Nders at NTCIR-13 Short Text Conversation

Han Ni, Liansheng Lin, Ge Xu

NetDragon Websoft Inc.

Dec. 2017

System Architecture

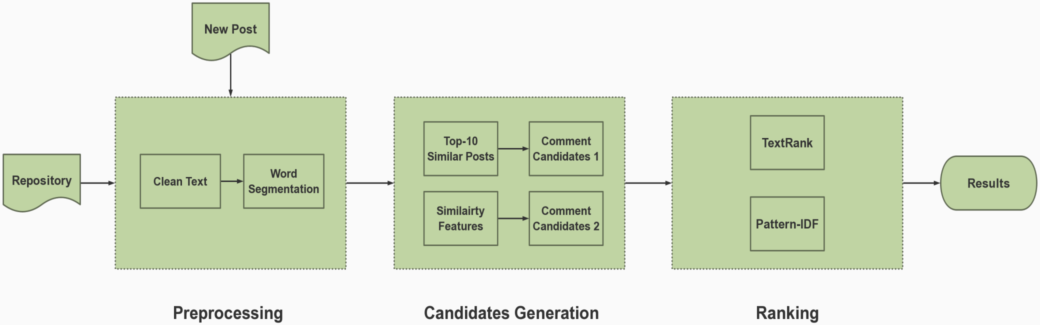

Figure 1: System Architecture

Preprocessing

- Traditional-Simplified Chinese conversion
- Convert Full-width characters into half-width ones
- Word segmentation (PKU standard)
- Replace number, time, url with token <_NUM>, <_TIME>, <_URL> respectively
- Filter meaningless words and special symbols

Short Text ID	test-post-10440
Raw Text	去到美國，还是吃中餐！宮保雞丁家的感覺～ Go to the USA, still eat Chinese food, Kung Pao Chicken, feeling like at home
Without T-S Conversion	去 到 美 國 , 还 是 吃 中餐 ! 宮 保 雞 丁 家 的 感 覺 ~
With T-S Conversion	去 到 美国 , 还 是 吃 中餐 ! 宫保鸡丁 家 的 感觉 ~
Clean Result	去 到 美国 还 是 吃 中餐 宫保鸡丁 家 的 感觉

Short Text ID	test-post-10640
Raw Text	汶川大地震9周年：29个让人泪流满面的瞬间。 9th Anniversary of Wenchuan Earthquake: 29 moments making people tearful
Without token replacement	汶川 大 地震 9 周年 : 29 个 让 人 泪流满面 的 瞬间 。
With token replacement	汶川 大 地震 <_NUM> 周年 : <_NUM> 个 让 人 泪流满面 的 瞬间 。
Clean Result	汶川 大 地震 <_NUM> 周年 <_NUM> 个 让 人 泪流满面 的 瞬间

Similarity Features

- TF-IDF
- LSA (Latent Semantic Analysis)
- LDA (Latent Dirichlet Allocation)
- Word2Vec (skip-gram)
- **LSTM-Sen2Vec**

We combine each post with its corresponding comments to be a document, then train LSA and LDA models on these documents.

LSTM

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * tanh(C_t) \quad (6)$$

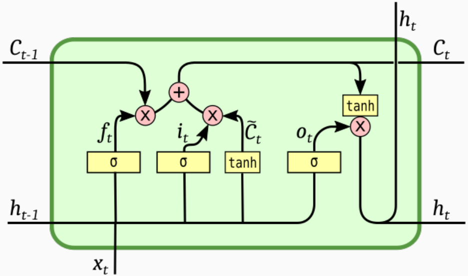

Figure 2: The LSTM Cell

Mikolov, Toma's. Statistical Language Models Based on Neural Networks. Ph.D. thesis, Brno University of Technology.(2012)

Zaremba, Wojciech, I. Sutskever, and O. Vinyals. Recurrent Neural Network Regularization. Eprint Arxiv (2014).

Attention weight

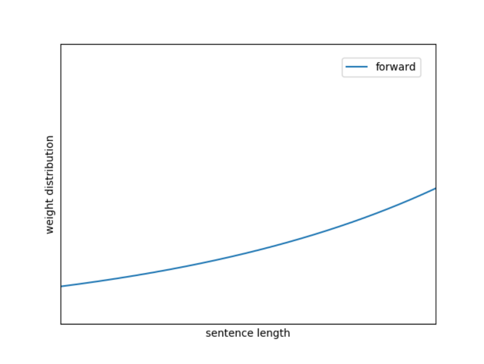

Figure 3: Unidirectional weight distribution

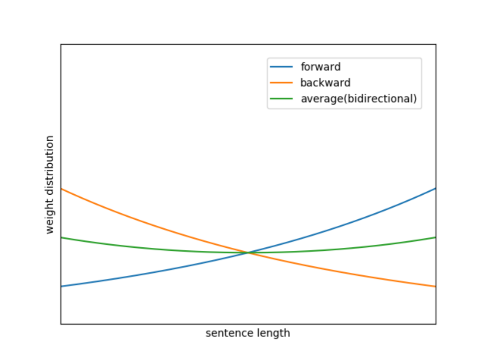

Figure 4: bidirectional weight distribution

LSTM-Sen2Vec

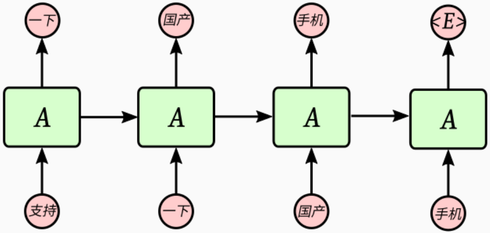

Figure 5: The Unidirectional LSTM

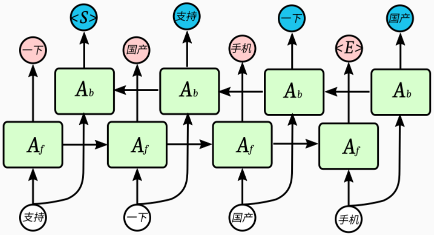

Figure 6: The Traditional Bidirectional LSTM

LSTM-Sen2Vec

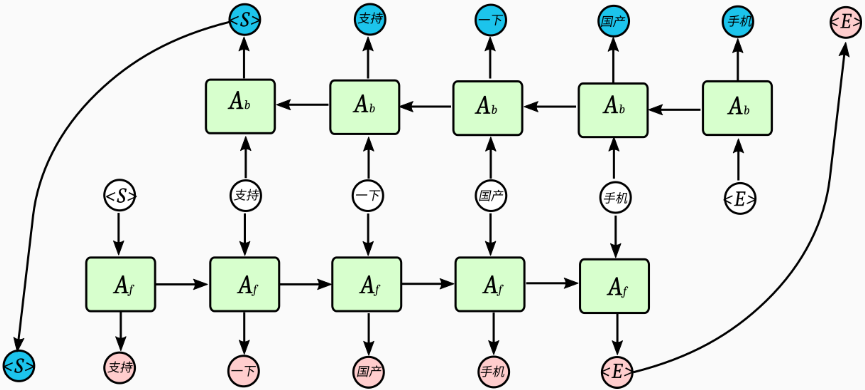

Figure 7: The Modified Bidirectional LSTM

Candidates Generation

- Similar Posts

$$Score^1_{q,p}(q,p) = Sim_{LDA}(q,p) * Sim_{W2V}(q,p) * Sim_{LSTM}(q,p) \tag{7}$$

$$Score^2_{q,p}(q,p) = Sim_{LSA}(q,p) * Sim_{W2V}(q,p) * Sim_{LSTM}(q,p) \tag{8}$$

- Comment Candidates

$$Score^1_{q,c}(q,c) = Sim_{LSA}(q,c) * Sim_{W2V}(q,c) \tag{9}$$

$$Score^2_{q,c}(q,c) = Sim_{LDA}(q,c) * Sim_{W2V}(q,c) \tag{10}$$

Ranking

- TextRank (Words as vertices)
- Pattern-IDF
- Pattern-IDF + TextRank (Sentences as vertices)

Formally, let $G = (V; E)$ be a undirected graph with the set of vertices V and and set of edges E, where E is a subset of $V \times V$. For a given V_i, let $link(V_i)$ be the set of vertices that linked with it. The score of a vertex Vi is define as follow:

$$WS(V_i) = (1 - d) + d * \sum_{j \in link(V_i)} w_{ij} * WS(V_j) \tag{11}$$

Where d is a damping factor[1] that is usually set to 0.85.

[1] Brin, Sergey, and L. Page. The anatomy of a large-scale hypertextual Web search engine. International Conference on World Wide Web Elsevier Science Publishers B. V. 1998:107-117.

TextRank - Vertices and Edges

- Vertices: each unique word in candidates
- Edges: a co-occurrence relation
- Weighted by: word2vec similarity between two words and the number of their cooccurrences

TextRank - Calculate Iteratively

For N candidates, k words in total, we construct $k \times k$ matrix M.
$M_{ij} = cnt * sim(D_i, D_j)$. Then we compute iteratively

$$R(t+1) = \begin{bmatrix} (1-d)/k \\ (1-d)/k \\ \ldots\ldots\ldots \\ (1-d)/k \end{bmatrix} + d \begin{bmatrix} M_{11} & M_{12} & M_{13} & \ldots & M_{1k} \\ M_{21} & M_{22} & M_{23} & \ldots & M_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{k1} & M_{k2} & M_{k3} & \ldots & M_{kk} \end{bmatrix} R(t)$$

Stop when $|R(t+1) - R(t)| < \epsilon$, $\epsilon = 10^{-7}$.
Here, cnt refers to the number of co-ocurrences within a sentence for D_i and D_j.

TextRank - Ranking

Since we get the score $R(D_i)$ for each word D_i in candidates, the score for each comment candidate c is calculated as:

$$Rank_{TextRank}(c) = \frac{\sum_{D_i \in c} R(D_i)}{len(c)} \tag{12}$$

Here, $len(c)$ refers to the number of words in comment c.

Pattern-IDF

For word D_i (minor word) in corresponding comment given word D_j (major word) in the post, we define (D_j, D_i) as a pattern.

Inspired by the IDF, we calculate the Pattern-IDF as:

$$PI(D_i|D_j) = 1/\log_2 \frac{count_c(D_i) * count_p(D_j)}{count_{pair}(D_i, D_j)} \tag{13}$$

Here $count_c$ refers to the number of occurrence in comments, $count_p$ in posts, $count_{pair}$ in post-comment pair. The PI whose $count_{pair}(D_i, D_j)$ less than 3 are eliminated.

Pattern-IDF

Let $X = \frac{count_c(D_i) * count_p(D_j)}{count_{pair}(D_i, D_j)}$, then $X \in [1, \infty)$.

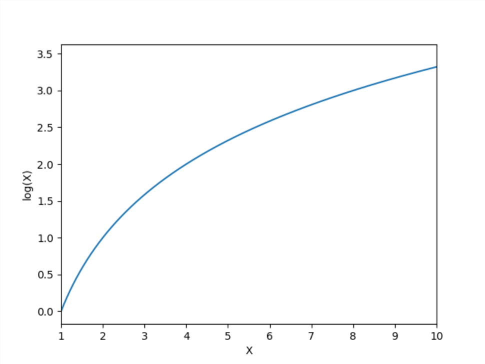

Figure 8: log(X)

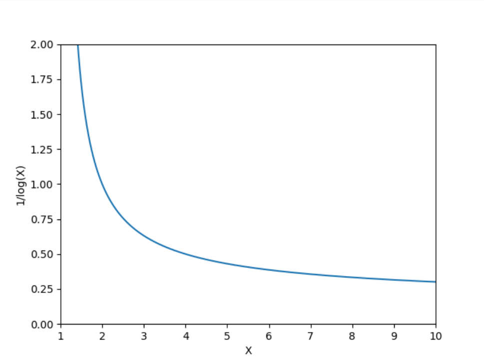

Figure 9: 1/log(x)

PI - Example

Table 1: The example of Pattern-IDF

MajorWord	*MinorWord*	*PI*
中国移动 (China Mobile)	接通 (connect)	0.071725
中国移动	cmcc	0.067261
中国移动	资费 (charges)	0.062408
中国移动	营业厅 (business hall)	0.059949
中国移动	漫游 (roamimg)	0.059234
...	...	...
中国移动	我 (me)	0.028889
中国移动	是 (be)	0.027642
中国移动	的 (of)	0.026346

Table 2: The entropy of Pattern-IDF for each Major Word

MajorWord	H
眼病 (eye disease)	0.889971
丰收年 (harvest year)	0.988191
血浆 (plasma)	1.033668
脊椎动物 (vertebrate)	1.083438
水粉画（gouache painting）	1.180993
...	...
现在 (now)	9.767768
什么 (what)	10.219045
是 (be)	10.934950

$$PI_{norm}(D_i|D_j) = \frac{PI(D_i|D_j)}{\sum_{i=1}^{n} PI(D_i|D_j)} \tag{14}$$

$$H(D_j) = -\sum_{i=1}^{n} PI_{norm}(D_i|D_j) \log_2 PI_{norm}(D_i|D_j) \tag{15}$$

PI - Ranking

For each comment c in candidates, given a query (new post) q, we calculate the score by PI as follow:

$$Score_{PI}(q, c) = \frac{\sum_{D_j \in q} \sum_{D_i \in c} PI(D_i|D_j)}{len(c) * len(q)} \tag{16}$$

Then we define rank score as follow:

$$Rank_{PI} = (1 + \frac{Score_{PI}(q, c)}{\max Score_{PI}(q, c)}) * Sim_{W2V}(q, c) * Sim_{LSA}(q, c) \tag{17}$$

TextRank + Pattern-IDF

In this method, We add each comment sentence in candidates as a vertex in the graph and use sentence Word2Vec similarity as edges between vertices in the graph.

For N candidates, we construct $N \times N$ matrix M.
$M_{ij} = Sim_{w2v}(candidate_i, candidate_j)$.

At time $t = 0$, We initiate a N-dimension vector P , here N is the number of comment candidates. And each entry of P is defined as the score of Pattern-IDF between the query (new post) q and corresponding comment c_i in candidates:

$$P_i = Score_{PI}(q, c_i) \tag{18}$$

TextRank + Pattern-IDF

Then we compute iteratively

$$R(t+1) = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \dots\dots\dots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} M_{11} & M_{12} & M_{13} & \dots & M_{1N} \\ M_{21} & M_{22} & M_{23} & \dots & M_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{N1} & M_{N2} & M_{N3} & \dots & M_{NN} \end{bmatrix} R(t)$$

Stop when $|R(t+1) - R(t)| < \epsilon$, $\epsilon = 10^{-7}$

Finally, we get the score P_i for each comment in candidates.

Experiment

- Nders-C-R5: LDA + Word2Vec + LSTM-Sen2Vec
- Nders-C-R4: LSA + Word2Vec + LSTM-Sen2Vec
- Nders-C-R3: R4 + TextRank (Words as vertices)
- Nders-C-R2: R4 + Pattern-IDF
- Nders-C-R1: R4 + Pattern-IDF + TextRank (Sentences as vertices)

Official Result

Table 3: The official results of five runs for Nders team

Run	Mean nG@1	Mean P+	Mean nERR@10
Nders-C-R1	0.4593	0.5394	0.5805
Nders-C-R2	0.4743	**0.5497**	**0.5882**
Nders-C-R3	0.4647	0.5317	0.5768
Nders-C-R4	**0.4780**	0.5338	0.5809
Nders-C-R5	0.4550	0.5495	0.5868
R2 vs. R4	↓0.77%	↑1.26%	↑2.02%

Questions?