Order Retrieval

Neil Rubens, Vera Sheinman, Takenobu Tokunaga, and Masashi Sugiyama

Department of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo 152-8552, Japan
neil@hrstc.org,{vera46,take}@cl.cs.titech.ac.jp,sugi@cs.titech.ac.jp

Abstract. Extensive work has been done in recent years on automatically grouping words into categories. For example, {*Wednesday*, *Monday*, *Tuesday*} could be grouped into a 'days of week' category. However, not only grouping the words, but also ordering them is important, e.g. *Monday*→*Tuesday*→*Wednesday*. The order relation is an important aspect that could be used to enrich existing ontologies, to determine the sequence of actions for planning tasks, and to determine the order of user's preferences for a set of items, etc. However, automatically determining the order relation seems to have been ignored. Pairwise similarity metric commonly used to cluster words may not be well suited for the ordering task. Therefore, we propose a new metric designed for the ordering task. We utilize statistical proximity features of the terms in the documents (in a large corpus) in order to determine the order relations between terms. The effectiveness of the proposed method is verified in experimental settings against orders provided by human subjects.

1 Introduction

Computational linguistics provides a large body of research on categorization [1,2,3,4]. For example, many lexical ontologies such as WordNet [5] provide hierarchical clustering of terms (concepts) based on lexical categorization. However, in some applications categorical information alone is not sufficient, as illustrated by the following examples:

- Which restaurant should the recommender system suggest based on customer's reviews that include one of the terms such as *{horrible, edible, good, delicious}*?
- What is the correct order of actions for baking a pie *{wash, cut, bake, serve}*?

To answer the above questions the information about the ordering of the terms is required. Words may be ordered by various semantic features. Some terms are ordered by the time feature e.g. *past, present, future*, by the dimensional feature e.g. *line, circle, sphere*, etc. [6]. We propose[1] to retrieve information about the order of the terms by utilizing the empirical observation that words that are part of sequence tend to appear sequentially in text (e.g. *one* tends to appear before *three*). We extract the information about the terms order, by utilizing statistical properties of the total and partial orders of the terms in the corpus.

[1] The source code, online application, the sequence dataset and additional materials are available at http://hrstc.org/or

T. Tokunaga and A. Ortega (Eds.): LKR 2008, LNAI 4938, pp. 310–317, 2008.

2 Related Work

Many works in computational linguistics explore semantic relatedness between words, clustering of words or concepts by shared meaning, and so forth. Some methods [1] extract the similarity information from high-quality ontologies constructed manually by experts, such as WordNet [5]. Others exploit various computational techniques to measure similarity in a large corpus, such as the Web [2]. Weeds and Weir [3] provide an excellent survey on distributional similarity techniques. However, we are not aware of studies that extract or represent the relation of order among words, or members in a cluster.

The similarity and clustering approaches in their current state are not suitable for the task of ordering (as confirmed by our evaluation in Section 4), since similarity information does not necessarily provide the order information. These approaches are complimentary to our work, in a sense, that we add directionality to undirected lexical sets.

3 Proposed Approach

Given a set of terms we want to find an order of terms that is representative of the corpus in which terms occur. In order to achieve this we analyze the order in which the terms appear in the documents (Section 3.1). The proposed approach is based on the empirical observations that words that are part of a sequence, tend to appear after each other in documents. This approach tends to achieve a high accuracy when there is sufficient data. However, the number of documents that contain all of the terms could be small or equal to zero (especially when the number of terms is high). This may make an estimation of the order unreliable. To cope with this, we propose another method that estimates the order of the terms from their partial orders (Section 3.2). That is, we estimate the relative order first (e.g. the term t_i usually occurs before the term t_j). Then by using the partial order we estimate the total order of terms in the sequence. In the following subsections we define the problem in a more formal way, and describe in detail proposed approaches and their advantages and limitations.

3.1 Total Order Sequence Retrieval

In this approach we try to find the most probable sequence of the terms, by estimating the probability of the complete sequence occurring in the corpus. Let us consider a corpus of documents D. A document $d \in D$ could be represented as a sequence of terms $w \in T$ in the order that they occur in the document:

$$d = (w_1, w_2, \ldots, w_l), \tag{1}$$

where l is the number of terms in the document. Let us introduce an operator '$\prec$' – 'precedes'. An expression $(t_i \prec t_j)$, where $t \in T$, means that in a given document, term t_i occurs before t_j, and that there could be zero or more terms between t_i and t_j. This operator could be applied to an arbitrary number of terms e.g. $(t_i \prec t_{i+1} \prec \ldots \prec t_{i+n})$. Let us also define an indicator I that determines whether a sequence S could be 'satisfied' in a document d. That is, for an arbitrary sequence $S = (t_1 \prec t_2 \prec \ldots \prec t_n)$, where $t \in T$ and n is the length of the sequence; we want to determine if for the given document d, there exist terms $w \in d$ that follow the same order as terms $t \in S$. If sequence $S = (t_1 \prec t_2 \prec \ldots \prec t_n)$ could be satisfied in a given document d, then $I(S, d) = 1$, if not then $I(S, d) = 0$. We can calculate the number of documents where the sequence S could be satisfied as:

$$df(S) = \sum_{d \in D} I(S, d), \tag{2}$$

where '*df*' stands for the *document frequency*. We can now estimate the probability of sequence S occurring in the corpus D as:

$$\hat{P}(S) = \frac{df(S)}{|D|}, \tag{3}$$

where $|D|$ is the number of the documents in the corpus. In practice, we can take advantage of the corpus index [7] in order to approximate the value of $df(S)$ directly. The problem of finding the most probable sequence can be formulated as:

$$\arg\max_S P(S). \tag{4}$$

In order to find the most probable sequence it is necessary to evaluate $\hat{P}(S)$ for all $n!$ permutations of the sequence S (where n is a number of terms in a sequence). This makes the total order approach intractable for longer sequences. The estimator $\hat{P}(S)$ is accurate in the cases when the number of documents that contains all of the terms in S (in arbitrary order) is high. However, in the cases where the number of documents containing *all* of the terms in S is small, the estimator $\hat{P}(S)$ may become unreliable as confirmed in Section 4. In the next subsection we propose a method that addresses this limitation.

3.2 Partial Order Sequence Retrieval

Evaluating all of the permutations obtained by total order approach (Section 3.1) may not be tractable for longer sequences. The full sequence information may also be biased when only a few documents are available. However, many documents that lack the total order information, do contain the partial order information. For these reasons, we estimate the total order of terms from partial orders. Hence, the implementation of the task presents a tradeoff among availability of information, precision, and efficiency.

We are trying to find the most probable sequence of terms $S = (t_1 \prec t_2 \prec \ldots \prec t_n)$ such that $\arg\max_S P(S)$. It is not necessary to extract the order from the documents that contain all of the terms in S (as in Section 3.1). We can extract the partial ordering information, even though not all of the terms appear in the document. For example, if t_j tends to occur before t_k in the documents, and t_k tends to occur before t_l, we can infer that $(t_j \prec t_k \prec t_l)$ is the most probable order. That is, we can reconstruct the total order of the terms by utilizing the partial order information.

We can apply the standard sorting algorithms in order to approximate the total order from the partial orders. When the sorting algorithm requests to compare arbitrary terms t_j and t_k we return $t_j \prec t_k$ (i.e. t_j precedes t_k) if $\hat{P}(t_j \prec t_k) \geq \hat{P}(t_k \prec t_j)$, and $t_k \prec t_j$ otherwise (where the probability of sequence occurring is calculated by Eq. (3)). Standard sorting algorithms, such as quicksort, are based on average $\mathcal{O}(n \log n)$ comparisons between pairs, and in the worst case $\mathcal{O}(n^2)$ comparisons (where n is the number of terms in the sequence). Note that sometimes the obtained partial order information may not be consistent. That is, we may have a conflicting information e.g $t_j \prec t_k$, $t_k \prec t_l$, but $t_l \prec t_j$. To cope with inconsistent cases, we terminate the algorithm after at most n^2 iterations and return a probable order.

When the sufficient number of the documents is available, the total order approach (Section 3.1) would be more accurate than the partial order approach. The partial order of the terms may change depending on the context in which they appear. The context of the documents used for the total order is less likely to change, since all of the sequence terms must appear in each of the documents.

4 Experimental Evaluation

In this section, we evaluate the performance of the proposed approaches (Section 3.1 and Section 3.2) and a baseline term clustering approach (Section 4.3).

4.1 Data Set

The dataset for the experiments is comprised of 73 sequences of words (Table 1) that were collected in the following manner. Five computer science students were the subjects of sequences collection. Two of them were native English speakers. Each one of the subjects suggested some sequences and verified all the sequences suggested by others. The sequences that were not unanimously agreed upon were removed from the test set, or changed to satisfy all the subjects.

4.2 Error Metric

To evaluate the quality of the acquired sequences in comparison with the sequences proposed by the human subjects, we use *Kendall tau* as a base measure [8]. The Kendall tau distance counts the number of pairwise disagreements between two lists. It counts the number of swaps necessary to place one list in the same order as the other list (normalized by the length of the list). The larger the distance is, the less similar the two lists are. The Kendal tau distance is expressed as:

$$KT(S_1, S_2) = \frac{|(t_i, t_j) : t_i \prec t_j, idx(t_i, S_1) < idx(t_j, S_1) \wedge idx(t_i, S_2) > idx(t_j, S_2)|}{n(n-1)/2}, \tag{5}$$

Table 1. Some of the sequences used in the evaluation. For the complete list please see http://hrstc.org/or.

Sun,Mercury,Venus,Earth,Mars,Jupiter,Saturn,Uranus,Neptune,Pluto
byte,kilobyte,megabyte,gigabyte,terabyte
a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z
dawn,morning,afternoon,evening,night
breakfast,lunch,dinner,supper
stand,walk,run
spotless,clean,dirty,filthy
wash,cut,bake,serve
baby,infant,child,teenager,adult,elderly
solid,liquid,gas
planet,solar system,galaxy,universe

where $|(t_i, t_j) : t_i \prec t_j, idx(t_i, S_1) < idx(t_j, S_1) \wedge idx(t_i, S_2) > idx(t_j, S_2)|$ is the cardinality of the set of tuples that have pairwise disagreement between two lists S_1 and S_2; $idx(t, S)$ refers to the index of the term t in the list S; and $n(n-1)/2$ is the length normalization factor.

In the current settings, a sequence is considered correct if the order of the terms is completely reversed. This is due to an unclear ascending or descending fashion of a sequence. Therefore, we modify Kendall tau (Eq. (5)) to treat the inverse transpositions equally to the original order:

$$KT'(S_1, S_2) = |2KT(S_1, S_2) - 1|. \tag{6}$$

By using a Kendall tau metric, we do not only judge whether the acquired order was right or wrong, but also approximate the quality of the order by a range of $[0, 1]$, where 1 stands for identical sequences or the inversely transposed sequence, and 0 for the non-correlated sequences.

4.3 Baseline Method

Baseline methods for evaluation are difficult to determine since we are not aware of any previous studies on automatic retrieval of orders. There have been related work on automatic term clustering [1,2,3,4]. However, many of these methods rely on the ontologies (e.g. WordNet [5]) or text corpora. Therefore, obtaining the relation between arbitrary terms may not be possible for these methods, unless given terms exist in an ontology or a corpus. Recently, a *normalized google distance* (NGD) method has been proposed to automatically extract term similarity information from the large corpus of data [2]. As the proposed approaches, NGD also utilizes statistical properties of the corpus such as document frequency and joint probability. For these reasons, we choose NGD as a baseline for the comparison with the proposed approaches (Section 3.1 and Section 3.2). NGD uses the pairwise similarity metric for the automatic terms clustering. The relation between terms is calculated as:

$$NGD(t_i, t_j) = \frac{\max\{\log df(t_i), \log df(t_j)\} - \log df(t_i, t_j)}{\log |D| - \min\{\log df(t_i), \log df(t_j)\}}, \tag{7}$$

where $df(t)$ is the number of documents that contain term t, and $|D|$ is the total number of documents in a corpus. Note that the NGD metric is symmetric i.e. $NGD(t_i, t_j) = NGD(t_j, t_i)$. Using NGD the sequence score is calculated by aggregating the NGD score for the term's relations in the order that they occur in the sequence, i.e.:

$$NGD(S) = \prod_{i=1}^{n-1} NGD(t_i, t_{i+1}), \tag{8}$$

where t_i refers to the ith term in the sequence S. The top sequence is determined as:

$$\arg\max_{S} NGD(S). \tag{9}$$

4.4 Results

We evaluate the performance of the proposed total order approach (Section 3.1) and the proposed partial order approach (Section 3.2) with the baseline NGD approach (Section 4.3). Methods are evaluated on the dataset described in Section 4.1. Although, these methods are extendable to any of the search engines that provide proximity search capability, in this paper we present experiments conducted with the use of Yahoo API [9]. We evaluate methods by using several metrics. The accuracy is measured by the number of correctly obtained orders. The quality of the obtained sequences is measured by a modified version of a Kendall tau (KT) distance metric (as described in Section 4.2). We also evaluate the quality of the sequence when sufficient information is available ($df > 50$). The performance of the methods in relation to the sequence length is measured by the average length of the correct sequence. Running time is measured by the computational complexity. Results of the evaluation are shown in Table 2. Proposed approaches outperform the baseline NGD method on all of the metrics. We discuss obtained results in more detail in the following paragraphs.

Table 2. Comparison of methods for order retrieval. Proposed methods are indicated by '*'. Best performing method for a given metric is indicated by '°'. For the sequence quality, a higher value of Kendall tau distance corresponds to a better sequence quality (score of 1 corresponds to the correct sequence). For details see Section 4.4.

	NGD	Total Order*	Partial Order*
Accuracy (binary loss function)	9.5%	35.6%	°39.7%
Avg. Sequence Quality (KT)	0.39	0.42	°0.63
Avg. Sequence Quality (KT, $df > 50$)	0.39	°0.96	0.63
Avg. length of correct sequences	3.3	4.3	°5.5
Computational Complexity	$\mathcal{O}(n!)$	$\mathcal{O}(n!)$	°$\mathcal{O}(n^2)$

Term Clustering Approach (Baseline): In this paragraph we examine performance of the baseline term clustering method NGD [2] (Section 4.3) on the task of term ordering. Term clustering (lexical similarity) approaches allow us to group terms together, although ordering of the terms is not intended, and may not be well suited for obtaining term ordering as confirmed by evaluation results (e.g. the accuracy of NGD is 9.5% in comparison to 35.6% and 39.7% of the proposed methods as shown in Table 2). NGD is not well suited for the ordering task, since it is symmetric (i.e. it does not contain directionality). That is, the relation between *Monday* and *Tuesday* is the same as the relation between *Tuesday* and *Monday*, i.e. $NGD(Monday, Tuesday) = NGD(Tuesday, Monday)$. Symmetricity of the NGD metric probably hinders its performance. In general, metrics that are used for clustering may not be well suited for ordering tasks. In addition, the computational complexity of applying pairwise similarity in order to find term's sequence is $\mathcal{O}(n!)$, which makes it not tractable for longer sequences.

Total Order Approach (Proposed): The proposed total order approach works well, when the sufficient number of documents is available (as discussed in Section 3.1). When sufficient data is available, the quality of the order acquired by this approach is impressively high, Kendall-Tau of 0.96 (max is 1) for the retrieved sequences, when the data from more than 50 documents was available i.e. $df > 50$. However, even in a huge corpus, not all the information is available, and in more than 58% of the cases it was difficult to acquire all the needed information. Out of the total of 73 sequences in the dataset, for the 38% of sequences there were no documents that contained all of the terms of the sequence i.e. $df = 0$; further 20% of the sequences had fewer than 50 documents i.e. $df < 50$. With $df < 50$ sequences , Kendall-Tau for the retrieved sequences was only 0.42.

Partial Order Approach (Proposed): The proposed partial order approach is able to construct a total order by utilizing the partial order information available in the documents. To illustrate usefulness of the partial order approach, consider the query "$wash * cut * bake * serve$" that represents rough order of actions for baking a dish. This query retrieves no documents i.e. $df = 0$. Its inverse transposition as well as other transpositions, all result in $df = 0$. However, by combining the available partial order information using the proposed approach returned the expected $wash, cut, bake, serve$ as the retrieved order of terms. The partial order approach achieves the best performance on most of the metrics. However, in cases when the sufficient information is available, it is outperformed by the total order approach.

5 Conclusion

The term order is an important relation that could be used in many practical applications e.g. to find a sequence of actions for planing tasks, to arrange terms by a certain feature (e.g. for recommender systems), etc. In this paper, we proposed to automatically determine the order of terms. We have shown that existing similarity metrics may not be well suited for the ordering task; and, in turn, proposed a metric that utilizes the statistical proximity information to retrieve the order of the terms.

There are several tradeoffs that should be taken into account when extracting the order of the terms. Examination of all possible permutations leads to $\mathcal{O}(n!)$ time complexity (where n is the length of a sequence) that is intractable even for relatively short sequences. We can reduce the time complexity to $\mathcal{O}(n^2)$ and maintain a fairly good accuracy, by utilizing the partial order to approximate the total order of the terms in the sequence.

Another tradeoff to consider is the availability of information as opposed to accuracy. When sufficient amount of information is available, the precision of the total order approach is excellent. However, in our experiments sufficient information was available in less than 50% of the cases. Using the partial order information such as the ordering between pairs of terms increases the recall but reduces the accuracy.

We hope there will be more works dealing with the term ordering. The source code, online application, the sequence dataset and additional materials used in this paper are made available at http://hrstc.org/or.

References

1. Budanitsky, A., Hirst, G.: Evaluating WordNet-based measures of lexical semantic relatedness. Computational Linguistics 32(1), 13–47 (2006)
2. Cilibrasi, R., Vitanyi, P.: The google similarity distance. IEEE Transactions on Knowledge and Data Engineering 19(3), 370–383 (2007)
3. Weeds, J., Weir, D.: Co-occurrence retrieval: A flexible framework for lexical distributional similarity. Computational Linguistics 31(4), 439–475 (2005)
4. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th conference on Computational linguistics, Nantes, France, pp. 539–545. Association for Computational Linguistics (1992)
5. Miller, G.A.: WordNet: A lexical database for English. Communications of ACM 38(11), 39–41 (1995)
6. Sheinman, V., Rubens, N., Tokunaga, T.: Commonly perceived order within a category. In: Aberer, K., et al. (eds.) ISWC 2007. LNCS, vol. 4825, Springer, Heidelberg (2007)
7. Wikipedia: Proximity search: usage in commercial search engines — Wikipedia, the free encyclopedia (2007) (Online, accessed 2007)
8. Kendall, M.: A new measure of rank correlation. Biometrika 30, 81–89 (1938)
9. Yahoo, `http://www.yahoo.com`