Ontology-based proximity search

Tuan M.V. Le
HCMC University of Tech. and John von Neumann Institute - VNUHCM
tuanminhlv@yahoo.com

Tru H. Cao
HCMC University of Tech. and John von Neumann Institute - VNUHCM
tru@cse.hcmut.edu.vn

Son M. Hoang
Ho Chi Minh City University of Technology
VNUHCM
minhson@cse.hcmut.edu.vn

Junghoo Cho
Department of Computer Science
UCLA
cho@cs.ucla.edu

ABSTRACT

This paper presents our developed general open source for ontology-based information retrieval to answer queries that involve named entities with their ontological features, namely, aliases, classes, and identifiers. We propose a novel approach for semantic search engines that exploit the ontology features of named entities in proximity search and develop an algorithm for computing dynamic distances between named entities and keywords in queries and documents. In particular, it deals with phrase and proximity queries for which the token-based lengths and positions of the queried named entities in a document may vary. The result provides a platform and library for implementing semantic search engines.

Categories and Subject Descriptors
H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval – *query formulation, retrieval models, search process.*

General Terms
Algorithms, Theory.

Keywords
Semantic search, phrase query, term proximity, named entity.

1. INTRODUCTION

To answer queries in a document retrieval system, search engines are expected to return the most relevant documents at the top of the result list ([13]). Users tend to query terms that often appear in phrases or are close to each other in a document in which distances between terms, called term proximity, are implicitly used. In such cases, documents that have terms occurring close to each other are often ranked higher as a result. Understanding the interaction of terms being constituents of phrases within queries ([11]) could be useful to improve the performance. Therefore, many research works have been looking into the term proximity to improve the retrieval precision of top returned documents ([1], [2], [9], [12], [14]). Although those researches used different approaches, they showed that retrieval effectiveness could be improved by integrating proximity scores into an existing retrieval model.

Besides studies focusing on term proximity, there were works in semantic search considering ontological features of named entities to enhance document retrieval effectiveness ([3], [4], [5]). Named entities (NE) are those that are referred to by names such as people, organizations, and locations ([10]). The work [3], for instance, explored combinations of ontological features and keywords to represent a high level semantics of queries and documents. However, it appears that no work in semantic search considers proximity queries involving named entities and keywords.

As a related work, [8] explored proximity measures between named entities and keywords in the field of expert search, i.e., the task of finding people who have skills and experience on a given topic. However, it was not for text retrieval.

For keyword-based search engines, term proximity scoring models consider the distances among terms in queries and documents in terms of tokens. For instance, a document is analyzed into tokens when being indexed; each token is given a position and the distance between two tokens is computed based on their positions. For example, consider the following text:

D: "*Cultural exhibitions on Ho Chi Minh City development and integration opened in Lam Son park*"

The text is analyzed into 15 tokens: "*Cultural*", "*exhibitions*", "*on*", "*Ho*", "*Chi*", "*Minh*", "*City*", "*development*", "*and*", "*integration*", "*opened*", "*in*", "*Lam*", "*Son*", "*park*" whose positions are from 0 to 14, respectively. Distances between these tokens can then be computed easily based on these positions. For example, the distance between "*on*" and "*development*" is 4 because there are 4 tokens "*Ho*", "*Chi*", "*Minh*", "*City*" between them.

However, when taking NEs into account, the distance in terms of token is no longer applicable. Every NE appearing in the document often covers more than one token. For example, in the text *D* above, there is the NE *Ho Chi Minh City* that contains 4 tokens. We denote the number of tokens covered by a NE as the length of that NE. Moreover, a NE may have different aliases of different lengths. Therefore it can have different lengths in a document. In addition, a class may have different instances of NEs, causing the same problem due to variable lengths. Consider the following phrase queries to see how different lengths of a NE make token-based term proximity not applicable:

*Q*1: "*Cultural exhibitions on Saigon City development*"

*Q*2: "*Cultural exhibitions on city development*"

In *Q*1, *Saigon City* is a NE containing 2 tokens. Here the query requires that the term "*on*" is right before and the term "*development*" is right after *Saigon City* in the document. In the text *D* above, there is the phrase "*Cultural exhibitions on Ho Chi Minh City development*". Since *Ho Chi Minh City* is another name of *Saigon City*, intuitively *D* should match with *Q*1. However, according to the token-based term proximity approach, *D* cannot be returned because it does not satisfy the proximity conditions of *Q*1. Specifically, *Saigon City* has length 2 in *Q*1, and thus the distance between the terms "*development*" and "*on*" in a document is required to be 2. However, since *Ho Chi Minh City* has length 4 in *D*, so distance between the terms "*development*" and "*on*" is 4, not 2. Therefore, *D* cannot match with *Q*1. We can see that this error occurs because a NE may appear with different names and different lengths in queries and documents. Although NEs in a query and a document can match to each other (as *Saigon City* and *Ho Chi Minh City* in this example), their lengths may be different and this causes token-based term proximity inapplicable.

In *Q*2, it does not mention a specific city but all NEs belonging to the class *City*. Since *Ho Chi Minh City* is a NE belonging to the class *City*, *D* intuitively matches with *Q*2. However, the distance between the terms "*development*" and "*on*" is 1 in *Q*2 and is 4 in *D*. Therefore, *D* cannot match with *Q*2. In general, there are many other cities besides *Ho Chi Minh City* and each of them may have a different length. Their different lengths make the distances between the terms before and after them different too. Therefore term matching is not straightforward as in the case of plain keyword matching. It must deal with variable and dynamic distances caused by NEs of different lengths.

The two examples show that token-based proximity matching cannot be applied to queries and documents containing NEs. In this paper, we do not propose a different scoring model to term proximity. The work of this paper is to consider and analyze dynamic distances between keywords and named entities, and then develop an algorithm to compute such a distance on fly. After distances are computed, any scoring model mentioned above can be applied for ranking retrieved documents. Until now, to the best of our knowledge, there is no retrieval system that supports proximity queries on distances between keywords and named entities.

We rely on S-Lucene ([3]), which is an extension of Lucene ([6]) for semantic search but does not address and support query term proximity involving NEs, to implement our algorithm. Section 2 provides a brief introduction to Lucene, in particular about computing static distances between keywords in Lucene. Section 3 presents the basis of indexing and searching in S-Lucene. Section 4 presents in detail the algorithm for computing dynamic distances between keywords and named entities and implementation of proximity search in S-Lucene. Section 5 draws some conclusions and future works.

2. PROXIMITY SEARCH IN LUCENE

Lucene ([6]) is an open source library for storing, indexing, and searching documents. Lucene scoring uses a combination of the Vector Space Model (VSM) and the Boolean model. In addition to single or multiple keyword search, Lucene also supports proximity search based on term proximity. Lucene supports exact phrase query search and proximity search with distances greater than 0, but does not require the order of keywords to be satisfied (using the term *slop* to mean *distance*).

In Lucene, the distance between term *A* and term *B* in a document is the number of steps needed to move *A* (or *B*) to its correct position as they appear in the query. For example, given the query *q* = "*A B*", i.e., *B* is right after *A*. Assuming that there are following documents where "_" represents an anonymous variable that can be bound to any term:

d_1: "*A* _ _ _ *B* _"
d_2: "*B* _ *A* _"

In d_1, to be as appearing in the query, *B* has to be moved as follows:

d_1: "*A* ↙ _ ↙ _ ↙ _ *B* _"

That is, it takes three steps to move *B* to the left to be right after *A*. In d_2, to be as appearing in the query, *B* has to be moved as follows:

d_2: "*B* ↘ _ ↘ *A* ↘ _"

That is, it takes three steps to move *B* to the right to be right after *A*. Above, in a document, *A* is fixed and *B* is moved. Alternatively, *B* can be fixed while *A* is moved as follows:

d_1: "*A* ↘ _ ↘ _ ↘ _ *B* _" (taking three steps)
d_2: "↙ *B* ↙ _ ↙ *A* _" (taking three steps)

In general, given a query having *n* terms $\{t_1,\ldots,t_n\}$, in a document any term $t \in \{t_1,\ldots,t_n\}$ can be fixed the other terms are moved to their right positions as appearing in the query. The movement distance of a term t_i with respect to the fixed term t_0 is calculated by the following formula:

$$step_i = (pd_i - pq_i) - (pd_0 - pq_0) \qquad (1)$$

where:
pd_i: position of term t_i in the document,
pq_i: position of term t_i in the query,
pd_0: position of term t_0 in the document,
pq_0: position of term t_0 in the query.

One can see that:
$step_i > 0$: t_i will be moved to the left,
$step_i < 0$: t_i will be moved to the right,
$step_i = 0$: t_i will not be moved.

With such distance movement calculation, Lucene's conditions for matching a document *d* and a proximity query *q* are as follows:

$slop = 0$: *d* matches with *q* if $\forall i$: $step_i = 0$
$slop > 0$: *d* matches with *q* if $maxstep \le slop$,
where $maxstep = \max_{1 \le i \le n}(step_i)$

Since there is more than one way of choosing a fixed term, the *maxstep* value may vary. Let $\Delta_i = pd_i - pq_i$ be the difference between the positions of term t_i in the document and in the query. Lucene can choose any term t_i satisfying $\Delta_i = \Delta_0 = min(\Delta_i)$, where $1 \le i \le n$ as a fixed term to calculate the *maxstep*. Then, $step_i = \Delta_i - \Delta_0$ is always greater than or equal to 0 for every term t_i, meaning that if term t_i is moved, it will be moved to the left. Generally, in Lucene, given a proximity query *q* having *n* terms $\{t_1,\ldots,t_n\}$ and $slop = s$, a document *d* matching with *q* must contain at least one set $\{t_1,\ldots,t_n\}$ and satisfies the condition that there exists a term t_i to be fixed so that any term t_k ($k \ne i$) is moved to the left by $step_k$ steps to arrive at its right position and $max(step_k) \le s$.

3. INDEXING WITH NAMED ENTITIES

Lucene efficiently supports indexing and searching based on keywords. However, semantic indexing and searching, in particular by named entities, require an extension of Lucene. This section presents how NEs are indexed along with keywords to perform searching with NEs. The extended Lucene is called S-Lucene ([3]).

3.1 Named entity recognition

Cultural exhibitions on Ho Chi Minh City development and integration opened in Lam Son park

Fig. 1. Text example

For searching documents based on named entities, a document has to be annotated to identify named entities occurring in them ([7]). At this stage, an annotation system is used to identify all named entities appearing in the document along with their aliases, class and super-classes. Figure 2 shows the result of annotating the text in Figure 1, where *Ho Chi Minh City* is recognized as of the class *City*, with the corresponding ID *city 123*. The results include extension of the text with aliases of *Ho Chi Minh City*, and its super-classes *Location* from the knowledge base and ontology of discourse.

```
<named-entities url=http://vn-kim.hcmut.edu.vn/demo.html>
  <named-entity startOffset="24" endOffset="39">
        <name>Ho Chi Minh City</name>
        <class>City</class>
        <uri>City_123 </uri>
        <aliases>
              <name>Ho Chi Minh City</name>
              <name>Saigon</name>
        </aliases>
        <classes>
              <class>City</class>
              <class>Location</class>
        </classes>
  </named-entity>
</named-entities>
```

Fig. 2. Recognized named entities

3.2 Indexing based on named entities and keywords

Ontological features of a named entity include its name appearing in a text, its most specific class, and its identifier if existing in the knowledge base of discourse. Users may search for documents by one or combination of these features. To search by these features, NE terms are introduced besides normal keyword terms. The structure of each NE term is the triple (*name*/*class*/*identifier*). Unspecified name, class, or identifier of a NE term is denoted by "*".

For indexing based on named entities and keywords, for each entity named *n* possibly with class *c* and identifier *id* in a document, the triples (*n*/*/*), (*/*c*/*), (*n*/*c*/*), (*alias*(*n*)/*/*), (*/*super*(*c*)/*), (*n*/*super*(*c*)/*), (*alias*(*n*)/*c*/*), (*alias*(*n*)/ *super*(*c*)/*), and (*/*/*id*) are added for the document ([3]). Here *alias*(*n*) and *super*(*c*) respectively denote any alias of *n* and any super-class of *c* in the ontology and knowledge base of discourse. Thus, indexed information of documents contain not only the keyword terms analyzed from the original documents but also additional NE triples. A field called *ne+kw* is created to store the keyword terms and NE triples. Storing these terms into the field consists of two steps. First, all keyword terms are added to the field. Second, all NE triples are added. For example, with the text mentioned above (Figure 1), the field *ne+kw* is first added with the keyword terms "*Cultural*", "*exhibitions*", "*on*", "*Ho*", "*Chi*", "*Minh*", "*City*", "*development*", "*and*", "*integration*", "*opened*", "*in*", "*Lam*", "*Son*", "*park*" whose positions are from 0 to 14, respectively. After that, it is added with additional NE triples of *Ho Chi Minh City* including "*ho chi minh city/*/**", "**/city/**", "*ho chi minh city/city/**", "**/*/city_123*", "*saigon/*/**", "*saigon/city/**", "**/location/**", "*ho chi minh city/location/**", "*saigon/location/**".

4. PROXIMITY SEARCH WITH NAMED ENTITIES

To search with named entities, all keywords and entities are unified and treated as generalized terms, where a term is counted either as a keyword or a named entity but not both. Each document is then represented by a single vector over that generalized term space. Document vector representation, filtering, and ranking are performed as in the traditional VSM. This section presents how proximity search are performed when combining named entities and keywords.

For proximity search with NEs, as noted in Section 1, the problem is that token-based positions are inadequate when taking NEs into account. To solve this problem we use another type of positions that we call NE-based positions when executing proximity queries containing NEs. As in Figure 1, we have the terms "*Cultural*", "*exhibitions*", "*on*", "*Ho*", "*Chi*", "*Minh*", "*City*", "*development*", "*and*", "*integration*", "*opened*", "*in*", "*Lam*", "*Son*", "*park*" whose token-based positions are from 0 to 14, respectively. NE-based positions are computed based on token-based positions as follows.

Given a term *A* in a document with a token-based position *x* and *N* NEs appearing on the left of *A*, the NE-based position of *A* is defined by:

$$x - \sum_{i=1}^{N} (NELength(NE_i) - 1) \qquad (2)$$

NELength(E) is the number of tokens that a NE *E* covers in the document. Note that the term *A* may be a keyword term or a NE term but not both. The token-based position of a NE term is the token-based position of the left-most token covered by the NE. For example, the token-based position of the NE *city 123* is 3, because its left most token is "*Ho*" whose token-based position is 3. The terms covered by a NE do not have NE-based positions and are virtually omitted from a document when performing proximity queries containing NEs.

In the text in Figure 1, the NE *city 123* has *NELength* of 4 because it covers the four tokens "*Ho*", "*Chi*", "*Minh*", "*City*" in the document. One can see that, when NE-based positions are applied, the distance between "*on*" and "*development*" is only 1. Meanwhile, using token-based positions, the distance between them is 4. It means that all tokens covered by a NE are treated as a single term and this term has a single position in the document. That is why we call it NE-based position. Note that term positions in a query must be processed in the same way as for documents. For example, in *Q*1, the NE-based positions of the terms "*Cultural*", "*exhibitions*", "*on*", "*Saigon City*" "*development*" are 0, 1, 2, 3, 4, respectively, because "*Saigon City*" is a NE whose *NELength* is 2. As such, in *Q*1 the distance between "*on*" and "*development*" is also 1 and thus the text *D* can now match with *Q*1.

```
Input: a query $q=\{t_1...t_n\}$ and a document $d$.
Output: Frequency of $\{t_1...t_n\}$ occurring in $d$ satisfying slop = 0.
1:  for each $t_i \in q$ do
2:     $pd \leftarrow$ first position of $P_d(t_i)$; $\Delta_i \leftarrow pd - pq_i$
3:  end for
4:  $freq \leftarrow 0$; $maxterm \leftarrow t_k$ where $\Delta_k = \max(\Delta_i)$
5:  do
6:     loop until $\min(\Delta_i) = \max(\Delta_i)$
7:        do
8:           $minterm \leftarrow t_m$ where $\Delta_m = \min(\Delta_i)$
9:           if next position of $P_d(minterm)$ exists then
10:             $pd \leftarrow$ next position of $P_d(minterm)$
11:             $\Delta_k \leftarrow pd - pq_k$ /* update the value $\Delta$ of minterm */
12:          else return freq end if
13:       while $\Delta_k < \Delta_m$ /* $\Delta_m$ is the value $\Delta$ of maxterm */
14:    end loop
15:    $freq \leftarrow freq + 1$
16: while
17: return freq
```

Fig. 3. Algorithm for matching *d* with *q* in the case *slop* = 0

Although NE-based positions have to be applied to process proximity queries containing NEs, token-based positions are still needed for purely keyword-based proximity search. For example, we may want to search for "*ho chi minh city development*" in the pure keyword form, treating all the tokens covered by a NE as separate keywords.

Since the number of NEs in a document is known at the time of indexing, NE-based positions can be computed at the indexing phase. In the query answering phase, if a query is purely keyword-based, then token-based positions are used; otherwise, NE-based positions are used.

For executing a NE-based query, NE-based positions are used in equation (1) introduced in Section 2 to compute the distances between NEs and keywords. Specifically, for a document *d* of length *l* (i.e., with *l* tokens) and a term *t*, the positions of *t* in *d* is denoted by $P_d(t) \subseteq \{1,\ldots,l\}$. Given a query $q=\{t_1 \ldots t_n\}$, the position of term $t_i \in q$ is denoted by pq_i. The algorithms for matching *d* with *q* in the cases *slop* = 0 and *slop* > 0 are respectively presented in Figure 3 and Figure 4. As mentioned at the end of Section 2, a document *d* matching with *q* must contain at least one set $\{t_1,\ldots,t_n\}$ that satisfies the *slop* condition. Here the algorithms do not halt when finding one such set, but they find all and return the number of times (frequency) the set $\{t_1,\ldots,t_n\}$ occurring in *d* and satisfying the *slop* condition. When the returned frequency is greater than 0, it means *d* matches with *q*; otherwise, *d* does not match with *q*. We note that after such distances and frequencies are computed, the ranking function of Lucene taking into account term proximity is employed for ranking retrieved documents.

```
Input: a query $q=\{t_1 \ldots t_n\}$, a document $d$, and $slop$.
Output: Frequency of $\{t_1 \ldots t_n\}$ occurring in $d$ satisfying $slop$ condition.
1:  $freq \leftarrow 0$; $done \leftarrow$ false
2:  for each $t_i \in q$ do
3:    $pd \leftarrow$ first position of $P_d(t_i)$; $\Delta_i \leftarrow pd - pq_i$
4:  end for
5:  do
6:    $maxterm \leftarrow t_k$ where $\Delta_k = \max(\Delta_i)$
7:    $minterm \leftarrow t_m$ where $\Delta_m = \min(\Delta_i)$
8:    $start \leftarrow \Delta_m$; $end \leftarrow \Delta_k$; $pos \leftarrow start$
9:    $next \leftarrow \min(\{\Delta_i \backslash \Delta_m\})$ /* next min value of $\Delta_i$ */
10:   while $pos \leq next$
11:       $start \leftarrow pos$
12:       if next position of $P_d(t_m)$ exists then
13:           $pos \leftarrow$ next position of $P_d(t_m)$
14:       else $done \leftarrow$ true end if
15:   end while
16:   $matchLength \leftarrow$ end – start
17:   if $matchLength \leq slop$ then
18:       $freq \leftarrow sloppyFreq(matchLength)$
19:       /* $sloppyFreq(matchLength) = 1.0f / (matchLength + 1)$ */
20:   end if
21:   if $pos > end$ then $end \leftarrow pos$ end if
22: while !$done$
23: return $freq$
```

Fig. 4. Algorithm for matching *d* with *q* in the case *slop* > 0

5. CONCLUSION AND FUTURE WORK

We have presented our developed general open source for semantic indexing and searching documents annotated with named entities. For proximity search with named entities, we have considered and analyzed dynamic distances between keywords and named entities in queries and documents, and have developed algorithms to compute such a distance on fly.

The novelty and contribution are that it can deal with phrase and proximity queries for which the token-based lengths and positions of the queried named entities in a document may vary. Therefore, it is useful for development of ontology-based search engines when distances between queried terms are significant.

In this work, we are still using the vector-based document ranking function of Lucene. Other ranking models are worth being explored to be integrated with our developed ontology-based proximity searching method. Besides, we have not considered semantic relations between keywords for which synonymous concepts may have different token-based lengths, and thus dynamic computation of distances between concepts for proximity queries is also needed. These are among the topics that we are currently investigating.

6. REFERENCES

[1] Büttcher, S., Clarke, C. L. A. *Efficiency vs. Effectiveness in Terabyte-Scale Information Retrieval*. In Proceedings of the 14th Text REtrieval Conference (TREC-2005), 2005.

[2] Büttcher, S., Clarke, C. L. A., Lushman, B. *Term proximity scoring for ad-hoc retrieval on very large text collections*. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 621–622, 2006.

[3] Cao, T. H., Le, C. K., Ngo, V. M. *Exploring Combinations of Ontological Features and Keywords for Text Retrieval*. In Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence, LNCS, Springer, pp. 603-613, 2008.

[4] Castells, P., Vallet, D., Fernández, M. *An Adaptation of the Vector Space Model for Ontology-Based Information Retrieval*. IEEE Transactions of Knowledge and Data Engineering, pp. 261-272, 2006.

[5] Gonçalves, A., Zhu, J., Song, D., Uren, V., Pacheco, R. *LRD: Latent Relation Discovery for Vector Space Expansion and Information Retrieval*. In Proceedings of the 7th International Conference on Web-Age Information Management, 2006.

[6] Gospodnetic, O., Hatcher, E. foreword by Cutting, D. *Lucene In Action*. A guide to the Java searrch engine, 2004.

[7] Nguyen, V. T. T., Cao, T. H. *VN-KIM IE: Automatic extraction of Vietnamese entities on the Web*, New Generation Computing Journal, Vol. 25, No. 3, pp. 277-292, 2007.

[8] Petkova, D., Croft, W. B. *Proximity-based document representation for named entity retrieval*. In Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, pp. 731-740, 2007.

[9] Rasolofo, Y., Savoy, J. *Term proximity scoring for keyword-based retrieval systems*. In Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-2003). LNCS, Springer, vol. 2633, pp. 207–218, 2003.

[10] Sekine, S. *Named Entity: History and Future*. Proteus Project Report, 2004.

[11] Silverstein, C., Henzinger, M., Marais, H. And Moricz, M.: *Analysis of a very large Web search engine query log*. ACM SIGIR Forum, Vol. 33, No. 1, pp. 6-12, 1999.

[12] Song, R., et al. *Viewing term proximity from a different perspective*. Technical Report MSR-TR-2005-69, Microsoft Research Asia, 2005.

[13] Spink, A. Wolfram, D., Jansen, B.J., Saracevic, T. *Searching the Web: The public and their queries*. Journal of the American Society for Information Science and Technology, Vol. 52, No. 3, pp. 226-234, 2001.

[14] Tao, T., Zhai, C. *An exploration of proximity measures in information retrieval.* In Proceedings of the 30th Annual SIGIR Conference on Research and Development in Information Retrieval, pp. 295-302, 2007.