

---

# Entendpreneur: Generating Humorous Portmanteaus using Word-Embeddings

---

**Jonathan A. Simon**  
Fin Exploration Company  
jonathan@finxpc.com

## Abstract

We present a novel algorithm for generating portmanteaus which utilizes word embeddings to identify semantically related words for use in the portmanteau construction. By simultaneously optimizing for phonetic closeness and phonetic novelty, this algorithm significantly outperforms related techniques on a subjective assessment of humor.

## 1 Introduction & Related Work

Pun generation systems have long been a topic of research interest, with applications including helping autistic children develop their language skills [1,2], generating names for companies [3], and simple entertainment [4,15]. Portmanteaus, a specific type of morphological pun, have received a large amount of focus in particular [5,6,7]. In prior work much of the emphasis was placed on constructing alignments between the pair of input strings. Deri and Knight (2015) use a multitape FST to generate these alignments, and Gangal, et al. (2017) use an LSTM neural network. While this works well in cases where there exists a natural alignment between two words (hungry/angry : hangry), for many words there exists no unambiguously good alignment (drunk/angry : drungry). Especially when the portmanteau is intended to be humorous, these imperfect matches ruin the overall effect. However a much larger set of portmanteaus can be constructed if we allow semantically related words to be used (drunk/angry : beeritable). Smith, et al.'s Nehovah (2014) system performs synonym lookups using WordNet [8], however in most cases synonym-matching is too constraining for this task. To the author's knowledge, the Entendpreneur system (entendre/entrepreneur) is the first published portmanteau generator to make use of word embeddings, with the resulting portmanteaus being subjectively more humorous than those produced by other systems.

## 2 Definitions

We define a *graph* to be a single English character, and a *phone* to be a single character from the ARPABET phonetic alphabet [14]. We define a *grapheme* to be a string of graphs, and a *phoneme* to be a string of phones. We define a *word* to be an aligned grapheme/phoneme pair. We define a *portmanteau* (PM) to be set of two words, together with a composite *phoneme portmanteau* ( $PM_{\text{phone}}$ ) and *grapheme portmanteau* ( $PM_{\text{graph}}$ ), constructed from the associated words' phonemes and graphemes respectively. See Supplementary Figure 1 for details.

## 3 Algorithm Overview

Given two input graphemes  $g^0, h^0$  the Entendpreneur algorithm performs the following steps:

1. Identify semantically-related graphemes  $\{g^1, \dots, g^m\}, \{h^1, \dots, h^m\}$  by performing nearest-neighbor lookups in the FastText word vector embedding space [9].

Table 1: Survey results

	Algorithm Name		Algorithm % Selected			
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2	n	p-value
Quality	Entrendrepreneur	Charmanteau	38.2%	61.8%	55	0.10
	Entrendrepreneur	Port Manteaux	48.1%	51.9%	54	0.45
Humor	Entrendrepreneur	Charmanteau	<b>64.4%</b>	35.6%	73	0.009
	Entrendrepreneur	Port Manteaux	<b>63.8%</b>	36.2 %	47	0.079

- Remove irregular capitalizations and tenses, and deduplicate the results, yielding new grapheme sets  $\{g^1, \dots, g^{n'}\}, \{h^1, \dots, h^{m'}\}$ .
- Map each grapheme  $g, h$  to its corresponding phoneme  $p, q$  using the CMU Pronouncing Dictionary [13], and align the constituent graphs and phones using an EM-based alignment algorithm [16].
- For each pair of phonemes  $p, q$  align the tail of  $p$  with the head of  $q$ , finding the alignment which minimizes the sum of the pairwise aligned phone distances.
- Two different values of  $PM_{\text{graph}}$  are consistent with the phoneme alignment. We choose the one which maximizes the likelihood that the reader will be able to correctly reconstruct  $g, h$  given  $PM_{\text{graph}}$ .
- The generated PMs are ordered according to the phonetic distance of the aligned phonemes  $d(p_{\text{overlap}}, q_{\text{overlap}})$  as well as the novelty of these phonemes  $p(p_{\text{overlap}}, q_{\text{overlap}})$ .

Steps (3-5) are also performed on the reverse-ordered graphemes  $h^0, g^0$ . See Supplementary Material for additional details.

#### 4 Assessing Portmanteau Humor and Quality

To assess the subjective quality and humor of the generated portmanteaus, questionnaires were assembled comparing the portmanteaus produced by Entrendrepreneur on random grapheme pairs to those produced by two other algorithms, Charmanteau [7] and Port Manteaux [15] (Supplementary Table 3). Twenty questionnaires containing a total of 200 different word pairs were generated, with each questionnaire given to one of 33 volunteers. The portmanteaus produced by Entrendrepreneur were judged to be significantly funnier than those produced by Charmanteau (two-tailed binomial  $p=0.009$ ,  $n=73$ ) and non-significantly funnier than those produced by Port Manteaux (two-tailed binomial  $p=0.079$ ,  $n=47$ ). The quality of the portmanteaus was not found to differ significantly between the algorithms (Table 1). See Supplementary Section 6.8 for details.

#### 5 Conclusions & Future Work

Using semantically related words in the construction of portmanteaus was found to produce significantly more humorous results. In particular, using cosine similarity in the FastText word vector embedding space as a measure of semantic relatedness was found to outperform related portmanteau generation methods in terms of humor, without any significant accompanying decrease in portmanteau quality. Possible extensions of this work include employing a more sophisticated phoneme alignment algorithm, and altering the result sorting criterion to include information related to the semantic distance between the portmanteau graphemes and the input graphemes.

#### References

- [1] Manurung, R., Ritchie, G., Pain, H., Waller, A., O'Mara, D., & Black, R. (2008) The Construction of a Pun Generator for Language Skills Development. *Applied Artificial Intelligence* **22**:841-869.
- [2] Shah, P.R., Shah, T.D., & Mali, S. (2017) Automated Pun Generator. *International Journal of Computer Applications* **166**(7).

- [3] Özbal, G. & Strapparava, C. (2012) A Computational Approach to the Automation of Creative Naming. *Proceedings of the Conference on International Language Resources and Evaluation*.
- [4] Benner, T. Pun Generator. <https://pungenerator.org>
- [5] Smith, M.R., Hintze, R.S. & Ventura, D. (2014) Nehovah: A Neologism Creator Nomen Ipsum. *Proceedings of the International Conference on Computational Creativity* 173-181.
- [6] Deri, A. & Knight, K. (2015) How to Make a Frenemy: Multitape FSTs for Portmanteau Generation. *Proceedings of the NAACL* 206-210.
- [7] Gangal, V., Jhamtani, H., Neubig, G., Hovy, E., & Nyberg, E. (2017) CharManteau: Character Embedding Models For Portmanteau Creation. *arXiv preprint arXiv:1707.01176*.
- [8] Miller, G.A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM* **38**(11):39-41.
- [9] Mikolov, T., Grave, E., Bojanowski, P., Puhersch, C., & Joulin, A. (2018) Advances in Pre-Training Distributed Word Representations. *Proceedings of the International Conference on Language Resources and Evaluation*.
- [10] Řehůřek, R. & Sojka, P. (2010) Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* 45-50.
- [11] Rijsbergen, C.J., Robertson, S.E., & Porter, M.F. (1980) New models in probabilistic information retrieval. London. *British Library Research and Development Report* **5587**.
- [12] Bird, S., Loper, E., & Klein, E. (2009) Natural Language Processing with Python.
- [13] Carnegie Mellon University Pronouncing Dictionary. <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>
- [14] ARPABET. <https://en.wikipedia.org/wiki/ARPABET>
- [15] Gerrish, S. & Beeferman, D. Port Manteaux. <https://www.onelook.com/pm>
- [16] Jiampoamarn, S., Kondrak, G., & Sherif, T. (2007) Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion. *Proceedings of the Association for Computational Linguistics* 372-379.
- [17] Jiampoamarn, S. & Kondrak, G. (2010) Letter-Phoneme Alignment: An Exploration. *Proceedings of the Association for Computational Linguistics* 780-788.
- [18] Treisman, A. (1985) Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing* **31**(2):156-177.

## 6 Supplementary Material

### 6.1 Algorithm Description

Given two input graphemes  $g^0, h^0$  the Entrendpreneur algorithm performs the following steps: (1) identify semantic neighbors, (2) lemmatization, stemming, and deduplication (3) grapheme-to-phoneme mapping, (4) phoneme portmanteau construction, (5) grapheme portmanteau construction, (6) result ordering. Steps (3-5) are also performed on the reverse-ordered graphemes  $h^0, g^0$ .

### 6.2 Identify Semantic Neighbors

Given two input graphemes  $g^0, h^0$  we find two corresponding sets of semantically related graphemes  $\{g^1, \dots, g^n\}, \{h^1, \dots, h^m\}$  by mapping each grapheme to its corresponding word vector in the FastText word vector embedding [9,10], and computing its cosine-distance to all other word-vectors in the space. We then take their  $n = m = 100$  nearest neighbors in this space to be their set of semantic neighbors.

### 6.3 Stemming, Lemmatization, and Deduplication

The set of semantic neighbor graphemes obtained from the FastText embedding often contain many strings with irregular capitalizations and tenses. These graphemes are mapped to their canonical English representations by identifying lemmas using WordNet and identifying stems using the Porter stemmer [11,12]. Any duplicate graphemes which are generated by these transformations are then removed from the neighbor set. This yields the new neighbor-grapheme sets  $\{g^1, \dots, g^{n'}\}, \{h^1, \dots, h^{m'}\}$ .

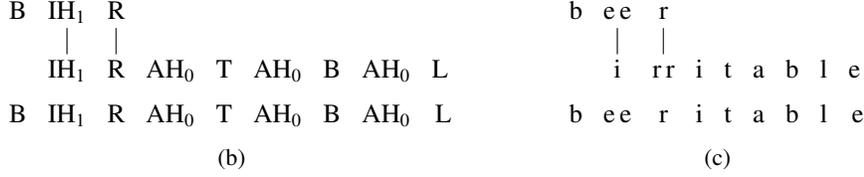
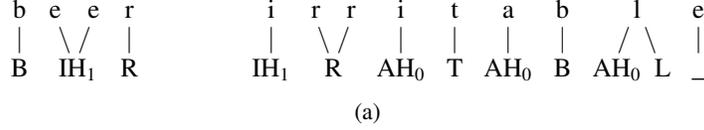


Figure 1: (a) grapheme-to-phoneme alignments for "beer" and "irritable", (b) phoneme-to-phoneme alignment with the resulting  $PM_{\text{phone}}$ , (c) grapheme-to-grapheme alignment with the resulting  $PM_{\text{graph}}$ .

## 6.4 Grapheme-to-Phoneme Mapping

The graphemes are then mapped to their corresponding phonemes using the CMU Pronouncing Dictionary [13], which associates each English-language grapheme with a phoneme string comprised of characters from the ARPABET character set [14]. In addition to the mappings of entire graphemes to phonemes produced by the CMU Pronouncing Dictionary, we further obtain mappings of individual graphs to phones using Jiampojarn, et al.'s (2007) EM-based L2P alignment algorithm [16], which was trained using the grapheme-phoneme pairs in the CMU Pronouncing Dictionary (Figure 1a). This associates to each grapheme  $g$  an aligned phoneme  $p$ , and to each grapheme  $h$  an aligned phoneme  $q$ .

## 6.5 Phoneme Portmanteau Construction

For each word pair  $(g, p), (h, q)$  we perform a 1-to-1 alignment of the head of the first phoneme string with the tail of the second phoneme string, finding the alignment which minimizes the sum of distances between the aligned phones (Figure 1b). The phone-level distance measure is defined as:

$$d(\text{phone}_1, \text{phone}_2) = \begin{cases} 0 & : \text{phone}_1 = \text{phone}_2 \\ 1 & : \text{Stressless}(\text{phone}_1) = \text{Stressless}(\text{phone}_2) \\ 2 & : (\text{phone}_1, \text{phone}_2) \in \text{NearMissConsonants} \\ 3 & : (\text{phone}_1, \text{phone}_2) \in \text{NearMissVowels} \\ \infty & : \text{otherwise} \end{cases}$$

For the sets of near-miss consonants and vowels, see Table 2. Vowel mismatches are penalized more heavily than consonant mismatches because it is the voiced vowel phone which primarily characterizes a syllable.

In addition to minimizing the summed phone distance, this alignment is subject to the constraints that at least 1 vowel phone and 1 consonant phone must be present in the phoneme overlap, and the number of phones in the overlap must be strictly less than  $\min(\text{len}(p), \text{len}(q))$ .

For an alignment having  $k$  overlapping phonemes, the resulting  $PM_{\text{phone}}$  will be one of either  $p_1 \dots p_{-1} q_{k+1} \dots q_{-1}$  or  $p_1 \dots p_{-k-1} q_1 \dots q_{-1}$ . Which one is chosen depends on the disambiguated  $PM_{\text{graph}}$ , discussed below.

## 6.6 Grapheme Portmanteau Construction

There can exist two different values of  $PM_{\text{graph}}$  for a given phoneme alignment. For example, in Figure 1c both "beeritable" and "birritable" are consistent with the alignment. In such cases we select the grapheme which maximizes the likelihood that the reader will correctly reconstruct  $g, h$  given  $PM_{\text{graph}}$ . The two possible representations of  $PM_{\text{graph}}$  are of the form

- (1)  $g_{\text{disjoint}} g_{\text{overlap}} h_{\text{disjoint}}$
- (2)  $g_{\text{disjoint}} h_{\text{overlap}} h_{\text{disjoint}}$

Table 2: Near-miss phones

	ARPABET		IPA	
	phone <sub>1</sub>	phone <sub>2</sub>	phone <sub>1</sub>	phone <sub>2</sub>
consonant	B	P	b	p
	D	DH	d	ð
	D	T	d	t
	DH	TH	ð	θ
	F	V	f	v
	SH	ZH	ʃ	ʒ
	CH	JH	tʃ	dʒ
	S	Z	s	z
vowel	AA	EH	ɑ	ɛ
	AH	UH	ʌ	ʊ
	EH	IH	ɛ	ɪ

In the case of (1) the entire  $g$  string is present in  $PM_{\text{graph}}$ , and therefore it is assumed that the reader will be able to uniquely identify  $g$  due to the perceptual "pop-out" effect [18]; the same holds for  $h$  in (2). Therefore the ease of reconstructibility is lower-bounded by  $p(h \mid \text{ends with } h_{\text{disjoint}})$  in (1) and by  $p(g \mid \text{starts with } g_{\text{disjoint}})$  in (2). Because the graphemes were generated through a form of loose word-association, we consider  $p(\cdot)$  to be uniform with support on all graphemes in the vocabulary. Therefore these conditional probabilities can be computed as:

$$(3) \quad p(h \mid \text{ends with } h_{\text{disjoint}}) = 1 / \#\{x \in \text{GraphemeVocab} \mid x \text{ ends with } h_{\text{disjoint}}\}$$

$$(4) \quad p(g \mid \text{starts with } g_{\text{disjoint}}) = 1 / \#\{x \in \text{GraphemeVocab} \mid x \text{ starts with } g_{\text{disjoint}}\}$$

If (3) > (4) then (1) is selected as  $PM_{\text{graph}}$ , otherwise (2) is selected.

## 6.7 Result Ordering

We consider the quality of a PM to be a function of how "unexpectedly" phonetically close  $p_{\text{overlap}}$  and  $q_{\text{overlap}}$  are. This is a function of the phonetic distance between  $p_{\text{overlap}}$  and  $q_{\text{overlap}}$ , as well as the commonality of the phonemes  $p_{\text{overlap}}$  and  $q_{\text{overlap}}$  in the vocabulary. We measure the commonality of a phoneme in the vocabulary as

$$p(p_{\text{overlap}}, q_{\text{overlap}}) = p(p_{\text{overlap}}) \times p(q_{\text{overlap}})$$

where

$$p(p_{\text{overlap}}) = \#\{y \in \text{PhonemeVocab} \mid y \text{ ends with } p_{\text{overlap}}\} / \#\text{PhonemeVocab}$$

$$p(q_{\text{overlap}}) = \#\{y \in \text{PhonemeVocab} \mid y \text{ starts with } q_{\text{overlap}}\} / \#\text{PhonemeVocab}$$

Take  $d^*$  to be the distance between  $p_{\text{overlap}}$  and  $q_{\text{overlap}}$ . Then we consider the best PM to be the one which is lowest in the lexical ordering induced by  $(d^*, p(p_{\text{overlap}}, q_{\text{overlap}}))$ . The generated PMs are then sorted from best to worst, with top results being returned.

## 6.8 Questionnaire Design

To assess the subjective quality and humor of the portmanteaus produced by Entendpreneur compared to those produced by Charmanteau [7] and Port Manteaux [15], 20 questionnaires were created. Each questionnaire was composed of 10 multiple-choice questions, the first five questions asking participants to select the "best portmanteau" of the two provided words, and the last five questions asking participants to select the "funniest combination" of the two provided words.

Table 3: Algorithm Outputs

Input		Result		
word <sub>1</sub>	word <sub>2</sub>	Entendpreneur	Port Manteaux	Charmanteau
angry	star	wrathlete	furiostar	anstar
hebrew	sensor	yahwearable	israelitesensor	hebrensensor
literary	cage	shackademic	heavyweightranslation	litercage
immune	orchestra	violinflamation	orchestrasponse	immunestra
prohibited	metallic	silverboten	prohibitalllic	protallic
intellectual	blond	tanalytical	blontellectual	intellond

The word pairs for each question were selected from among the 10k most common English words, removing all words where any of the following hold:

1. does not appear in WordNet
2. is shorter than 4 letters
3. is the name of a person or place
4. is pluralized

The word pairs were then constructed by randomly sampling "[adjective], [noun]" pairs from among the remaining valid words. For each question, 7 multiple choice options were provided in random order: 3 options were the top-3 results produced by Entendpreneur, 3 options were the top-3 results produced by one of either Charmanteau or Port Manteaux, 1 option was "Other" which the participant was instructed to select if they did not like any of the other 6 options. All questions which were answered with "Other" were removed from the results shown in Table 1. The participants completed the questionnaires using the website TypeForm. A sample question is shown in Figure 2.

immune, orchestra

A violinflamation

B orchestrasponse

C orchestrasponses

D operatect

E woodwindestructible

F orchestraceptors

G Other

Figure 2: Sample question.