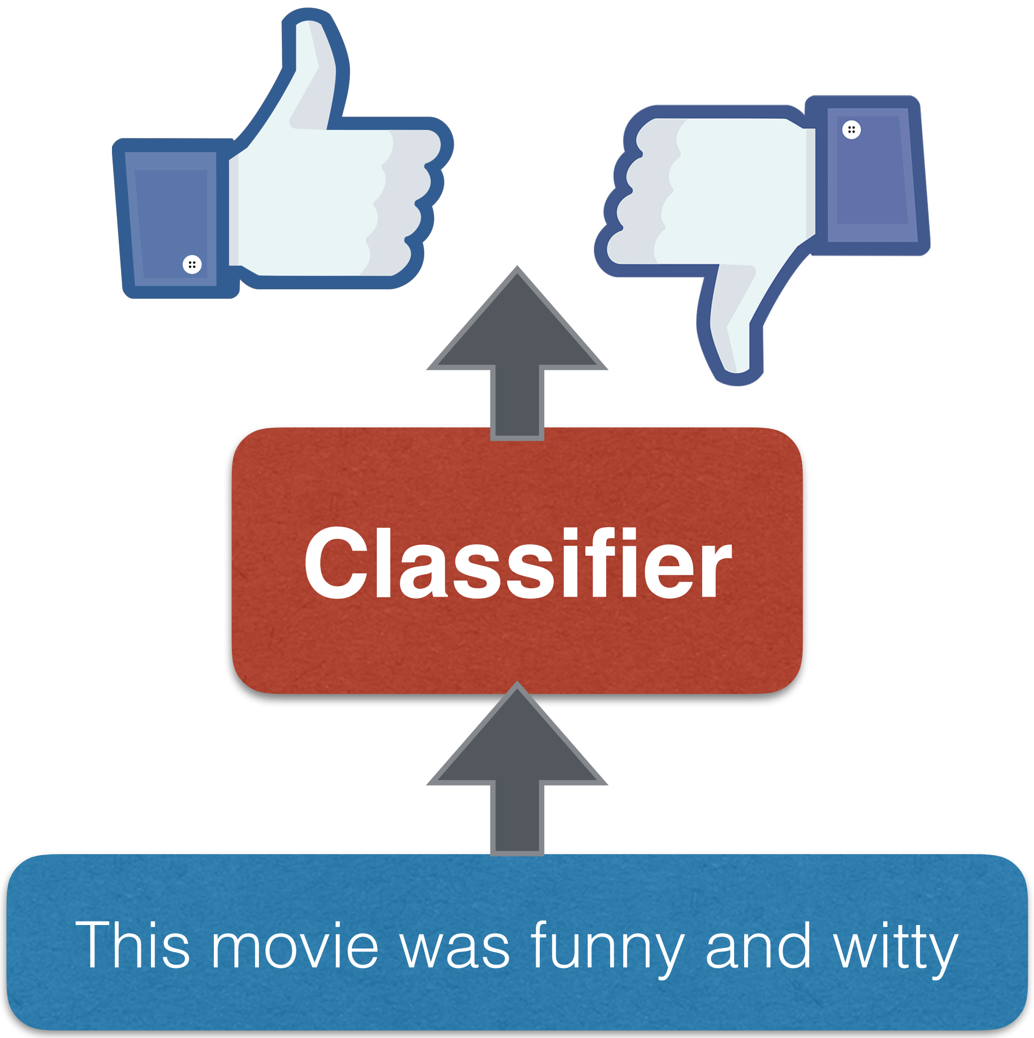
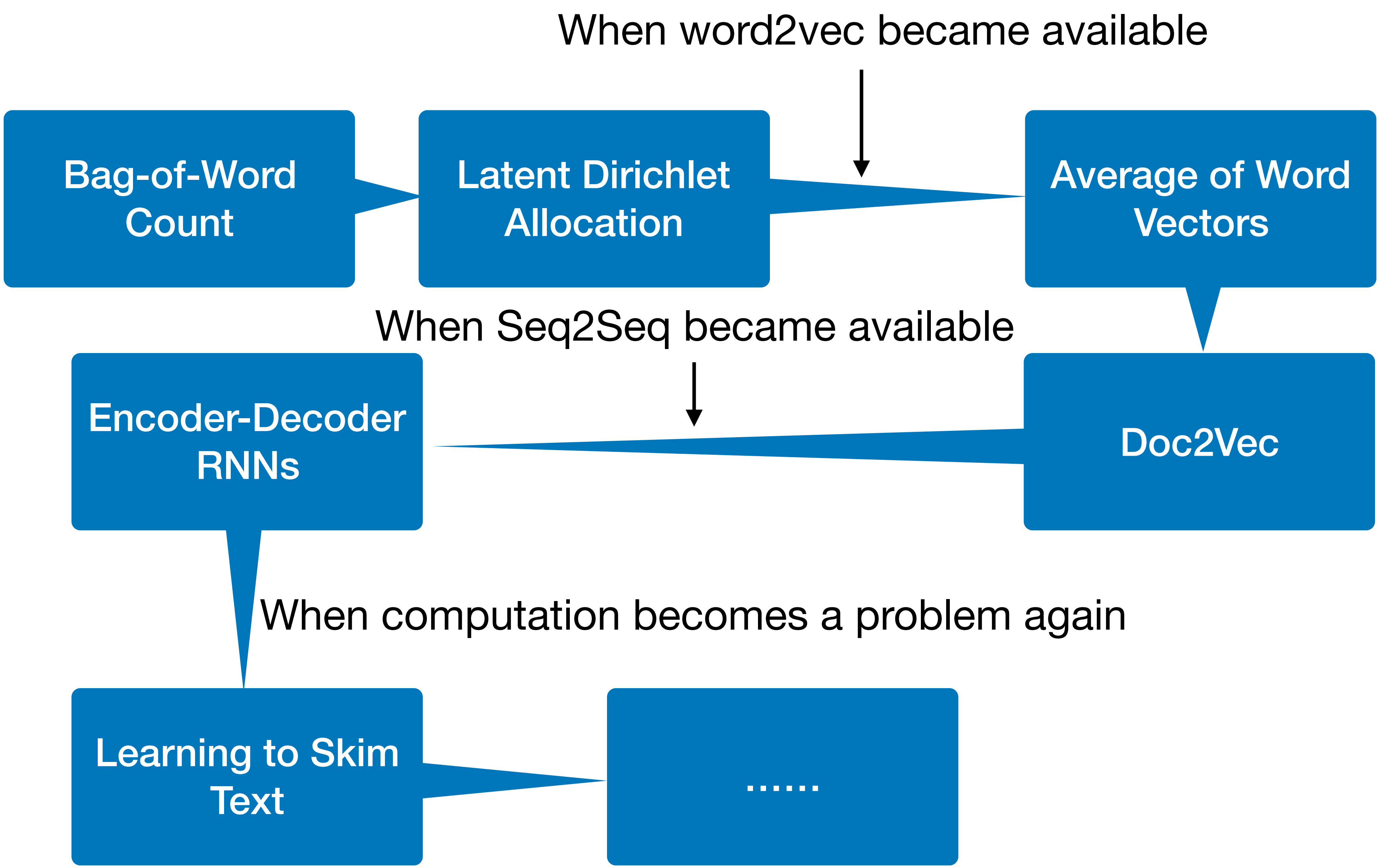
Sentence Representations

Vectors for Setences/Paragraphs

- Text Classification
- Text Summarization
- Question Answering
- Information Retrieval
- …



A Long History….



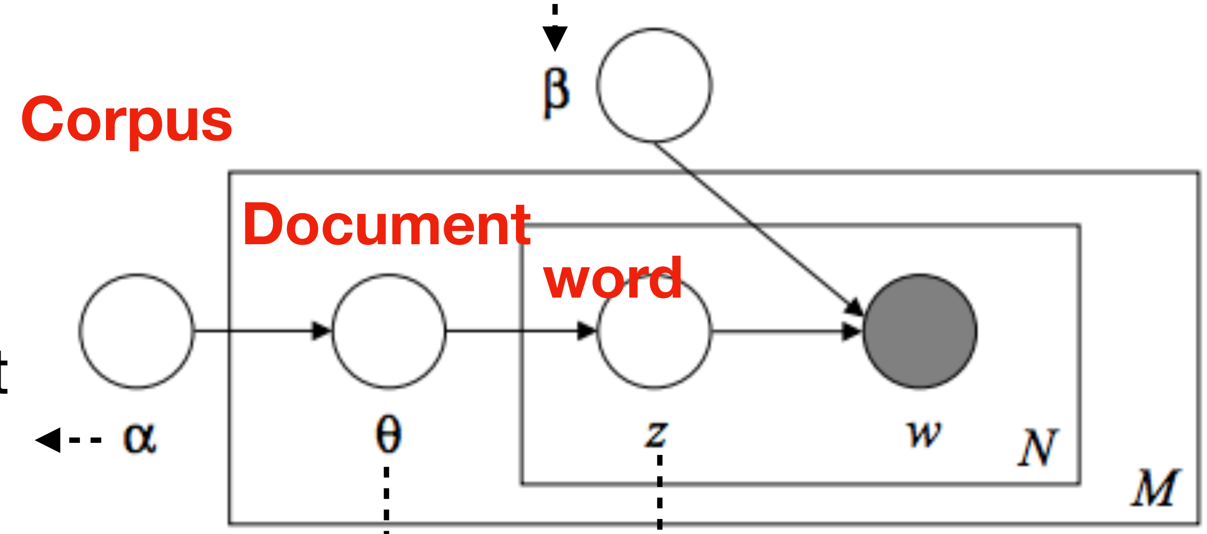
Bag-of-Words

- Raw count
- Tf-idf count
- Normalized count

$$\#(\cdot) \leftarrow \begin{cases} \#(\cdot) & \text{if } t = \text{—} \\ \log(1 + \#(\cdot)) & \text{if } t = \text{log} \\ \#(\cdot)^{2/3} & \text{if } t = \text{two-thirds} \\ \sqrt{\#(\cdot)} & \text{if } t = \text{sqrt} \end{cases}$$

Latent Dirichlet Allocation

LDA assumes a generative process for each document in a corpus.

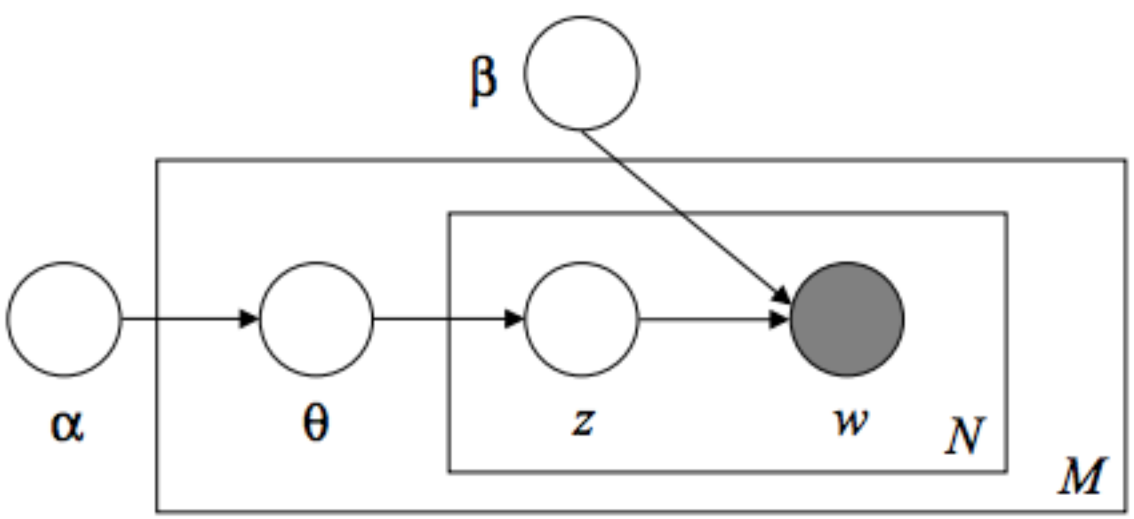


Reference: http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

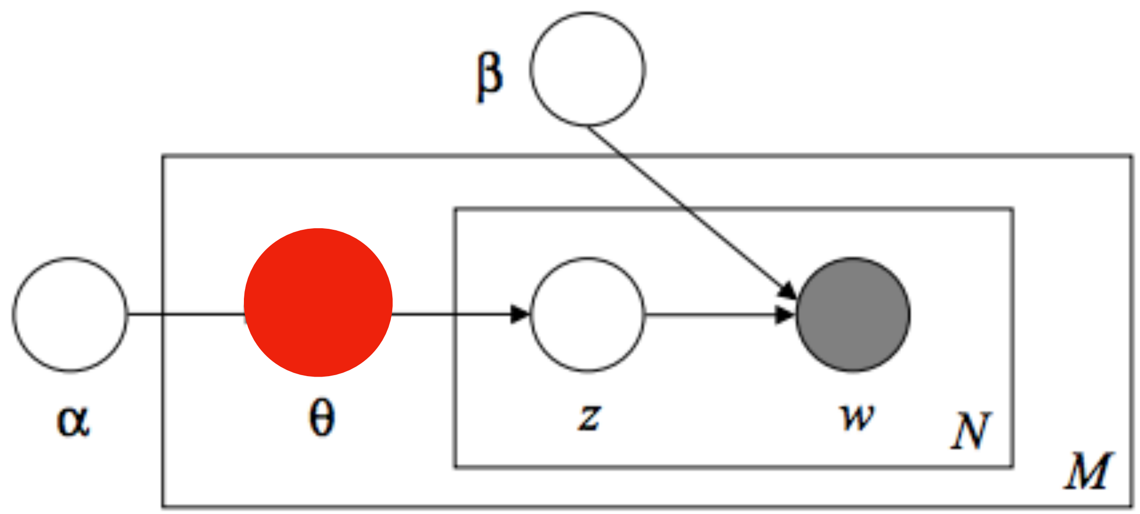
Latent Dirichlet Allocation

Generative Process:

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the N words w_n:
    (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
    (b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.



Vector Representations in LDA



The topic distribution for this document can be viewed as the vector representation.

Average of Word Vectors

- Pure Average

$$v(s) = \frac{1}{|s|} \sum_{w \in s} v(w)$$

- Weighted Average
  - tf-idf

$$v_{\mathrm{tf-idf}}(s) = \sum_{w \in s} \mathrm{tfidf}(w; s) v(w)$$

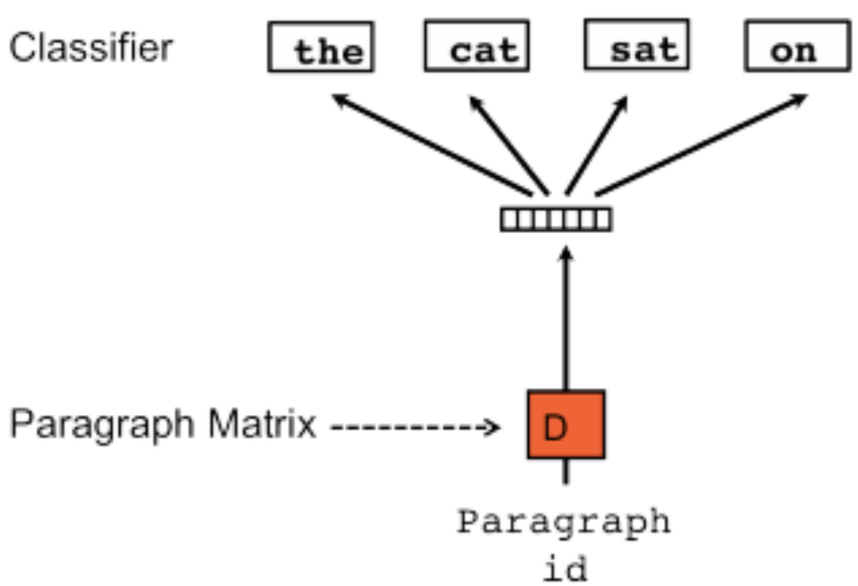
  - Soft-Inverse Frequency

$$v_{\mathrm{sif}}(s) = \sum_{w \in s} \frac{a}{a + p(w)} v(w), \qquad a \in [10^{-4}, 10^{-3}]$$

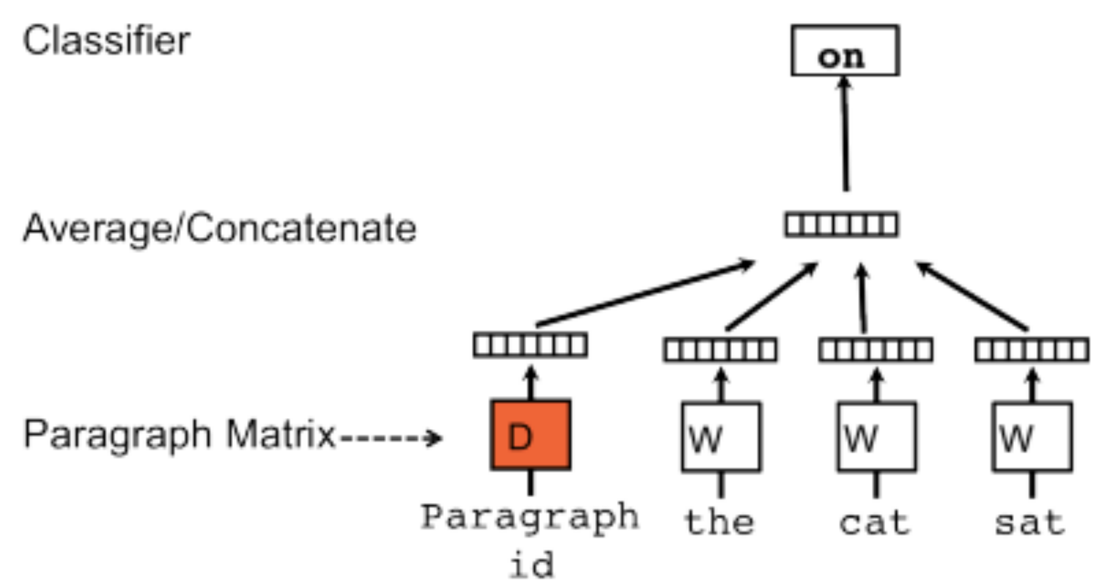
Reference: https://openreview.net/pdf?id=SyK00v5xx

Doc2Vec

Doc2Vec leverages the idea of word2vec.



PV-DBOW

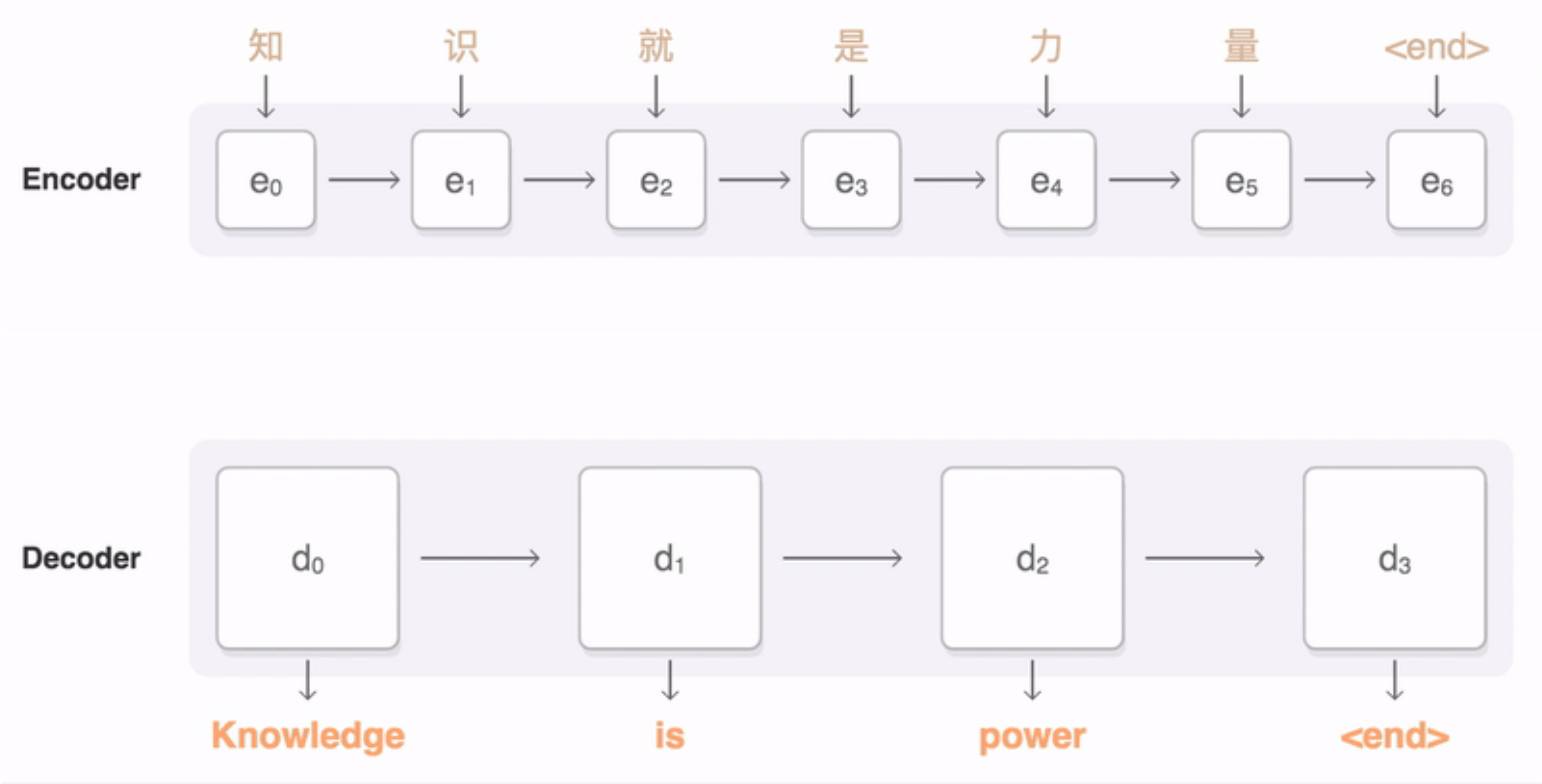


PV-DM

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

Encoder-Decoder

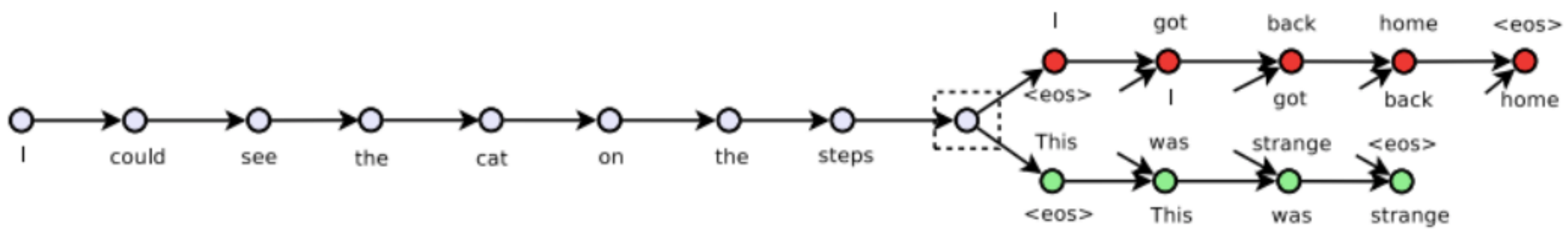
Everything starts with machine translation…



Reference: https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

Skip-Thought Vectors

Use current sentence to predict surrounding sentences.



Reference: https://arxiv.org/pdf/1506.06726.pdf

Skip-Thought Vectors

Encode the current sentence

$$h_i^{(t)} = \mathrm{RNN}(w_i^{(t)}, h_i^{(t-1)})$$

Vector for current sentence

$$h_i = h_i^{(-1)}$$

Decode previous/after sentences

$$\hat{h}_j^{(t)} = \mathrm{RNN}\left(\left[w_j^{(t-1)}, h_i\right], \hat{h}_j^{(t-1)}\right)$$

$$P(w_j^{(t)} | w_j^{(<t)}, h_i) \propto \exp(v(w_j^{(t)})^{\mathrm{T}} \hat{h}_j^{(t)})$$

Training objective:

$$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i)$$

Reference: https://arxiv.org/pdf/1506.06726.pdf

FastSent

RNN is too time-consuming..
Representing sentences by the sum of its words

S_{i-1} S_i S_{i+1}

$$s_i = \sum_{w \in S_i} v(w)$$

Training objective: $$\sum_{w \in S_{i-1} \cup S_{i+1}} \phi(\mathbf{s_i}, v_w)$$

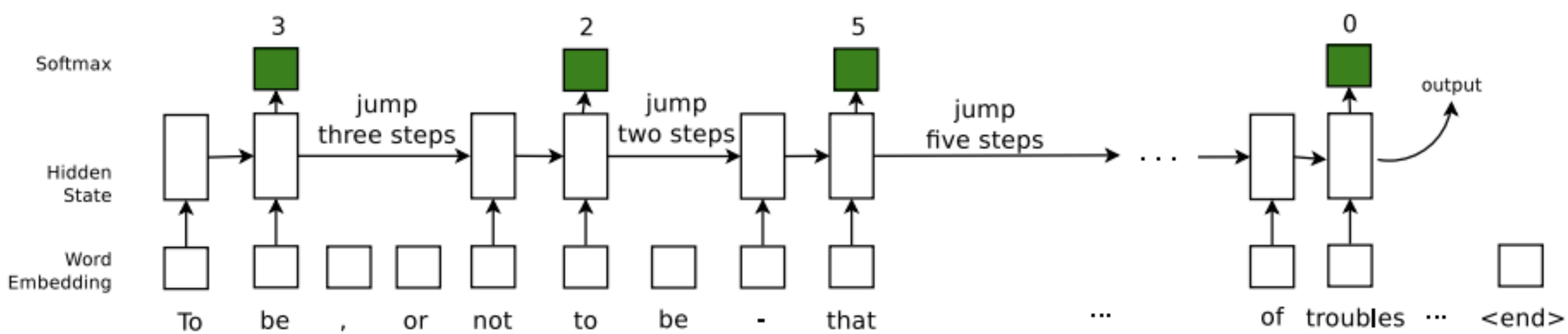
Another training objective: $$\sum_{w \in S_{i-1} \cup S_i \cup S_{i+1}} \phi(\mathbf{s_i}, v_w).$$

$\phi(v_1, v_2)$ is the softmax function.

Reference: https://arxiv.org/pdf/1602.03483.pdf

Another Way to Speedup RNN

- No need to read documents word by word
- After reading a few words, decide how many words to jump in the next round..



Reference: https://arxiv.org/pdf/1704.06877.pdf

Other Methods

- CNN-based sentence representation

  Reference: https://arxiv.org/abs/1408.5882

- Paraphrase motivated sentence representation

  Reference: https://arxiv.org/abs/1511.08198

- Jointly embedding sentences and words

  Reference: http://www.aclweb.org/anthology/P16-1089

- ……