实习实践报告：短文本对话

网龙网络公司
NETDRAGON WEBSOFT INC.

报告人：林连升
校内导师：叶东毅 教授
企业导师：关胤
实习单位：福建天晴在线互动科技有限公司

短文本对话

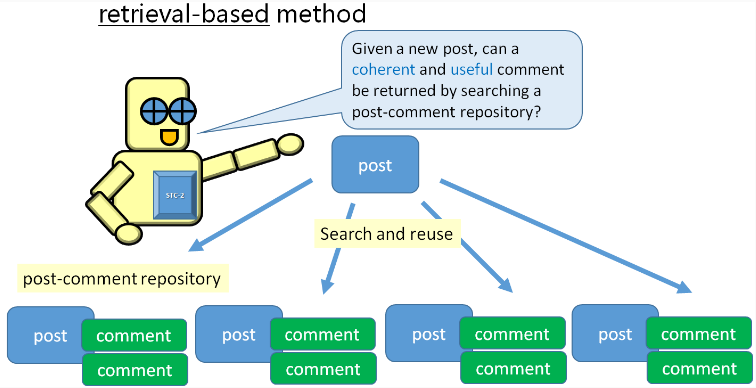

Figure 1: 基于检索的方法

相关性评估

评估检索出的评论主要遵循以下四个准则:

1. Fluent
2. Coherent: logically and topically relevant
3. Self-sufficient
4. Substantial

If either (1) or (2) is untrue, the retrieved comment should be labeled "L0"; if either (3) or (4) is untrue, the label should be "L1"; otherwise, the label is "L2".

Labels

Post	意大利禁区里老是八个人...太夸张了吧 There are always 8 Italian players in their own restricted area…Unbelievable!	Related Criteria	Labels
Comment1	我是意大利队的球迷，等待比赛开始。 I am a big fan of the Italy team, waiting for the football match to start	(2) Coherent	L0
Comment2	意大利的食物太美味了 Italian food is absolutely delicious.	(2) Coherent	L0
Comment3	太夸张了吧! Unbelievable!	(4) Substantial	L1
Comment4	哈哈哈仍然是0：0。还没看到进球。 Haha, it is still 0:0, no goal so far.	(3) Self-sufficient	L1
Comment5	这正是意大利式防守足球。 This is exactly the Italian defending style football game	——	L2

Figure 2: 一条微博以及人工标注的五条候选评论

System Architecture

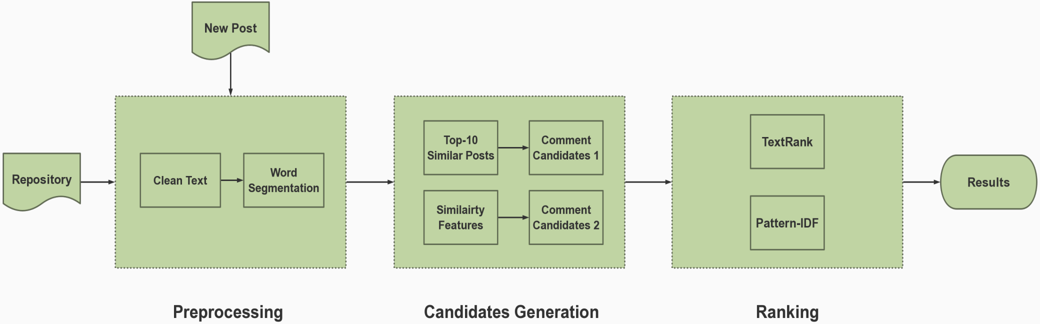

Figure 3: System Architecture

文本预处理

- 繁简转换
- 全角转半角
- 分词（pynlpir, PKU 标准）
- token 替换（<_NUM>, <_TIME>, <_URL>）
- 过滤无意义的词和特殊符号

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

相似度特征

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

我们将每条微博和它的所有评论合并成一个文档，然后对这些文档训练 LSA 和 LDA 模型。其他模型是以句子作为输入。

LSTM

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

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

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

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

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

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

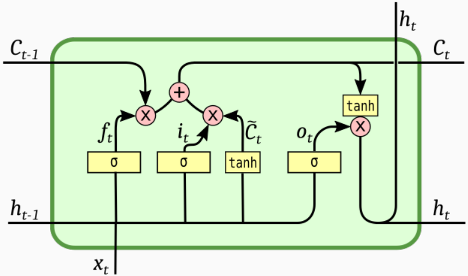

Figure 4: LSTM 单元

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

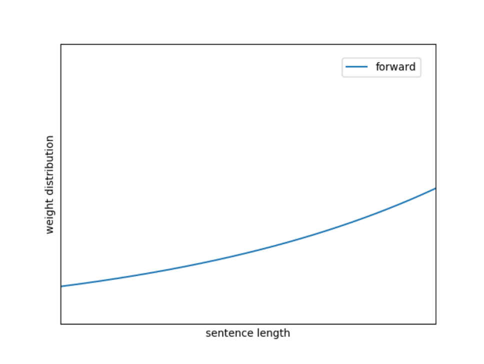

Figure 5: Unidirectional weight distribution

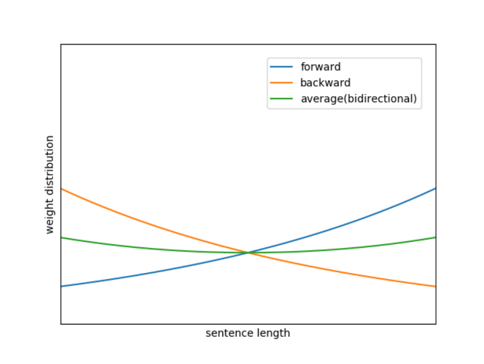

Figure 6: bidirectional weight distribution

LSTM-Sen2Vec

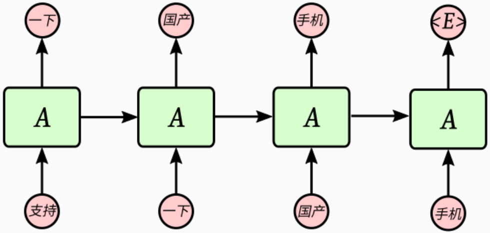

Figure 7: The Unidirectional LSTM

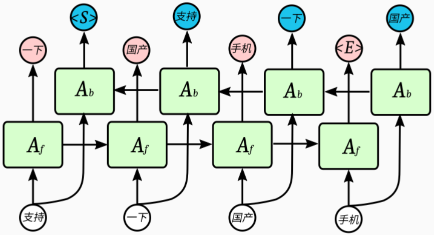

Figure 8: The Traditional Bidirectional LSTM

LSTM-Sen2Vec

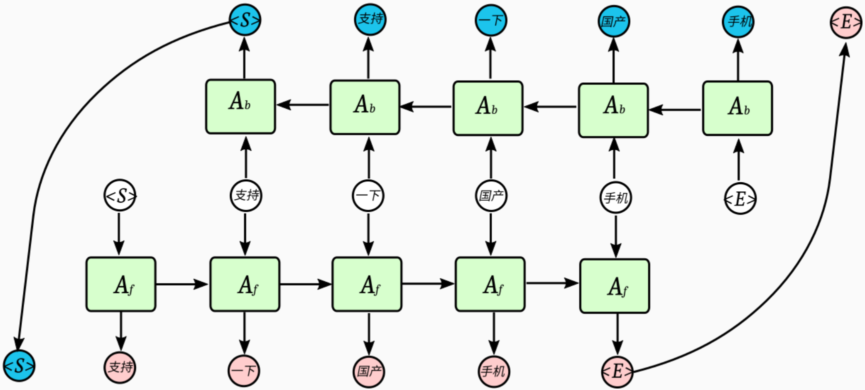

Figure 9: The Modified Bidirectional LSTM

生成候选集

- **相似的微博**

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

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

- **候选评论**

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

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

排序

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

TextRank - 一个基于图的排序算法

设 $G = (V; E)$ 为一个由点集 V 和边集 E 构成的无向图，其中 E 是 $V \times V$ 的子集. 对于一个给定的点 V_i, 设 $link(V_i)$ 为与其相连的点的集合. 则一个点 V_i 的分数定义如下:

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

其中 d 是一个 damping factor[1]，通常设为 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: 候选评论中的每个词
- Edges: 共现关系
- Weighted by: word2vec 相似度和共现次数

对于 N 个候选评论, k 个词语, 我们构建一个 $k \times k$ 的矩阵 M. 其中 $M_{ij} = cnt * sim(D_i, D_j)$. 然后我们迭代计算

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

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

这里, cnt 表示 D_i 和 D_j 在一个句子中共现的次数.

既然我们得到候选评论中每个词 D_i 的分数 $R(D_i)$, 那么每个候选评论 c 的分数可以按下列式子计算:

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

这里, $len(c)$ 表示候选评论中词语的个数.

Pattern-IDF

对于微博中的一个词语 D_j 和相应评论中的一个词语 D_i，我们定义 (D_j,D_i) 为一个 pattern.

受到 TF-IDF 的启发，我们定义 Pattern-IDF 为:

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

这里， $count_c$ 表示词语出现在评论中的次数，$count_p$ 表示词语出现在微博中的次数，$count_{pair}$ 表示 (D_j,D_i) 的个数. $count_{pair}(D_i, D_j)$ 小于 3 的 PI 将被移除.

Pattern-IDF

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

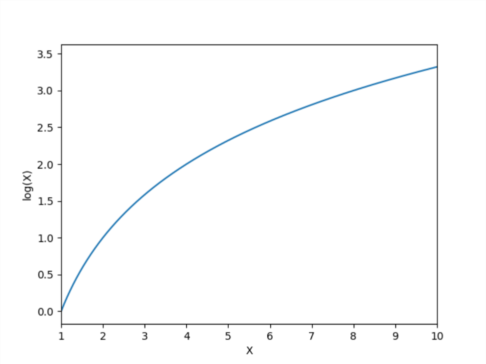

Figure 10: log(X)

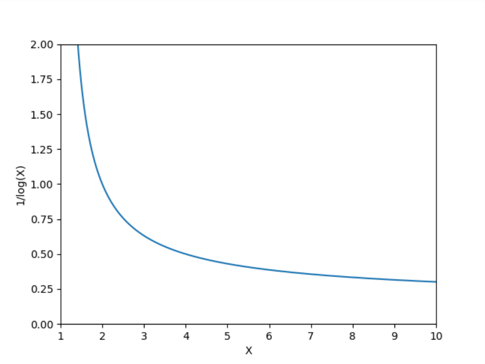

Figure 11: 1/log(x)

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

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

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

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:

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

Then we compute iteratively

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

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)

官方结果

Table 3: The official results

Run	Mean nG@1	Mean P+	Mean nERR@10
Team 1	0.5867	0.6670	0.7095
Team 2	0.5080	0.6080	0.6492
Team 3	0.4980	0.5818	0.6105
Nders-C-R1	0.4593	0.5394	0.5805
Nders-C-R2	0.4743	**0.5497(5th)**	**0.5882(5th)**
Nders-C-R3	0.4647	0.5317	0.5768
Nders-C-R4	**0.4780(4th)**	0.5338	0.5809
Nders-C-R5	0.4550	0.5495	0.5868
R2 vs. R4	↓0.77%	↑2.98%	↑1.26%

Questions?