Deep learning

Word Embedding

Hamid Beigy

Sharif university of technology

November 22, 2019

Table of contents

Table of contents

Introduction

1. How to represent a word?
   - Represent words as atomic symbols such as talk, university, building.
   - Represent word as a one-hot vector such as

$$university = (\underset{egg}{0}, \underset{student}{0}, \underset{talk}{0}, \underset{university}{1}, \underset{building}{0}, \ldots, \underset{buy}{0})$$

2. Issues with on-hot representation
   - How large is this vector? dimensionality is large; vector is sparse
   - Representing new words (any idea?).
   - How measure word similarity?

Distributional representation

1. Linguistic items with similar distributions have similar meanings (words occur in the same contexts probably have similar meaning).

$$\textit{university} = (\underset{\textit{egg}}{0.2}, \underset{\textit{student}}{0.1}, \underset{\textit{talk}}{0.12}, \underset{\textit{university}}{0.38}, \underset{\textit{building}}{0.2}, \ldots, \underset{\textit{buy}}{0.12})$$

2. Word meanings are vector of basic concept.
3. What are basic concept?
4. How to assign weights?
5. How to define the similarity/distance?

How to use word vectors?

1. Distance/similarity
   - Cosine similarity: Word vector are normalized by length

$$\cos(u, v) = \frac{\langle u, v \rangle}{\|u\| \|v\|}$$

   - Euclidean distance:

$$d(u, v) = \|u - v\|^2$$

   - Inner product: This is same as cosine similarity if vectors are normalized

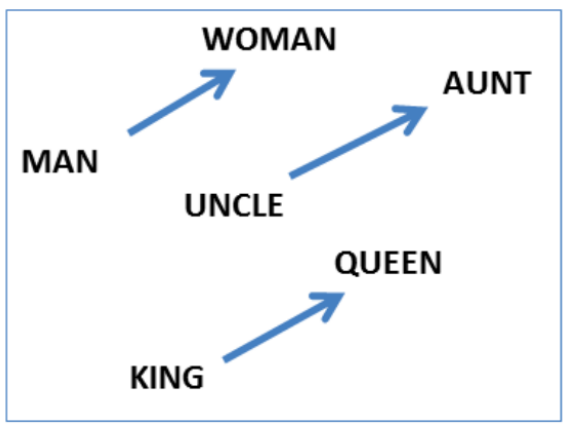
$$d(u, v) = \langle u, v \rangle$$

2. Choosing the right similarity metric is important.

How to use word vectors?

1. Word analogy

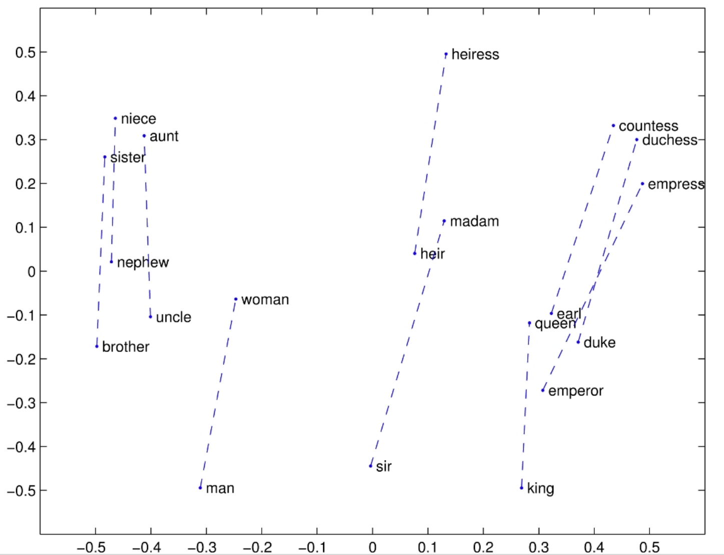
$$v_{man} - v_{woman} + v_{uncle} \sim v_{aunt}$$



How to learn word vectors?

1. What are basic concept?
   - We want that the number of basic concepts to be small and
   - Basis be orthogonal
2. How to assign weights?
3. How to define the similarity/distance such as cosine similarity?

Distributional representation (example)



Term-document incidence matrix

Example

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Anthony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Entry is 1 if term occurs. Example: Calpurnia occurs in *Julius Caesar*.
Entry is 0 if term doesn't occur. Example: Calpurnia doesn't occur in *Tempest*.
Each term is represented as a vector of bits.

Term weighting

1. Evaluation of how important a term is with respect to a document.
2. First idea: the more important a term is, the more often it appears: *term frequency*

$$tf_{t,d} = \sum_{x \in d} f_t(x) \text{ where } f_t(x) = \begin{cases} 1 & \text{if } x = t \\ 0 & \text{otherwise} \end{cases}$$

3. The order of terms within a doc is ignored

Inverse Document Frequency

1. *Inverse document frequency* of a term t:

$$idf_t = log \frac{N}{df_t} \qquad \text{with } N = \text{collection size}$$

2. Rare terms have high *idf*, contrary to frequent terms
3. Example (Reuters collection):

Term t	df_t	idf_t
car	18165	1.65
auto	6723	2.08
insurance	19241	1.62
best	25235	1.5

4. In tf-idf weighting, the weight of a term is computed using both *tf* and *idf*:

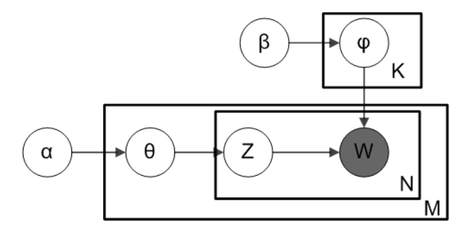
$$w(t,d) = tf_{t,d} \times idf_t \qquad \text{called } tf - idf_{t,d}$$

Dimensionality reduction

1. we don't need all of the dimensions that represent a word, only the most important ones.
2. There are several techniques such as
   - Principle Component Analysis (PCA): The most important dimensions contain the most variance
   - Latent Semantic Analysis (LSA): Project terms and documents into a topic space using SVD on term-document (co-occurrence) matrix.
   - Low-rank Approximation
3. Can we learn the dimensionality reduction from texts?

Latent Dirichlet allocation

1. Assumes generative probabilistic model of a corpus[1].
2. Documents are represented as distribution over latent topics, where each topic is characterized by a distribution over words.



[1] Blei, David M.; Ng, Andrew Y.; Jordan, Michael, "Latent Dirichlet Allocation". Journal of Machine Learning Research. 3 (45): pp. 993-1022, 2003.

Table of contents

Language modeling

1. An language model is a model for how humans generate language.
2. The quality of language models is measured based on their ability to learn a probability distribution over words in vocabulary V.
3. Language models generally try to compute the probability of a word w_t given its $n-1$ previous words, i.e. $p(w_t|w_{t-1}, \cdots w_{t-n+1})$.
4. Applying the chain rule and Markov assumption, we can approximate the probability of a whole sentence or document by the product of the probabilities of each word given its n previous words:

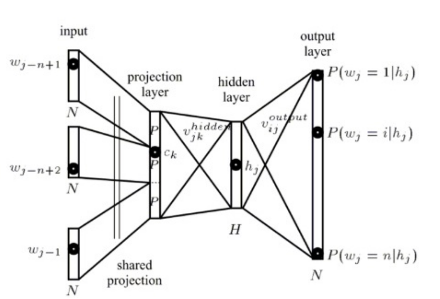
$$p(w_1, \cdots, w_T) = \prod_i p(w_i|w_{i-1}, \cdots, w_{i-n+1})$$

5. In n-gram based language models, we can calculate a word's probability based on the frequencies of its constituent n-grams:

$$p(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{count(w_{t-n+1}, \cdots, w_{t-1}, w_t)}{count(w_{t-n+1}, \cdots, w_{t-1})}$$

Neural Probabilistic Language Model

1. In NNs, we achieve the same objective using the softmax layer[2].

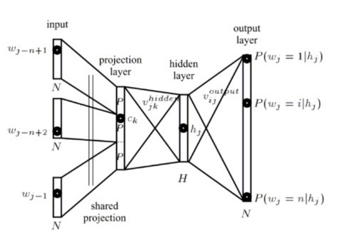


$$p(w_t | w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(h^\top v'_{w_t})}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})}$$

[2] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 3, 1137-1155.

Neural Probabilistic Language Model

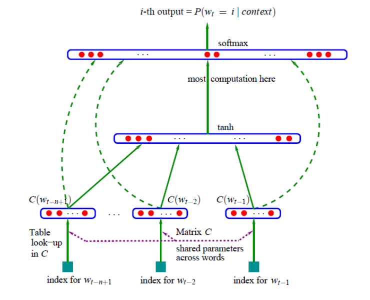
1. In this model



2. The inner product $h^\top v'_{w_t}$ computes the (unnormalized) log-probability of word w_t , which we normalize by the sum of the log-probabilities of all words in V.
3. h is the output vector of the penultimate network layer, while v'_w is the output embedding of word w, i.e. its representation in the weight matrix of the softmax layer.

Neural Probabilistic Language Model

1. In NNs, we achieve the same objective using the softmax layer[3].



2. Associate each word in vocabulary a distributed feature vector.
3. Learn both the embedding and parameters for probability function jointly.

[3] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 3, 1137-1155.

Table of contents

Word2vec algorithm

1. Proposed by Mikolov et. al. and widely used for many NLP applications (in two papers).
2. Key features
   - Uses neural networks to train word / context classifiers (feedforward neural net)
   - Uses local context windows (environment around any word in a corpus) as inputs to the NN
   - Removed hidden layer.
   - Use of additional context for training LMs.
   - Introduced newer training strategies using huge database of words efficiently.

Word2vec algorithm

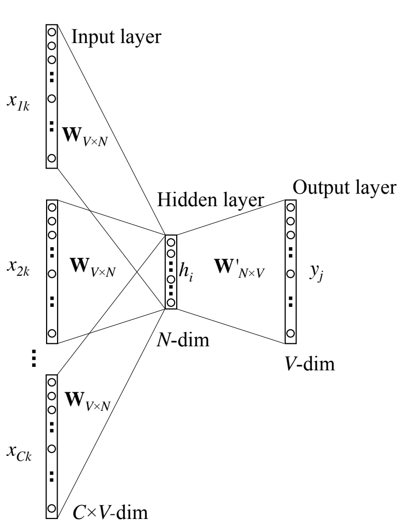
1. In their first paper, Mikolov et al. propose two architectures for learning word embeddings that are computationally less expensive than previous models[4].
2. In their second paper, they improve upon these models by employing additional strategies to enhance training speed and accuracy[5].

[4] Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1-12.

[5] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS, 1-9.

Continuous Bag-of-Words

1. Mikolov et al. thus use both the n words before and after the target word w_t to predict it.
2. They call this continuous bag-of-words (CBOW), as it uses continuous representations whose order is of no importance.



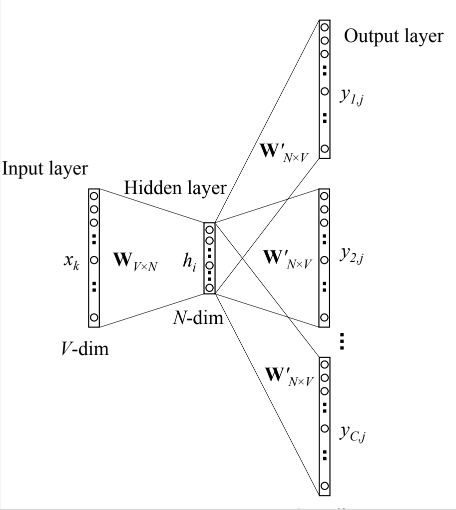
Continuous Bag-of-Words objective function

1. The objective function of CBOW in turn is

$$I(\theta) = \sum_{t \in Text} \log P(w_t | w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n})$$

Skip-gram

1. Instead of using the surrounding words to predict the center word as with CBOW, skip-gram uses the centre word to predict the surrounding words.



Skip-gram objective function

1. The skip-gram objective thus sums the log probabilities of the surrounding n words to the left and to the right of the target word w_t to produce the following objective function.

$$l(\theta) = \sum_{t \in Text} \sum_{-n \leq j \leq n, \neq 0} \log P(w_{t+j} | w_t)$$

Table of contents

Global Vectors for Word Representation[6]

1. Skip-gram doesn't utilize the statistics of corpus since they train on separate local context windows instead of on global co-occurrence counts.
2. The statistics of word occurrences in a corpus is the primary source of information available to all unsupervised methods for learning word representations.
3. Let X_{ij} be the number of times word j occurs in the context of word i.
4. Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of word i.
5. Let $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$ be the probability that word j appear in the context of word i.

[6] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. Proc. of Conference on Empirical Methods in Natural Language Processing, 1532-1543.

Global Vectors for Word Representation

1. Co-occurrence probabilities for target words ice and steam with selected context words from a 6 billion token corpus.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

2. Considering two words $i = ice$ and $j = steam$ and study their relationship using various probe words, k.
3. For words k related to ice but not steam, say $k = solid$, we expect the ratio $\frac{P_{ik}}{P_{jk}}$ will be large.
4. For words k related to steam but not ice, say $k = gas$, we expect the ratio $\frac{P_{ik}}{P_{jk}}$ will be small.
5. For words k like water or fashion, that are either related to both ice and steam, or to neither, the ratio should be close to one.

Global Vectors for Word Representation

1. This argument suggests that the appropriate starting point for word vector learning should be with ratios of co-occurrence probabilities rather than the probabilities themselves.
2. Noting that the ratio $\frac{P_{ik}}{P_{jk}}$ depends on three words i, j, k, the most general model takes the form of
$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$
where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ are separate context word vectors.
3. We would like F to encode $\frac{P_{ik}}{P_{jk}}$ in the word vector space.
4. Since vector spaces are inherently linear structures, the most natural way to do this is with vector differences. Hence
$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Global Vectors for Word Representation

1. Parameters of F are vectors while the right-hand side is a scaler.
2. F can be a complicated function such as a neural network, but a simplified function can be used also.

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

3. For word-word co-occurrence matrices, the distinction between a word and a context word is arbitrary. Hence, we are free to exchange the two roles, i.e. $w \leftrightarrow \tilde{w}$ and $X \leftrightarrow X^T$.
4. Hence, model should be invariant under this relabeling. Thus

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{X_{ik}}{X_i}$$

5. $F = \exp$ is the solution of the above equation. Hence,

$$w_i^T \tilde{w}_k = log P_{ik} = \log X_{ik} - \log X_i$$

Global Vectors for Word Representation

1. We have,
$$w_i^T \tilde{w}_k = log P_{ik} = \log X_{ik} - \log X_i$$
2. The above equation would exhibit the exchange symmetry if not for the $\log X_i$ on the right-hand side.
3. This term is independent of k so it can be absorbed into a bias b_i for w_i.
4. Adding an additional bias $\tilde{b}_k$ for $\tilde{w}_k$ restores the symmetry. Hence
$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik}$$

Glove cost function

1. In an ideal setting, where you have perfect word vectors, the following expression will be zero.

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik} =$$

2. Hence, we can define the objective function as

$$J(w_i, \tilde{w}_k) = \left(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik} \right)^2$$

3. Now, the final cost function is

$$J(w_i, \tilde{w}_k) = \sum_{i,k=1}^{|V|} f(X_{ik}) \left(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik} \right)^2$$

where f is a weighting function, which is defined manually.

Glove Results

1. GloVe becomes a global model for unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks.
2. Advantages
   - Fast training
   - Scalable to huge corpora
   - Good performance even with small corpus, and small vectors
   - Early stopping. We can stop training when improvements become small.
3. Drawbacks
   - Uses a lot of memory: the fastest way to construct a term-cooccurence matrix is to keep it in RAM as a hash map and perform cooccurence increments in a global manner
   - Sometimes quite sensitive to initial learning rate

Table of contents

FastText

1. One drawback of the Word2vec and Glove is the fact that they dont handle out-of-vocabulary.
2. FastText introduced the concept of subword-level embeddings, based on the skip-gram model, but where each word is represented as a bag of character -grams[7].
3. A vector representation is associated to each character -gram, and words are represented as the sum of these representations.
4. This allows the model to compute word representations for words that did not appear in the training data.

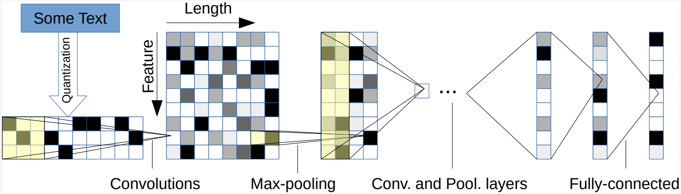
[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, Enriching Word Vectors with Subword Information, Transactions of the Association for Computational Linguistics, Vol. 5, pp. 135–146, 2017.

FastText

1. Each word w is represented as a bag of character n-gram, plus a special boundary symbols $<$ and $>$ at the beginning and end of words, plus the word w itself in the set of its n-grams.
2. Taking the word where and $n = 3$ as an example, it will be represented by the character n-grams:
   $<$ wh, whe, her, ere, re$>$ and the special sequence $<$ where $>$.
3. In practice, they extracted all the n-grams for n greater or equal to 3 and smaller or equal to 6.

Character-level Convolutional Networks[8]

1. A list of character are defined 70 characters which including 26 English letters, 10 digits, 33 special characters and new line character.
2. The network architectures are: 9 layers deep with 6 convolutional layers and 3 fully-connected layers.

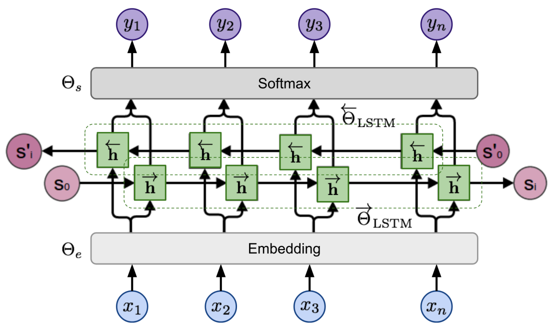


[8]Xiang Zhang, Junbo Zhao, and Yann LeCun, Character-level Convolutional Networks for Text Classification, NIPS 2015.

Table of contents

Embeddings from Language Model (ELMo) [9]

1. ELMo learns contextualized word representation by pre-training a language model in an unsupervised way.



[9] Matthew Peters, et. al., Deep Contextualized Word Representations, Proc. of Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018.

Embeddings from Language Model (ELMo)

1. The bidirectional Language Model (biLM) is the foundation for ELMo.
2. While the input is a sequence of n tokens, $(x_1, \ldots, x_n)$, the language model learns to predict the probability of next token given the history.
3. In the forward pass, the history contains words before the target token,

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

4. In the backward pass, the history contains words after the target token,

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_{i+1}, \ldots, x_n)$$

5. The predictions in both directions are modeled by multi-layer LSTMs with hidden states.

Embeddings from Language Model (ELMo)

1. The model is trained to minimize the negative log likelihood (= maximize the log likelihood for true words) in both directions:

$$\mathcal{L} = -\sum_{i=1}^{n} \Big(\log p(x_i \mid x_1, \ldots, x_{i-1}; \Theta_e, \overrightarrow{\Theta}_{\text{LSTM}}, \Theta_s) + \log p(x_i \mid x_{i+1}, \ldots, x_n; \Theta_e, \overleftarrow{\Theta}_{\text{LSTM}}, \Theta_s) \Big)$$

Embeddings from Language Model (ELMo)

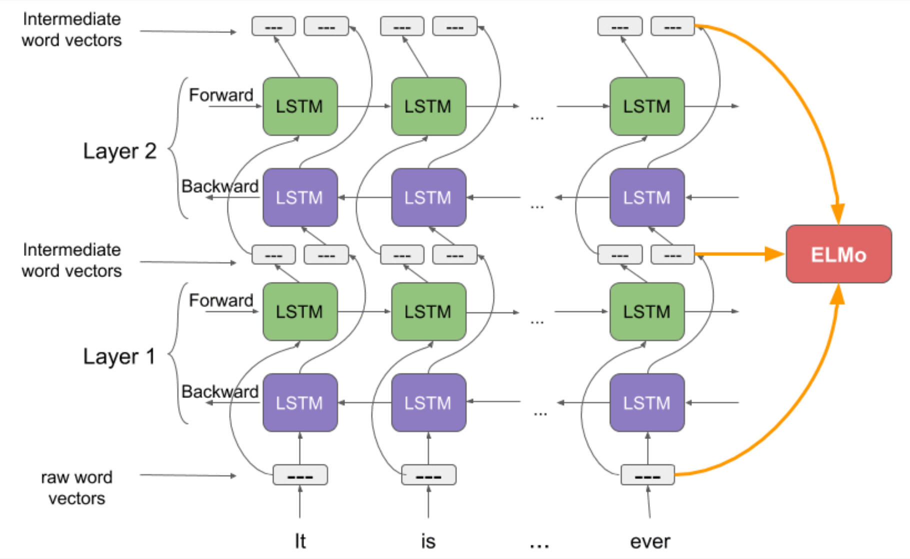
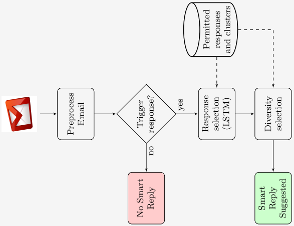


Table of contents

Smart Reply[10]

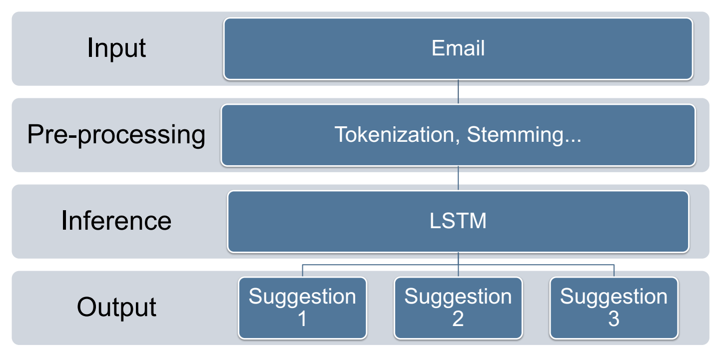
1. An end-to-end method for automatically generating short email responses, called Smart Reply.
2. It generates semantically diverse suggestions that can be used as complete email responses with just one tap on mobile.
3. The system is currently used in In- box by Gmail and is responsible for assisting with 10% of all mobile responses.



[10] Anjuli Kannan et. al., Smart Reply: Automated Response Suggestion for Email, SIGKDD 2016.

Smart Reply Architecture

1. A first version



Smart Reply Architecture

1. Issues Response diversity, Quality control, and Computing costs

Unnormalized Responses
Yes, I'll be there.
Yes, I will be there.
I'll be there.
Yes, I can.
What time?
I'll be there!
I will be there.
Sure, I'll be there.
Yes, I can be there.
Yes!

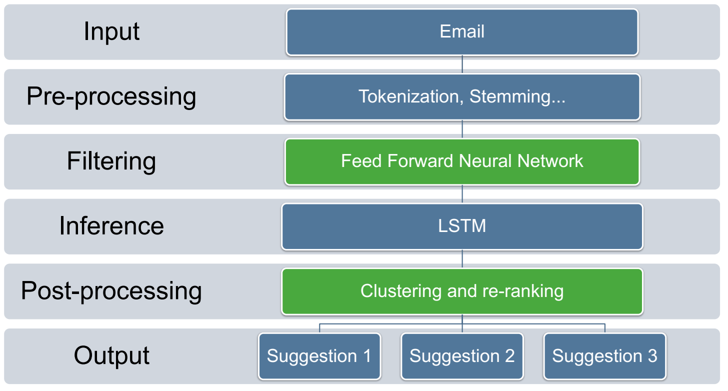
Smart Reply Architecture

1. Issues Response diversity, Quality control, and Computing costs

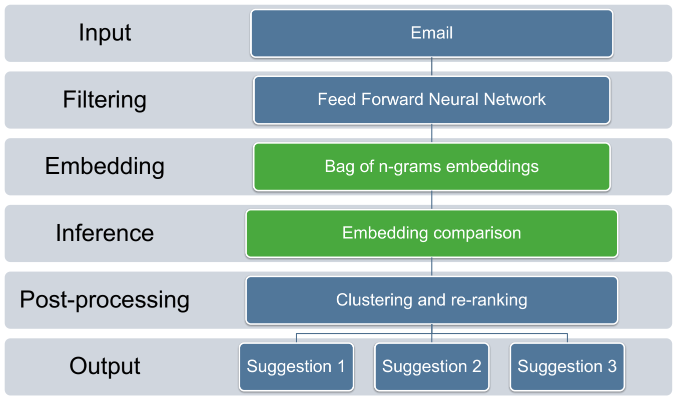
Unnormalized Responses
Yes, I'll be there.
Yes, I will be there.
I'll be there.
Yes, I can.
What time?
I'll be there!
I will be there.
Sure, I'll be there.
Yes, I can be there.
Yes!

Smart Reply: A better architecture

1. Issues
   - Response diversity :semi supervised clustering (human in the loop)
   - Quality control : restrict output sentences to a subset
   - Computing costs : triggering NN system eliminates 90% of messages



Smart Reply: An optimized version[11]



[11] Matthew Henderson, et al., Efficient Natural Language Response Suggestion for Smart Reply, arXiv 2017.

Table of contents

Reading I

1. Blei, David M.; Ng, Andrew Y.; Jordan, Michael, "Latent Dirichlet Allocation". Journal of Machine Learning Research. 3 (45): pp. 993-1022, 2003.
2. Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 3, 1137-1155.
3. Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proc. of the International Conference on Learning Representations (ICLR 2013), 1-12.
4. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS, 1-9.

Reading II

5. Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532-1543.
6. Di Carlo, V., Bianchi, F., and Palmonari, M. (2019). Training Temporal Word Embeddings with a Compass. Proc. of the AAAI Conference on Artificial Intelligence, 33(01), 6326-6334.
7. White board notes.