An Introduction of Sentence Embedding

Fan Jia

September 20, 2019

Contents

Bag of Words(BOW)

- Model based on Statistics
  - One-Hot
  - Bag of Words(BOW)
  - TF-IDF
- Model based on Word Vector
  - Average on Word Vector(Word2Vec, Glove, FastText)
  - Weighted Average based on TF-IDF

Bag of Words(BOW)

- One-Hot

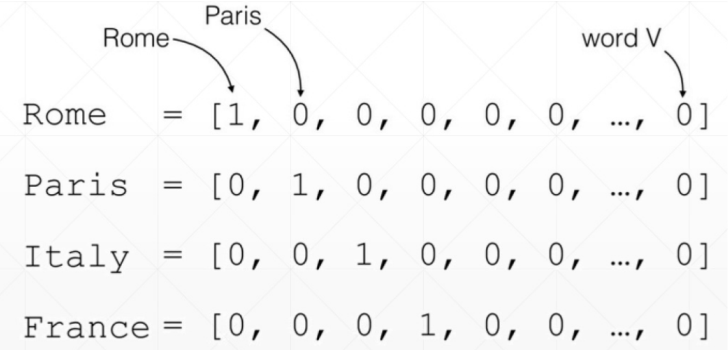


Figure: ont-hot encoding

- Bag of Words(BOW)

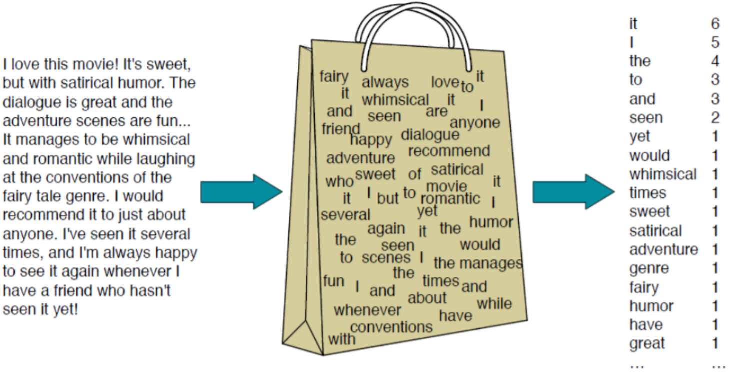


Figure: Bag of Words encoding

Bag of Words(BOW)

- Bag of Words(BOW)

Article ID	biolog	biopsi	biolab	biotin	almost	cancer-surviv	cancer-stage	Article Class
00001	12	1	2	10	0	1	4	breast-cancer
00002	10	1	0	3	0	6	1	breast-cancer
00014	4	1	1	1	0	28	0	breast-cancer
00063	4	0	0	0	0	18	7	breast-cancer
00319	0	1	0	9	0	20	1	breast-cancer
00847	7	2	0	14	0	11	5	breast-cancer
03042	3	1	3	1	0	19	8	lung-cancer
05267	4	4	2	6	0	14	11	lung-cancer
05970	8	0	4	9	0	9	17	lung-cancer
30261	1	0	0	11	0	21	1	prostate-cancer
41191	9	0	5	14	0	11	1	prostate-cancer
52038	6	1	1	17	0	19	0	prostate-cancer
73851	1	1	8	17	0	17	3	prostate-cancer

Figure: frequency matrix

Bag of Words(BOW)

- Bag of Words(BOW)

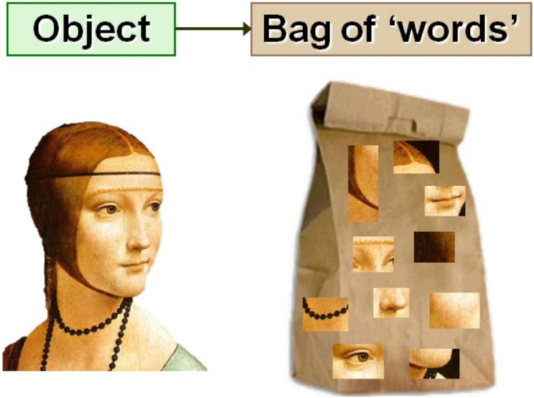


Figure: Bag of Words Encoding

Bag of Words(BOW)

- TF-IDF(Term Frequency–Inverse Document Frequency)
    - TF: Term Frequency

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

    - IDF: Inverse Document Frequency

$$IDF(t) = log\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}$$

Bag of Words(BOW)

- Pros and Cons
  - Pros
    Fast calculation
  - Cons
    Poor performance in Sentiment Analysis because of the omission of words' order

Deep Learning Methods

- Deep Averaging Network(DAN)
- CNN for Sentence Modeling
- RNN for Sentence Modeling

- Deep Averaging Network(DAN): comparable sentiment accuracies to syntactic functions

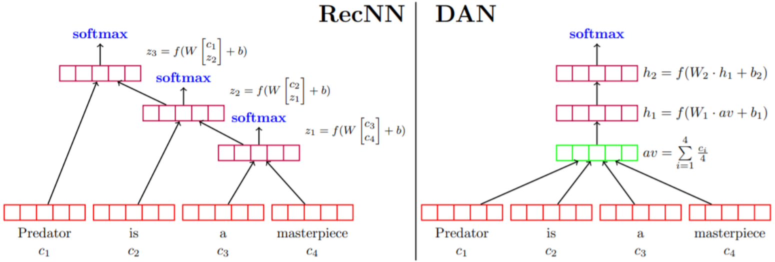


Figure 1: On the left, a **RecNN** is given an input sentence for sentiment classification. Softmax layers are placed above every internal node to avoid vanishing gradient issues. On the right is a two-layer **DAN** taking the same input. While the **RecNN** has to compute a nonlinear representation (purple vectors) for every node in the parse tree of its input, this **DAN** only computes two nonlinear layers for every possible input.

Reference:ACL, Deep Unordered Composition Rivals Syntactic Methods for Text Classification,2015

Deep Learning Methods

- Deep Averaging Network(DAN): comparable sentiment accuracies to syntactic functions

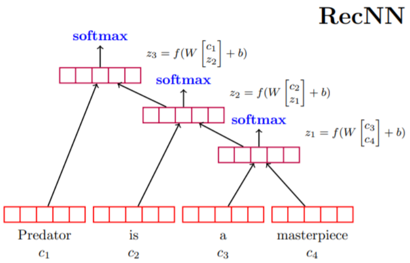
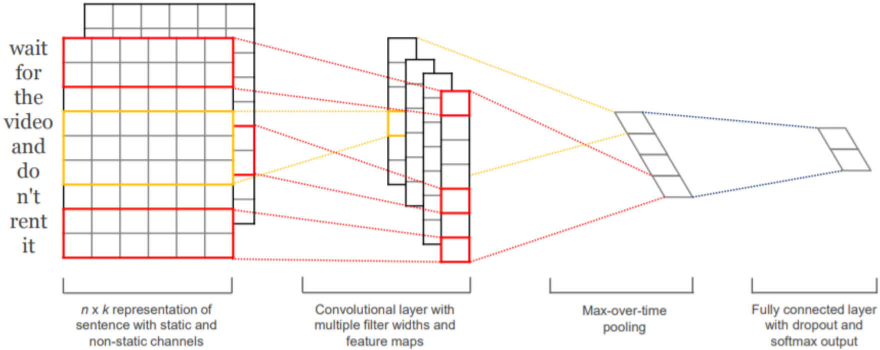


Figure 1: On the left, a **RecNN** is given an input sentence for sentiment classification. Softmax layers are placed above every internal node to avoid vanishing gradient issues. On the right is a two-layer **DAN** taking the same input. While the **RecNN** has to compute a nonlinear representation (purple vectors) for every node in the parse tree of its input, this **DAN** only computes two nonlinear layers for every possible input.

https://github.com/miyyer/dan

keras:https://github.com/aravindsiv/dan_qa

- CNN for Sentence Modeling



Reference:Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

Deep Learning Methods

- CNN for Sentence Modeling
  - Input Layer($n \times k$)
    - static
    - non static:BP fine tune
  - Convolution Layer($h \times k$)
    - Feature Map
  - Pooling layer
    - Max-over-time Pooling
  - Fully Connected Layer(FC) + Softmax
    - Softmax
    - Dropout(Dropout=0.5, L2<=3)

  `https://github.com/yoonkim/CNN_sentence`

Bag of Words(BOW)

- RNN for Sentence Embedding
  - RNN for Long Term Sequence Modeling
  - LSTM(Long-Short Term Memory)

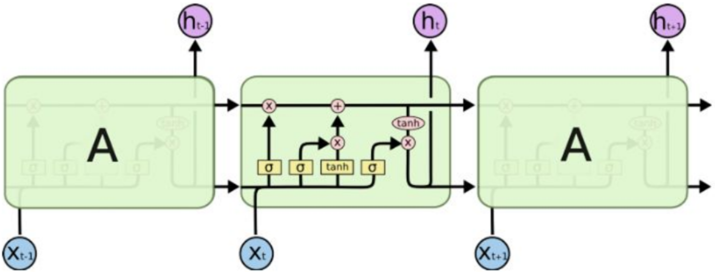


Figure: LSTM

Bag of Words(BOW)

- Pros and Cons
  - Capture the features of sequence
  - supervised task
  - inefficiently transferrable

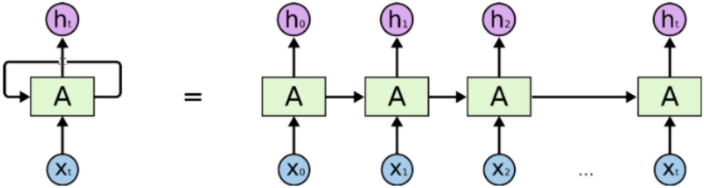


Figure: RNN

2017 Supervised Learning of Universal Sentence Representations from Natural Language Inference Data
https://blog.csdn.net/triplemeng/article/details/82106615

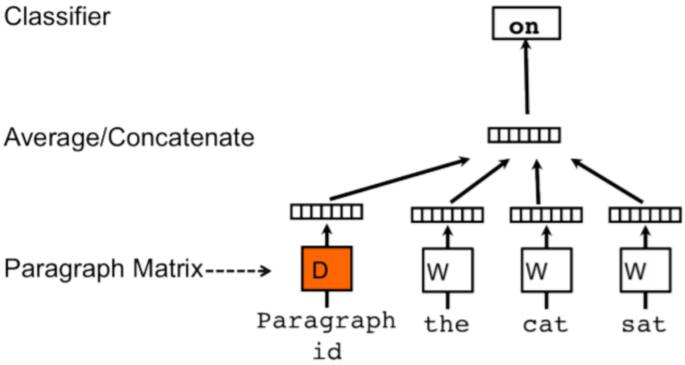
Doc2Vec

- Doc2Vec(Paragraph2Vec, Sentence Embeddings)
  - PV-DM Distributed Memory Model of paragraph vector
    - similar to CBOW of Word2Vec
  - PV-DBOW Distributed Bag of Words of parageaph vector
    - similar to Skip-Gram of Word2Vec
  - gensim.models.doc2vec

`https://cs.stanford.edu/~quocle/paragraph_vector.pdf`

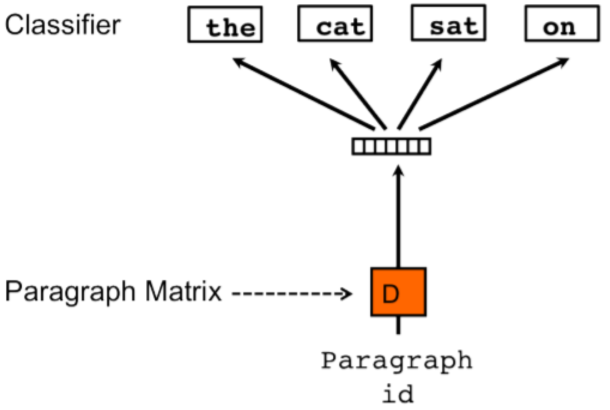
Doc2Vec

- PV-DM Distributed Memory Model of paragraph vector



Doc2Vec

- PV-DBOW Distributed Bag of Words of parageaph vector



Sentence2Vec

- SIF(Smooth Inverse Frequency)
- Skip-though Vectors
- Quick-Thought Vectors

Sentence2Vec

- SIF(Smooth Inverse Frequency)

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.

Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

1: **for all** sentence s in $\mathcal{S}$ **do**

2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$

3: **end for**

4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector

5: **for all** sentence s in $\mathcal{S}$ **do**

6: $\quad v_s \leftarrow v_s - uu^\top v_s$

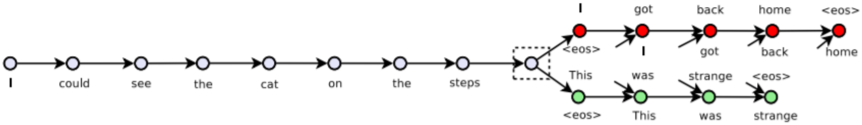
7: **end for**

---

https://github.com/peter3125/sentence2vec

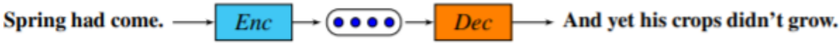
Sentence2Vec

- Skip-though Vectors(NIPS15)

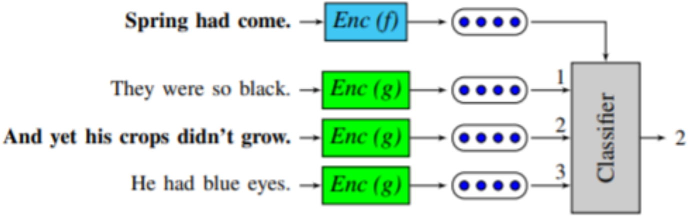


https://github.com/tensorflow/models/tree/master/research/skip_thoughts

- Quick-Thought Vectors(2018)



(a) Conventional approach

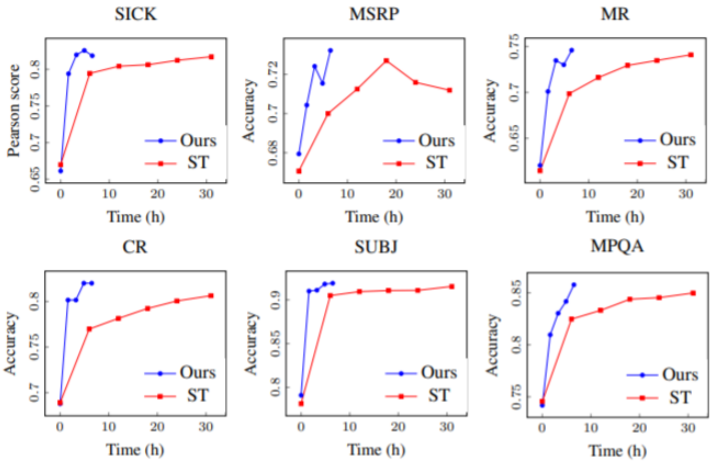


(b) Proposed approach

https://arxiv.org/pdf/1803.02893.pdf

Sentence2Vec

- Quick-Thought Vectors(2018): outperform and fast



Power Mean

- Power Mean(2018):much harder-to-beat baseline

$$\left(\frac{x_1^p + \cdots + x_n^p}{n}\right)^{1/p}; \quad p \in \mathbb{R} \cup \{\pm\infty\}$$

for a sequence of numbers $(x_1, \ldots, x_n)$. This generalized form retrieves many well-known means such as the arithmetic mean ($p = 1$), the geometric mean ($p = 0$), and the harmonic mean ($p = -1$). In the extreme cases, when $p = \pm\infty$, the power mean specializes to the minimum ($p = -\infty$) and maximum ($p = +\infty$) of the sequence.

$$\mathbf{s}^{(i)} = H_{p_1}(\mathbf{W}^{(i)}) \oplus \cdots \oplus H_{p_K}(\mathbf{W}^{(i)})$$

https://github.com/UKPLab/arxiv2018-xling-sentence-embeddings

Power Mean

- Power Mean(2018):much harder-to-beat baseline

Model	Σ	AM	AC	CLS	MR	CR	SUBJ	MPQA	SST	TREC
Arithmetic mean										
GloVe (GV)	77.2	50.0	70.3	76.6	77.1	78.3	91.3	87.9	80.2	83.4
GoogleNews (GN)	76.1	50.6	69.4	75.2	76.3	74.6	89.7	88.2	79.9	81.0
Morph Specialized (MS)	73.5	47.1	64.6	74.1	73.0	73.1	86.9	88.8	78.3	76.0
Attract-Repel (AR)	74.1	50.3	63.8	75.3	73.7	72.4	88.0	89.1	78.3	76.0
GV $\oplus$ GN $\oplus$ MS $\oplus$ AR	**79.1**	**53.9**	**71.1**	**77.2**	**78.2**	**79.8**	91.8	**89.1**	**82.8**	**87.6**
power mean [p-values]										
GV $[-\infty, 1, \infty]$	77.9	54.4	69.5	76.4	76.9	78.6	92.1	87.4	80.3	85.6
GN $[-\infty, 1, \infty]$	77.9	55.6	71.4	75.8	76.4	78.0	90.4	88.4	80.0	85.2
MS $[-\infty, 1, \infty]$	75.8	52.1	66.6	73.9	73.1	75.8	89.7	87.1	79.1	84.8
AR $[-\infty, 1, \infty]$	77.6	55.6	68.2	75.1	74.7	77.5	89.5	88.2	80.3	89.6
GV $\oplus$ GN $\oplus$ MS $\oplus$ AR $[-\infty, 1, \infty]$	80.1	58.4	71.5	77.0	78.4	80.4	**93.1**	88.9	83.0	90.6
$\rightarrow$ with z-norm[†]	**81.1**	**60.5**	**75.5**	**77.3**	**78.9**	**80.8**	93.0	**89.5**	**83.6**	**91.0**
Baselines										
GloVe + SIF	76.1	45.6	72.2	75.4	77.3	78.6	90.5	87.0	80.7	78.0
Siamese-CBOW	60.7	42.6	45.1	66.4	61.8	63.8	75.8	71.7	61.9	56.8
Sent2Vec	78.0	52.4	**72.7**	75.9	76.3	80.3	91.1	86.6	77.7	**88.8**
InferSent	**81.7**	**60.9**	72.4	**78.0**	**81.2**	**86.7**	**92.6**	**90.6**	**85.0**	88.2

Other Methods

- Attention-based Models
  - self-attention
  - Learning Sentence Representation with Guidance of Human Attention IJCAI
  - Hierarchical Attention
- Multi-task Learning
  - Multi-task Learning
  - Universal Sentence Encode(Google)
- from Conversations