Commonly Perceived Order within a Category

Vera Sheinman, Neil Rubens, and Takenobu Tokunaga

Department of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8550 Japan
{vera46,take}@cl.cs.titech.ac.jp
neil@hrstc.org

Abstract. Lexical ontologies cluster words by categories. Rarely they provide information about the inner relationships between words in a single category, particularly the order relation. We discuss the human perception of order among members of a single semantic category and introduce a novel computational method, Word Sequences, that arranges words by a commonly perceived order. We use proximity-search to acquire properties of order from the English webpages. The effectiveness of the proposed method is verified in a preliminary experimental settings against orders provided by human subjects. The method might be applied for enrichment of existing ontologies with order relation, and might be used for planning tasks, recommendation systems and second language acquisition purposes.

Key words: lexical acquisition, second language acquisition, ontology, typicality, common order

1 Introduction

Cognitive Linguistics provides a large body of research on categorization [1]. Many categories or lexical sets do not have a particular order among their members. Others are commonly perceived as ordered. Days of week, for instance, is a lexical set commonly ordered as *Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday* in English.

Lexical ontologies, such as WordNet [2] represent information on lexical relations, however, they do not provide order information among each category's members, such as co-hyponyms. We propose a method to enhance the existing relationships, by order information.

What is "commonly perceived" is rather vague. For the purpose of this study we approximated this notion by collection of sequences that seemed "common" to the authors of this paper and three additional human subjects. We approximate "common" by gathering the knowledge from the Web that averages out and emphasizes the common human beliefs about the world.

The proposed method orders a given set of words as illustrated in the schematic in Fig. 1. It is based on the observation that words that are part of common sequences tend to appear sequentially in texts (e.g. *one* tends to appear before

three in human expression). We extract the information about the appearance of the words by utilizing statistical properties of the partial orders of the terms of the corpus' documents, in particular, document frequency (*df*).

In many cases only partial ordering of a sequence is available from a single resource. Sometimes, the full sequence information available is misleading when it is biased to particular pages. While most of the pages lack the total order information, but do have partial. For this reason, we estimate the total order of terms from partial orders.

In the following subsections we discuss the varieties of existing lexical sequences and provide an overview of possible applications of the proposed method.

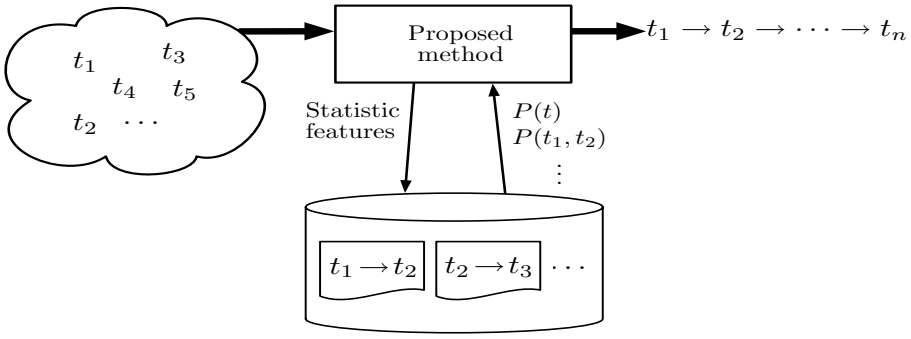


Fig. 1. Proposed sequence ordering method

1.1 Sequence Types

Some words are typically ordered by a certain semantic feature. Consider the order by the time feature *past, present, future* and *breakfast,lunch,dinner,supper*, or the dimensional feature *line, circle, sphere*. Some words are ordered by an arbitrary parameter, such as the lexicographic order. Some words are ordered compositionally, from the most general term to the most specific one, or vice versa, as in *universe, galaxy, solar system, planet earth*. Some sequences are well defined, consider *winter, spring, summer, autumn*, while others are much more fuzzy, as in *one, couple, few, several, many, lots*.

The differences in the types of sequences may affect the extraction results and require an extensive research. Also, some items may be ordered differently in different domains. In this study, we try to extract the most salient order of any type, leaving the differences for later investigation.

1.2 Applications

Having a computational method that will sort given lexical sets by commonly-perceived order will be beneficial for recommendation systems, planning appli-

cations, First and Second Language Acquisition, and for cognitive research in general.

A possible use is in recommendation systems. Content-based recommendation systems [3], [4], [5] attempt to predict user's preferences for items based on their descriptions. Which restaurant should the system recommend to users from reviews that include *horrible, edible, good, delicious*? Which teacher should it choose, *incompetent, adequate, skilful*, or *masterful*? What should it learn from the clue-words *loath, dislike, like, fond, love*?

Planning tasks deal with sorting of actions by the order of their performance [6], [7]. Machine, or a robot that constructs a plan for basic human actions, may use an automatic method to extract the order that is a matter of common sense for people. For instance, the rough plan for baking a pie will comprise the stages of washing the ingredients, cutting them, baking them and serving the pie. Our method approximates the typical order of the words as *wash, cut, bake, serve*, and could assist in knowledge base construction for planning tasks.

A method for extraction of the common order might serve as a textbook authoring tool for teachers, and as an exploratory device for students. Several language textbooks for children and beginner second language learners organize vocabulary in lexical sets [8]. One of the tasks that is commonly used is ordering actions or words in the new language. For instance, New English File Elementary textbook [9] provides the following exercise for students.

Example 1. What's the next word? Example: one, two, three

1. ten, twenty, _____
2. Monday, Tuesday, _____
3. July, August, _____
4. spring, summer, _____

Combination with WordSets for SLA WordSets [10] method builds lexical sets suitable for language learners. Given several input example words, such as *Monday* and *Tuesday*, it will provide the user with a set of words so that:

- semantically similar to examples he knows. For instance, *January* is not a good output word, since it is not a day of week.
- similar to the given examples by their level of typicality of usage. Both *Monday* and *Tuesday* are very basic and prototypical words, and so is *Saturday*, but not *Sabbath*.

The filtering provided by WordSets protects the learners from receiving redundant information, when they search for words similar to the examples they already know. It may also provide a useful extension for a dictionary or serve textbooks authors allowing them to extract lexical sets, suitable for learners level.

Consider a learner that knows the words *Monday* and *Sunday*. The combined method will provide the learner with similar words that are from the same conceptual category of days of week, omitting such non-typical words as *Sabbath*

that will overwhelm the learner with unneeded information. The method for ordering these words will suggest the user that these words are typically ordered as *Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.* Combination of the two methods will provide further enhancement for thesauri and textbook authoring, for exercises such as in Example 1.

2 Related Work

Much work in Computational Linguistics explores lexical relatedness between words, clustering of words or concepts by shared meaning, and so forth. Some methods [11] extract the similarity information from high-quality ontologies constructed manually by experts, such as WordNet [2]. Others exploit various computational techniques to measure similarity in a large corpus, such as the Web [12]. Weeds and Weir [13] provide an excellent survey on distributional similarity techniques. Some works aim to distinguish a specific relation between words, hyponym-hypernym relation, being particularly popular [14]. However, we are not aware of studies that extract or represent the relation of order among similar words, or members in a cluster.

The similarity and clustering approaches in their current state are not suitable for the task of ordering as shown by our evaluation in section 4.2. These approaches are complementary to our work, in a sense, that we add directionality to the undirected lexical sets that may be acquired using one of them.

Inkpen and Hirst [15] introduce a method to acquire information about the differences among sets of near-synonyms, such as *error, mistake, slip and blunt.* They present a pattern-based approach to gather detailed information on the differences among synonyms from a dictionary. Their approach is relevant to our work and its incorporation may be useful in our future work. Our work is different from theirs in that we focus on the order relation between any given similar words.

Recent works such as [12] show the advantages of using *document frequency* measure provided by most of the search engines to measure similarity between two words. We also use *df*, but extract the commonly perceived order relation among given words.

3 Proposed Approach

We are looking for a sequence S of terms as $\{t_i\}_{i=1}^{n}$ to maximize a relation R that gives the highest value to the most probable sequence. We formulate the problem in terms of graph theory. It is equivalent to a Hamiltonian path problem [16]. Let terms $\{t_i\}_{i=1}^{n} = \{v_i\}_{i=1}^{n} = T = V$ be the vertices in the graph $G = (V, E)$. The weight of an arbitrary edge $(v_j, v_k) = e_{j,k} \in E$, where $v_j, v_k \in V$ and $v_j \neq v_k$ is defined by a score function as $R(e_{j,k}) = R(v_j, v_k)$. In graph G there is an edge between every two vertices $\forall_{v_j,v_k} \exists (v_j, v_k)$. The path (sequence) S is defined as the set of edges that connects terms in the desired

order. The goal is to find the sequence S such that the path score is maximized, i.e. $\arg\max_S R(S) = \prod_{e \in S} R(e)$.

3.1 Implementation

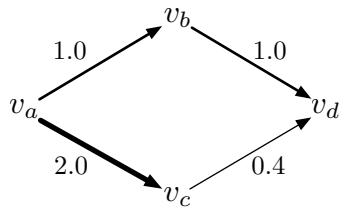
Modern search engines provide implicit or explicit proximity search capability. In our method, we assume the availability of proximity search; in particular, we use the operator roughly equivalent to FollowedBy($term_1$, $term_2$). Yahoo [17] and Google [18] search engines, each in a slightly different manner provide the binary wildcard ($*$) operator to represent up to two words between items. Although, this method is extendable to any of the search engines that provide proximity search capability, in this paper we present experiments conducted with Yahoo. We calculate the probability of occurrence of "$t_j * t_k$" as

$$P(\text{“}t_j * t_k\text{”}) = \frac{df(\text{“}t_j * t_k\text{”})}{|D|} \tag{1}$$

where $|D|$ is the total number of pages indexed by the search engine in use; and $df(\text{“}t_j * t_k\text{”})$ is the number of the documents that contain "$t_j * t_k$" in its text.

3.2 Sequence Score Estimation

In this section we define a method that assigns a score to the sequence of terms S. We calculate the score of the sequence as the aggregation of its pair wise relations. The simplest choices to aggregate relations are sum or product operators. Using the product as the aggregation operator allows us to penalize sequences that contain weak relations, resulting in more stable sequences scored higher, than by sum aggregator, as shown in the example in Fig. 2. Therefore, we define the sequence score as $R(S) = \prod_{e \in S} R(e)$.



$score_{prod}(R(v_a, v_b), R(v_b, v_d) = 1 > score_{prod}(R(v_a, v_c), R(v_c, v_d) = 0.8$
$score_{sum}(R(v_a, v_b), R(v_b, v_d) = 2 < score_{sum}(R(v_a, v_c), R(v_c, v_d) = 2.4$

Fig. 2. Illustration of stability preference by product aggregation

3.3 Optimization

In the task of obtaining term ordering, efficiency is an important issue. If all the permutations are evaluated, time complexity of the algorithm is $\mathcal{O}(n!)$, equivalent to the NP-Complete Asymmetric Travel Salesman Problem (ATSP). Hence, the implementation of the task presents a trade-off among availability of information, precision, and efficiency. Tractable methods for approximating order are described in the following subsections.

Time Complexity Optimization (TCO) In order to approximate the order in a feasible manner, given a sequence of terms to order, we sort them using one of the available sorting algorithms. Standard sort algorithms, such as quick-sort are based on $\mathcal{O}(n \log n)$ comparisons between pairs. For a pair of terms (t_j, t_k), if $P(\text{“}t_j * t_k\text{”})$ is bigger than $P(\text{“}t_k * t_j\text{”})$ we assume that the term t_j tends to precede the term t_k and return $t_j < t_k$ for the sorting algorithm, and vice versa. The efficiency of this approach is $\mathcal{O}(n^2)$ in the worst case. This approach is equivalent to a greedy solution of ATSP, and may not be optimal in terms of precision. Therefore, we present an extension to this approach in section 3.3.

Precision Optimization (PO) In order to improve the precision of the results acquired by the time-complexity optimization, once the sorted sequence is acquired, we use it as the heuristic to check all the permutations that are within a single transposition of neighboring terms in it. Due to the many times unclear ascending or descending fashion of a given sequence, the inverse transposition of the sorted sequence and all its permutations within a single transposition are tested. The permutations that are acquired using the described heuristics are sorted by their score as described in section 3.2. The procedure may be illustrated by the following example. Given a lexical set *stand, walk, run*, we sort it and acquire additional permutations using the sorted result as heuristics.

1. run, stand, walk - acquired by the sorting procedure
2. stand,run, walk - *stand* and *run* transposed
3. run, walk, stand - *walk* and *stand* transposed
4. walk, stand, run - inverse transposition
5. stand, walk, run - *stand* and *walk* transposed after inversion
6. walk, run, stand - *run* and *stand* transposed after inversion

We acquire the information about the document frequency of pairs from the Web. The total amount of pairs is $n * (n - 1)$, for n as number of members in a set.

$$P(\text{“}stand * walk\text{”})^1 = \frac{df(\text{“stand * walk”})}{|D|} = \frac{300{,}000}{10 \times 10^9} = 3 \times 10^{-5}$$

[1] The *df* numbers provided here may change as they are dependent on the index state of the search engine on a certain day. In this example they are true for April 26, 2007

$P(\text{“}walk * stand\text{”}) = 1.26 \times 10^{-5}$
$P(\text{“}stand * run\text{”}) = 4.69 \times 10^{-6}$
$P(\text{“}run * stand\text{”}) = 1.16 \times 10^{-5}$
$P(\text{“}walk * run\text{”}) = 7.86 \times 10^{-5}$
$P(\text{“}run * walk\text{”}) = 8.75 \times 10^{-5}$

The acquired sequences are ranked according to their scores as follows.

1. run, walk, stand - 1.10×10^{-9}
2. stand, walk, run - 2.37×10^{-9}
3. run, stand, walk - 3.48×10^{-10}
4. stand, run, walk - 4.10×10^{-10}
5. walk, stand, run - 5.91×10^{-11}
6. walk, run, stand - 9.20×10^{-11}

The ranking shows how likely is each order. The reverse sequence $run, walk, stand$ was ranked as the top-sequence, i.e. the most probable. Second top-ranked is the intended sequence $stand, walk, run$. In the example above, there were only 3 members in the sequence, so all the possible permutations were examined. However, for any given n, this approach will examine no more than n^2 sequences, and request information about $\frac{n!}{(n-2)!}$ pairs, keeping it computationally feasible.

3.4 Straightforward Approach (SA)

Our task is construction of total order from partial order information. In order to evaluate total order acquisition from total order information, we define the straightforward approach (SA) in this section.

In SA we check the *df* of the query “$t_1 * t_2 * ... * t_n$”. To keep the polynomial runtime of the procedure, we extract the *df* for the intended ordered sequence, and all the other orders that were acquired as the result of the precision optimization (section 3.3). In the best case the correctly ordered sequence or its inverse transposition receives the highest value of *df*.

4 Evaluation

The evaluation presented in this paper is preliminary. The test set for the experiments comprised 73 sequences of words that were collected in the following manner. Three 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 verifies all the sequences by others. The sequences that were not unanimously agreed upon were removed from the test set, or changed to satisfy all the subjects. All the sequences that appear in this paper as examples are taken from that test set.

There is no previous study on automatic sorting of sets that simulates human way of sorting. Therefore, baseline for evaluation is difficult to determine. We show good precision results that are superior to an undirectional approach of

lexical similarity measuring of Normalized Google Distance (NGD) as described in 4.2 and to a straightforward approach of checking the document frequency of the sequence as a proximity-search query. The comparative results of NGD, the proposed straightforward approach (SA), the proposed time-complexity optimization (TCO), and the proposed precision optimization (PO) are shown in Table 1.

Table 1. Comparison of methods for commonly perceived order extraction

	NGD	SA	TCO	PO
Avg. length of correct top-sequences	3.3	4.3	5.5	5.1
Correctly top-ranked sequences	7 (9.5%)	26 (35.6%)	29 (39.7%)	23 (31.5%)
Kendall Tau for the top-sequence	0.39	0.42	0.63	0.66

NGD: Normalized Google Distance
SA: Straightforward Approach
TCO: Time-Complexity Optimization
PO: Precision Optimization

4.1 Evaluation Method

To evaluate the quality of the acquired sequences in comparison with the sequences proposed by the human subjects, we use Kendall Tau measure [19] normalized to reward the inverse transpositions equally to the original order. Using Kendall Tau, 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 $[0, 1]$. 1 stands for identical sequences or for the inversely transposed sequence, and 0 for the non-correlated sequences. To evaluate an order acquired by our approach, we measure the distance between the expected correct sequence s_1 and the acquired order of the same sequence s_2 by equation (2).

$$KT(s_1, s_2) = abs(\frac{4|(i,j) : i < j, s_1(i) < s_1(j) \wedge s_2(i) > s_2(j)|}{n(n-1)}) - 1 \qquad (2)$$

4.2 Comparison with Lexical Similarity Measures

In this section we examine performance of the lexical similarity measures to the task of term ordering. Lexical similarity approaches allow to group terms together, however ordering of the terms is not intended, and not suited for obtaining term ordering as confirmed by our evaluation results. We use Normalized Google Distance (NGD) [12] since it allows to determine the relation between any two terms. We chose it for comparison, because it also uses *df* in Web search engines to extract similarity information.

For each ordered sequence acquired by precision optimization, we calculated the product of the NGD values among its pairs, similarly to the product we

compute using our method. Then we resorted the sequences, ranking them proportionally to the products of their NGD values. We calculated Kendall Tau distance between the top-sequence in terms of NGD values product and the expected sequence for each lexical set in our test set.

4.3 Comparison with Straightforward Approach

In the Straightforward Approach, out of 73 sequences, 28 sequences, 38% had $df = 0$. Further 20% had $df < 50$. The quality of the order acquired by this approach is impressively high, Kendall-Tau of 0.96 for sequences that were scored as top with $df > 50$. However, even in a huge corpus, not all the information is available, and in more than 50% of the cases it was difficult to acquire all the needed information from a single webpage for all the top-sequences. With $df < 50$ sequences , Kendall-Tau for top scored sequences was merely 0.42.

To illustrate usefulness of partial-information based methods, consider the query "$wash * cut * bake * serve$" that represents rough order of actions for baking a dish. The query results in $df = 0$. Its inverse transposition as well as other transpositions, all resulted in $df = 0$. However, using the proposed method of combining the partial information using the precision optimization returned the expected $wash, cut, bake, serve$ as the top result.

4.4 Sequence Types Analysis

The sequences were manually categorized (see 1.1). The highest accuracy for the proposed method was achieved within the largest group of 22 sequences ordered by time. 18 of them (82%) were sorted correctly by the proposed method. Numerical sequences, such as *first, second, ..., tenth* and spatial sequences, such as *Sun, Mercury, Venus, Mars* also showed good results. 7 sequences were categorized in each group and the proposed method sorted 5 (71%) correctly in each. Compositionally ordered sequences, however, imposed a difficulty for our method, out of 7 collected sequences that were categorized as such, none were ordered correctly. The high quality for time-based sequences might be explained by tendency of terms to appear chronologically in texts. Compositional sequences or meronym-holonym chains that might not be as easy to determine by order of appearance. We plan to investigate the characteristics of various sequence types further in the future.

5 Discussion

We introduced common word sequences and the applications of their automatic extraction in planning, education, and recommendation systems. Based on the observation that words that are part of a sequence, tend to appear after each other in text, we used the available proximity search functionality to extract order information. In some cases words may be mentioned in a different order,

however, these biases are averaged out by the huge size of the Web. The results were better than a baseline of NGD. They might have been affected by inaccuracies in the proximity search functionality. Experiments with various implementations of proximity search are needed.

There are several trade-offs that should be taken into account when finding the optimal sequence based on probabilities. Examination of all possible permutations leads to $\mathcal{O}(n!)$ time complexity that is hard in terms of computation for sequences of as small as 7 items, and unfeasible for relatively long sequences. We show that we can keep fairly good precision, performing the ordering task heuristically with only a polynomial time complexity.

Another trade-off to consider is the availability of information as opposed to precision. As we have shown using the straightforward method, its precision was superb, but the *df* information was available only in 50% of the cases. Using the partial information such as the ordering between pairs of words increases the recall. The proposed method provides the lexical equivalent of total order construction given partial order information.

Word Sequences provides sorting of a given sequence of items. We plan to expand its functionality to extract probable sequence members, given only its start and end-points. Items similar to the start and end-points might be extracted from ontology such as WordNet, or by combination with WordSets method that will not only extract similar members from an appropriate lexical set, but also provide the user with differentiating information between more useful and more obscure words for learners.

References

1. Croft, W., Cruse, A.: Cognitive Linguistics. Cambridge University Press, UK (2004)
2. Miller, G.A.: Wordnet: a lexical database for english. ACM **38**(11) (1995) 39–41
3. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on **17**(6) (2005) 734–749
4. Balabanovic, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. ACM **40**(3) (1997) 66–72
5. Pazzani, M., Billsus, D.: Learning and revising user profiles: The identification of interesting web sites. Machine Learning **27** (1997) 313–331
6. Tate, A., E.: Advanced planning technology – the technological achievements of the arpa/rome laboratory planning initiative. Menlo Park, CA: AAAI Press (1996)
7. Newell, A., Simon, H.A.: General problem solver – a program that simulates human thought. In E. A. Fiegenbaum and J. Feldman, Eds., Computers and Thought. New York: McGraw Hill (1963) 279–293
8. Nation, P.: Learning vocabulary in lexical sets: Dangers and guidelines. TESOL **9**(2) (2000) 6–10
9. C. Oxenden, C.L.K., Seligson, P.: New English File Elementary. Oxford University Press (2004)
10. Sheinman, V., Tokunaga, T.: Wordsets: Finding lexically similar words for second language acquisition. In: RANLP, Borovets, Bulgaria (September 2007) 530–536

11. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics **32**(1) (2006) 13–47
12. Cilibrasi, R., Vitanyi, P.: The google similarity distance. Knowledge and Data Engineering, IEEE Transactions on **19**(3) (March 2007) 370–383
13. Weeds, J., Weir, D.: Co-occurrence retrieval: A flexible framework for lexical distributional similarity. Computational Linguistics **31**(4) (2005) 439–475
14. Hearst, M.: Automatic acquisition of hyponyms from large text corpora. In: ACL. (1992) 539–545
15. Inkpen, D., Hirst, G.: Building and using a lexical knowledge base of near-synonym differences. Computational Linguistics **32**(2) (2006) 223–262
16. Ore, O.: A note on hamiltonian circuits. American Mathematical Monthly **67, 55** (1960)
17. Yahoo: http://www.yahoo.com
18. Google: http://www.google.com
19. Kendall, M.: A new measure of rank correlation. Biometrika **30** (1938) 81–89