualization. If one is better than the other, it polls the highest from the two and adds this as the current rank for this generator's element. Counting how often a generator generates elements at a certain rank, then visually shows how well their generated portmanteaus

perform compared to the other. The results in Figure 1 show what we expected, namely that Charmanteau performs better than the simple algorithm. The chart makes this clear by showing that Charmanteau peaks at the 3rd rank, and the simple generator at 1st, but also at the 9th position, and the fact that most of the mass of the simpler algorithm is ranked lower than that of Charmanteau. This is because the simple generator generates much more candidates (at least five per input source words, while Charmanteau always generates exactly five), and thus on average worse ones while also being able to generate some good portmanteaus that Charmanteau can not or did not. This is in line with Figure 1, where we can see that according to the human evaluation, Charmanteau indeed generates 59.89% of the highest-ranking portmanteaus, while the simple generator only 54.40%, and also much more of the very bad and fewer higher-ranking ones. While we used ranked versions of both generators in the human evaluation, the fact that the ranker and the human evaluation both confirm that the simple generator performs worse (as it is designed to do), helps answer Q3. This in combination with Figure 2 answers Q3: yes, the ranker can help identify and visualize the better performance of Charmanteau compared to the simple generator. This fact is a promising result for the automatic evaluation of generators, given the easy-to-create ranker training dataset.

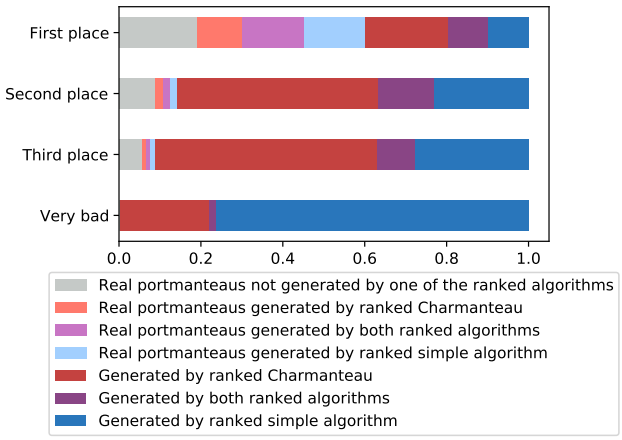

Figure 2: Human evaluation annotations for the portmanteau sources, with at most seven portmanteaus per group of source words (existing portmanteau plus at most three of each source, with potentially fewer due to both generating the same portmanteau).

Dataset Granularity & Human Evaluation

As mentioned in the Data section, we use the human annotations to create datasets for training a portmanteau ranker to evaluate the best dataset granularity and the need for human-annotated labels. Linking the resulting labels with the source of the portmanteau (Figure 2), we can see that while most of the highest-ranked portmanteaus were the real portmanteau, the generators also came up with high-ranking portmanteaus that weren't the real portmanteau. An example of a generated portmanteau that was ranked first, is *brocket* for *broken* + *bracket* (whereas the real one is *broket*. An example of a portmanteau labeled *"very bad"* for this pair was *bket*.

The human annotations were used to create three new datasets. One dataset is similar to the first ranker using only labels reflecting which portmanteau is used in reality (called "Only real"), one with only the highest-ranking ones weighted as 1, and all others as 0 (named "Only highest"), and the third dataset (called "All annotations") using weight 3 for first place, 2 for second place, 1 for third place, 0 for unlabeled and -2 for the "very bad" label. This extra information about many possible pairs would in theory help ranking portmanteaus in a more detailed way. We trained the same XGBoost ranker on our three new datasets. Table 2 shows that the ranker trained on only the real portmanteaus can still predict which one is performing best with 1.44% less accuracy than the one trained on the truly highest-ranking portmanteaus. This answers Q4, as just using real portmanteaus seems to work about as well as annotated high ranking portmanteaus. This can be partially explained by the fact that properties of real portmanteaus are closely correlated to the first place ranking portmanteaus, and that 60.03% of the real portmanteaus are also labeled best (Figure 2), thus validating our assumption for generating negative examples.

To answer Q5, we compared the last two rankers, which use a different label coarseness. We can see that adding more fine-grained annotations does not significantly improve the performance in predicting the best portmanteaus. Additionally, the higher number of 4th and lower-ranking real portmanteaus could indicate light degradation of the model for predicting the best portmanteaus.

	1st	2nd	3rd	4th	≥5th
Only real	42.58%	19.14%	12.92%	11.96%	13.40%
Only highest	44.02%	20.10%	12.44%	10.53%	12.92%
All annotations	44.98%	18.66%	9.57%	11.96%	14.83%

Table 2: Comparing how often the highest-ranking portmanteau according to human evaluation is ranked as the best portmanteau out of the seven possibilities for each portmanteau group. Ties were broken not in favor of the actual highest-ranking (worst case rank).

Code and Data

We released our code and data on `https://github.com/larapollet/portmanteau-ranker`.

Future Work

It would be interesting to validate if this approach also works for improving other types of generation, such as other types of wordplay e.g. acronyms and anagrams, or even non-textual domains. For example, by taking existing, funny anagrams, generating some other ones with a simple generator, and learning to rank anagrams, which in turn improves the average quality of the outputs the simple generator by

only outputting the best few. Another interesting further extension would be to allow for ranking portmanteaus based on semantically related source words and thus have a model ranking more diverse portmanteaus, similar to the Frenemy and Entrendrepreneur generators.

Conclusion

We created a model for automatically ranking portmanteaus. We showed that using only real portmanteaus, and generating other portmanteaus with a particular generator, can be used as a training dataset for a ranker that in turn can help improve the average quality of the outputs of that generator. This relies on the assumption that real portmanteaus are generally better than other possible portmanteaus, which we confirmed in our human evaluation. We also found that using human-annotated portmanteaus only slightly improved the quality compared to using only using labels reflecting whether or not the portmanteaus were used in real life. We also found that the largest increase in performance mostly came from only using the label for the best portmanteaus. Both these findings give rise to an optimistic view about the ease of data collection when replicating this model to improve the output quality of other types of text generators.

Acknowledgments

We would like to thank the volunteers for labeling the portmanteaus. Thomas Winters is a fellow of the Research Foundation-Flanders (FWO-Vlaanderen, 11C7720N). Pieter Delobelle was supported by the Research Foundation - Flanders (FWO) under EOS No. 30992574 (VeriLearn) and also received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

References

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Deri, A., and Knight, K. 2015. How to make a frenemy: Multitape fsts for portmanteau generation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 206–210.

Gangal, V.; Jhamtani, H.; Neubig, G.; Hovy, E.; and Nyberg, E. 2017. Charmanteau: Character embedding models for portmanteau creation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2917–2922.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Schiavoni, S.; Maggi, F.; Cavallaro, L.; and Zanero, S. 2014. Phoenix: Dga-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 192–211. Springer.

Simon, J. A. 2018. Entendrepreneur : Generating humorous portmanteaus using word-embeddings. In *Second Workshop on Machine Learning for Creativity and Design (NeurIPS 2018)*.

Smith, M.; Hintze, R. S.; and Ventura, D. 2014. Nehovah: A neologism creator nomen ipsum. In *Proceedings of the Fifth International Conference on Computational Creativity*.