JMLR: Workshop and Conference Proceedings 20 (2011) 351–366 Asian Conference on Machine Learning

Summarization of Yes/No Questions Using a Feature Function Model

Jing He HEJING2929@GMAIL.COM and **Decheng Dai** DECHENG@GOOGLE.COM
Google Inc.

Editor: Chun-Nan Hsu and Wee Sun Lee

Abstract

Answer summarization is an important problem in the study of Question and Answering. In this paper, we deal with the general questions with "Yes/No" answers in English. We design 1) a model to score the relevance of the answers and the questions, and 2) a feature function combining the relevance and opinion scores to classify each answer to be "Yes", "No" or "Neutral". We combine the opinion features together with two weighting scores to solve this problem and conduct experiments on a real word dataset. Given an input question, the system firstly detects if it can be simply answered by "Yes/No" or not, and then outputs the resulting voting numbers of "Yes" answers and "No" answers to this question. We also first proposed the accuracy, precision, and recall to the "Yes/No" answer detection.

Keywords: Answer Summarization, Yes/No Questions

1. Introduction

Search engines and community question answering(CQA) sites have been very popular recently, and are used by a large amount of users every day to find out the answers to a variety of questions. In the most common question-and-answering sites such as Yahoo! Answers and WikiAnswers, a user posts a question and waits for other users to answer it. Also, the user can search for all the answers to a certain question in the system and get tremendous results. As there are too many search results and existing answers to a question, answer summarization becomes a key problem in the task of answer retrieval(see (Liu et al., 2008; Filho et al., 2006; Tang et al., 2010)).

During the past few years, document summarization, especially for the web documents, has become a hot research area in natural language processing, web mining and information retrieval (see (Shen et al., 2007; Haghighi and Vanderwende, 2009; Wan et al., 2007; Svore et al., 2010)). Since document summarization already has some matured and suabilityessful solutions, a natural approach on attacking the answer summarization problem is to utilize the existing tools for solving the document summarization problem. However, when applying document summarization into the community question answering datasets to summarize the existing answers, the situation becomes different. Unlike news articles, the answers to the same question might have different expressions, different keywords and even various of opinions. The traditional methods in news summarization couldn't work in answer summarization. While there are lots of additional information in the CQA sites, such as the grade of the user who provides the answer, and whether the answer is selected

as the "Best Answer" or not, with which we can obtain the training data set more easily. But overall, answer summarization is harder than multi-document summarization since the summary cannot simply be a composition of the sentences with highest scores as in the articles.

There is a large class of questions we call them *yes/no qeustions*, which are seeking for a clear "Yes" or "No" answer. Our study shows that this special type of questions is one of the most common question type in CQA sites. In this paper we deal with a subproblem of answer summarization, focusing on the *yes/no questions*. Most of the yes/no questions can be simply answered by a "Yes" or "No". So the purpose of summarization is clearly to denote each answer as a "Yes" answer or a "No" answer. It is simpler than general answer summarization which is aiming to give a clear and coherent answer. For instance, given a certain yes/no question, the system will scan all its answers and output a summary like: 6 answers say "Yes" answers while 5 answers say"No"(some ambiguous answers are denoted as "Neutral"). The main contributions of this paper are as follows:

1. We formulate the problem of detecting whether an answer to a general question is a "Yes" answer, or a "No" answer, or a neutral one.

2. We propose a model to calculate the relevance for asymmetric documents. And we also combined this model as one of the features to give a weight for each sentences in the answers.

3. We propose three features, including the relevance feature, the sentiment-word score feature and the position weight feature, for classifying the answers to general questions. We then integrate these three features to obtain a novel feature function that classifies the answers into three categories: "Yes", "No" and "Neutral".

4. In the experimental part, we are the first to implement a real yes/no question summarization system, which reaches our first-proposed accuracy, precision, and recall.

Related Work. Answer summarization originates from multi-document summarization, whose task is to create concise abstracts from several sources of information. This problem initially comes from the need of generating a brief report of various news articles to save the time for reading news; see e.g. (McKeown and Radev, 1995). Systems for multi-document summarization include a topic detection and tracking system that generates cluster centroids (Radev et al., 2000), and an extraction-based multi-document summarization system (yew Lin and Hovy, 2002), etc.

Answer summarization can be regarded as a multi-document summarization problem, in which the question corresponds to the topic of the documents and the existing answers are treated as parallel multi-documents. A summarization is a brief report combining all the factual information and the opinions contained in the answers.

The existing question-and-answering systems can be divided into two classes: the systems for answering factual questions (see (et al., 2002)) and those for extracting sentiment answers (see (Yu and Hatzivassiloglou, 2003; Li et al., 2009)). The corresponding answer summarization tasks for CQA sites also has two aspects: factual answer summarization and sentiment/opinion answer summarization (Li et al., 2009). Our work in the present paper is different from both of the aforementioned aspects. Firstly, "Yes/NO" answers can be a true

fact. For example, if the question is "Is Obama the president of the U.S. now?", a possible answer could be "Yes" which is a factual answer. Correspondingly, if the question is "Is Obama a qualified president of the U.S.?", the answer "Yes, I think he is a good president." will be a sentiment one. Thus, our problem occupies an overlapping of both factual and sentiment answer summarization. In this sense, our solution comprises of both the sentiment words detection and factual words detection, which makes ours different from previous work.

This paper is organized as follows: in section 2 we give a more detailed definition to the yes/no questions and have a introduction to the model and the features. In section 3 we discuss the details of the features and show how to calculate the feature function. Then the experimental results are given in section 4.2. We conclude the whole paper in section 5.

2. Basic Framework

2-optional opinion seeking questions(we call them "yes/no question's' in this paper) are those can be answered by a short but clear supportive answers or objections. For instance, "Is a nutritionist a doctor?" is a *yes/no question*.

2.1. Introduction of the Basic Framework

The framework consists of 2 parts: a yes/no question detector and an opinion classification model. The system takes a question with several answers as inputs. First, it will detect whether the question is a yes/no question. If it is indeed a desired question, our system will classify the answers using the score computed by our feature function.

We use patterns to match *yes/no questions* and the detector is decried in detail in section 3.1. An opinion classification method can be characterized by the model and how its parameters are estimated. With classic feature-based approaches, classification is performed using a linear model computed over a specified feature space. Let q, a denotes a question and an answer, and we suppose the answer consists of a list of sentences $s_1, ..., s_k$. We extract the following features from each question-sentence pair (q, s),

1. $\text{rel}(q, s) \in [0, 1]$: The relevance score of the sentence s to the given question q;
2. $\text{pos}(s) \in [0, 3]$: The position score of the given sentence s;
3. $\text{yesno}(s) \in [-3, 3]$: This score to represent s is likely to be a positive or negative sentence;
4. $\text{sen}(s) \in [-3, 3]$: The sentiment in the sentence, positive value represent a supportive sentiment and vice versa;
5. $\text{sen}'(s) \in [-1, 1]$: An extended sentiment feature which is similar to $sen(s)$ with a higher coverage but lower precision.

Here we only give the meaning of the features but the detailed description is in the next section. These features are combined by Eq.3.

$$F(q,a) = \sum_{1 \le i \le k} \mathrm{rel}(q, s_i) \cdot pos(s_i) \cdot (\alpha \cdot yesno(s_i) + \beta \cdot sen(s_i) + \gamma \cdot sen'(s_i)) \quad (1)$$

The feature functions are learnt from unlabeled CQA dataset while the three parameters α, β, γ in Function(1) are by linear SVM using a labeled training set. Since in real dataset some answers are ambiguous, we apply the following function as our final classifier.

$$ret(q,A) = \begin{cases} 1, \text{ if } F(q,A) \ge \delta; \\ 0, \text{ if } -\delta < F(q,A) < \delta; \\ -1, \text{ if } F(q,A) \le -\delta. \end{cases} \quad (2)$$

$$out(q) = (pos = |\{A : ret(q,A) = 1\}|, neg = |\{A : ret(q,A) = -1\}|) \quad (3)$$

Finally, we outputs $out(q)$ as the voting results to the question. We will explain each feature of the model in detail in the next section.

3. Implementing the Feature Function Model

This section is organized in the order of the data flow in the system. We will first introduce how we detect yes/no questions; and then explain the five functions in Eq. 1 in detail. Training the parameters are introduced in the last subsection.

3.1. Detection of the General Questions

Detecting the yes/no questions is the first step of summarization. By annotating question types for over $5,000$ questions manually, we found that the following rule of detecting yes/no questions in English is efficient, that is, the question should start with three types of words,

1. Be verbs: S_{be} = { am, is, are, been, being, was, were }.
2. Modal verbs: S_{model} = {can, could, shall, should, will, would, may, might }.
3. Auxiliary verbs: S_{aux} = { do, did, does, have, had, has }.

Using simply the regular expression "$[S_{be}|S_{model}|S_{aux}]$.*+ ?" to detect obtains a quite good start point of the detection problem. It achieve a 80% precision with over 90% recall. However, there are two kinds of trivial mistakes this method would make. The first kind of error is alternative questions, as in community question answering(CQA) data there are a lot of alternative questions, such as "Is he married or not?" and "Will the concern be on May 23rd or June 1st?". The answers of the alternative questions will not be simply a "Yes" nor a "No" so we should drop these alternative questions.

Another type of error in the detecting process is the questions might start with "Do you know ...", "Does anyone know ..." In CQA. One kind of most popular questions might be "Can anybody tell me who is the president of the U.S.?" or "Does anyone know how much Bill Gates earns a year?" Using proper regular expression can filter over 90% of this kind of questions whose answer will not be "Yes" nor "No". The system will accept questions satisfy "$[S_{be}|S_{model}|S_{aux}]$.*+ ?" but drop the sentences match: "$[S_{be}]$ [a-z]* [or] [a-z]*" and

"[a-z]* [anyone | anybody][a-z]*[tell | know][a-z]*". Evaluating by another labeled 5000-question test set, this method detects 487 yes/no questions in it. The precision and recall are 91% and 87%, respectively.

3.2. Relevance Scores of the Question and Answer

Before going to extract the users' sensitive from the sentences, a key step is to extract the important sentences from an answer. Many signals can be used to extract key sentences but the relevant score is one of the most significant features in sentence importance extraction. Especially in user generated content(UGC), the content is very noisy so that filtering irrelevant answers is very important in our problem. Evaluating by a dataset contains 1000 questions and over 3000 answers, we found that over 43% of the sentences are irrelevant to the questions they are answering. Specially for the answers have more than 5 sentences(we call them long answers), over 64% of the sentences are irrelevant to their corresponding questions. So filtering the irrelevant sentences in the answer and weighting the relevance of the sentences to the questions are very important in opinion extractions.

We denote $\mathcal{D} = \{w_1, ..., w_m\}$ as the finite dictionary contains all words in our dataset and $m = |\mathcal{D}|$ is the size of the dictionary. The "bag-of-words" model is an assumption widely used in natural language processing. In this assumption, a sentence or a document is represented as an unordered collection of words, disregarding grammar and even word order (Lewis, 1998). A sentence s is denoted as a vector $s = (w_1, ..., w_m)$, in which $w_i \in \mathbb{Z}$ is the number of occurrences of the i-th word in the sentence.

We use the relevance score to measure the "topic distance" of two documents, as it is used in other problems. Classic vector space models use weighted word counts(e.g.TF-IDF) as well as the cosine similarity of their bag-of-words vectors as relevant function (Salton and McGill, 1986). Such models perform well in lots of applications, but suffers from the fact that only exact matches words are closely related to the similarity. Unfortunately, this is a false assumption in most of the web QA data. Another fact that we can't directly use $q\dot{a}$ as the relevance score is that the two document sets(the question set and the answer set) are not symmetric. For example, some words, like "what" and "how", should have a higher occurrent frequency in the questions than in the answers.

In our problem, we measure the similarity of a question and an answer by two parts: the first part is a word-vector cosine similarity score which is the widely used TF-IDF relevant score, while the second part is a more complicated model to measure the topic similarity of them. Formally speaking, the relevance is defined as follows.

$$\mathrm{rel}(q, a) = q \bigodot a + q\mathbf{M}a^T \tag{4}$$

where $q = (t_1, ..., t_m), t_i \in \mathbb{Z}$ is the question vector and $a = (e_1, ..., e_m)$ is the answer vector. $\mathbf{M}$ is a matrix of size $m \times m$ and $q \bigodot a$ is the TF-IDF relevant score defined as follows,

$$q \bigodot a = \frac{\sum_{1 \le i \le m} \mathrm{idf}(w_i) t_i e_i}{\sum_{i=1}^{m} \mathrm{idf}(w_i) t_i \sum_{i=1}^{m} \mathrm{idf}(w_i) e_i} \tag{5}$$

in which, $|q| = \sum_{1 \le i \le m} t_i$ and $|a| = \sum_{1 \le i \le m} e_i$ are the lengths of the question and the answer, respectively. Intuitively speaking, the matrix $\mathbf{M}$ is a measurement of the topic

distance of the two sentences (the question and the the answer); $\mathbf{M}_{i,j}$ measures the "topic distance" of two words w_i and w_j.

The first part is the cosine distance of the question and the answer. But as discussed before, the question and the answer might fall into two different feature spaces, which means the inner product can't measure their relevance correctly. We use a matrix $\mathbf{M}$ in the second term to measure the topic distance of the question space and the answer space. The distance of q and a is measured via the words in them. The matrix $\mathbf{M}$ can be explain as the transformation matrix from the question space to the answer space. For each sentence q in the question, $\mathbf{M} : q \to a$ maps it to the feature space the answers, and the insight meaning of $\mathbf{M}_{i,j}$ is the relevance of w_i(a word in the question) with w_j (a word in the answer). Although $\mathbf{M}$ can be learnt by neural network, we simply divide $\mathbf{M}$ two two simpler matrices, by letting $\mathbf{M} = \mathbf{U}^T\mathbf{V}$. That is,

$$f(q, a) = \mathbf{U}q(\mathbf{V}a)^T \tag{6}$$

$$\text{rel}(q, a) = q \bigodot a + f(q, a) \tag{7}$$

in which, $\mathbf{U}$ and $\mathbf{V}$ are matrices of size $k \times m$(k is a feasible parameter). $\mathbf{U}$ can be imagined as the function to transform a word in the question to a set of topics: for a question q, $\mathbf{U}q$ maps it to a vector of size k to represent the topics this question related to. In our experiments, we optimize the parameter k to obtain a reasonable trade-off between performance and accuracy.

Training such model could take many forms. Providing a large dataset that contains over millions of questions and even more user generated answers, we build the matrices $\mathbf{U}$ and $\mathbf{V}$ in a typical "learning to rank" way. Suppose we are given a set of questions $q_1, ..., q_t$ and the corresponding answers to q_i are $\{a_{i,1}, ..., a_{i,k_i}, i = 1...t\}$. The goal is to give the q_i and its answers $a_{i,1}, ..., a_{i,k_i}$ a higher ranking than any other answers, that is, $f(q_i, a_{i,k'}) > f(q_i, a_{j,k''})$ for all $i \neq j$. For this purpose we trained the $\mathbf{U}$, $\mathbf{V}$ by minimizing the following function (see (Bai et al., 2009)),

$$\sum_{1 \leq i \neq j \leq M} \left(\sum_{1 \leq m \leq k_i} \sum_{1 \leq l \leq k_j} \max(0, 1 - f(q_i, a_{i,m}) + f(q_i, a_{j,l})) \right) \tag{8}$$

To minimize Eq.8, we train this iteratively: one picks a random question q and one of its answer a^+. (q, a^+) is a positive training data to the model; while for a^- which is not an answer to q, (q, a^-) is a negative pair to the model. By randomly picking q, a^+ and q^-, we make a gradient step by the following updates:

$$\mathbf{U} \leftarrow \mathbf{U} + \delta\mathbf{V}(a^+ - a^-)q^T \text{ , if } 1 - \text{rel}(q, a^+) + \text{rel}(q, a^-) > 0 \tag{9}$$

$$\mathbf{V} \leftarrow \mathbf{V} + \delta\mathbf{U}q(a^+ - a^-)^T \text{ , if } 1 - \text{rel}(q, a^+) + \text{rel}(q, a^-) > 0 \tag{10}$$

We begins on two random sparse matrices $\mathbf{U}$ and $\mathbf{V}$ which satisfy $||\mathbf{U}||_F = 1$ and $||\mathbf{V}||_F = 1$. Here $||\mathbf{U}||_F$ is the Frobenius norm of $\mathbf{U}$,

$$||\mathbf{U}||_F = \sqrt{\sum_{i=1}^{k} \sum_{j=1}^{m} |u_{i,j}^2|} = \sqrt{\text{tr}(\mathbf{U}^T\mathbf{U})}$$

Obviously, this iteration is to minimize the Eq.8, which is to maximize the gap between $f(q, a^+)$ and $f(q, a^-)$. In our experiments we found that the iteration always converges to a result we don't expect. The reason is that the frequent words are involved in many questions/answers and so they are enhanced in many iterations, so that they always have a larger norm than other words. And the variety of negative examples (q, a^-) is very important in the training process even for a small set of training data, so that we have to prepare a lot of negative pairs (q, a^-) for each q. The quantity of negative pairs make the training process converges quite slowly. To reduce the cost of enumerating negative pairs, we use a simpler but more direct method, in which we don't need negative pairs at all, that is,

$$\text{Minize:} \quad F(\mathcal{Q}) = \sum_{1 \le i \ne j \le M} \Big(\sum_{1 \le m \le k_i} \sum_{1 \le l \le k_j} f(q_i, a_{i,m}) - f(q_i, a_{j,l}) \Big), \tag{11}$$

$$\text{Subject to:} \quad ||(\mathbf{U}^T)_j||_2 = ||(u_{1,j}, ..., u_{k,j})||_2 = \sqrt{\sum_{i=1}^{k} u_{i,j}^2} = 1,$$

$$\text{and} \quad ||(\mathbf{V}^T)_j||_2 = ||(v_{1,j}, ..., v_{k,j})||_2 = \sqrt{\sum_{i=1}^{k} v_{i,j}^2} = 1$$

$$\text{Each iteration:} \quad \mathbf{U} \leftarrow \mathbf{U} + \delta(\mathbf{V}a^+ - \mathbf{U}q)q^T - \delta'(\mathbf{V}a^+ - \mathbf{U}e)e^T, \tag{12}$$

$$\mathbf{V} \leftarrow \mathbf{V} + \delta(\mathbf{U}q - \mathbf{V}a^+)(a^+)^T - \delta'(\mathbf{U}q - \mathbf{V}e)e^T,$$

$$(\mathbf{U}^T)_j = (\mathbf{U}^T)_j \Big/ ||(\mathbf{U}^T)_j||_2, \tag{13}$$

$$(\mathbf{V}^T)_j = (\mathbf{V}^T)_j \Big/ ||(\mathbf{V}^T)_j||_2$$

in which, $\mathcal{Q}$ is the question set and $e = (1, ..., 1)^T$ is the unit vector of size m and $\delta' \ll \delta$ are iteration parameters.

The idea of the iteration is simply enhancing the relationship of q and a^+ and in the meanwhile punish all other words. In experiments, we compared Eq.9 and Eq.12 and found that they have the same accuracy in the large enough training set, but Eq.12 takes converge much more quickly. To overcome the second shortage of the training process of Eq.9(the frequent words have a larger norm, and when doing the matching, they dominate the results), we apply a simple normalization(in Eq.13) after every iteration such that each column of the two matrices $\mathbf{U}$ and $\mathbf{U}$ has a unified $\mathcal{L}_2$ norm.

Applying the improved iteration improves the training speed a lot. It improves the speed by at least 10 times with no harm to the accuracy. For more experiment results, please refer to section 4.2.

3.3. Yes/No Scores of the Question-and-Answer Pair

We design an answer dictionary that to some extent indicates whether the answer should be Yes, No, or Neutral. The human-made dictionary consists of 19 English words such as "Yes", "No", "Not" with corresponding scores which reflects the extent of positiveness

or negativeness. We design the Yes/No keywords and their scores by extracting the most popular keywords in the strong positive/negative answers in our labeled dataset. The score is designed according to their popularity in strong positive/negative answers. For example: yes: 3, yep: 2, no: -3, nope: -2, not: -2.

The labeled dataset we used to extract keywords only contains hundreds of questions. It is quite small compared to the large CQA data which contains millions of data. In the real user data, our human-made 19-word directory definitely have a low coverage, although it is accurate. Some words appear frequently on web but never appears in our labeled data, including but not only some typical typos (e.g. "noooo", "wooooooow") and abbreviations (e.g. "of coz"). To extend the human-made directory to contains as much as web keywords, we look into the concurrences of the words in the same paragraph. Supposing the directory is $\mathcal{D}_{yesno} = \{u_1, ..., u_n\}, n = 19$ and their associated scores are $s(u_i) \in [-3, 3]$, we utilize the following equation to calculate an approximate score to every word.

$$s(w) = \sum_{i=1}^{n} \frac{\text{co-ocr}(w, u_i)s(u_i)}{\log idf(w)}, \text{ for all } w \notin \mathcal{D}_{yesno} \tag{14}$$

in which, co-ocr(w, u_i) is the number of the concurrences of w and u_i in the same paragraph.

After all words are scored, we import the words in the top 100 and last 100 into the dictionary as positive and negative words, respectively. Since the data is noisy, the importation is monitored by manual review and the irrelevant words aren't imported. Since the Eq.14 doesn't guarantee any scope of $s(w)$, we normalize the $s(w)$'s to $[-2, 2]$ by a linear rescaling. Finally, we combine the 19-word directory together with the extended key words as our final dictionary $\mathcal{D}'_{yesno} = \{u'_1, ..., u'_{n'}\}$.

The Yes/No score yesno(a) is calculated directly by summing up the Yes/No scores of all the words in the answer,

$$\text{yesno}(a) = \sum_{\substack{u'_i \in \mathcal{D}'_{yesno} \\ u'_i \in a}} \frac{s(u'_i)}{|a|}$$

The value of Yes/No score indicates whether the sentence should be Yes, No, or Neutral. Also this value will be multiplied by the relevance score of the sentence corresponding to its question and the position score that reveals the importance of the sentence in the answer.

3.4. Sentiment Scores of the Question-and-Answer Pair

We investigate hundreds of question-and-answer pairs and notice that the sentiment words are the key indicators for the polarity of answers, especially the words appearing in both the questions and answers. In order to get the sentiment scores of a certain question to its corresponding answer, we use an English sentiment words dictionary called AFINN, which is generated by (Nielsen", 2011). The dictionary contains 1032 English words with their polarity scores. We want to compute the similarity score of the QA pair based on the sentiment words. Generally, the sentences and answers are considered as a vector of the TF-IDF scores of m sentiment words:

$$d_j = (tf_{1,j} \cdot idf_1, tf_{2,j} \cdot idf_2, \ldots, tf_{m,j} \cdot idf_m).$$

3.4.1. TF-IDF Score

The TF-IDF score (Term Frequency – Inverse Document Frequency) is a weight often used in information retrieval and text mining. In our system, it is applied as a statistical measure in constructing the sentiment term vector of a certain question-and-answer pair to evaluate the importance of the words to the sentences. The TF-IDF weight increases proportionally to the number of times a word appears in the sentence but is offset by the document frequency of the word in the whole corpus. By multiplying the term frequency and the inverse document frequency, we get the following formula for calculating the TF-IDF:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad \text{and} \quad idf_i = \log \frac{|\mathcal{D}|}{|\{d_j : t_i \in d_j\}|},$$

and,

$$\text{TF-IDF}_{i,j} = tf_{i,j} \cdot idf_i = \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \log \frac{|D|}{1 + |\{d_j : t_i \in d_j\}|}.$$

A high weight in TF-IDF is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out too common terms.

3.4.2. Sentiment Scores

We use the cosine measure, which is defined as the cosine value of the angle between two sentence vectors, to calculate the sentiment similarity scores of the given question and answer. The higher the sentiment similarity score is, the higher probability that the answer will be a positive/Yes answer.

$$\text{sen}(q, a) = \frac{q \cdot a}{|q| \cdot |a|}. \tag{15}$$

Here $q \cdot a$ means the vector dot product of q and a, and $|q|$ denotes the length of the vector q. Note that in the original definition of sentiment words, there are two types of words: the positive sentiment words and the negative ones. But in Eq.15 we drop the types of the sentiment words and the reason we can drop the types is we are not interested in the sentiment of the answer but in if the answers agree on the question or not.

3.4.3. Extended Sentiment Word Dictionary

Notice that the English sentiment word dictionary AFINN is made by human and the coverage is not sufficient for our application. Therefore, we design a method to extend the sentiment word dictionary by the mutual information scores (MI scores) of any two words. We use Eq.16 and Eq.17 to calculate all the sentiment scores in the word dictionary. After that, we keep all the words with absolute score larger than some prescribed bound to produce the extended sentiment word dictionary. The function $\text{sen}'(q, a)$ is computed similarly with $\text{sen}(q, a)$ but using the extended sentiment word directory.

$$MI(w_1, w_2) = \log \frac{p(w_1, w_2)}{p(w_1) \cdot p(w_2)} \tag{16}$$

and

$$sentiment\_score(w_i) = \sum_{\text{for all words } w_j} MI(w_i, w_j) \cdot sentiment\_score(w_j) \tag{17}$$

3.5. Position Scores of the Sentence

Besides the above features, position is also an important feature in our system. In traditional multi-article summarizations, position information is very important to detect the key points in the articles (Radev et al., 2000; McKeown and Radev, 1995). The situation is the same in answer summarization. The third important features used is the position score of each sentence. We all know that in English the first and last sentences are more likely to be conclusion sentences, while the middle sentences are less important. The appearances of the words "Yes", "No" in the first/last sentence is a very strong signal that this answer is support the question or not and so they have huge power to determine the polarity of the answer. In our model we use three different position weights as an additional feature in the function. The three position weights, w_{first}, w_{last} and w_{middle} is learned with the exhaustive search method given the training set.

The relevance score and the position score give a "confidence score" to the sentence, while the other features give a score to tell whether this sentence is a supporting opinion to the question or not. We use a simple linear function to combine the two factor together as described in Eq.1. Some other models such as binary decision tree are also tried but the best performances are achieved by feature function.

3.6. Learning the Parameters in the Feature Function Model

There are three parameters in the linear feature function model, namely α,β and γ. The SVM learning algorithm is used to train the parameters. The training dataset is a list of labeled question-answer pairs; each pair is associated with a label $+/-$ to indicate it is a positive pair or a negative one. In our training process, each QA pair is converted to a feature list using the definitions above. We utility a linear SVM to learn the three parameters α,β and γ to classify the positive and negative pairs.

4. Experiments

The experiments consist of two parts. The first part is to evaluate the relevance model in section 3.2; the second part is the subjective/objective answers detection model. All the experiments are conducted on a server with an Intel(R) Pentium(R) D CPU with 3.00 GHz and 2.99 GHz, and 16 GB of RAM. We choose C++ as the programming language because it has advantages in the processing speed.

4.1. Evaluations on the Asymmetric Relevance Model

We conducted experiments on real user community QA data. We sampled millions of docs from our indexed QA pages and extract the QA content from the pages. All QA pages are already parsed, the text of the questions, the best answers are all well annotated. The extraction keeps only the text of the question and its corresponding best answer. All other related information, like non-best answers, user information, time and locale are ignored. We use the question and the corresponding best answer as positive examples (q, a_q^+); and the questions combined with answers from other pages are considered as negative examples, which are denoted as (q, a_q^-). Here we apply an assumption to the QA data, that is, the best answers are highly relevant to the questions. The assumption is convincing since in most of

RunningTime	100 examples	1K examples	5K examples	10K examples	100K examples
Eq.9	2 secs/iter	52 secs/iter	5mins/iter	$> 30mins$/iter	$> 5hr$/iter
Eq.12	0.1secs/iter	0.3secs/iter	1.2 secs/iter	3secs/iter	$3mins$/iter

Table 1: Running time of the Relevance Model

the CQA sites, the best answer is selected by the asker. Regardless of some spams(they are easily filtered by user comments), most of the best-answers are high-quality training data to the relevance model. Our goal is to give (q, a_q^+) a higher score than (q, a_q^-) for each question q. Denote the question set as $\mathcal{Q}$, the accuracy in our evaluation is defined accordingly as follows, $acc = |\{q \in \mathcal{Q} | F(q, a_q^+) > F(q, a_q^-),\ \text{for all } a_q^+, a_q^-\}|/|\mathcal{Q}|$.

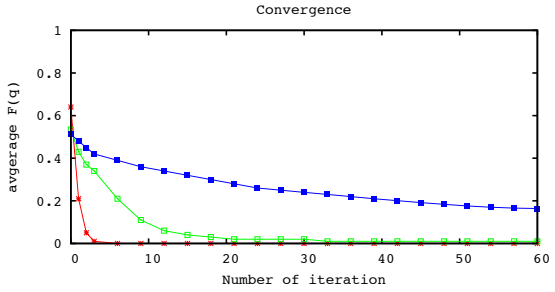

Figure 1: Convergence of Different Size of Training Sets (red: size=100; green: size = 1K; blue: size=10K)

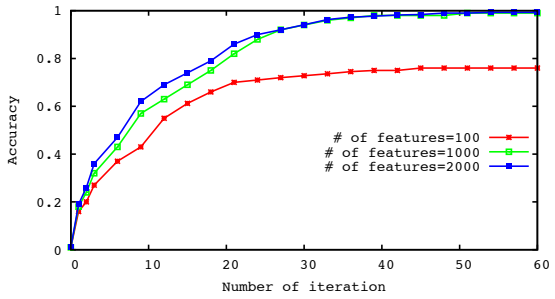

Figure 2: The impact of number of features

4.1.1. Experiment Results

The running time of the original relevant model using iteration Eq.9 and the improved model using Eq.12 are compared in Table 1. The experiment shows that the running speed is improved by more than 100 times in large training datasets.

Data Set	#Questions	#Answers	Yes	No	Neutral
Training	200	477	142	102	233
Testing	400	988	257	220	511

Table 2: Overview of the Date Set

Fig.1 shows the convergence speed of the relevant model. The x-axe is the number of iterations, while the y-axe is the target function, which is defined as $F(\mathcal{Q})/|\mathcal{Q}|$(slightly different to Eq,11). In this experiment, we set the feature number to $\mathcal{Q}/5$. The red line is the case we only use 100 quesitons as training data. For this simple dataset, the function converge very quickly. For a larger training set with 1000 questions, the model converge in roughly 30 iterations(the green curve); In a large dataset with $10k$ questions, which is almost in the same scale as our real dataset, the model converge in roughly 300 iterations(blue curve). For small cases, the model converges very quickly because the bag-of-words vectors are sparse. That is, for a small enough dataset, It is very likely that each question has a "special" word that never occurs in other questions. These "special"words will become a significant feature to the question because using this word it can easily identify the question. But in the real dataset, this situation is not true. So in a large enough dataset(e.g. the one we used to generate the blue curve) it converges much slower than small sets. We also consider the impact of the number of features. And Fig.2 shows that for reasonable setting of features, this model converges to a very good accuracy.

4.2. Subjective Answer Detection

In order to test the performance of our learning model, given a certain question, we ask two human annotators to annotate all the existing answers and classify them into three subjects: yes, no, and neutral. The two annotators are consistent on 78% of the answers and we take this part as our training and testing data set. The accuracy, precision and recall are given in the next subsections. In our application, we do not care too much about the neutral answers, but the precision and recall of the "Yes" and "No" answers are more important for us. Therefore we take all the three values as the evaluation criteria.

4.2.1. Testing Data Sets

To evaluate the results of our "Yes/No" answer summarization, we use 600 question-and-answer pairs with 600 questions and 1465 corresponding answers crawled from several famous question-and-answering sites such as Yahoo! Answers and WikiAnswers. In Table 2, an overview of the training and testing data is given.

4.2.2. Evaluation Results

We test our feature function model over 400 questions with 988 corresponding answers. The baseline is a simplifed system using only $F(q,a) = (yesno(a))$ and it only reaches a precision of 51% and a recall of 55%. Adding sentiment words score $F(q,a) = \text{yesno}(a) + \text{sen}'(q,a)$ improves the model slightly. After adding the relevance feature, the extended dictionary feature and the position weight feature, the accuracy, precision and recall increase by 6%,

System	Accuracy	Precision	Recall
Baseline	58%	51%	55%
+ sentiment	58%	52%	57%
+ sentiment, pos/rel weighting(complete model)	64%	56%	57%

Table 3: Detailed Results

Result \ Real	#Yes	#No	#Neutral
#Yes	148	31	102
#No	24	125	55
#Neutral	85	64	354

Table 4: Detailed Results

5% and 2%, respectively. In the final system with the The overall accuracy is 64%, the precision of the "Yes" and "No" answers is 56%, and the recall of the "Yes" and "No" answers is 57%. The detailed results are shown in Table 4. The high precision proves that our model is convincing, and the recall being 56% shows that we successfully hunt out a large amount of the existing answers. Table 5 shows some examples of the outputs of our summarization system.

4.2.3. Error Case Study

There are still some mistakes that the feature function model can not fix. An example is that, when the question is "Is Obama the president of the US?", a positive answer might be "I don't think he is not." The feature function model considers this answer to be negative as there are both "not" and "don't" in the answer. However, this example is a double negative sentence and should be counted as a "Yes" answer. Dealing with such cases requires more complex semantic analysis, and we need to parse the sentence to get a grammar tree. Nonetheless, parsing and semantic analysis will bring in new errors and make this problem even more complex. We will further study this method in the future.

Another error case is more complex. For example, the question is "Is someone married?" while there is a definitely "Yes" answers appearing like "If he is not married, I'll buy you a month of dinner.". Human can understand this answer easily but the computer cannot because it requires deeper understanding and further reasoning of languages. With the development of natural language understanding and artificial intelligence, these problems may be solved in the future.

5. Conclusion

We first proposed the answer summarization problem for the yes/no questions as well as a model to classify the subjective/objective answers. Our model integrates answer-keywords, QA-relevance, answer-opinion information. Empirical evaluation demonstrated the merits of our approach.

In this paper we also introduced a novel relevant model for asymmetric documents, especially for question-answers relevance computation. We also improved the iteration function

Question: can a vegetarian eat chicken flavored ramen noodles
Answers # 1 i believe it has chicken power which is chicken so i consider it not being vegetarian
Answers # 2 you can certainly eat the noodles alone but the chicken flavoring actually has chicken in it ...
Answers # 3 i m pretty sure they use chicken fat in that so no
Answers # 4 yes i do all the time it is not chicken at all just some artificial extract someone put together and called it chicken
Summary: # of "Yes": 2, # of "No": 2, # of "Neutral": 0
Question: is a nutritionist a doctor
Answers # 1 well no/nutritionists are not doctors/
Answers # 2 no a nutritionist is basically a dietician/they have medical knowledge but specify ...
Answers # 3 generally a nutritionist has a 4 year degree in nutrition/ a nutritionist can get a ph d but that is not the same as a medical doctor
Answers # 4 no you need to do a 2 or a 3 year course/ my dietician is better than most docs in diagnosing
Answers # 5 no/you do a science in nutrition degree/you cannot prescribe drugs/ you do not perform surgery/it takes 4 years
Answers # 6 a brain doc i think
Summary: # of "Yes": 1, # of "No": 3, # of "Neutral": 2

Table 5: Examples of the Output of the System

to the relevant model for the cases that only the positive training examples are guaranteed, and the evaluation on real user generated contents(UGC) show that our iteration functions improve the performance of the model significantly.

As the future work, we plan to add this "Yes/No" answer summarization process to the summarization of all the answers to a certain question and embed it in a real QA system. In this case, the summarization will consist of two parts: the first is the voting results of "Yes" and "No" numbers, and the second part contains simple facts, opinions and other summarized answers. Thus the system will give the user a clear overview of the existing answers or search results to his/her question.

References

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Corinna Cortes, and Mehryar Mohri. Polynomial semantic indexing. In *Advances in Neural Information Processing Systems 22*, pages 64–72. 2009.

Sameer Pradhan et al. Building a foundation system for producing short answers to factual questions. In *TREC'02*, 2002.

Pedro Paulo Balage Filho, Vincius Rodrigues de Uzda, Thiago Alexandre Salgueiro Pardo, and Maria das Graas Volpe Nunes. Experiments on applying a text summarization system for question answering. In *CLEF'06*, pages 372–376, 2006.

Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 362–370, Stroudsburg, PA, USA, 2009.

David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. pages 4–15, 1998.

Fangtao Li, Yang Tang, Minlie Huang, and Xiaoyan Zhu. Answering opinion questions with random walks on graphs. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL '09, pages 737–745, Stroudsburg, PA, USA, 2009.

Yuanjie Liu, Shasha Li, Yunbo Cao, Chin-Yew Lin, Dingyi Han, and Yong Yu. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 497–504, Stroudsburg, PA, USA, 2008.

Kathleen McKeown and Dragomir R. Radev. Generating summaries of multiple news articles. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 74–82, New York, NY, USA, 1995.

"F. Å. Nielsen". Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2011. Informatics and Mathematical Modelling, Technical University of Denmark.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pages 21–30, Stroudsburg, PA, USA, 2000.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document summarization using conditional random fields. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2862–2867, San Francisco, CA, USA, 2007.

Krysta Marie Svore, Lucy Vanderwende, and Christopher J. C. Burges. Enhancing single-document summarization by combining ranknet and third-party sources. In *EMNLP-CoNLL*, pages 448–457, 2010.

Yang Tang, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. Summarizing similar questions for chinese community question answering portals. In *Proceedings of the 2010 Second International Conference on Information Technology and Computer Science*, pages 36–39, Washington, DC, USA, 2010.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2903–2908, San Francisco, CA, USA, 2007.

Chin yew Lin and Eduard Hovy. From single to multi-document summarization: A prototype system and its evaluation. In *In Proceedings of the ACL*, pages 457–464. MIT Press, 2002.

Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, pages 129–136, Stroudsburg, PA, USA, 2003.