CharManteau: Character Embedding Models For Portmanteau Creation

Anonymous EMNLP submission

Abstract

Portmanteaus are a morphological word formation phenomenon where two or more words are combined into a new word. We propose character-level neural sequence-to-sequence methods for learning the task of portmanteau generation. Our end-to-end embedding-based neural approaches outperform a prior state-of-the-art FST-based baseline with respect to ground truth, besides being judged superior in human evaluations. To enable further research on this problem, we will release our dataset of 1642 portmanteaus, which is almost four times larger than the previously existing dataset.

1 Introduction

Portmanteaus [1] [2] (also referred to as blends (Algeo, 1977)) are found not just in English, but several widely-spoken languages such as Bahasa Indonesia (Dardjowidjojo, 1979), Russian (Shaw et al., 2014), Farsi (Megerdoomian and Hadjarian, 2010), Modern Hebrew (Bat-El, 1996; Berman, 1989) and Spanish (Piñeros, 2004). Portmanteaus have become quite frequent in modern-day social media conversations, news reports and advertising, with portmanteaus such as *Brexit* (Britain + Exit), *Bregret* (Britain + Regret), *Trumponomics* (Trump + Economics) and *Feminazi* (Feminist + Nazi) garnering widespread popularity [3]. Their shorter length makes them ideal for headlines, hashtags and adphrases. They also aid in the increasingly difficult task of finding a novel, unclaimed brand-name [4], since most of the vocabulary words are already claimed.

Unlike other, more well-defined morphological phenomenon such as inflection (tense and number changes), portmanteau generation is difficult to capture using a set of rules. For instance, (Shaw et al., 2014) note that the similarity of the portmanteau with respect to its root words (*"position faithfulness"*) depends on several factors such as prosody of resultant portmanteau (How good it sounds) and semantic head of the portmanteau (Which word is the head and which one is the modifier?).

This paper makes following contributions.

1. We curate and share a novel dataset of 1602 portmanteaus from multiple , disparate online sources such as the RICE neologisms Database, Urban Dictionary, Wikipedia and Wiktionary. Our dataset is a superset of the dataset of (Deri and Knight, 2015), which contains 402 examples from Wikipedia.
2. Unlike (Deri and Knight, 2015), our approaches do not make use of external resources/lexicons/G2P converters such as the CMU Pronunciation Dictionary to get additional information such as phonetics or syllables. This makes our model generalizable to languages (as well as root words, which may themselves be slang or out-of-dictionary) where such resources are unavailable, or which have different amounts of alphabet-phoneme correspondence.
3. Most importantly, our experiments show

[1] https://en.wikipedia.org/wiki/Portmanteau

[2] Portmanteau is itself formed from French *porter*(carry) and *manteau* (cloak)

[3] https://www.washingtonpost.com/blogs/compost/post/just-say-no-to-portmanteaus/2012/06/21/gJQAQfqetV_blog.html?utm_term=.cf38a00023f3

[4] https://www.nytimes.com/2015/01/18/magazine/the-weird-science-of-naming-new-products.html?_r=0

that character-embedding based neural approaches perform better than a FST-based baseline, thereby demonstrating that such methods can be used effectively in a morphological task.

4. Our approaches are end-to-end trainable, unlike (Deri and Knight, 2015), which has 5 FSTs trained and tuned separately.
5. We perform and share the results of a human annotated study on AMT, where we compare the outputs of our neural methods and the baseline system of (Deri and Knight, 2015) in the case where both are different from the ground truth.

2 Related Work

The work closest to ours is by Deri and Knight (2015). Their work is on the same problem of predicting portmanteau given two words. They propose a multi-tape FST model. In contrast, we uses a neural sequence to sequence model which can be trained end to end. Moreover, we work with a much bigger data set than proposed by them.

Ozbal and Strapparava (2012) aim to generate new words to describe a product given the category of the product to be advertised and the properties to be emphasized. However, their method is limited to manually hand-crafted rules as compared to our data driven approach. Also their focus is on names for products and brands, while ours is a more generic case.

Generating portmanteau from two given words can be viewed as a sequence-to-sequence class of problems. Recently, neural approaches have been used in solving sequence-to-sequence problems (Sutskever et al., 2014) such as Machine Translation.

Prior works, such as (Faruqui et al., 2016), have demonstrated the efficacy of neural approaches for morphological tasks such as inflection. However, our problem is more challenging due the fact that we have to work with lesser training data (unlike inflection tables in their case).

3 Dataset

We curate and share a dataset of 1642 distinct portmanteau examples in English language. These examples were manually collected from multiple online sources, a non-exhaustive list of which is mentioned below:

- Urban Dictionary [5]
- Wiktionary
- Wikipedia
- Rice University's Neologism Database [6], created by Kemmer et al (Kemmer, 2003)
- American Dialect Society's shortlists for Word of The Year
- Birmingham City University's Neologism Lists [7] from 1994 to 2012. This list has itself been curated from neologisms appearing in British newspapers such as *The Guardian*.

Thereafter we removed the duplicates and those examples in which 3 or more words were joined to generate a new word. Average length (number of characters) for first word across the data set is 6.41 with variance of 6.51, while for second word is 7.32 with variance of 4.98.

We observed that many of the words in the set can be generated possibly by concatenating prefix of first word with a suffix of the second word. Across the entire dataset, we found that 84.7% percentage of the data points fall in this category.

4 Methodology

4.1 Description of neural model

4.1.1 Forward Architecture

This is a sequence to sequence attentional model. The input sequence is the root words $x; y$, while the output sequence is the portmanteau z. This model directly attempts to learn the distribution $P(z|x; y)$. The network architecture we use is an attentional sequence-to-sequence models, where both the encoder and decoder are LSTMs (Hochreiter and Schmidhuber, 1997). We use a bidirectional encoder (Bahdanau et al., 2014), which is known to work well for sequence-to-sequence problems with similar token order, which is roughly true in our case. **TODO:Rationale for Bidirectional**.

4.1.2 Backward Architecture

This sequence-to-sequence model tries to learn a model in the reverse direction, i.e the probability $P(x; y|z)$ that the given parent words were generated from the portmanteau. Note that, to

[5] http://www.urbandictionary.com/

[6] http://neologisms.rice.edu/index.php?a=index&d=1

[7] http://rdues.bcu.ac.uk/neologisms.shtml

use this model for giving the probability score, we need to couple it with a character-sequence model $P(z)$, which is a probability distribution over all character sequences $z \in C^*$, where C is the character set of the language. We can then use the Bayes rule $P(z|x,y) = \frac{P(x;y|z)P(z)}{P(x;y)}$ to get $argmax_z P(z|x;y) = argmax_z P(x;y|z)P(z)$. This model has two advantages

1. The reverse direction model (or alignment model) gives higher probability to those portmanteaus from which one can discern the parent words easily.

2. Analogous to a language model which assigns a probability to every sentence or word sequence, we have a "character-sequence" model which assigns a probability to any word/character sequence. The character-sequence model $P(z)$ can be trained on the vocabulary of the language using a next-character prediction cross-entropy loss. The likelihood of a word (character sequence) z is factorized as $P(z) = \Pi_{i=1}^{i=|z|} P(z_i|z_1^{i-1})$, where z_j^i is the i-j subsequence of z

4.2 The Importance Of Attention

Since the portmanteau has high fidelity in terms of characters towards its parent words, it is critical that it is able to observe all the characters in the parent sequence. This makes it very important to have an attention based model, since just passing on the encoder state is insufficent for the decoder to access the exact input sequence. We use dot product attention since it does not add any parameters to our model. From the results in table 2, we can observe that the plain encoder-decoder (attention-less) models performs very poorly.

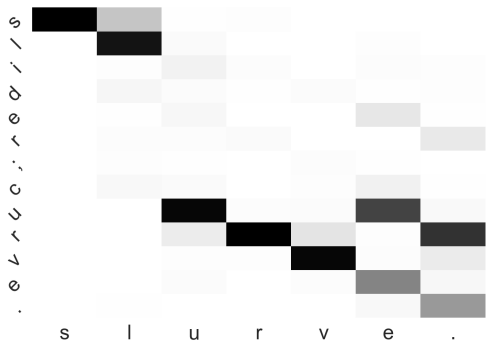

Figure 1: Attention matrix while correctly generating *slurve* from *slider* and *curve* using the forward model. *;* and . are the separator and stop characters respectively

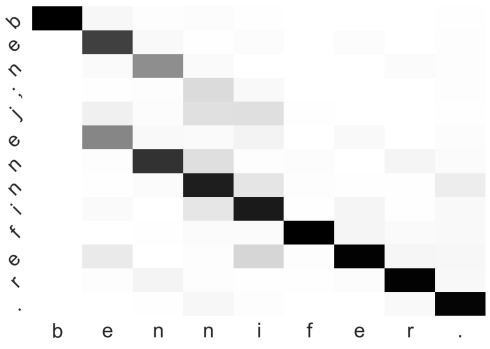

Figure 2: Attention matrix while correctly generating *bennifer* from *ben* and *jennifer* using the forward model. *;* and . are the separator and stop characters respectively

4.3 Initialization of Character Embeddings

Since portmanteau of two English words typically sounds *English-like*, pre-trained character embeddings on English language words can be useful. This also helps to mitigate learning character embeddings based on our rather limited dataset. We use character embeddings learnt on using a LSTM recurrent neural model on words from CMU dictionary (Weide, 1998). Each word is a sequence of characters, and the model will predict next character in sequence conditioned on previous characters in the sequence. The dictionary contains 134,000 thousand words. The objective function used was cross-entropy loss on probability of next predicted character.

Model	Ensemble	Pretraining	Exact Matches	Edit Distance
Knight	-	-	**45.39%**	1.59
Forward+Greedy	No	No	22.00	1.98
Forward+Greedy	No	Yes	28.00	1.905
Forward+Beam2	No	No	13.25	2.47
Forward+Beam2	No	Yes	15.25	2.37
Forward-Attention	No	Yes	6.75	3.78
Forward-Attention	No	No	6.50	3.76
Reverse-Attention	No	Yes	5.00	3.95
Reverse-Attention	No	No	4.75	3.98
Forward	No	No	30.25%	1.64
Forward	No	Yes	32.88%	1.53
Forward	Yes	Yes	42.25%	1.335
Forward	Yes	No	41.25%	1.34
Reverse	No	No	37.00%	1.53
Reverse	No	Yes	42.25%	1.35
Reverse	Yes	Yes	**48.75%**	**1.12**
Reverse	Yes	No	**46.5%**	**1.24**

Table 1: Results on 10-Fold Cross-Validated[9] Test Knight Data

Model	Ensemble	Pretraining	Exact Matches	Edit Distance
Knight			**386**	2.32
Forward	No	Yes	309	2.13
Reverse	No	Yes	315	2.14
Forward	No	No	305	2.32
Reverse	No	No	309	2.17
Forward	Yes	No	382	1.98
Forward	Yes	Yes	354	2.04
Reverse	Yes	No	**388**	**1.96**
Reverse	Yes	Yes	**401**	**1.96**

Table 2: Results on Held-Out Non-Knight Data (1223 Examples)

4.4 Decoding Mechanisms

4.4.1 Greedy

4.4.2 Exhaustive Generation

As mentioned earlier, many of the portmanteau were observed to be concatenation of prefix of first word and suffix of last word. We therefore generate candidate outputs which follow such a rule. Thereafter we score these candidates with the decoder and output the one with the maximum score.

4.5 Ensembling

Given that our training data is quite small in size, we expect ensembling (Breiman, 1996) (especially bagging-like approaches) to help reduce model variance and improve performance. We train an ensemble of models, each with a random 80-20 split of the training set, with the first 80% used for training the network and the remaining 20% used for early stopping. The models are ensembled by averaging their log probability for each candidate, in the generate-and-rank mechanism at test time.

Note that the optimal ensemble size and training sizes are all found based on the validation folds performance in the cross-validation experiment on the Knight dataset.

5 Experiments

5.1 Human Annotation Experiments

Upon inspection of outputs, we observed that in many cases output from our system seemed good in spite of high edit distance with respect to ground truth. Such aspect of an output *seeming good* is not captured satisfactorily by measures like edit distance. To investigate and compare the errors made by our model and that of the baseline system, we designed and conducted a human evaluation task on Amazon Mechanical Turk [10].

In the survey we ask showed human annotators outputs from our system and that of baseline. We ask them to judge which one is better where *better* means the word 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 (see Figure for example). To avoid any ordering bias, we randomly shuffled the order of two portmanteau between our system and that of baseline. Additionally, with every HIT, one of the questions had two options as gold truth and a very poorly scored candidate. This was included to have a check if the annotator have understood the task, and has worked diligently. If the annotator selects the non-gold option as better, then we do not consider ratings by him.

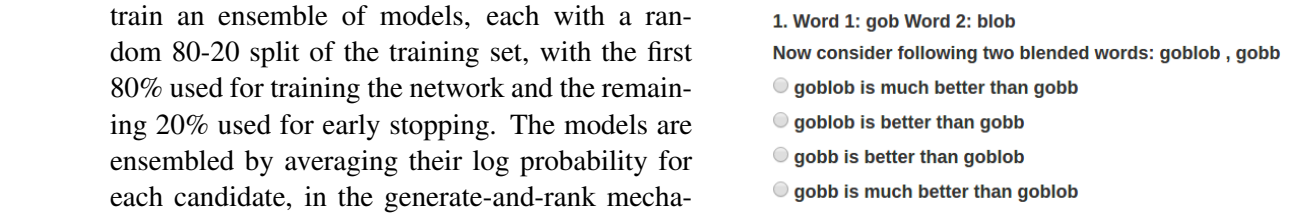

Figure 3: Example question in a HIT (Human Intelligence Task). Every HIT had 6 such questions.

As can be seen in Table 3, output from our system was labelled much better by humans as compared to the baseline system.

6 Conclusion

We have proposed an end-to-end neural system to model portmanteau generation.Our experiments show the efficacy of proposed system in predicting portmanteau given the parent words. Our methods

[10] https://requester.mturk.com/

Judgement	Fraction
Much Better (1)	72
Better (2)	108
Worse (3)	125
Much Worse (4)	125

Table 3: AMT annotator judgements on whether our system's proposed portmanteau is better or worse compared to the baseline

are language independent, and learn from existing list of portmanteaus. We conclude that pretraining character embeddings on dictionary of known English words helps the model. Through human evaluation we show that predictions by our model are much better than that of the baseline system. We will release our datatset and code to encourage further research on the phenomenon of portmanteaus. An obvious extension to our work is to try similar models on a variety of different languages.

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 preprint 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.

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 *HLT-NAACL*. 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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Suzanne Kemmer. 2003. Schemas and lexical blends. *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE SERIES 4* pages 69–98.

Karine Megerdoomian and Ali Hadjarian. 2010. Mining and classification of neologisms in persian blogs. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*. Association for Computational Linguistics, pages 6–13.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 703–711.

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.

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

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