Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information

ACL 2018 Best Paper

Sudha RaoHal Daume´ III

University of Maryland, College Park,

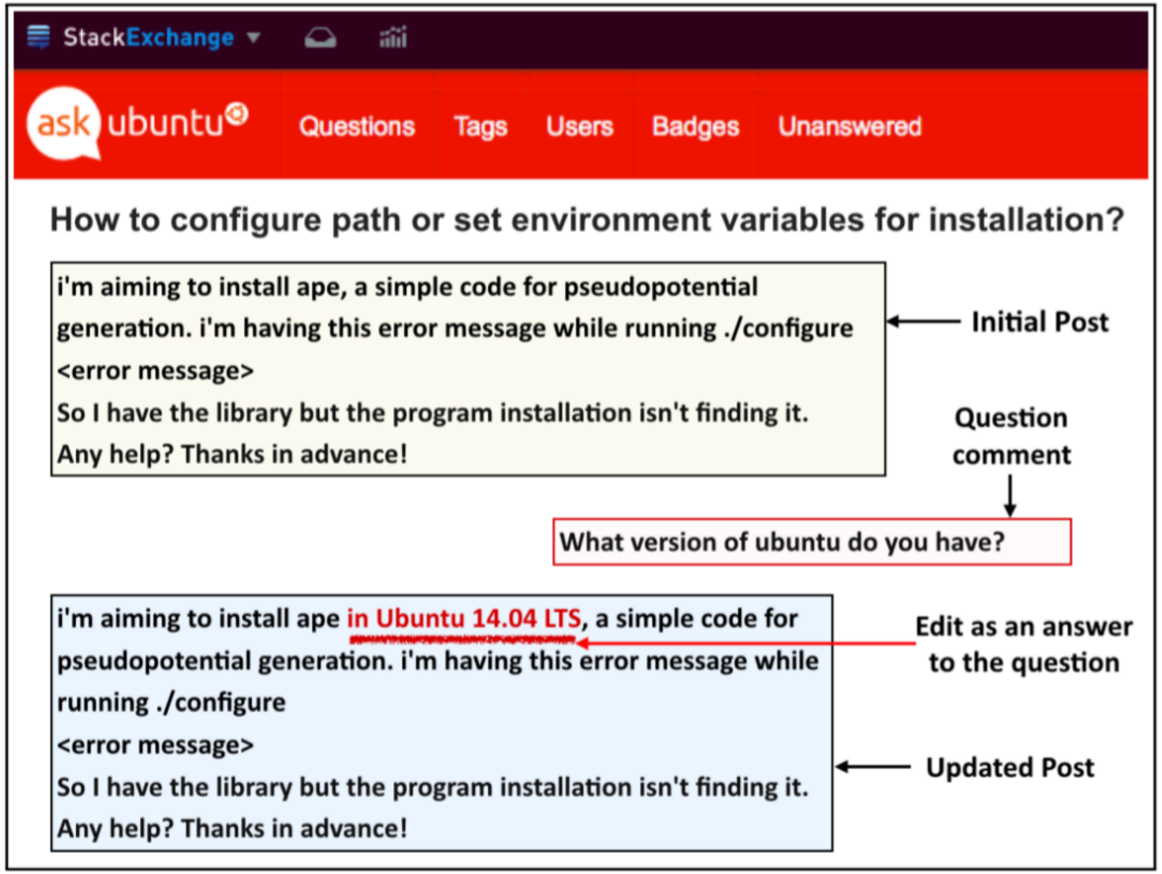
University of Maryland, College Park , Microsoft Research, New York City

Motivation

In some qa scenes， post is so little information that it's hard to answer.

Teach machine to ask those clarification questions : design a model to rank a candidate set of clarification questions by their usefulness to the given post

Possible use case ∶ while user writing their post, a system suggests a shortlist of questions asking for information.



Contribution

- 1. A novel neural-network model for addressing the task of ranking clarification question built on the framework of expected value of perfect information
- 2. A novel dataset, derived from StackExchange, that enables us to learn a model to ask clarifying questions by looking at the types of questions people ask.

Model description

- Inspired by the theory of expected value of perfect information (EVPI)
- EVPI is a measurement of: if I were to acquire information X, how useful would that be to me?

$$\mathbf{EVPI}(q_i|p) = \sum_{a_j \in A} \mathbb{P}[a_j|p, q_i]\mathbb{U}(p + a_j)$$

A good question is the one whose
***likely answer* will be useful!**

Model description

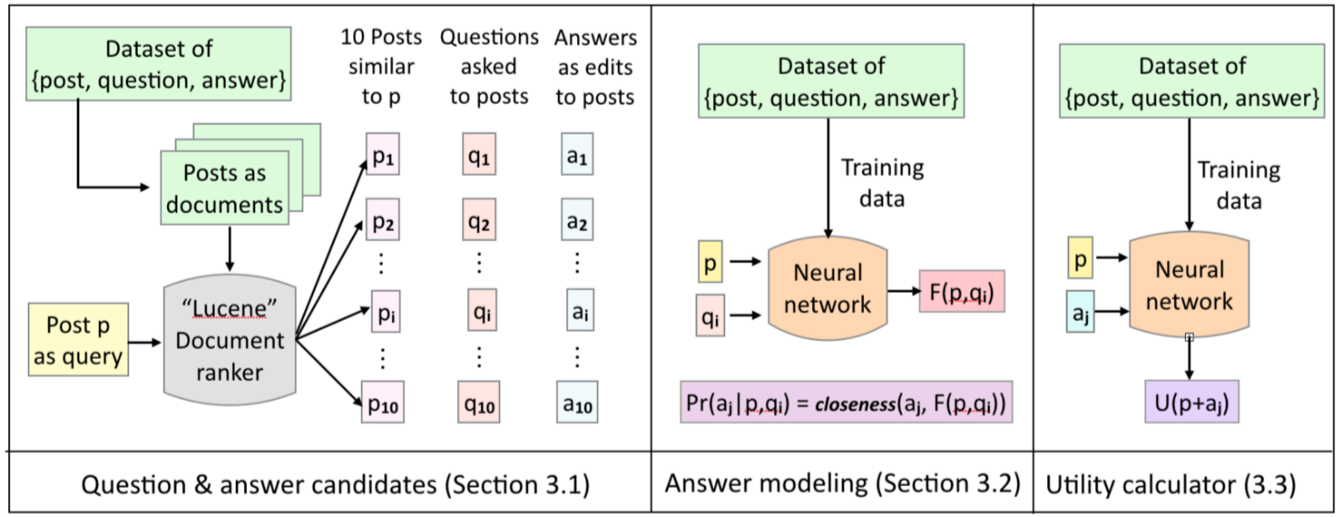


Figure 2: The behavior of our model during test time: Given a post p, we retrieve 10 posts similar to post p using Lucene. The questions asked to those 10 posts are our question candidates Q and the edits made to the posts in response to the questions are our answer candidates A. For each question candidate q_i, we generate an answer representation $F(p, q_i)$ and calculate how close is the answer candidate a_j to our answer representation $F(p, q_i)$. We then calculate the utility of the post p if it were updated with the answer a_j. Finally, we rank the candidate questions Q by their expected utility given the post p (Eq 1).

Model description

1.Question & answer candidate generator

2. Answer modeling

$$dist(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j) = 1 - cos\_sim(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j)$$

$$\mathbb{P}[a_j|p, q_i] = \exp^{-dist(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j)} * cos\_sim(\hat{q}_i,\ \hat{q}_j) \tag{2}$$

3. Utility calculator

$$\mathbb{U}(p_i + a_j) = \sigma(F_{util}(\bar{p}_i, \bar{q}_j, \bar{a}_j))$$

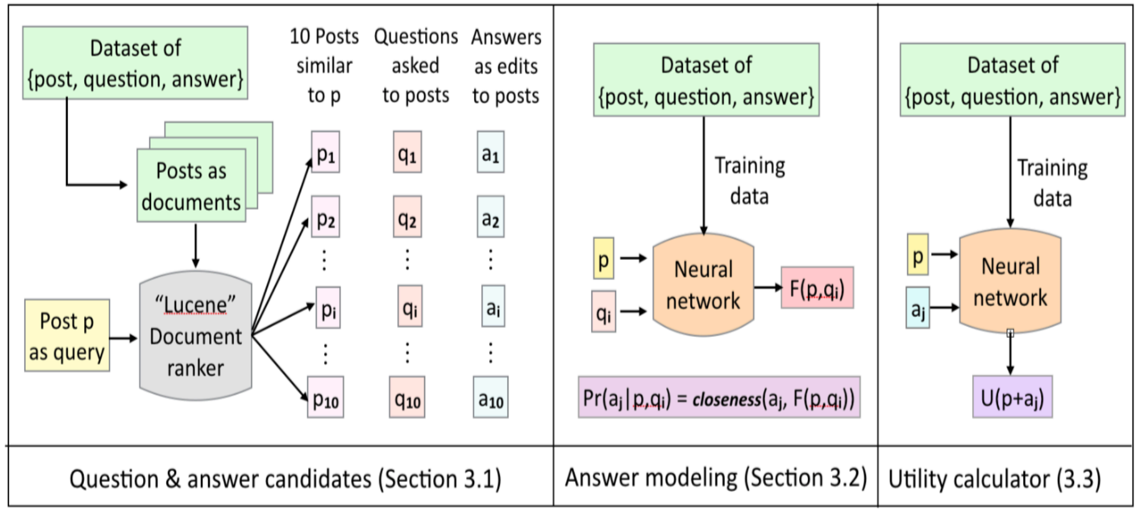
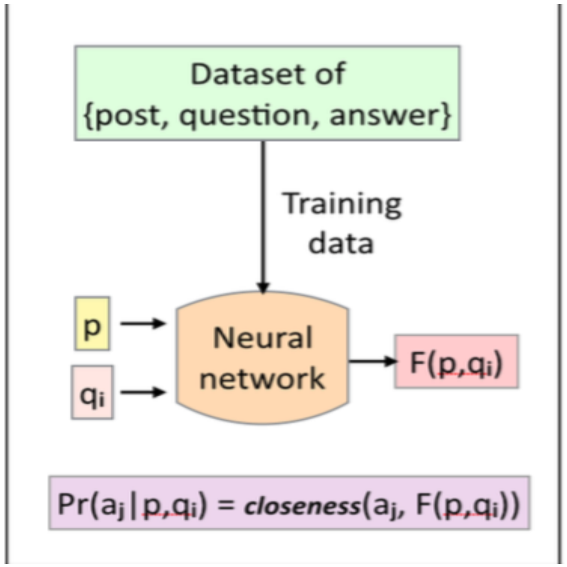


Figure 2: The behavior of our model during test time: Given a post p, we retrieve 10 posts similar to post p using Lucene. The questions asked to those 10 posts are our question candidates Q and the edits made to the posts in response to the questions are our answer candidates A. For each question candidate q_i, we generate an answer representation $F(p, q_i)$ and calculate how close is the answer candidate a_j to our answer representation $F(p, q_i)$. We then calculate the utility of the post p if it were updated with the answer a_j. Finally, we rank the candidate questions Q by their expected utility given the post p (Eq 1).

Training

Answer modeling:

$$\text{loss}_{\text{ans}}(p_i, q_i, a_i, Q_i) = dist(F_{ans}(\bar{p}_i, \bar{q}_i),\ \hat{a}_i) + \sum_{j \in Q} \Big(dist(F_{ans}(\bar{p}_i, \bar{q}_i),\ \hat{a}_j) * cos\_sim(\hat{q}_i,\ \hat{q}_j) \Big)$$



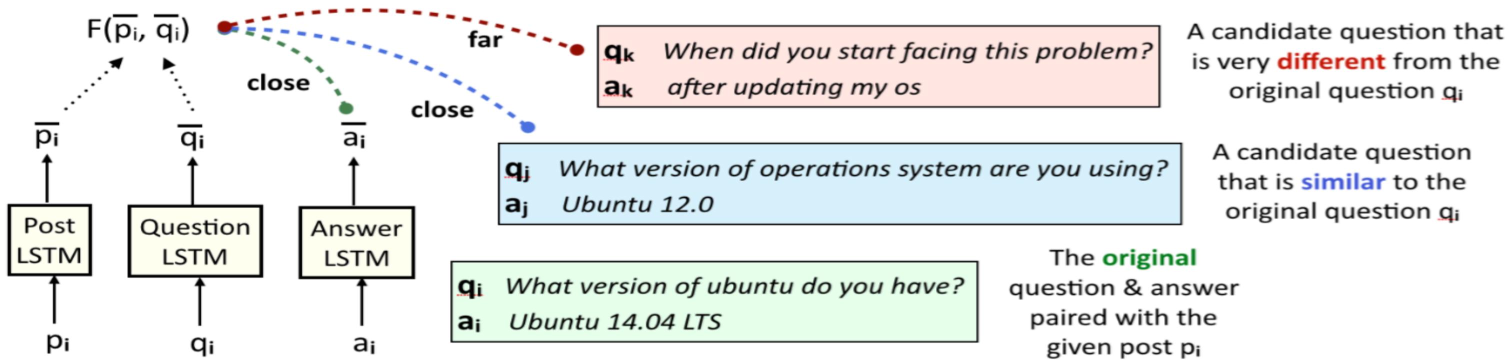
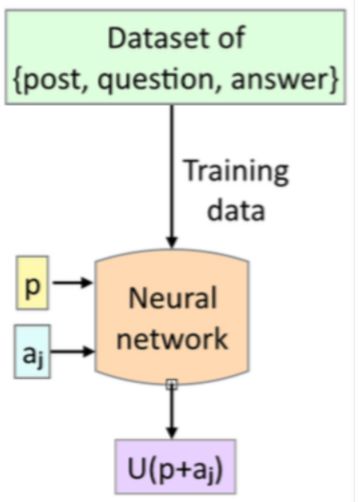


Figure 3: Training of our answer generator. Given a post p_i and its question q_i, we generate an answer representation that is not only close to its original answer a_i, but also close to one of its candidate answers a_j if the candidate question q_j is close to the original question q_i.

Training

Utility calculator :

$$\mathrm{loss}_{\mathrm{util}}(y_i, \bar{p}_i, \bar{q}_j, \bar{a}_j) = y_i \log(\sigma(F_{util}(\bar{p}_i, \bar{q}_j, \bar{a}_j))) \tag{4}$$



joint neural network model

$$\sum_i \sum_j \mathrm{loss}_{\mathrm{ans}}(\bar{p}_i, \bar{q}_i, \bar{a}_i, Q_i) + \mathrm{loss}_{\mathrm{util}}(y_i, \bar{p}_i, \bar{q}_j, \bar{a}_j)$$

Data creation

- 1. Extract posts
- 2. Extract questions
- 3. Extract answers

    (a) Edited post

    (b) Response to the question

Experimental results

Model	$B1 \cup B2$ p@1	p@3	p@5	MAP	$V1 \cap V2$ p@1	p@3	p@5	MAP	Original p@1
Random	17.5	17.5	17.5	35.2	26.4	26.4	26.4	42.1	10.0
Bag-of-ngrams	19.4	19.4	18.7	34.4	25.6	27.6	27.5	42.7	10.7
Community QA	23.1	21.2	20.0	40.2	33.6	30.8	29.1	47.0	18.5
Neural (p, q)	21.9	20.9	19.5	39.2	31.6	30.0	28.9	45.5	15.4
Neural (p, a)	24.1	**23.5**	20.6	41.4	32.3	**31.5**	29.0	46.5	18.8
Neural (p, q, a)	25.2	**22.7**	**21.3**	42.5	34.4	**31.8**	**30.1**	47.7	20.5
EVPI	**27.7**	**23.4**	**21.5**	**43.6**	**36.1**	**32.2**	**30.5**	**49.2**	**21.4**