Learning in a Pairwise Term-Term Proximity Framework for Information Retrieval

Ronan Cummins
Digital Enterprise Research Institute
NUI Galway, Ireland
ronan.cummins@deri.org

Colm O'Riordan
Dept. of Information Technology
NUI Galway, Ireland
colmor@it.nuigalway.ie

ABSTRACT

Traditional ad hoc retrieval models do not take into account the closeness or proximity of terms. Document scores in these models are primarily based on the occurrences or non-occurrences of query-terms considered independently of each other. Intuitively, documents in which query-terms occur closer together should be ranked higher than documents in which the query-terms appear far apart.

This paper outlines several term-term proximity measures and develops an intuitive framework in which they can be used to fully model the proximity of all query-terms for a particular topic. As useful proximity functions may be constructed from many proximity measures, we use a learning approach to combine proximity measures to develop a useful proximity function in the framework. An evaluation of the best proximity functions show that there is a significant improvement over the baseline ad hoc retrieval model and over other more recent methods that employ the use of single proximity measures.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models, Search Process*

General Terms

Experimentation, Performance

Keywords

Information Retrieval, Learning to Rank, Proximity

1. INTRODUCTION

Ad hoc retrieval functions typically use occurrences of the query-terms in the document and collection to determine the usefulness (or weight) of each term in a particular document. These weights are aggregated and normalised for a specific document so that a final score can be assigned to the document for a particular topic. Vector space models [17], probabilistic models [9] and language models [12] all simplify their approaches by adopting the term-independence assumption. This assumption ignores the relationship between individual terms with regard to *proximity*, *position* and *synonimity*. We tackle the problem of incorporating the first of these characteristics, that of the proximity between terms, with regard to document scoring in the traditional ad hoc retrieval task in an intuitive information retrieval (IR) model.

The contributions of this paper are four-fold:

- We outline an extensive list of term-term proximity measures which are heuristically motivated. We also perform an analysis of the independent measures to identify their correlation to relevance in the top ranked documents.
- We develop an intuitive framework for the proximity model into which useful term-term proximity functions can be incorporated.
- We employ the use of a machine learning approach to search through the space of term-term proximity functions.
- We evaluate the best learned proximity functions on test data to show that they achieve a significant increase in performance.

The remainder of the paper is organised as follows: Section 2 summarises previous work into proximity in IR. We outline the strengths and limitations of previous work. Section 3 outlines an extensive list of possible term-term proximity measures. We also perform an analysis of each of the independent measures with respect to relevance. Section 4 details the framework into which a possible proximity function may be placed in order to complete the IR model. Section 5 presents the experimental setup and retrieval functions used. Section 6 presents the experimental results. Our conclusions and future work are detailed in section 7.

2. RELATED RESEARCH

There has been many recent attempts to incorporate proximity into IR models [4, 3, 18, 2, 1, 11]. The relatedness of terms in a semantic sense can be mapped to a proximity measure [3]. Proximity in IR has been researched for some time and semantic proximity (or related-ness between terms) has been incorporated into retrieval systems using query expansion techniques to help overcome the problem of term mismatch.

However, as we wish to explore the actual closeness of terms in a segment of text, we define proximity as the similarity of query-terms within a sample of text. Thus, if the proximity between two terms is 0, they are not in the same grouping. This same grouping may be defined at different levels of granularity, for example, at the document level. This paper deals with proximity within documents only. The underlying hypothesis being that documents in which query-terms appear closer together are more useful to the user (i.e. have a higher degree of relevance).

Boolean (or set-based) retrieval models often have operators to measure the proximity of certain terms within documents. Work has been presented [2] that makes use of a fuzzy set theoretic measure of proximity in a Boolean model of retrieval. This framework is elegant and presents results for a number of different system parameter settings. Previous research [14] has used a window or passage method to determine proximity within a certain threshold. The work shows that proximity can increase performance on small collections.

Proximity information is incorporated into an existing ad hoc retrieval function (used as an underlying framework) to improve the performance of short queries [15]. They create a proximity function at the sentence level, whereby if two query terms appear within the same sentence the document score will be increased. This score is also subject to the distance between the terms in the sentence. They show that modest improvements on larger ad hoc retrieval TREC data can be achieved. More recently some approaches have been successful in employing proximity into a number of keyword based retrieval functions [18]. The work shows that a search of the space of proximity functions is non-trivial. They show significant improvements for short queries.

However, we have identified several limitation of the research conducted to date. Some of the research looks at only the closest pair of query-terms in a document and modifies the document score accordingly, while ignoring other query-terms in the document [18]. A complete proximity function should include the relationships between all query-terms. In much, if not all, of the literature, only short queries are used. It has also been shown that longer queries are more difficult to improve using a proximity measure [1]. The use of shorter queries simplify the number of term-term proximity relationships, as there may be more complex interactions when there are several term matches between a query and document. It is not clear if significant increases in retrieval effectiveness can be achieved using longer queries. This question has been largely unexplored.

Most of the research to-date has only constructed proximity functions from independent proximity measures. It is a simpler task to measure independently the benefit of individual proximity measures to retrieval performance in a specific framework. However, a better proximity function may well consist of a combination of proximity measures and normalisation factors. A proximity function which uses multiple measures of proximity may outperform these simpler approaches. In much of the research, the models adopted augment a traditional keyword-based retrieval model with a proximity function in an unmotivated, ad-hoc and non-intuitive manner.

This paper aims to address all of the aforementioned issues. We aim to create a proximity function which deals with all possible query-terms, without the need for arbitrary thresholds or parameters and which fits neatly into an intuitive framework for retrieval.

3. PROXIMITY MEASURES

This section outlines several individual term-term proximity measures, measures which capture proximity of all terms in the query and also outlines some normalisation measures. For the term-term proximity measures outlined, it is necessary that the measure is symmetrical. For a specific term-term proximity measure ($pm(a,b)$) which measures the proximity between term a and b, we wish to find measures where $pm(a,b) = pm(b,a)$. This definition of proximity is intuitive for proximity as defined in this work (although it may not be a valid assumption if one is to define proximity in terms of synonimity or semantic relatedness).

We wish to search for proximity functions which incorporate relationships between all query-terms and as such we first consider the pairwise similarity between terms. Therefore, when considering two terms a and b occurring in a document D, we consider measures which can be calculated using both terms' position vectors. We define a position vector as the list of integer positions in the document D where a term occurs. The following sample document D will be used to explain how each of the proximity measures can be calculated for a query Q with query-terms a and b:

```
     1  2  3  4  5  6  7  8  9  10 11 12 13 14
D = {a  b  c  d  a  b  d  e  f  g  h  a  i  j}
Q = {a  b}
```

For simplicity, we ignore paragraph and sentence boundaries. Therefore, the positions of each term reflect the actual ordering in which the terms occur in the document. Let pos_a^D denote the vector of integer positions of a in document D and let tf_a^D be the term-frequency of a in document D. Therefore, $pos_a^D = \{1, 5, 12\}$, $pos_b^D = \{2, 6\}$, $tf_a^D = 3$ and $tf_b^D = 2$. In this work, we are aiming to develop measures which capture proximity information from all of the query terms.

The 12 measures in this section relate to proximity either implicitly or explicitly. Many of these measures can be easily calculated using the position vectors of each term. The term-term proximity measures 1 to 6 deal with the distance between the positions of a pair of terms in a document. These measures can be seen as explicitly capturing proximity in some way. Measures 7 and 8 may capture proximity implicitly by combining the term-frequencies of each term in the document. Measures 9 and 10 capture information regarding the terms in the entire query. Measures 11 and 12 are potentially useful normalisation measures.

1. $\mathbf{min\_dist(a, b, D)}$ is defined as the minimum distance between any occurrences of a and b in D. In the example, *min_dist* is 1 (i.e. $2 - 1$) and can be calculated from the position vectors. The main intuition behind this measure is that if any occurrence of term a is close to any occurrence of term b, it indicates a relatedness between the terms. It would be expected that there is an inverse correlation between the *min_dist* measure of all query-terms and relevance.

2. **diff_avg_pos(a, b, D)** is defined as the difference between the average positions of a and b in D. This measure first calculates the average position of each of the terms using the position vectors and then uses the difference as a measure of proximity. In the given example, $diff\_avg\_pos$ is 2 (i.e. $((1+5+12)/3)-((2+6)/2)$). It indicates where each term tends to occur (e.g. one term may tend to occur near the beginning of the document, while the other may tend to occur near the end of document). This measure makes use of position information about all occurrences of both query-terms.

3. **avg_dist(a, b, D)** is defined as the average distance between a and b for all position combinations in D (with time complexity $O(tf_a^D \times tf_b^D)$). In the example, the distances from the first occurrence of a (in position 1) to all occurrences of b are: $\{1$ and $5\}$. This is computed for the next occurrence of a (in position 5) and so on. *avg_dist* for the example is $((1+5)+(3+1)+(10+6))/(2 \cdot 3) = 26/6 = 4.33$. It sums every possible combination of distance between a and b and averages the result. This measure will promote terms that consistently occur close to one another in a localised area. For example, if a single paragraph has multiple occurrences of both query terms, this distance will be reduced.

4. **avg_min_dist(a, b, D)** is defined as the average of the shortest distance between each occurrence of the least frequently occurring term and any occurrence of the other term. In the example, b is the least frequently occurring term so $avg\_min\_dist = ((2-1)+(6-5))/2 = 1$. It can be seen that in two cases a and b occur very close together in D. These terms may constitute a phrase. If this phrase occurs multiple times in a document but far apart in that document, the two previously introduced measures would unfairly penalise the relationship simply because the occurrences of the entire phrase are far apart (i.e. all occurrences are not localised). The first measure (*min_dist*) would not sufficiently reward the number of times the entire phrase occurs. The factor used to average the measure is the frequency of the least frequently occurring term. This is used so that each occurrence of term b (i.e. the least frequently occurring) is matched only once (this also ensures symmetry for this measure). In the example, the occurrence of a at position 12 maybe completely unrelated to b (superfluous to the relationship between a and b) and is ignored by the measure.

5. **match_dist(a, b, D)** is defined as the smallest distance achievable when each occurrence of a term is uniquely matched to another occurrence of a term. For the previous distance function, the occurrence of a term may be used twice in the computation of the relationship (if it is an occurrence of the most frequently occurring term). However, if indeed each term is treated as having a pair, each occurrence of the least frequently occurring term should be paired with one distinct occurrence of the second term. The problem can be posed as follows; what is the best way to match the occurrences of pairs of terms so as to minimise the total distance between the pairs? This problem can be solved in exponential time using a dynamic programming algorithm. Fortunately, the frequencies of the query-terms in the document are relatively small so that this calculation is feasible on TREC documents. In the above example, the answer is the same as the previous measure as $match\_dist = ((2-1)+(6-5))/2 = 1$. The frequency of the least frequently occurring term is used in averaging the score.

6. **max_dist(a, b, D)** is the maximum distance between any two occurrences of a and b. In the example $max\_dist = (12-6) = 6$. This may be a useful measure of distance or may be a useful normalisation factor for some of the other proximity measures.

7. **sum(tf(a), tf(b))** is defined as the sum of the term-frequencies of a and b in D. This measure gives an implicit indication of the proximity of both terms. If this measure is high, the probability of closer occurrences of terms is automatically higher. In the example used, *sum* is 5 (i.e. $3+2$) for document D.

8. **prod(tf(a), tf(b))** is defined as the product of the term-frequencies of a and b. This measure also gives an implicit indication the proximity of both terms. If this measure is high, the probability of closer occurrences of terms is again automatically higher. In the example, *prod* is 6. Furthermore, these two measures (*sum* and *prod*) can be combined to give an indication of the equality of pairwise occurrences of terms in the entire document. For example, if

$$\sqrt{prod(tf(a), tf(b))}/sum(tf(a), tf(b)) = 0.5$$

then both terms occur an equal number of times possibly indicating a closeness between the terms. If it is considerably less than 0.5, one term is far more frequent.

9. **fullcover(Q, D)** is defined as the length of the document that covers all occurrences query-terms. This measure has previously been defined as span [18]. The value of $fullcover$ is 12 in the example given. This measure is a query specific measure as it incorporates all terms in the query.

10. **mincover(Q, D)** is defined as the length of the document that covers all query-terms at least once. The value of *mincover* is 2 in the example given. This will be the same as min_dist+1 for a two-term query. The intuition for this, and the preceding measure, is that if all the query terms reside in a smaller segment of text, it indicates that the document contains a segment that has a high probability of relevance and thus, the document has a higher probability of relevance.

11. **dl(D)** is defined as the length of the document and is a factor useful for normalisation in IR. It may be very important in the scaling or normalisation of some of the proximity measures introduced here. For example, shorter documents are more likely to have closer term proximities. In the example outlined earlier, $dl(D)$ is 14.

12. **qt(Q, D)** is defined as the number of unique terms that match both document and query. In [18], it is shown that *mincover* is not inversely correlated with relevant

documents (indeed the opposite). This is because of the fact that *mincover* will be also be smaller for documents with fewer query terms (i.e. a document matching only one query term has a *mincover* of 1 by definition). Thus, it is suggested that *mincover* should be normalised by the number of distinct matching terms between a document and query (i.e. $qt(Q, D)$). The results of this intuition are more promising (as a weak inverse correlation is uncovered). However, this normalisation factor was incorporated in an unguided manner and indeed better normalisation may lead to a proximity function with superior performance. In the example outlined earlier, qt is 2.

It can be noted that some of these proximity measures are similar to those in [18]. However, all the measures in that work are query specific, meaning that they only calculate the proximity for the closest pair of terms in a query or use a type of *span* measure to capture the minimum portion of text which covers occurrences of the query terms. In fact, of the 12 measures outlined here, only *mincover* and $fullcover$ are the same measures used previously [18].

3.1 Correlations of Measures

In this section, we perform an analysis of each of the independent proximity measures to predict which measures may be most useful when incorporated into a proximity measure.

3.1.1 Collections Used

For our analysis and subsequent experiments, we use the FBIS, FT, FR collections from TREC disks 4 and 5 as test collections and topics. Some characteristics of the collections are indicated in Table 1. For each set of topics we create a short query set, consisting of the title field of the topics and a medium length query set, consisting of the title and description fields. We also use the OHSUMED collection and its topics. Table 1 shows some of the characteristics of the collections used in this research. We stemmed the collections using Porter's algorithm [13] and removed standard stop-words.

3.1.2 Analysis

Our aim is to incorporate a proximity function into an existing term-weighting scheme. Therefore, we can view the problem as performing re-ranking on the top N documents from an initial ranked list using a proximity function on the query-terms. This also ensures that these N documents have an ample supply of query-terms. Consequently, we perform an analysis of the top 1000 documents from a retrieval run and examine the correlation between the measures outlined in the previous section and the relevant and non-relevant documents in this set of documents.

Table 2 shows the average values of the individual measures per query across all of the test collections (for relevant and non-relevant documents respectively). For example, for short queries, the average *min_dist* between query terms in the relevant documents is 39.2, while it is 172.4 in the non-relevant documents. The percentage of queries for which this inverse relationship holds is also indicated. For example, in 83.5% of short queries, the *min_dist* is a smaller value in the relevant documents than in the non-relevant documents across all of the collections. However, on the collections used we have determined that the consistency of the correlation and the average proximity measure in the relevant and non-relevant document does not always agree. For example, consider the *avg_dist* proximity measure for the medium length queries. We can see that the measure seems *directly* correlated with relevance as the *avg_dist* averaged over all queries is higher in the relevant documents (i.e. 250.0 compared to 235.1). However, we can see that in fact it is inversely correlated with relevance in 68% of the queries. Thus, there is a difference between the strength of the difference of each proximity measure in the relevant and non-relevant documents and the degree to which they are correlated with relevance. The asterisk (*) indicates that there are certain collections in which the consistency of correlation does not agree with the average difference in magnitude of the measure in the relevant and non-relevant document sets.

Of the six pairwise proximity measures, we can see that *min_dist*, *avg_min_dist* and *match* seem to have the strongest inverse correlation with relevance. Conversely, we can see that the qt measure is directly correlated with relevance. In only 18.5% of queries, the number of distinct query terms (qt) is smaller in the relevant documents than in the number of non-relevant documents for short queries.

However, it should be noted that some of the proximity measures may be correlated with each other implicitly and a combination of such measures may not increase performance. Some measures may be correlated to relevance because they are highly correlated with the original ranking (which is correlated to relevance). Since ranking functions tend to promote shorter documents with more occurrences of query terms, many measures of proximity will be implicitly correlated to relevance. As a result, incorporating them in a ranking function may bring about no increase in performance (MAP). Due to the complexity of the problem, the potential number of measures and multitude of combinations that may exist, it is difficult to perform an in-depth analysis of the potential benefit of each proximity measure in isolation.

4. PROXIMITY RETRIEVAL MODEL

Our goal is to create a model of retrieval which incorporates proximity into an intuitive model. Therefore, we extend a vector (or bag of words) model into one which exploits proximity or closeness between pairs of terms. As most traditional keyword models of retrieval can be viewed as vector type approaches, we extend this model so that documents and queries are viewed as matrices.

Consider the following 3×3 matrix representing a document D matching 3 query-terms (Q). The document may have many terms but we only consider terms that match the query in the scoring process. Let the diagonals be the relatedness between a term and itself and the non-diagonal entries be the proximity (or closeness) between a pair of distinct terms. The query representation is kept simple due to the relative size of the query compared to the document. Thus, the following document representation is equivalent to a vector based model as term-independence is assumed (i.e. the proximity between two distinct terms is 0 and $w(t_1)$ is some tf-idf type score for each term).

$$D = \begin{pmatrix} w(t_1) & 0 & 0 \\ 0 & w(t_2) & 0 \\ 0 & 0 & w(t_3) \end{pmatrix}$$

Table 1: Test Collections

	Documents		Topics			
	#	Avg. Length	Range	#	title	title+desc
					Avg. Length	
LA	131,896	225	351-450	95	2.4	9.5
FBIS	130,471	241	301-450	116	2.4	10.4
FR	55,630	344	251-450	95	2.6	10.6
OHSUMED	293,856	91	1-63	63	-	7
FT-TRAIN	69,507	191	301-400	55	2.2	9.5

Table 2: Inverse correlations over all collections

	title only			title+desc		
	REL	NON-REL	%Qrys	REL	NON-REL	%Qrys
min_dist	39.2	172.4	*83.5%*	55.7	87.8	*80.3%*
$diff\_avg\_pos$	176.7	366.5	*75.8%*	165.0	175.4	*74.6%*
avg_dist	270.7	455.7	*69.8%*	250.0	235.1	*68.0%**
avg_min_dist	65.5	190.5	*82.1%*	77.8	101.7	*77.3%**
$match\_dist$	86.0	205.8	*78.3%*	96.2	113.6	*76.2%*
max	564.6	647.7	*58.7%**	471.1	376.3	*60.5%**
sum	13.0	7.5	*32.2%*	10.2	6.4	*33.9%*
$prod$	94.6	45.3	*38.0%*	80.3	23.7	*39.7%*
$fullcover$	720.5	875.4	*61.8%**	684.2	539.7	*59.4%**
$mincover$	49.5	181.9	*81.7%*	160.5	180.5	*72.3%*
dl	971.8	1410.5	*73.2%*	814.4	730.6	*69.4%**
qt	2.1	2.0	*18.5%*	3.3	2.7	*21.0%*

If we assume that the weights of the diagonal entries are weights in a traditional retrieval model (e.g. tf-idf or $BM25$), the model can be simply extended by defining the non-diagonal elements as a useful proximity function ($p()$). Thus, the proximity function is some combination of the set of 12 proximity measures introduced earlier that define term-term closeness or proximity as defined in this work. Again, we can assume a simple weighting on the query-terms due to the relative size of the query in comparison to the document (even for a query of 10 or so terms). Thus, the representation of a document in our model is as follows:

$$D = \begin{pmatrix} w(t_1) & p(t_2,t_1) & p(t_3,t_1) \\ p(t_1,t_2) & w(t_2) & p(t_3,t_2) \\ p(t_1,t_3) & p(t_1,t_2) & w(t_3) \end{pmatrix}$$

where $w()$ is a standard term-weighting scheme and gives us the weight of term in a document and $p()$ is a proximity function. Usually, a document can be represented by a vector and the final score is some aggregation of those weights. Therefore, the entire score of the document can now be defined as the sum of all combinations of term-term relationships:

$$S(D,Q) = \sum_{i \in Q \cap D} \sum_{j \in Q \cap D} \left| \begin{array}{ll} w(i) & \forall i = j \\ p(i,j) & \forall i \neq j \end{array} \right| .$$

which aggregates all of the elements of the document matrix (as we use simple weighting on the query terms). As all measures of proximity introduced earlier are symmetrical, $p(i,j)$ will equal $p(j,i)$ and each term-term proximity score in the framework is counted twice. We allow this as the framework can then also be used to model semantic similarity (i.e. where $p(i,j) \neq p(j,i)$) using a different set of semantically related term-term measures.

While this framework, like many others, has no theoretical basis, it is an intuitive extension of a vector based approach. Indeed, there is no theoretical basis for mapping documents into a Euclidean space at all. If we assume that proximity has no effect or that term occurrence is independent we can set $p(i,j) = 0 \; \forall i \neq j$ and recover a standard vector based retrieval model. Now, assuming that proximity is an important measure in a retrieval model, we need to instantiate the proximity function $p()$ in our framework. Previous research has indicated that this is a non-trivial problem and developed a constraint which helped to guide the search for a useful function [18]. However, this constraint is only useful when using the individual measures for proximity in isolation. We wish to find a useful proximity function, by combining some or all of the 12 proximity measures identified earlier. In order to achieve this, we employ the use of genetic programming (GP) [10] and search the space of proximity functions in a guided manner, as GP has previously been used in IR to search for useful ranking functions [19, 5]. The next section introduces this GP process and experiments which aim to instantiate useful proximity functions in our framework.

5. EXPERIMENTAL SETUP

In this section, we outline the benchmarks and the initial term-weighting schemes into which we can incorporate a proximity function. We also provide a brief description of the GP process before outlining the settings for the GP process for which we use to create proximity functions.

5.1 Benchmarks

As a benchmark against which to test our approach we use the traditional BM25 scheme.

$$BM25(Q,D) = \sum_{t \in Q \cap D} \left(\frac{tf_t^D \cdot log(\frac{N - df_t + 0.5}{df_t + 0.5}) \cdot tf_t^Q)}{tf_t^D + k_1 \cdot ((1-b) + b \cdot \frac{dl}{dl_{avg}})} \right) \quad (1)$$

where tf_t^D is the frequency of term t in document D, dl is the document length, df_t is the document frequency of term t and dl_{avg} is the average document length in the entire collection. k_1 is the term-frequency influence parameter which is set to 1.2 by default. The query term weighting used here (tf_t^Q) is slightly different to the original weighting method proposed [16] but has been used successfully in many studies [7]. b is the document normalisation influence tuning parameter and has a default value of 0.75.

We use the $BM25$ matching function and the proximity function previously developed by Tao [18] as another benchmark. The proximity function $t()$ is as follows:

$$t() = log(\alpha + exp(-min\_dist(Q,D))) \quad (2)$$

where $min\_dist(Q,D)$ is the minimum distance between any occurrence of any two query-terms in the document[1]. It is stated that α set to 0.3 provides stable performance and thus we use this setting for our experiments [18].

The term-weighting scheme upon which we base the proximity functions in these experiments is based on a previously learned formula [6] and is as follows:

$$ES(D,Q) = \sum_{t \in Q \cap D} \left(\frac{tf_t^D \cdot tf_t^Q}{tf_t^D + 0.45 \cdot \sqrt{\frac{dl}{dl_{avg}}}} \cdot \sqrt{\frac{cf_t^3 \cdot N}{df_t^4}} \right) \quad (3)$$

where cf_t is the frequency of t in the entire collection. The choice of underlying ranking function is not crucial for the development of a proximity function. Although the performance of a specific instantiation of a proximity functions is dependent on the underlying ranking function, it is our intent to show that the process can find useful proximity functions and that these are indeed general across different test collections (not that the proximity functions are general for a multitude of ranking functions). The process outlined here can be repeated for different underlying ranking functions, like $BM25$, but this experimentation lies outside the scope of this paper.

The benchmark proximity functions can be instantiated by simply adding $t()$ to the score of the original ranking function. We will label these functions $BM25 + t()$ and $ES + t()$. In this section, we have outlined two ranking functions ($BM25$ and ES) and a baseline proximity function ($t()$) which can be used with each.

5.2 GP process

GP is a heuristic stochastic search algorithm, inspired by natural selection, and is useful for navigating large complex search spaces. Initially, a population of solutions is created randomly (although other approaches seed the initial population with known solutions). The solutions are encoded as trees. Each tree (genotype) contains nodes which are either functions (operators) or terminals (operands). Each solution is rated based on how it performs in its environment. This is achieved using a fitness function. Having assigned the fitness values, selection can occur. Individuals are selected for reproduction based on their fitness value. There are various different methods used to select individuals but all are based in some way on the fitness of the individual.

As a result, fitter solutions will be selected more often. Once selection has occurred, recombination can start. Recombination creates new solutions for the next generation by use of crossover and mutation. Recombination occurs until the population is replaced by new solutions. The process usually ends when a predefined number of generations is complete. Some important parameters in the GP process are the population size, the number of generations for which to run the GP, the function set and the terminal set. Genetic programming has been shown to be useful for finding solutions in complex search spaces such as combining sources of evidence in IR. One of the big advantages conferred by the GP approach over other machine learning solutions is that the resulting learned solutions are available in readable format rendering them amenable to further analysis.

5.3 Training

We used a subset of the Financial Times collection as a training collection for the GP (FT-TRAIN in Table 1). We took 69,500 documents and 55 queries (in the range of topics 301-400). We used 25 title only queries and 30 title and description queries. The slightly longer queries will help the GP to learn general term-term proximity functions as there are more pairwise interactions in these longer queries ($(n^2-n)/2$ possible relationships for each query). The fitness function used in the experiments is MAP as it is a stable measure of IR performance. In fact, we learn a function which is twice the value produced by a proximity function for a pair of terms in the the framework outlined earlier (i.e. $2 \times p()$). Therefore, we can score the proximity part of each document in $(n^2 - n)/2$ steps.

We ranked the documents using the ES scheme and calculated all the proximity measures for the top 3000 documents. We used all of the proximity measures previously introduced as input terminals to the GP (which was developed using existing software [8]) along with three constants for used for scaling (i.e. $\{1, 10, 0.5\}$) . We also used the following functions as inputs to the GP : $+$, $-$, $\times$, $/$, $\sqrt{\ }$, $sq()$, $log()$. We then ran the GP six times with an initial random population of 2000 for 30 generations using an elitist strategy (i.e. where the best individual is automatically copied to the next generation). We then chose the best function from each of the six runs as a proximity function (labelled $p1()$ to $p6()$ in our experiments).

6. EXPERIMENTAL RESULTS

In this section, we outline the results of our experiments. We conducted a Wilcoxon signed-rank test on the proximity functions when compared to the underlying ranking function. For the following results, † and ‡ indicate that the results are significant at the 0.05 and 0.01 level respectively. The best result on a collection is in bold text. In all cases we are testing the significance with respect to the underlying ranking function. For example, $BM25 + t()$ is compared to the $BM25$ function, while $ES + t()$ is compared to the ES function.

6.1 Training

Table 3 shows the results of the best three runs of the

[1]This is different that the $min\_dist(a,b,D)$ defined in this work which specifies which pair of query-terms to use.

GP on the training data. The only two formula which are significant on the training data are $p5()$ and $p6()$. On the training data, the $t()$ function does not lead to a significant increase although a small increase is seen when compared to both underlying ranking functions.

Table 3: MAP on Training Data

	TRAINING
$BM25$	0.2999
$BM25 + t()$	0.3007
ES	0.3273
$ES + t()$	0.3280
$ES + p2()$	0.3435
$ES + p5()$	**0.3466** †
$ES + p6()$	0.3451 †

The three best functions produced by the GP are as follows:

$$2 \times p2() = log(\frac{10}{min\_dist}) + 5 \cdot \frac{prod}{avg\_dist} + \sqrt{\frac{10}{min\_dist}}$$

$$\begin{aligned} 2 \times p5() &= ((((((\frac{log(fullcover)}{min\_dist^2} + \frac{10}{sum}) \\ &\cdot min\_dist) - 0.5)/(min\_dist^2)) + (((log(0.5)) \\ &+(\frac{prod}{avg\_dist}))/0.5))/min\_dist) - 0.5 \end{aligned}$$

$$\begin{aligned} 2 \times p6() &= ((3 \cdot log(\frac{10}{min\_dist}) + log(prod + \frac{10}{min\_dist}) \\ &+\frac{10}{min\_dist} + \frac{prod}{sum \cdot qt})/qt) \\ &+\frac{prod}{avg\_dist \cdot min\_dist} \end{aligned}$$

From inspection, it can be seen that *min_dist* appears in all of the functions found by the GP. *avg_dist* also appears in all of the best functions suggesting that there is also benefit in incorporating this measure into proximity functions. The best two functions ($p_5()$ and $p_6()$) also make use of the *sum* and *prod* measures. However, as is the case with any machine learning approach, it is possible to have over trained, or over-fitted, for the data. In order to properly test the usefulness of the learned functions, we need to test over unseen data. The next section compares the chosen functions on unseen test data.

6.2 Tests on Unseen Data

The results on the test data show that measures of proximity and the information contained therein are useful in increasing performance of IR systems. We attain a notable improvement over existing benchmarks by learning a suitable means to combine intuitive proximity measures. This is reflected in the increased MAP scores obtained (Table 4). We can see that the performance of $p6()$ in terms of MAP is significantly better than that of its underlying function on most test collections. $p5()$ also shows significant improvement on many of the collections. We can see that there is little or no improvement for $p2()$ which mirrors the results of the significance tests on the training data.

The $t()$ function also performs poorly on many of the collections especially for longer queries. This shows that it is important to incorporate the proximity relationships of all the query-terms into one function for these longer queries. The $t()$ function only incorporates the relationship between the closest query-term pair in the document to augment the retrieval score. For longer queries, this becomes problematic as the closest pair of query-terms may be less important terms with respect to the entire query.

We also compare the benchmarks with the best evolved proximity function ($p6()$) in terms of the precision achieved at 10 documents (Table 5). Again, an improvement is achieved over the standard ranking functions.

7. CONCLUSION

We have outlined an extensive list of measures that may be used to capture the notion of proximity in a document. We have indicated the potential correlation between each of the individual measures and relevance. From this analysis, we have indicated that *min_dist* is highly correlated with relevance which corresponds with previous work [18]. We outline an IR framework which incorporates the term-term similarities of all possible query-term pairs. This framework leads to a ranking function into which a proximity function can be placed. We adopt a population based learning technique (GP) which learns useful proximity functions in the framework developed. Using this learning approach, we develop six functions, three of which are presented here.

Finally, we evaluate three of our learned proximity functions on test data and show that they outperform previous benchmarks, particularly for longer queries. It is interesting to note that the proximity functions presented in this paper achieve an improvement for both short and longer queries. Previous approaches have failed to demonstrate an improvement using proximity measures for longer queries. We believe this is due to limitations of the chosen proximity measures used in previous research; previously used measures did not take into account evidence provided by all the proximity relationships among query terms. We have also included measures which consider more than just a limited subset of the query terms proximity scores. Such measures (e.g *avg_dist*) take into account proximity scores between all occurrences of query terms.

The research described in this paper demonstrates that it is possible to use combinations of proximity measures to improve the performance of IR systems for both short and long queries. Future work will involve further exploration of the learned functions and other types of proximity that could be used in the framework.

8. ACKNOWLEDGMENTS

The work presented in this paper has been funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

9. REFERENCES

[1] Jing Bai, Yi Chang, Hang Cui, Zhaohui Zheng, Gordon Sun, and Xin Li. Investigation of partial query proximity in web search. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1183–1184, New York, NY, USA, 2008. ACM.

Table 4: MAP on Test Data

	title only queries			title and description queries			
	LA	FBIS	FR	LA	FBIS	FR	OHSUMED
$BM25$	0.1917	0.2298	0.2626	0.2256	0.2475	0.2694	0.3102
$BM25 + t()$	0.1956	0.2273	0.2571	0.2273 †	0.2490 ‡	0.2563	0.3113
ES	0.2099	0.2666	0.2772	0.2293	0.2920	0.2792	0.3314
$ES + t()$	0.2169	0.2678 †	0.2615	0.2306 ‡	0.2686	0.2290	0.3327 ‡
$ES + p2()$	0.2152	0.2746	0.2820	0.2394 †	0.3001	0.2848	0.3367
$ES + p5()$	0.2208	**0.2797** ‡	0.2788	0.2363	**0.3009** †	0.2826	0.3290
$ES + p6()$	**0.2233**	0.2770 ‡	**0.2834** ‡	**0.2420** ‡	0.2979	**0.2901** ‡	**0.3371** †

Table 5: P@10 on Test Data

	title only queries			title and description queries			
	LA	FBIS	FR	LA	FBIS	FR	OHSUMED
$BM25$	0.2610	0.2974	0.1705	0.2894	0.3206	0.1831	0.5015
ES	0.2642	0.3146	**0.1894**	0.3042	0.3094	0.1926	0.5015
$ES + t()$	0.2715	0.3155	0.1831	**0.3052**	0.2974	0.18	0.5
$ES + p6()$	**0.2778**	**0.3318**	0.1873	0.3021	**0.3215**	**0.2021**	**0.5079**

[2] Michel Beigbeder and Annabelle Mercier. An information retrieval model using the fuzzy proximity degree of term occurences. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1018–1022, New York, NY, USA, 2005. ACM.

[3] M. P. S. Bhatia and Akshi Kumar Khalid. Contextual proximity based term-weighting for improved web information retrieval. In *KSEM*, pages 267–278, 2007.

[4] Stefan Büttcher, Charles L. A. Clarke, and Brad Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 621–622, New York, NY, USA, 2006. ACM.

[5] Ronan Cummins and Colm O'Riordan. Evolving local and global weighting schemes in information retrieval. *Information Retrieval*, 9(3):311–330, 2006.

[6] Ronan Cummins and Colm O'Riordan. An axiomatic study of learned term-weighting schemes. In *Learning to Rank for Information Retrieval - Workshop, SIGIR '07*, Amsterdam, Netherlands, 8-10 August 2007.

[7] Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 480–487. ACM Press, 2005.

[8] Adam Fraser. Genetic programming in c++. Technical Report Technical Report 040, University of Salford, Cybernetics Research Institute, 1994.

[9] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[10] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[11] Seung-Hoon Na, Jungi Kim, In-Su Kang, and Jong-Hyeok Lee. Exploiting proximity feature in bigram language model for information retrieval. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 821–822, New York, NY, USA, 2008. ACM.

[12] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.

[13] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[14] Bruno Pôssas, Nivio Ziviani, and Jr. Wagner Meira. Enhancing the set-based model using proximity information. In *SPIRE 2002: Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, pages 104–116, London, UK, 2002. Springer-Verlag.

[15] Yves Rasolofo and Jacques Savoy. Term proximity scoring for keyword-based retrieval systems. In *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, 2003. Proceedings*, page 79, 2003.

[16] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC-3. In *In D. K. Harman, editor, The Third Text REtrieval Conference (TREC-3) NIST*, 1995.

[17] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[18] Tao Tao and ChengXiang Zhai. An exploration of proximity measures in information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 295–302, New York, NY, USA, 2007. ACM.

[19] Andrew Trotman. Learning to rank. *Inf. Retr.*, 8(3):359–381, 2005.