CharManteau: Character Embedding Models For Portmanteau Creation

Varun Gangal [*], Harsh Jhamtani [*], Graham Neubig, Eduard Hovy, Eric Nyberg
Language Technologies Institute,
Carnegie Mellon University
{vgangal,jharsh,gneubig,hovy,ehn}@cs.cmu.edu

Abstract

Portmanteaus are a word formation phenomenon where two words are combined to form a new word. We propose character-level neural sequence-to-sequence (S2S) methods for the task of portmanteau generation that are end-to-end-trainable, language independent, and do not explicitly use additional phonetic information. We propose a noisy-channel-style model, which allows for the incorporation of unsupervised word lists, improving performance over a standard source-to-target model. This model is made possible by an exhaustive candidate generation strategy specifically enabled by the features of the portmanteau task. Experiments find our approach superior to a state-of-the-art FST-based baseline with respect to ground truth accuracy and human evaluation.

1 Introduction

Portmanteaus (or lexical blends Algeo (1977)) are novel words formed from parts of multiple root words in order to refer to a new concept which can't otherwise be expressed concisely. Portmanteaus have become frequent in modern-day social media, news reports and advertising, one popular example being *Brexit* (Britain + Exit). Petri (2012). These are found not only in English but many other languages such as Bahasa Indonesia Dardjowidjojo (1979), Modern Hebrew Bat-El (1996); Berman (1989) and Spanish Piñeros (2004). Their short length makes them ideal for headlines and brandnames (Gabler, 2015). Unlike better-defined morphological phenomenon such as inflection and derivation, portmanteau generation is difficult to capture using a set of rules. For instance, Shaw et al. (2014) state that the composition of the portmanteau from its root words depends on several factors, two important ones being maintaining prosody and retaining character segments from the root words, especially the head. An existing work by Deri and Knight (2015) aims to solve the problem of predicting portmanteau using a multi-tape FST model, which is data-driven, unlike prior approaches. Their methods rely on a grapheme to phoneme converter, which takes into account the phonetic features of the language, but may not be available or accurate for non-dictionary words, or low resource languages.

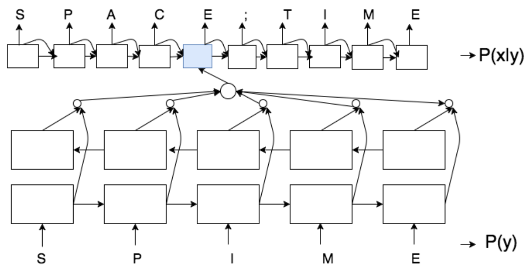

Figure 1: A sketch of our BACKWARD, noisy-channel model. The attentional S2S model with bidirectional encoder gives $P(x|y)$ and next-character model gives $P(y)$, where y (*spime*) is the portmanteau and $\boldsymbol{x} = \text{concat}(\boldsymbol{x}^{(1)}, \text{“;”}, \boldsymbol{x}^{(2)})$ are the concatenated root words (*space* and *time*).

Prior works, such as Faruqui et al. (2016), have demonstrated the efficacy of neural approaches for morphological tasks such as inflection. We hypothesize that such neural methods can (1) provide a simpler and more integrated end-to-end framework than multiple FSTs used in the previous work, and (2) automatically capture features such as phonetic similarity through the use of character embeddings, removing the need for explicit

[*] denotes equal contribution

grapheme-to-phoneme prediction. To test these hypotheses, in this paper, we propose a neural S2S model to predict portmanteaus given the two root words, specifically making 3 major contributions:

- We propose an S2S model that attends to the two input words to generate portmanteaus, and an additional improvement that leverages noisy-channel-style modelling to incorporate a language model over the vocabulary of words (§2).
- Instead of using the model to directly predict output character-by-character, we use the features of portmanteaus to exhaustively generate candidates, making scoring using the noisy channel model possible (§3).
- We curate and share a new and larger dataset of 1624 portmanteaus (§4).

In experiments (§5), our model performs better than the baseline Deri and Knight (2015) on both objective and subjective measures, demonstrating that such methods can be used effectively in a morphological task.

2 Proposed Models

This section describes our neural models.

2.1 Forward Architecture

Under our first proposed architecture, the input sequence $\boldsymbol{x} = \text{concat}(\boldsymbol{x}^{(1)}, \text{``;''}, \boldsymbol{x}^{(2)})$, while the output sequence is the portmanteau $\boldsymbol{y}$. The model learns the distribution $P(\boldsymbol{y}|\boldsymbol{x})$.

The network architecture we use is an attentional S2S model (Bahdanau et al., 2014). We use a bidirectional encoder, which is known to work well for S2S problems with similar token order, which is true in our case. Let $\overrightarrow{LSTM}$ and $\overleftarrow{LSTM}$ represent the forward and reverse encoder; $e_{enc}()$ and $e_{dec}()$ represent the character embedding functions used by encoder and decoder The following equations describe the model:

$$\begin{aligned}
h_0^{\overrightarrow{enc}} &= \overrightarrow{0}, h_{|x|}^{\overleftarrow{enc}} = \overrightarrow{0} \\
h_t^{\overrightarrow{enc}} &= \overrightarrow{LSTM}(h_{t-1}^{enc}, e_{enc}(x_t)) \\
h_t^{\overleftarrow{enc}} &= \overleftarrow{LSTM}(h_{t+1}^{enc}, e_{enc}(x_t)) \\
h_t^{enc} &= h_t^{\overrightarrow{enc}} + h_t^{\overleftarrow{enc}} \\
h_0^{dec} &= h_{|x|}^{enc} \\
h_t^{dec} &= LSTM(h_{t-1}^{dec}, [\text{concat}(e_{dec}(y_{t-1}), c_{t-1})]) \\
p_t &= \text{softmax}(W_{hs}[\text{concat}(h_t^{dec}, c_t)] + b_s)
\end{aligned}$$

The context vector c_t is computed using dot-product attention over encoder states. We choose dot-product attention because it doesn't add extra parameters, which is important in a low-data scenario such as portmanteau generation.

$$\begin{aligned}
a_i^t &= dot(h_t^{dec}, h_i^{enc}), \alpha^t = \text{softmax}(a^t) \\
c_t &= \sum_{i=1}^{i=|x|} \alpha_i^t h_i^{enc}
\end{aligned}$$

In addition to capturing the fact that portmanteaus of two English words typically sound *English-like*, and to compensate for the fact that available portmanteau data will be small, we pretrain the character embeddings on English language words. We use character embeddings learnt using an LSTM language model over words in an English dictionary,[1] where each word is a sequence of characters, and the model will predict next character in sequence conditioned on previous characters in the sequence.

2.2 Backward Architecture

The second proposed model uses Bayes's rule to reverse the probabilities $P(\boldsymbol{y}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})}{P(\boldsymbol{x})}$ to get $\text{argmax}_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) = \text{argmax}_{\boldsymbol{y}} P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})$. Thus, we have a reverse model of the probability $P(\boldsymbol{x}|\boldsymbol{y})$ that the given root words were generated from the portmanteau and a character language model model $P(\boldsymbol{y})$. This is a probability distribution over all character sequences $y \in A^*$, where A is the alphabet of the language. This way of factorizing the probability is also known as a *noisy channel model*, which has recently also been shown to be effective for neural MT (Hoang et al. (2017), Yu et al. (2016)). Such a model offers two advantages

1. The reverse direction model (or alignment model) gives higher probability to those portmanteaus from which one can discern the root words easily, which is one feature of good portmanteaus.
2. The character language model $P(\boldsymbol{y})$ can be trained on a large vocabulary of words in the language. The likelihood of a word y is factorized as $P(y) = \Pi_{i=1}^{i=|y|} P(y_i|y_1^{i-1})$, where $y_j^i = y_i, y_{i+1} \dots y_j$, and we train a LSTM to maximize this likelihood.

[1] Specifically in our experiments, 134K words from the CMU dictionary (Weide, 1998).

3 Making Predictions

Given these models, we must make predictions, which we do by two methods

Greedy Decoding: In most neural sequence-to-sequence models, we perform auto-regressive greedy decoding, selecting the next character greedily based on the probability distribution for the next character at current time step. We refer to this decoding strategy as GREEDY.

Exhaustive Generation: Many portmanteaus were observed to be concatenation of a prefix of the first word and a suffix of the second. We therefore generate all candidate outputs which follow this rule. Thereafter we score these candidates with the decoder and output the one with the maximum score. We refer to this decoding strategy as SCORE.

Given that our training data is small in size, we expect ensembling (Breiman, 1996) to help reduce model variance and improve performance. In this paper, we ensemble our models wherever mentioned by training multiple models on 80% subsamples of the training data, and averaging log probability scores across the ensemble at test-time.

4 Dataset

The existing dataset by Deri and Knight (2015) contains 401 portmanteau examples from Wikipedia. We refer to this dataset as D_{Wiki}. Besides being small for detailed evaluation, D_{Wiki} is biased by being from just one source. We manually collect D_{Large}, a dataset of 1624 distinct English portmanteaus from following sources:

- Urban Dictionary[2]
- Wikipedia
- Wiktionary
- BCU's Neologism Lists from '94 to '12.

Naturally, $D_{Wiki} \subset D_{Large}$. We define $D_{Blind} = D_{Large} - D_{Wiki}$ as the dataset of 1223 examples not from Wikipedia. We observed that 84.7% of the words in D_{Large} can be generated by concatenating prefix of first word with a suffix of the second.

[2]Not all neologisms are portmanteaus, so we manually choose those which are for our dataset.

Model	Attn	Ens	Init	Search	Matches	Distance
BASELINE	-	-	-	-	**45.39%**	1.59
FORWARD	✓	×	×	GREEDY	22.00%	1.98
	✓	×	✓	GREEDY	28.00%	1.90
	✓	×	×	BEAM	13.25%	2.47
	✓	×	✓	BEAM	15.25%	2.37
	✓	×	×	SCORE	30.25%	1.64
	✓	×	✓	SCORE	32.88%	1.53
	✓	✓	✓	SCORE	42.25%	1.33
	✓	✓	×	SCORE	41.25%	1.34
	×	×	✓	SCORE	6.75%	3.78
	×	×	×	SCORE	6.50%	3.76
BACKWARD	✓	×	×	SCORE	37.00%	1.53
	✓	×	✓	SCORE	42.25%	1.35
	✓	✓	✓	SCORE	**48.75%**	**1.12**
	✓	✓	×	SCORE	**46.50%**	**1.24**
	×	×	✓	SCORE	5.00%	3.95
	×	×	×	SCORE	4.75%	3.98

Table 1: 10-Fold Cross-Validation results, D_{Wiki}. *Attn, Ens, Init* denote attention, ensembling, and initializing character embeddings respectively.

5 Experiments

In this section, we show results comparing various configurations of our model to the baseline FST model of Deri and Knight (2015) (BASELINE). Models are evaluated using exact-matches (*Matches*) and average Levenshtein edit-distance (*Distance*) w.r.t ground truth.

5.1 Objective Evaluation Results

In *Experiment 1*, we follow the same setup as Deri and Knight (2015). D_{Wiki} is split into 10 folds. Each fold model uses 8 folds for training, 1 for validation, and 1 for test. The average (10 fold cross-validation style approach) performance metrics on the test fold are then evaluated. *Table 1* shows the results of *Experiment 1* for various model configurations. We get the BASELINE numbers from Deri and Knight (2015). Our best model obtains 48.75% *Matches* and 1.12 *Distance*, compared to 45.39% *Matches* and 1.59 *Distance* using BASELINE.

For *Experiment 2*, we seek to compare our best approaches from *Experiment 1* to the BASELINE on a large, held-out dataset. Each model is trained on D_{Wiki} and tested on D_{Blind}. BASELINE was similarly trained only on D_{Wiki} , making it a fair comparison. Table 2 shows the results[3]. Our best model gets *Distance* of 1.96 as compared to 2.32 from BASELINE.

We observe that the *Backward* architecture performs better than *Forward* architecture, confirming our hypothesis in §2.2. In addition, ablation results confirm the importance of attention, and

[3]For BASELINE (Deri and Knight, 2015), we use their trained model from `http://leps.isi.edu/fst/step-all.php`

Model	Attn	Ens	Init	Search	Matches	Distance
BASELINE	-	-	-	-	**31.56%**	2.32
FORWARD	✓	×	✓	SCORE	25.26%	2.13
	✓	×	×	SCORE	24.93%	2.32
	✓	✓	×	SCORE	31.23%	1.98
	✓	✓	✓	SCORE	28.94%	2.04
BACKWARD	✓	×	✓	SCORE	25.75%	2.14
	✓	×	×	SCORE	25.26%	2.17
	✓	✓	×	SCORE	**31.72%**	**1.96**
	✓	✓	✓	SCORE	**32.78%**	**1.96**

Table 2: Results on D_{Blind} (1223 Examples). In general, BACKWARD architecture performs better than FORWARD architecture.

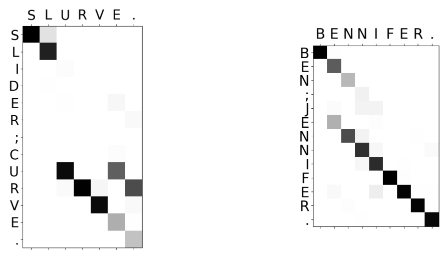

Figure 2: Attention matrices while generating *slurve* from *slider;curve*, and *bennifer* from *ben;jennifer* respectively, using *Forward* model. **;** and **.** are separator and stop characters. Darker cells are higher-valued

initializing the word embeddings. We believe this is because portmanteaus have high fidelity towards their root word characters and its critical that the model can observe all root sequence characters, which attention manages to do as shown in Fig. 2.

5.1.1 Performance on Uncovered Examples

The set of candidates generated before scoring in the approximate SCORE decoding approach sometimes do not cover the ground truth. This holds true for 229 out of 1223 examples in D_{Blind}. We compare the FORWARD approach along with a GREEDY decoding strategy to the BASELINE approach for these examples.

Both FORWARD+GREEDY and the BASELINE get 0 *Matches* on these examples. The *Distance* for these examples is 4.52 for BASELINE and 4.09 for FORWARD+GREEDY. Hence, we see that one of our approaches (FORWARD+GREEDY) outperforms BASELINE even for these examples.

5.2 Significance Tests

Since our dataset is still small relatively small (1223 examples), it is essential to verify whether BACKWARD is indeed statistically significantly better than BASELINE in terms of *Matches*.

Input	FORWARD	BACKWARD	GROUND TRUTH
shopping;marathon	shopparathon	shoathon	shopathon
fashion;fascism	fashism	fashism	fashism
wiki;etiquette	wikiquette	wiquette	wikiquette
clown;president	clowident	clownsident	clownsident

Table 3: Example outputs from different models (Refer to appendix for more examples)

Judgement	Percentage of total
Much Better (1)	29.06
Better (2)	29.06
Worse (3)	25.11
Much Worse (4)	16.74

Table 4: AMT annotator judgements on whether our system's proposed portmanteau is better or worse compared to the baseline

In order to do this, we use a paired bootstrap[4] comparison (Koehn, 2004) between BACKWARD and BASELINE in terms of *Matches*. BACKWARD is found to be better (gets more *Matches*) than BASELINE in 99.9% ($p = 0.999$) of the subsets.

Similarly, BACKWARD has a lower *Distance* than BASELINE by a margin of 0.2 in 99.5% ($p = 0.995$) of the subsets.

5.3 Subjective Evaluation and Analysis

On inspecting outputs, we observed that often output from our system seemed good in spite of high edit distance from ground truth. Such aspect of an output *seeming good* is not captured satisfactorily by measures like edit distance. To compare the errors made by our model to the baseline, we designed and conducted a human evaluation task on AMT.[5] In the survey, we show human annotators outputs from our system and that of the baseline. We ask them to judge which alternative is *better* overall based on following criteria: 1. It is a good shorthand for two original words 2. It sounds better. We requested annotation on a scale of 1-4. To avoid ordering bias, we shuffled the order of two portmanteau between our system and that of baseline. We restrict annotators to be from Anglophone countries, have HIT Approval Rate $> 80\%$ and pay 0.40$ per HIT (5 Questions per HIT).

As seen in Table 4, output from our system was labelled better by humans as compared to the baseline 58.12% of the time. Table 3 shows outputs from different models for a few examples.

[4] We average across $M = 1000$ randomly chosen subsets of D_{Blind}, each of size $N = 611$ ($\approx 1223/2$)

[5] We avoid ground truth comparison because annotators can be biased to ground truth due to its existing popularity.

6 Related Work

Özbal and Strapparava (2012) generate new words to describe a product given its category and properties. However, their method is limited to hand-crafted rules as compared to our data driven approach. Also, their focus is on brand names. Hiranandani et al. (2017) have proposed an approach to recommend brand names based on brand/product description. However, they consider only a limited number of features like memorability and readability. Smith et al. (2014) devise an approach to generate portmanteaus, which requires user-defined weights for attributes like *sounding good*. Generating a portmanteau from two root words can be viewed as a S2S problem. Recently, neural approaches have been used for S2S problems (Sutskever et al., 2014) such as MT. Ling et al. (2015) and Chung et al. (2016) have shown that character-level neural sequence models work as well as word-level ones for language modelling and MT. Zoph and Knight (2016) propose S2S models for multi-source MT, which have multi-sequence inputs, similar to our case.

7 Conclusion

We have proposed an end-to-end neural system to model portmanteau generation. Our experiments show the efficacy of proposed system in predicting portmanteaus given the root words. We conclude that pre-training character embeddings on the English vocabulary helps the model. Through human evaluation we show that our model's predictions are superior to the baseline. We have also released our dataset and code[6] to encourage further research on the phenomenon of portmanteaus. We also release an online demo [7] where our trained model can be queried for portmanteau suggestions. An obvious extension to our work is to try similar models on multiple languages.

Acknowledgements

We thank Dongyeop Kang, David Mortensen, Qinlan Shen and anonymous reviewers for their valuable comments. This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

[6]https://github.com/vgtomahawk/Charmanteau-CamReady

[7]http://tinyurl.com/y9x6mvy

References

John Algeo. 1977. Blends, a structural and systemic view. *American speech*, 52(1/2):47–64.

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

Outi Bat-El. 1996. Selecting the best of the worst: the grammar of Hebrew blends. *Phonology*, 13(03):283–328.

Ruth Berman. 1989. The role of blends in Modern Hebrew word-formation. *Studia linguistica et orientalia memoriae Haim Blanc dedicata. Wiesbaden: Harrassowitz*, pages 45–61.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv:1603.06147*.

Soenjono Dardjowidjojo. 1979. Acronymic Patterns in Indonesian. *Pacific Linguistics Series C*, 45:143–160.

Aliya Deri and Kevin Knight. 2015. How to make a frenemy: Multitape FSTs for portmanteau generation. In *Proceedings of NAACL-HLT*, pages 206–210.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological Inflection Generation using Character Sequence to Sequence Learning. In *Proceedings of NAACL-HLT*, pages 634–643.

Neal Gabler. 2015. The Weird Science of Naming New Products. *New York Times - http://tinyurl.com/lmlq7ex*.

Gaurush Hiranandani, Pranav Maneriker, and Harsh Jhamtani. 2017. Generating appealing brand names. *arXiv preprint arXiv:1706.09335*.

Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. 2017. Decoding as Continuous Optimization in Neural Machine Translation. *arXiv:1701.02854*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv:1511.04586*.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of ACL*, pages 703–711. Association for Computational Linguistics.

Alexandra Petri. 2012. Say No to Portmanteaus. *Washington Post* - `http://tinyurl.com/kvmep2t`.

Carlos-Eduardo Piñeros. 2004. The creation of portmanteaus in the extragrammatical morphology of Spanish. *Probus*, 16(2):203–240.

Katherine E Shaw, Andrew M White, Elliott Moreton, and Fabian Monrose. 2014. Emergent faithfulness to morphological and semantic heads in lexical blends. In *Proceedings of the Annual Meetings on Phonology*, volume 1.

Michael R Smith, Ryan S Hintze, and Dan Ventura. 2014. Nehovah: A neologism creator nomen ipsum. In *Proceedings of the International Conference on Computational Creativity*, pages 173–181.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Neural information processing systems*, pages 3104–3112.

R Weide. 1998. The CMU pronunciation dictionary, release 0.6. *Carnegie Mellon University*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2016. The Neural Noisy Channel. *arXiv:1611.02554*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv:1601.00710*.