RelationListwise for query-focused multi-document summarization

Wenpeng Yin   Lifu Huang   Yulong Pei   Lian′en Huang
The Shenzhen Key Lab for Cloud Computing Technology & Applications (SPCCTA)
Shenzhen Graduate School
Peking University, Shenzhen 518055, P.R. China
{mr.yinwenpeng,warrior.fu,paul.yulong.pei}@gmail.com, hle@net.pku.edu.cn

Abstract

Most existing learning to rank based summarization methods only used content relevance of sentences with respect to queries to rank or estimate sentences, while neglecting sentence relationships. In our work, we propose a novel model, RelationListwise, by integrating relation information among all the estimated sentences into listMLE-Top K, a basic listwise learning to rank model, to improve the quality of top-ranked sentences. In addition, we present some unique sentence features as well as a novel measure of sentence semantic relation, aiming to enhance the performance of training model. Experimental results on DUC2005-2007 standard summarization data sets demonstrate the effectiveness of our proposed method.

KEYWORDS: Query-focused multi-document summarization, Listwise, Sentence relation.

1 Introduction

In this paper, we focus on the task of producing extraction-based query-focused multi-document summaries given a collection of documents, which is usually considered as a sentence ranking problem. Typically, ranking methods calculate the combinational effects of various features which are designed to identify the different aspects of sentences and/or their relevance to queries. Yet so far not much attention has been paid to it. Most commonly, the features are simply combined by a linear function in which the weights are assigned manually or tuned experimentally. In the past, machine learning approaches have been successfully applied in extractive summarization (Ouyang et al., 2007; Shen and Li, 2011), and a new research branch named "learning to rank" has emerged. Its objective is to explore how the optimal weights can be obtained automatically by developing learning strategies. However, previous work mainly considered the content relevance of sentences with respect to certain query while ignoring the relationships among sentences. In this paper, we try to study how to use sentence relatedness to improve the performance of a ranking model. Further, we notice that many learning to rank algorithms have been proposed in recent literature, and these algorithms can be categorized into three types: pointwise, pairwise, and listwise approaches. The pointwise and pairwise approaches transform ranking problem into regression or classification on single object and object pairs respectively, while neglecting the fact that ranking is a prediction task on a list of objects. In listwise approach, object lists instead of object pairs are used as instances in learning, and the major task is how to construct a listwise loss function, representing the difference between the ranking list output by a ranking model and the ranking list given as ground truth. Experimental results showed that listwise approach usually outperforms pointwise and pariwise approaches (Cao et al., 2007; Qin et al., 2008). Accordingly, we mainly concentrate on developing listwise learning to rank in our summarization task.

More exactly, taking into account the specific scenario of summarization, it's better to base our work on a variant of basic listwise training model: ListMLE Top-K presented in (Xia et al., 2009). It's because that we usually only need to select a small amount of sentences to construct a summary. ListMLE Top-K , a modification of basic listwise algorithm for more suitability in many real ranking problems where the correct ranking of the entire permutation is not needed, could help us to improve the ranking accuracies of top-K sentences. Based on that, our novel RelationListwise function, having absorbed sentence affinity information, is formed to learn the optimal feature weights.

Apparently, how to design appropriate sentence features and measure sentence similarity matter greatly in influencing the system performance. In most existing approaches about feature design, the authors tended to only consider factors that were supposed to reflect the bias of sentences towards a query while neglecting such a possibility: overestimating the relationship with a query might lead to the competitiveness among relevant sentences, which would eventually do harm to the summary quality. For this reason, some extra processing was usually conducted during sentence selection because some top-ranked sentences usually can not be used to construct a high-quality summary directly. For example, (Shen and Li, 2011) defined a total of 20 sentence features for its ranking SVM model. Whereas, those features were produced by paying no attention to avoiding similar sentences to get close scores. Therefore, in order to keep low information redundancy in the finally generated summary, those authors had to apply diversity penalty algorithm to update the sentence orders which resulted from the training function directly. Namely, some previous work had to take two steps to construct a high-quality summary, including initial sentence ranking coming from ranking model and sentence order adjustment. Hence, a strong motivation comes to our mind: designing sentence features considering the query-biased factors as well as the competitiveness coming from similar sentences, aiming to produce a high-quality summary through directly selecting some top-ranked sentences.

As for sentence similarity, existing literatures have presented various methods to deal with it. Nevertheless, hard matching of words is commonly a primary obstacle in judging whether two sentences are semantically related. For example, while $wonderful$ and $amazing$ are semantically close, they are treated completely different in many existing methods, such as tf-isf (term frequency and inverse sentence frequency) based cosine measure. Motivated by that, we utilize Log-Bilinear Document Model, proposed in (Maas et al., 2011), to achieve the identification of semantically similar words. As a result, sentences with no same words but similar ones could also be considered having a certain degree of semantic similarity. Extensive experiments on DUC2005-2007 standard summarization data sets are performed and their results point out the good performance of RelationListwise in this task.

Since our proposed training model is on the basis of sentence similarity and sentence features, in following sections, after giving related work in Section 2, we first elaborate how to derive sentence similarity in Section 3 and biased features in Section 4, respectively. Then, detailed description of using sentence similarity to improve listMLE top-K is presented in Section 5. Section 6 shows training data generation. Experiments and results are given in Section 7.

2 Related work

We first introduce some supervised learning approaches applied in query-biased summarization. Then, some typical work about feature design and sentence similarity follows.

Supervised learning approaches have been successfully applied to query/topic-biased summarization. (Zhao et al., 2005) applied the Conditional Maximum Entropy, a classification model, on the DUC 2005 query-based summarization task. (Ouyang et al., 2007) used support vector regression (SVR), a pointwise ranking algorithm, to relate the “true" score of the sentence to its features. (Jin et al., 2010) presented a systematic study of comparing different learning to rank algorithms and comparing different selection strategies for multi-document summarization. Whereas, it focused on the simple comparison of some basic models with no optimization. (CHALI and HASAN, 2011) had deeply investigated and compared the effects of using different automatic annotation techniques on different supervised learning approaches, including SVMs, HMMs, CRFs, and MaxEnt, in the domain of query-focused multi-document summarization. (Shen and Li, 2011) explored the use of ranking SVM, a pairwise learning to rank model, for obtaining credible and controllable solutions for feature combinations. Our main contributions not only lie in our unique design of sentence features and sentence similarity measure, more importantly, we integrate sentence relationships with listwise to improve the ranking model while above literatures ignored the relation information among those sentences

With regard to feature design, (Li et al., 2009) treated summarization as a supervised sentence ranking process, where coverage, balance and novelty properties were incorporated. Whereas, it focused on generic summarization rather than query-biased situation. (Wan et al., 2007) gave explicit definitions of biased information richness and novelty, then, it proposed to compute biased information richness using manifold-ranking process (Zhou et al., 2004), and a modified MMR algorithm was applied to keep low information redundancy in generated summaries. In (Wei et al., 2008), authors proposed query-sensitive sentence similarity. Only the overlapping of topic-relevant contents was penalized when applying MMR algorithm. In addition, clustering techniques were commonly adopted to identify different and novel aspects of documents (Wan and Yang, 2008). The method presented in (Li et al., 2010) is slightly similar with our work for it considered novelty, coverage and balance wholly. However, sentence features of existing literatures were usually acquired asynchronously. For instance, $novelty$ and $balance$ in (Li et al., 2010) were achieved as an optimization process after authors have identified part of high-quality sentences through the effects of other features.

Calculating sentence similarity appears in many applications. tf-isf based cosine measure is widely used to determine the lexical similarity of two sentences. Whereas, high dimensionality and high sparsity usually lead to disappointing performance. (Erkan, 2006) proposed a graph-based sentence ranking model: Biased LexRank, where edge weight or called sentence similarity was acquired using generation probability between two sentences based on a (unigram) language model. (Islam and Inkpen, 2008) determined sentence similarity by combining string similarity, semantic similarity and common-word order similarity with normalization. The similarity between two short text snippets in (Quan et al., 2010) was calculated based on their common terms and their distinguishing terms relationship which was discovered by examining their probabilities under each topic. Some literatures opted to first deal with word similarity calculation, then combine the result with sentence structure information. For example, (Li et al., 2006) first derived word semantic similarity from a lexical knowledge base, modeling common human knowledge about words in a natural language, and a corpus, adapting to the specific application area. Secondly, it considered the impact of word order on sentence meaning. The derived word order similarity measured the number of different words as well as the number of word pairs in a different order. In (Zhang et al., 2011), word similarity only considered the spellings and ignored the semantic meanings of words. Structure information considered the orders of words and the distances between words, and it ignored the syntactic information of sentences. (Yin et al., 2012) used a similar way with (Quan et al., 2010) to determine word relatedness while its structural similarity of sentences was acquired via longest common subsequence (LCS), weighted longest common subsequence (WLCS) and skip-bigram co-occurrence statistics, respectively.

3 New measure of sentence similarity

Hard matching between words has long been an obstacle in determine the relatedness of two sentences. For example, considering following two sentences:

s_1 : employee enjoy happy *holiday* s_2 : employee enjoy happy *vacation*

where words $holiday$ and $vacation$ would be treated with no relation in traditional VSM based cosine measure. Whereas, they are semantically related very much in the real context. Hence, in our perspective, before computing sentence similarity, we should first solve this problem: identifying the semantic relatedness of words.

3.1 Capturing semantic similarities of words

Authors in (Maas et al., 2011) presented an algorithm to acquire word semantic similarity through learning word vectors via an unsupervised probabilistic model of documents. While it is common to represent words as indices in a vocabulary, but this fails to capture the rich relational structure of the lexicon. Vector-based models do much better in this regard. They encode continuous similarities between words as distance or angle between word vectors in a high-dimensional space. Next, we briefly introduce that algorithm.

For a document (e.g., d), a probabilistic model is constructed using a continuous mixture distribution over words indexed by a multi-dimensional random variable θ, then the probability of d is determined using a joint distribution over d and θ. As many common treatments did, the algorithm also puts an assumption of independence for words given θ. The probability of a document d is as follows

$$p(d) = \int p(d,\theta)d\theta = \int p(\theta)\prod_{i=1}^{N} p(w_i|\theta)d\theta \tag{1}$$

where N is the number of words in d and w_i is the i^{th} word. θ has a Gaussian prior.

Then define the conditional distribution $p(w_i|\theta)$ using a log-bilinear model (Maas and Ng, 2010) with parameters R and b. R is virtually a word representation matrix $R \in \mathbb{R}^{(\beta \times |V|)}$ where the β-dimensional vector representation of each word w in vocabulary V corresponds to that word's column in R, i.e., $\phi_w = R_w$. The random variable θ is also a β-dimensional vector ($\theta \in \mathbb{R}^{\beta}$), indicating the weights of the β dimensions of words' representation vectors. In addition, a bias b_w is introduced for each word to capture differences in overall word frequencies. The energy assigned to a word w, given these model parameters, is

$$E(w;\theta,\phi_w,b_w) = -\theta^T\phi_w - b_w \tag{2}$$

After normalization, we obtain the distribution $p(w|\theta)$,

$$\begin{aligned} p(w|\theta;R,b) &= \frac{\exp(-E(w;\theta,\phi_w,b_w))}{\sum_{w'\in V}\exp(-E(w';\theta,\phi_{w'},b_{w'}))} \\ &= \frac{\exp(\theta^T\phi_w + b_w)}{\sum_{w'\in V}\exp(\theta^T\phi_{w'} + b_{w'})} \end{aligned} \tag{3}$$

Apparently, for a given θ, a word w's occurrence probability is related to how closely its representation vector ϕ_w matches the scaling direction of θ. Finally, maximum likelihood learning is exploited for this model when given a set of unlabeled documents D. In maximum likelihood learning we maximize the probability of the observed data given the model parameters. Here, we omit the learning details[1].

Having obtained vector representations of words, we could determine the semantic relatedness (SR for short) of two terms (e.g., w_1 and w_2):

$$SR(w_1,w_2) = \phi_{w_1}\cdot\phi_{w_2} \tag{4}$$

3.2 Sentence similarity identification

Based on word relatedness, we construct a word connectivity graph, and use PageRank algorithm, with normalized words' term frequencies as prior distribution, to determine words' importance, e.g., the importance score of word w is denoted as $imp(w)$, and importance score of sentence s is: $Imp(s) = \sum_{w\in s} imp(w)\cdot\sqrt{c_s(w)}$, where $c_s(w)$ is the times of w occurring in

[1]For more details, please refer to (Maas et al., 2011)

s. Note that we have filtered out stop words. We use $\sqrt{\{\cdot\}}$ to reduce the influence of repeated words instead of using sentence length to divide the aggregate score of a sentence, because we believe that a sentence with more important words deserves high importance. Based on our algorithm, a sentence with lots of unimportant words will not get a high importance score.

Given two sentences such as s_1 and s_2, our next step is to find for each word a in one sentence the corresponding word a^*, in the other sentence, that maximizes their mutual semantic relatedness (e.g., a in s_1, a^* in s_2 and vice versa).

$$a^* = \arg\max_{b \in s_2} SR(a, b) \tag{5}$$

Then, we average the semantic relevance scores from all terms in sentence s_1, with reference to their best matches in sentence s_2, as shown in Equation 6.

$$\zeta(s_1, s_2) = \frac{\sum_{w_i \in s_1} imp(w_i) \cdot SR(w_i, w_i^*)}{\sum_{w_i \in s_1} imp(w_i)} \tag{6}$$

We do the same for the opposite direction (i.e., from the words of s_2 to the words of s_1) to cover the cases where the two sentences are not equally important or they receive different similarities from each other. Finally, we derive the similarity between sentences s_1 and s_2 as:

$$sim(s_1, s_2) = \frac{Imp(s_1) \cdot \zeta(s_1, s_2) + Imp(s_2) \cdot \zeta(s_2, s_1)}{Imp(s_1) + Imp(s_2)} \tag{7}$$

4 Feature design

In the case of query-sensitive summarization, we conclude that qualified summary sentences should mainly meet the following typical demands: query-biased relevance (Shen and Li, 2011; Otterbacher et al., 2005), biased information richness (Wan et al., 2007) and biased novelty (Wan et al., 2007). Query-biased relevance requires that the sentences in the summary must overlap with the query in terms of topical content. Query-biased information richness denotes the information degree of a sentence with respect to both the sentence collection and the query. Query-biased information novelty is used to measure the content uniqueness of a sentence based on that sentence's capability in differentiating itself from other sentences as well as responding to the demands of the query. According to above definitions, we design multiple sentence features corresponding to them, respectively.

4.1 Four kinds of relevance

Given a sentence s, we exploit following information available in the DUC2005-2007 datasets:

- t_s: Title of the document containing sentence s.
- d_s: The document containing sentence s.
- c_s: The document cluster containing sentence s.
- q_s: Query of the document collection containing sentence s.

Noting that we do not conduct sentence segmentation if the query consists of more than one question, instead we treat it as a single, long sentence. Consequently, we calculate the following four sentence features using similarity measure discussed in Section 3:

- $r_{t,s}$: Relevance between sentence s and the title of the document to which s belongs.
- $r_{d,s}$: Relevance between sentence s and the document to which s belongs.
- $r_{c,s}$: Relevance between sentence s and the document cluster to which s belongs.
- $r_{q,s}$: Relevance between sentence s and query q.

4.2 Biased information richness (BIR)

Given a sentence collection and a query q, the BIR of sentence s is used to indicate the information degree of s with regard to both the sentence set and q, i.e., the richness of information contained in the sentence s biased towards q.

This feature score for each sentence is obtained via a variant version of the manifold-ranking process proposed in (Zhou et al., 2004). Points $\{s_0, s_1, \cdots, s_n\}$ denote the query statement (s_0) and all the sentences in the document collection ($\{s_i|1 \leq i \leq n\}$) in a manifold space. The ranking function is denoted by $f = [f_0, f_1, ..., f_n]$. (Wan et al., 2007) hypothesized that all the sentences had blank prior knowledge so their initial scores were all set to zero. Whereas in this study, it is rational to treat the query-sentence relevance discussed in Section 4.1 as prior knowledge of sentences. Since s_0 denotes the query description, the initial score vector of these sentences is y =$[y_0, y_1, ..., y_n]$, where $y_0 = 1$ and y_i = sim(s_i, s_0) (1≤ i≤ n). The manifold ranking can be performed iteratively using the following equation:

$$f(k+1) = \alpha S f(k) + (1-\alpha) y \tag{8}$$

where S is the symmetrically normalized similarity/relevance matrix as for $\{s_0, s_1, \cdots, s_n\}$, trade-off parameter α is set to 0.6, and k indicates the k^{th} iteration. Obviously, modified initial scores will exert a greater influence to sentence scores than the settings in (Wan et al., 2007) at each step of the iteration process. After convergence, let $f_i^\star$ denotes the limit of the sequence $\{f_i(t)\}$, then the BIR of sentence s_i is:

$$BIR(s_i) = f_i^\star \quad (1 \leq i \leq n) \tag{9}$$

4.3 Biased information novelty (BIN)

In our perspective, those sentences, owning relative high BINs and picked out to generate summary, must have information redundancy as low as possible meanwhile meet the user's information need, expressed by a query, as much as possible. Satisfaction of only one of them will certainly be off the original intention of biased novelty. Hence, we employ DivRank, proposed in (Mei et al., 2010), to acquire this sentence property. DivRank uses a *vertex-reinforced random walk* model to rank graph nodes based on a diversity based centrality. The basic assumption in DivRank is that the transition probability from a node to another is reinforced by the number of previous visits to the target node. Let $p_T(u, v)$ be the transition probability from any state u to any state v at time T. We can define a family of time-variant random walk processes in which $p_T(u, v)$ satisfies

$$p_T(u,v) = (1-\lambda) \cdot p^*(v) + \lambda \cdot \frac{p_o(u,v) \cdot N_T(v)}{D_T(u)} \tag{10}$$

where $D_T(u) = \sum_{v \in V} p_o(u,v) N_T(v)$. Here, $N_T(v)$ is the number of times that node v has been visited up to time T and $p^*(v)$ is a distribution which represents the prior preference of visiting vertex v. In our task, $p^*(v)$ is set to the normalized similarity between sentence v and the query q, i.e., $p^*(v) = \widetilde{sim}(v,q)$. $p_o(u,v)$ is the *organic* transition probability prior to any reinforcement, which can be estimated in a regular time-homogenous random walk, such as

$$p_o(u,v) = \begin{cases} \gamma \cdot \frac{sim(u,v)}{degree(u)} & \text{if } u \neq v; \\ 1-\gamma & \text{otherwise} \end{cases} \tag{11}$$

If the network is ergodic, after a sufficiently large T, the reinforced random walk defined by Equation 10 also converges to a stationary distribution $\pi(v)$. That is

$$\pi(v) = \sum_{u \in V} p_t(u,v)\pi(u), \qquad \forall t \geq T \tag{12}$$

$\pi(v)$ is then used to denote the BIN of sentence v. In experiments, $\lambda = 0.9$ and $\gamma = 0.25$.

5 RelationListwise ranking function construction

Inspired by the work in (Zhou et al., 2011), which described a general ranking function with relationship information among objects, we modify that model specifically for our summarization task. Firstly, we define some notations used in this section. Query q is associated with a sentence collection $S = \{s_1, s_2, \cdots, s_n\}$, and S is associated with a set of judgments $Y = \{y_1, y_2, \cdots, y_n\}$. Here, n denotes the number of sentences in that collection, and y_i is the relevance judgment of sentence s_i with respect to query q. We could also treat y_i to be the position of sentence s_i in ranking list. Exactly, each sentence s_i is represented as a feature vector $\mathbf{x}_i = \Phi(q, s_i)$, where the acquisition of those features is presented in Section 4. In whole, we can see query q corresponds to a set of sentences S, a set of features vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, a set of judgements Y, and R, the affinity matrix among sentences in S.

Let $g(\mathbf{x}_i, \mathbf{w})$ denote the basic ranking function of Listwise method, i.e.,

$$g(\mathbf{x}_i, \mathbf{w}) = <\mathbf{x}_i, \mathbf{w}> = \mathbf{x}_i \cdot \mathbf{w} \tag{13}$$

where vector $\mathbf{w}$ is unknown, and is exactly what we want to learn. In this paper, $g(\mathbf{x}_i, \mathbf{w})$, meaning the content relevance of s_i with regard to query q, is defined as a linear function, namely taking the inner product between vector $\mathbf{x}_i$ and $\mathbf{w}$. Based on this step, we use following formula to derive the final RelationListwise ranking score of sentence s_i $(1 \leq i \leq n)$, denoted as $f_{\mathbf{w}}(\mathbf{x}_i, R)$, by integrating its initial Listwise ranking score (i.e., $g(\mathbf{x}_i, \mathbf{w})$) with its neighbors' Listwise scores:

$$f_{\mathbf{w}}(\mathbf{x}_i, R) = (1-\tau)g(\mathbf{x}_i, \mathbf{w}) + \tau \sum_{j \neq i}^{n} g(\mathbf{x}_j, \mathbf{w}) \cdot \tilde{R}^{(i,j)}) \tag{14}$$

where $\tilde{R}^{(i,j)} = \frac{R^{(i,j)}}{\sum_{z \neq i} R^{(i,z)}}$ denotes the normalized similarity between sentence s_i and s_j. The second item of Equation 14 can be interpreted as following: if the relevance score of s_j with query q is high and s_j is very related with s_i , then the relevance value between s_i and q will be increased significantly, and vice versa. In Equation 14 we can find that the prestige of sentence s_i is decided not only by the content of itself, but its neighbors' prestige. The coefficient τ is the weight of relation information (the second item of Equation 14). We can change its value to adjust the contribution of similarity information to the whole ranking value. In our experiment, we set it to 0.5.

5.1 RelationListMLE Top-K probability optimization

Before presenting our training algorithm, it's worth mentioning that a crucial difference between summarization scenario and most ranking problems is that summarization task casts more emphasis on the top-ranked sentences in the final list. Because of the length limit of a summary, most sentences assigned relatively low ranking scores will not be selected to construct a summary. Therefore, the correct ranking of the entire sentence permutation is not needed, our goal is to improve the ranking accuracies of top-K sentences (here we set K to a constant on condition that we are confident that K selected sentences satisfy the demand of summarization task about summary length, e.g., $K = 20$ in our experiments).

There are many training algorithms to learn listwise ranking function, such as Likelihood loss, Cosine loss and Cross entropy loss, among which likelihood loss has been proved to have the most comprehensive properties and its corresponding learning algorithm is called ListMLE (Xia et al., 2008). Accordingly, we integrate K value with listMLE to learn our proposed RelationListwise ranking function. This kind of solution was indicated by (Xia et al., 2009) to be more suitable for some real ranking applications where top-K objects are the focus. For convenience, we name it *RelationListMLE Top-K* probability optimization.

Our proposed optimization method also uses stochastic gradient descent algorithm to search the local minimum of loss functions. The stochastic gradient descent algorithm is described as Algorithm 1.

Algorithm 1: Stochastic Gradient Descent

```
Input: training data $\{\{X_1, Y_1, R_1\}, \{X_2, Y_2, R_2\}, \cdots, \{X_n, Y_n, R_n\}\}$
Parameter: learning rate $\eta$, tolerance rate $\varepsilon$
Initialize parameter $\mathbf{w}$
repeat
    for $i = 1$ to $n$ do
        (1)Compute score of each sentence $j$ with current $\mathbf{w}$ using Equation 14
        (2)Compute the gradient $\Delta\mathbf{w}$ with current $\mathbf{w}$ using Equation 15
        (3)Update $\mathbf{w}$=$\mathbf{w}$-$\eta \times \Delta\mathbf{w}$
    end for
    Compute likelihood loss:
        $L = -\sum_{i=1}^{n} \log \prod_{c=1}^{K} \frac{\exp(f_{\mathbf{w}}(\mathbf{x}_{iy_i^c}, R_i))}{\sum_{t=c}^{n_i} \exp(f_{\mathbf{w}}(\mathbf{x}_{iy_i^t}, R_i))}$
until change of likelihood loss is below $\varepsilon$ times the previous loss
Output: $\mathbf{w}$
```

In RelationListMLE-Top-K, the gradient of likelihood loss $L(f_{\mathbf{w}}(X_i, R_i), Y_i)$ with respect to w_j can be derived as Equation 15:

$$\Delta w_j = \frac{\partial L(f_{\mathbf{w}}(X_i, R_i), Y_i)}{\partial w_j} \tag{15}$$

$$= \sum_{c=1}^{K} \{ \frac{\sum_{t=c}^{n_i} [\exp(f_{\mathbf{w}}(\mathbf{x}_{iy_i^t}, R_i)) \cdot \frac{\partial f_{\mathbf{w}}(\mathbf{x}_{iy_i^t}, R_i)}{\partial w_j}]}{\sum_{t=c}^{n_i} \exp f_{\mathbf{w}}(\mathbf{x}_{iy_i^t}, R_i)} - \frac{\partial f_{\mathbf{w}}(\mathbf{x}_{iy_i^c}, R_i)}{\partial w_j} \}$$

where

$$\frac{\partial f_{\mathbf{w}}(\mathbf{x}_{ir}, R_i)}{\partial w_j} = (1-\tau)\mathbf{x}_{ir}^j + \tau \sum_{p=1,p\neq r}^{n_i} \mathbf{x}_{ip}^j \tilde{R}_i^{(r,p)} \tag{16}$$

and $\mathbf{x}_{ir}^j$ is the j^{th} element in $\mathbf{x}_{ir}$

6 Training data construction

To apply learning to rank in summarization, we should have a labeled training collection in the form of $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i$ is a sentence (or sentence feature vector) in sentence set S and y_i is the ranking of the sentences. In addition, the relation information among sentences must be constructed too. To estimate the ranking score of a sentence s given the human summary H, we implement manifold-ranking algorithm to achieve our goal, treating H as a query vertex in graph. The reason for using manifold-ranking algorithm rather than directly calculating the relevance of a sentence towards the query (i.e., H) lies in that manifold-ranking process considers the query-sentence relatedness as well as the inter-sentence affinities. It matches our intention of improving the basic listwise with relationships among sentences.

Whereas, it is worth mentioning that we do not treat the human summary H wholly as a long sentence to participate in the computing of query-sentence similarities. Instead, we treat H as a sentence set and our goal is to find a sentence, from H, that has the relatively maximum similarity with an estimated sentence in S. This is because that if a sentence is similar with a summary sentence, it is also supposed to have the potential to become a summary sentence even though it might have no similarity with other sentences in H at all. So, during the manifold-ranking process, similarity between a pair of sentences is acquired via the method discussed in Section 3 while H-sentence relevance is obtained as follows:

$$rel(s,H) = \max_{r\in H} sim(s,r) \tag{17}$$

We name our method for training data generation $sent\_manifold$. Additionally, biased LexRank (Erkan, 2006) is also a feasible solution to rank sentences for training data construction. Its primary idea is to generate a prior distribution for the objects in traditional random walk to reflect the bias degree of objects towards certain query. We will conduct experiments to compare our approach with some representative alternatives.

7 Experimental study

7.1 Data sets and evaluation metrics

We use the popular query-focused summarization benchmark data sets DUC2005[2], DUC2006[3] and DUC2007[4] for our experiments. Each of them consists of document sets and reference/human summaries. For documents, we use the OpenNLP[5] to detect and tokenize sentences. Stop words are removed and remaining words are stemmed using Porter stemmer[6]. In experiments, DUC2005 is used to train the model tested on DUC2006, and DUC2006 is used to train the model tested on DUC2007. Table 1 gives a short summary of the three data sets.

[2]http://www-nlpir.nist.gov/projects/duc/duc2005/tasks.html
[3]http://www-nlpir.nist.gov/projects/duc/duc2006/tasks.html
[4]http://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html
[5]http://opennlp.sourceforge.net/
[6]http://tartarus.org/ martin/PorterStemmer/

	DUC2005	DUC2006	DUC2007
Cluster number	50	50	45
Documents per cluster	25-50	25	25
Summary length limit	250 words	250 words	250words

Table 1: Summary of datasets

We use ROUGE (Lin, 2004) (version 1.5.5) toolkit[7] to measure the summarization performance. In experiments, we report three widely adopted F-measure metrics: ROUGE-1, ROUGE-2 and ROUGE-SU4, among which ROUGE-N means n-gram recall, and ROUGE-SU4 is based on unigram plus skip-bigram match with maximum skip distance of 4.

7.2 Experimental results

7.2.1 Comparison among some typical supervised summarization systems

First, we compare RelationListwise with some competitive and typical supervised summarization algorithms and three top systems of DUC. (1)Ranking-SVM: applying ranking-SVM directly; (2) Ranking-SVM-CSL: Ranking-SVM with Cost Sensitive Loss, proposed in (Shen and Li, 2011) (3)SVR: learning a regression model using SVM, presented in (Ouyang et al., 2007); (4)Listwise: similar to our RelationListwise while taking no account of sentence relationship; (5)top three systems with the highest ROUGE scores that participated in the DUC2006 (S12, S23, S24) and the DUC2007 (S4, S15, S29) for comparison, respectively.

Systems	ROUGE-1	ROUGE-2	ROUGE-SU4
SVR	0.41813	0.09492	0.15116
Ranking-SVM	0.42014	0.09713	0.15326
Ranking-SVM-CSL	0.42179	0.10332	0.15377
S23	0.40973	0.09785	0.14562
S12	0.41053	0.09633	0.15074
S24	0.41081	0.09857	0.15248
Listwise	0.42716	0.10387	0.16008
RelationListwise	**0.43066**	**0.10852**	**0.16324**

Table 2: F-measure comparison on DUC2006

Systems	ROUGE-1	ROUGE-2	ROUGE-SU4
SVR	0.43821	0.11997	0.16508
Ranking-SVM	0.44514	0.12213	0.17326
Ranking-SVM-CSL	0.44839	0.12332	0.17377
S4	0.43603	0.11785	0.17162
S29	0.43159	0.12048	0.17374
S15	0.44481	0.12907	0.17748
Listwise	0.45283	0.12667	0.17549
RelationListwise	**0.45852**	**0.13091**	**0.17824**

Table 3: F-measure comparison on DUC2007

Tables 2 and 3 present the performance of these systems with the metrics ROUGE-1, ROUGE-2, and ROUGE-SU4. From the results we can observe that in this task, listwise based methods

[7]http://www.isi.edu/licensed-sw/see/rouge/

(RelationListwise and Listwise) generally outperform pairwise methods (Ranking-SVM and Ranking-SVM-CSL), and the latter outperforms SVR, a pointwise learning to ranking. Even through (Shen and Li, 2011) developed cost sensitive loss to improve basic ranking SVM, its results are still inferior to that of Listwise based methods, which indicates the correctness of our choosing Listwise learning to rank as basic training model. More importantly, taking into account the sentence relatedness indeed improves the performance of $Listwise$. As the statistics show, $RelationListwise$ outperforms $Listwise$ over all three metrics.

7.2.2 Validation of feature design

Further, in order to investigate the effectiveness of combining our designed features, we compare our method RelationListwise with some baselines which mainly consider individual features: (1)Rel: a method considering only the sentence relevance towards a query and choosing the most relevant sentences to produce summary until length limit is reached. (2)Rel+MMR: similar with (1) except that we use MMR algorithm to control redundancy. It denotes a system considering query-biased relevance as well as information novelty. (3)Coverage: a baseline clustering-based method. It clusters sentences and selects the most relevant sentences from different clusters. Note that the clustering operation is carried to select sentences that have low degree of information overlap. So this baseline is similar with (2) for considering both relevance and novelty. (4)Manifold: ranking the sentences according to the manifold ranking scores and select top-ranked sentences to construct summary directly, where the parameter $\alpha = 0.5$. It corresponds to feature BIR. (5)Manifold+MMR: similar with (4) except to reduce redundancy via MMR algorithm (Wan et al., 2007). (4)Diversity: selecting sentences according to their query-biased diversity scores acquired using DivRank (Mei et al., 2010). It represents the feature: BIN. Tables 4 and 5 show their comparison results.

Systems	ROUGE-1	ROUGE-2	ROUGE-SU4
Rel	0.36775	0.07092	0.12777
Rel+MMR	0.37328	0.07109	0.12884
Manifold	0.38827	0.08028	0.13349
Coverage	0.39004	0.08394	0.13705
Diversity	0.39052	0.08814	0.13721
Manifold+MMR	0.39116	0.08741	0.13729
RelationListwise	**0.43066**	**0.10852**	**0.16324**

Table 4: Comparison results of feature design on DUC2006

Systems	ROUGE-1	ROUGE-2	ROUGE-SU4
Rel	0.38985	0.10075	0.13108
Rel+MMR	0.39938	0.1033	0.14501
Manifold	0.40214	0.10131	0.14833
Coverage	0.41243	0.11196	0.15537
Diversity	0.41440	0.11261	0.15502
Manifold+MMR	0.42015	0.11327	0.15936
RelationListwise	**0.45852**	**0.13091**	**0.17824**

Table 5: Comparison results of feature design on DUC2007

Obviously, the statistics point out the improvement of our approach combining multiple task-specific features over those baselines. While some reference systems involve more than one information aspect, such as *Rel+MMR* and *Manifold+MMR*, their performances are still limited.

7.2.3 Competitiveness of our similarity measure

In Section 7.2.1, experimental results have validated our proposal that exploiting sentence relation to improve the overall training model. Nevertheless, they could not point out the superiority of our designed measure of sentence similarity. Hence, we keep consistency for our algorithm framework except to replace the part of calculating sentence similarity. Since there are lots of existing sentence similarity measures, and it's hard to compare qualities of them all, we just select following typical alternatives: (1)cosine measure; (2)wordSimi_sentStruct: The measure proposed in (Yin et al., 2012), which determined words semantic similarity by computing the cosine value of the words' distribution representations over latent topics, and identified sentence structure similarity using LCS and etc.; (3)geneProb: Generation probability method presented in (Erkan, 2006). Note that generation probability is not necessarily symmetric, we just average the mutual generation probabilities of two sentences as their final similarity value in experiments. We provide their comparison statistics in Tables 6-7.

Systems	ROUGE-1	ROUGE-2	ROUGE-SU4
Listwise	0.42716	0.10387	0.16008
cosine measure	0.42906	0.10659	0.16239
geneProb	0.42923	0.10671	0.16246
wordSimi_sentStruct	0.43004	0.10726	0.16307
RelationListwise	**0.43066**	**0.10852**	**0.16324**

Table 6: Comparison results of sentence similarity measures on DUC2006

Systems	ROUGE-1	ROUGE-2	ROUGE-SU4
Listwise	0.45283	0.12667	0.17549
cosine measure	0.45626	0.12819	0.17634
geneProb	0.45693	0.12873	0.17616
wordSimi_sentStruct	0.45793	0.12956	**0.17833**
RelationListwise	**0.45852**	**0.13091**	0.17824

Table 7: Comparison results of sentence similarity measures on DUC2007

The two tables demonstrate the influence of different sentence similarity measures on our approach. Among the four kinds of measures, $geneProb$ is close with $cosine$ with slight improvement while $wordSimi\_sentStruct$ is more competitive to our proposed similarity measure. Except that $wordSimi\_sentStruct$ performs slightly better than $RelationListwise$ in ROUGE-SU4 over DUC2007, our proposed measure is more superior in other metrics. Note that we also put method $Listwise$ in the tables, and yet its performance is relatively poor compared with ones involving inter-sentence impacts. It further validates that sentence relatedness is worth considering when dealing with sentence ranking problem.

7.2.4 Training data generation comparison

In this section, we empirically investigate the effects of different strategies for training data generation. In Section 6, we have given the reason why choose to compute the similarity of an estimated sentence towards one sentence in H instead of the whole H. Additionally, we also come up with a novel approach for measuring sentence similarity in Section 3. In general, we could treat reference summary H as (1): a long and single sentence (labeled as *summ*), or (2): a sentence set (labeled as $sent$); meanwhile, ranking methods could be classified into (1): manifold, (2): biased LexRank, and (3)simi: computing the similarities of estimated sentences towards the summary directly. So, we can combine them into 6 kinds of pairs: $summ\_manifold$, $summ\_biasedLexRank$, $summ\_simi$, $sent\_manifold$, $sent\_biasedLexRank$ and $sent\_simi$. Remember that $sent\_manifold$ is our proposed method for training data construction.

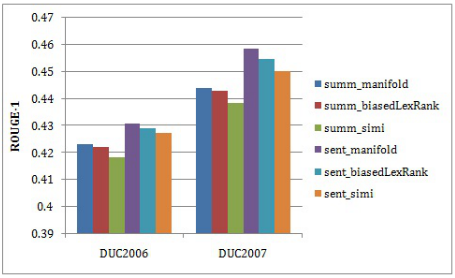

Figure 1: Performance comparison of methods about training data generation.

The comparison results are shown in Figure 1. From the comparison, we observe that:(1)graph-based methods, including $\{\cdot\}\_manifold$ and $\{\cdot\}\_biasedLexRank$, both outperform $\{\cdot\}\_simi$ methods which rank sentences by computing sentences' similarities towards the reference summary directly. It might result from that $\{\cdot\}\_simi$ solution is too simple to take into consideration the prestige of sentences in documents. (2) Using a sentence as the reference is much better than using the whole summary. As the figure shows, all $sent\_\{\cdot\}$ methods outperform $summ\_\{\cdot\}$. This may due to the fact that in constructing training data, we aim to judge the ability of a sentence to be a summary sentence, and it's best to be treated as the ability for the estimated sentence to *replace* certain sentence in H. Therefore, it is more rational to compare our estimated sentence with the sentences in H one by one rather than with the whole H. For example, if a sentence in H (e.g., h_i) has a high similarity (e.g., 0.9) with an estimated sentence s, then s is supposed to be able to replace h_i as a summary sentence. Whereas, if a long sentence s' is also very relevant to the whole H while having very low similarities with sentences in H, it is still not considered to be a good summary sentence.

Conclusion and perspectives

In this work, we propose a novel model named RelationListwise for query-biased multi-document summarization task. More specifically, through defining some unique sentence features and designing a creative measure for sentence relatedness, we integrate sentence relation information with listwise learning to rank to automatically learn feature weights. Experimental results suggest that our modification of basic listwise is considerably in favor of generating high-quality summaries. In future work, we will use more complex features and try some new summarization tasks.

Acknowledgments

This work is financially supported by HGJ Grant 2011ZX01042-001-001 and NSFC Grant 60933004.

References

Cao, Z., Qin, T., Liu, T., Tsai, M., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM.

CHALI, Y. and HASAN, S. (2011). Query-focused multi-document summarization: automatic data annotations and supervised learning approaches. *Natural Language Engineering*, 1(1):1–37.

Erkan, G. (2006). Using biased random walks for focused summarization. *Ann Arbor*, 1001:48109–2121.

Islam, A. and Inkpen, D. (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10.

Jin, F., Huang, M., and Zhu, X. (2010). A comparative study on ranking and selection strategies for multi-document summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 525–533. Association for Computational Linguistics.

Li, L., Zhou, K., Xue, G., Zha, H., and Yu, Y. (2009). Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of the 18th international conference on World wide web*, pages 71–80. ACM.

Li, X., Shen, Y., Du, L., and Xiong, C. (2010). Exploiting novelty, coverage and balance for topic-focused multi-document summarization. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1765–1768. ACM.

Li, Y., McLean, D., Bandar, Z., O'Shea, J., and Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1138–1150.

Lin, C. (2004). Rouge: A package for automatic evaluation of summaries. In *Proceedings of the workshop on text summarization branches out (WAS 2004)*, volume 16.

Maas, A., Daly, R., Pham, P., Huang, D., Ng, A., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*.

Maas, A. and Ng, A. (2010). A probabilistic model for semantic word vectors. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*.

Mei, Q., Guo, J., and Radev, D. (2010). Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1009–1018. ACM.

Otterbacher, J., Erkan, G., and Radev, D. (2005). Using random walks for question-focused sentence retrieval. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 915–922. Association for Computational Linguistics.

Ouyang, Y., Li, S., and Li, W. (2007). Developing learning strategies for topic-based summarization. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 79–86. ACM.

Qin, T., Zhang, X., Tsai, M., Wang, D., Liu, T., and Li, H. (2008). Query-level loss functions for information retrieval. *Information Processing & Management*, 44(2):838–855.

Quan, X., Liu, G., Lu, Z., Ni, X., and Wenyin, L. (2010). Short text similarity based on probabilistic topics. *Knowledge and information systems*, 25(3):473–491.

Shen, C. and Li, T. (2011). Learning to rank for query-focused multi-document summarization. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 626–634. IEEE.

Wan, X. and Yang, J. (2008). Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM.

Wan, X., Yang, J., and Xiao, J. (2007). Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of IJCAI*, volume 7, pages 2903–2908.

Wei, F., Li, W., Lu, Q., and He, Y. (2008). Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 283–290. ACM.

Xia, F., Liu, T., and Li, H. (2009). Top-k consistency of learning to rank methods. *Advances in Neural Information Processing Systems*, pages 2098–2106.

Xia, F., Liu, T., Wang, J., Zhang, W., and Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM.

Yin, W., Pei, Y., Zhang, F., and Huang, L. (2012). Automatic multi-document summarization based on new sentence similarity measures. In *Proceedings of the 2012 Pacific Rim International Conference on Artificial Intelligence*, pages 832–837.

Zhang, J., Sun, Y., Wang, H., and He, Y. (2011). Calculating statistical similarity between sentences. *Journal of Convergence Information Technology*, 6(2).

Zhao, L., Huang, X., and Wu, L. (2005). Fudan university at duc 2005. In *Proceedings of DUC*, volume 2005.

Zhou, D., Ding, Y., You, Q., and Xiao, M. (2011). Learning to rank documents using similarity information between objects. In *Neural Information Processing*, pages 374–381. Springer.

Zhou, D., Weston, J., Gretton, A., Bousquet, O., and Scholkopf, B. (2004). Ranking on data manifolds. *Advances in neural information processing systems*, 16:169–176.