# 1 Introduction

Term dependency and term proximity are important signals in ad-hoc information retrieval (IR) tasks. Intuitively, we can say that a document in which query terms appear closer together (e.g. in a sentence, passage, or paragraph etc.) is more relevant than the document in which they are far apart. In this report we look at several traditional and neural models which aim to better capture and utilize these signals without sacrificing other important signals such as exact and soft match. Specifically, we consider a traditional sequential dependency modeling (SDM) approach as presented in [1], proximity approaches presented in [2], and the more recent neural IR approach presented in [3], [4], and [5].

The rest of the report is organized as follows: Section 2 compares term dependency modeling of ConvKNRM [5], PACRR [4], and MatchPyramid [3]. Section 3 compares the three neural models with traditional MRF model. Finally, section 4 presents an axiomatic analysis two new models under consideration: PACRR and MRF/SDM.

# 2 Term dependency and term proximity in neural IR models

## 2.1 MatchPyramid

MatchPyramid (MP) [3] takes inspiration from computer vision approaches and poses text matching as an image recognition problem [6] by treating the similarity matrix between embeddings of document terms and query terms as an image. Two types of convolution filters ($1 \times n$ and $n \times n$) are considered in this model and applied to this matrix to capture relevance matching patterns.

It is clear that sequential dependencies and term proximity are encoded in the learned filter patterns. However, MP only uses one type of kernel at a time i.e. either a $1 \times n$ kernel or a $n \times n$ kernel, where $n$ is 1,3, or 5. This greatly reduces the representational power of the model compared to PACRR and ConvKNRM (both discussed later). Using one value of $n$ means that the model is only using n-terms features for that specific value of $n$. For example, if model is using a $3 \times 3$ filter only, it will miss out on the exact or soft-matches between 1,2,4, and 5 terms. Additionally, using features of only 3 (corresponding to variant which uses only $3 \times 3$ kernel) or 5 terms (corresponding to variant which uses only $5 \times 5$ kernel) imposes a very strong assumption that matches occur in groups of 3 or 5 only, which has very little grounds in reality where matches can happen anywhere in document and between any number of terms.

Another short-coming of the model is that it treats unigram exact and soft match signals as optional. These signals are only considered in the $1 \times 1$ kernel variant. This is in stark contrast compared to all the other models considered in this report which take care to include unigram matching signals. By doing this, the model fails to learn the unique importance of exact matching signals in ad-hoc IR. Hence, we expect PACRR,

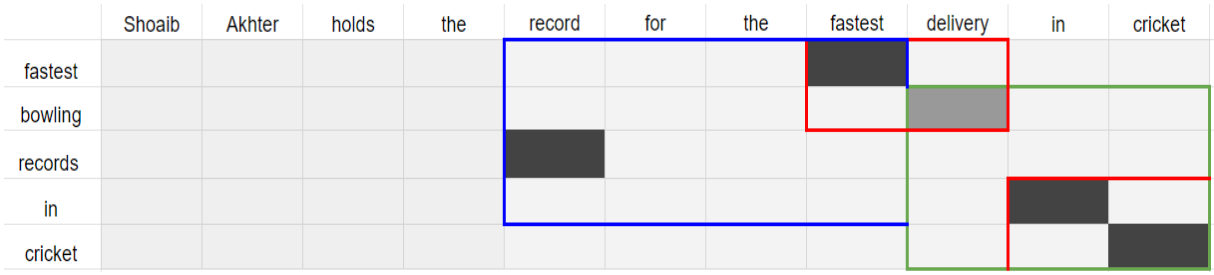


Figure 1: A $4 \times 4$ Kernels (blue) captures a proximity match, Two $2 \times 2$ Kernels (red) capture sequential dependence bigram (soft)-match, a $3 \times 3$ kernel (green) captures proximity match which overlaps with $2 \times 2$ kernel's stronger match

and ConvKNRM to perform better as they consider multi-scale n-term features together with unigram exact and soft-matches and thus have access to more information.

## 2.2 PACRR

PACRR [4] also uses a similarity matrix between $q$ and $d$ as a basis like MP. It however builds a much richer representation with an aim to capture term dependence and term proximity without sacrificing unigram matches.

PACRR captures term dependency and term proximity via a set of $n \times n$ filters, without building an embedding-like representation as in ConvKNRM (discussed later). Several kernels of size $2 \times 2$, $3 \times 3$, $4 \times 4$ are applied to the similarity matrix, where it is assumed that each kernel specializes in picking out one specific type of match. Some examples of possible matches for a $3 \times 3$ kernel are shown in Fig. 2. Each kernel of size $n \times n$ is responsible for capturing n-gram and n-term exact and soft-match patterns. Application of these kernels on the similarity matrix results in an activation map which can be considered as a response of the kernel w.r.t the existence of a particular matching pattern. For example, a kernel like the one shown in part (ii) of Fig. 2 would give rise to an activation map which encodes "3-term out of order" matches.

Observing the possible kernels that the model can learn from data we can see that it captures term dependency in a relatively transparent way through specialized kernels. The term proximity capability is however of very limited scope (considering only 4-term window). Consider the query âĂIJ**fastest bowling records in cricket**", instead of exactly matching the whole phrase (which is very unlikely) we are happy if the query terms appear near each other. So the document âĂIJShoaib Akhter holds the **record** for the **fastest** delivery in **cricket**âĂİ will match the query, even though there is no exact phrase match. A segment of similarity matrix and how kernels representing term dependency and term proximity contribute to score is shown in Fig. 1.

## 2.3 ConvKNRM

ConvKNRM [5] builds on a previous model [7] and adds the ability of n-gram soft-matching to the model. It constructs embeddings for n-grams by fusing together the

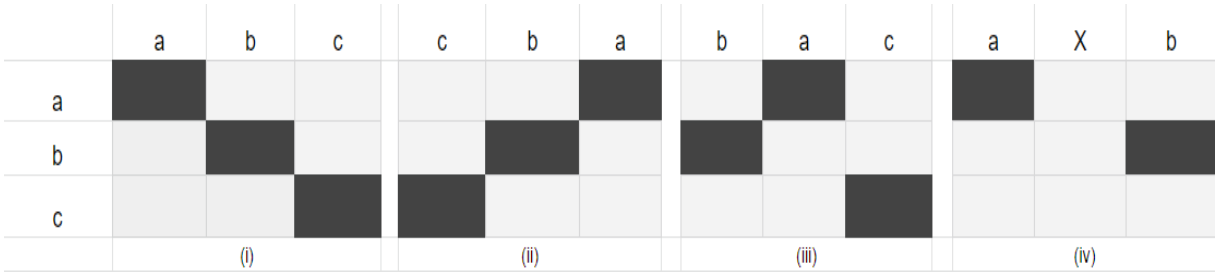


Figure 2: A visualisation of some possible $3 \times 3$ matching patterns along with their interpretations. Case **(i)** corresponds to exact or soft 3 term sequence match, Case **(ii)** represents exact or soft 3 term reverse match, Case **(iii)** shows unordered 3 term exact or soft match, and Case **(iv)** can be considered a 2 term proximity match in presence of an irrelevant term

embeddings for individual terms. It takes a sequence of $h$ words and learns a combination matrix (filters) which fuses together the embeddings of individual terms into a unified $F$-dimensional $h$-gram embedding ($F$ is the number of filters). However, it only considers embeddings for the contagious sequence of $h$ terms. Therefore, if words never occur together (or not often enough) in an $h$-window, model will not be able to learn a good composed embedding for those words or phrases.

Intuitively, one useful constraint (that is not explicitly enforced or guaranteed by model) is that the cosine similarity of embedding for bigram [a,b] (formed via combination of [a] and [b]) must be closer to a trigramâĂŹs embedding which contains [a,b] or its semantic equivalents (in any order). This will allow for modeling limited term proximity if we assume that the model learns to do this during training. Hence, we can also match 2 different bigrams for similarity: âĂIJ**good movie**âĂİ and âĂIJ**amazing film**âĂİ.

One useful consequence of having a unified representation for $h$-grams is that the model can benefit from n-gram cross-matching. As related concepts can be described by words with different lengths e.g âĂIJconvolutional neural networkâĂİ and âĂIJdeep learningâĂİ. Other than this, cross-matching can also be used for matching abbreviations and acronyms e.g. âĂIJ**MPI**âĂİ and âĂIJ**Max Planck Institute**âĂİ or âĂIJ**UN**âĂİ and âĂIJ**United Nations**âĂİ. This kind of n-gram cross match will be difficult to express in PACRR's representation.

ConvKNRM considers upto trigram matches ($h_{max} = 3$) whereas PACRR considers max $4$-gram matches ($l_g = 4$, corresponding to $4 \times 4$ filters). That is, term proximity window in ConvKNRM is even more limited than PACRR. One possible reason for using fewer terms in ConvKNRM might be that the embeddings gets increasingly fragmented and begin to lose any semantic meaning for higher values of $h$.

Hence, we see that compared to the approach followed in [2], where proximity information was used as an addon on top of relevance scores produced by models, the considered models include this capacity within themselves and can learn to make use of proximity and dependency signals.

# 3 Comparison of neural models with MRF/SDM

In [1] authors use the formalism of markov random field (MRF) to model the dependencies between query terms. Both document and query terms are represented as nodes of a graph and dependency relationships are modeled via an edge between the nodes. Authors presented three components of the model: full independence, sequential dependence, and full dependence which correspond to bag-of-words uni-gram matches, bi-gram and bi-term matches, and general n-gram matches respectively.

There are equivalent components in both PACRR, ConvKNRM, and to some extent MP which mirror the functionality of MRF. For instance, MRF models unigram matches through the 2-cliques in full-independence component. This is equivalent to PACRR's original similarity matrix in which each terms is considered individually. In ConvKNRM the same function is performed by $h = 1$ and exact matching bin with $\mu = 1.0$. MP considers this in $1 \times 1$ conv kernel variant.

The sequential dependence between 2 adjacent query terms is represented in 3-cliques (2 query terms, and document). It is used to model bigram and biterm matches and takes into account sequential dependence and ordered proximity of 2 query term through its âĂIJordered windowâĂİ and âĂIJunordered windowâĂİ, respectively. Thus, the query âĂIJ**fifa worldcup**âĂİ will match âĂIJ**fifa** organizes the football **worldcup** every 4 yearsâĂİ and it will also match âĂIJthe football **worldcup** is organized by **fifa** every 4 yearsâĂİ in which terms appear out of order. In PACRR this is done via $2 \times 2$ kernels, and in ConvKNRM dependence is modeled by $h = 2$ and exact matching bin with $\mu = 1.0$. Under the assumption that composed embeddings are not senstive to term orders, cross-matching can be used to model proximity. MP does not have $2 \times 2$ kernels, so their is no direct way to replicate this in MP. This sequential dependence can be extended to more than 2 terms. So, in PACRR this is accomplished via $3 \times 3$ and $4 \times 4$ kernels and by $5 \times 5$ kernels in MP. In ConvKNRM $h = 3$ gives 3-gram embeddings which can be used to do the same.

The full dependence model captures ordered and unordered matches for the power set of $n$ query terms within a window. It is trivial to see that this approach suffers from combinatorial explosion for larger queries and hence is of limited practical use. A more practical approach is to combine signals from a limited power set of query terms, as done via $n \times n$ where $n \in [2, 3, 4, 5]$ in PACRR and through $h \in [1, 2, 3]$ in ConvKNRM.

One of the most obvious limitations of MRF model is the lack of soft matching capability. Thus, soft-matching models in the example of 1 will be able to related the query âĂIJ**cricket bowling speed record**âĂİ with document âĂIJShoaib Akhter holds the **record** for the **fastest delivery** in **cricket**âĂİ, where "delivery" is similar to "bowling" and "speed" is similar to "fastest". Other than this, the model also treats all types of dependency and proximity within a class of matches (ordered, unordered) as having the same weight or importance. The learnable parameters in nerual IR models allow the models to learn the importance of each type of match (kernel in PACRR, combination matrix in ConvKNRM) from data.

# 4 Axiomatic analysis of SDM model

### Term Frequency Constraints

**TFC1:** SDM model captures unigram (exact) matches through its full independence variant. Thus, it can be trivially seen that whenever a document $d_1$ has more matches of query term $q$ than another document $d_2$, it will be ranked higher. This is further reinforced by authors as they assign highest weight to these kind of unigram term frequency matches.

**TFC2:** The model explicitly incorporates term frequency discounting in the form of $log$ sum of term frequencies. This ensures diminishing returns for increasing number of matches.

### Term Discrimination Constraints

Term discrimination or rarity information is available in the factor $\frac{cf_w}{|C|}$, but it is not used in a discriminatory sense. It could be fixed simply by inverting the factor so it becomes $\frac{|C|}{cf_w}$ , a term discrimination statistic.

### Length Normalization Constraints

**LNC1:** The factor $\frac{tf_w}{|D|}$ ensures that this constraint is satisfied. If an unrelated term is added to document, it will decrease the score because of increased length.

**LNC2:** In case of LNC2, we see that concatenating a document with itself k times will result in a k-fold increase in the count part of all three model components. This, combined with TFC1 ensures that the score will increase. However, the discounting effect of logarithmic scaling and length normalization makes sure that this does not lead to a blowup in relevance score.

### TF-Length Constraint

Two documents $d_1$ and $d_2$ which have different lengths and where $d_1$ was constructed by adding more query terms to $d_2$ will contribute more to the frequency counts in all 3 components (FI, SD, FD) of models. Thus, $d_1$ will rank higher than $d_2$.

# 5 Axiomatic analysis of PACRR

### Term Frequency Constraints

**TFC1:** A document which contains $n + 1$ query terms will register more matches to the original similarity matrix compared to a document containing $n$ matches. Also,

depending on where precisely the additional matching term occurs, it will contribute positively to the score e.g. by turning a weaker proximity match to a strong n-gram match.

**TFC2:** In PACRR, unbounded increase in term frequency does not lead to (monotonically increasing) diminishing returns. Rather we observe a hard limit on contribution of additional term matches in the form of $k$-max pooling. Thus after $k$ matches, an additional match is no longer considered informative of relevance.

### Term Discrimination Constraints

PACRR passes the normalized IDF of query term to the LSTM layer which produces global relevance score. Hence, the network has access to discrimination statistics and can be reasonably expected to learn to give more importance i.e. produce a higher score for matches of rare terms.

### Length Normalization Constraints

**LNC1:** PACRR is made robust to document length because of two pooling layers which keep only the most relevant (filter-pooling) and the strongest (k-max pooling) signals. Given these components we can at least say that addition of an irrelevant term will not result in increasing the score. However, if the added term has a soft-match with a query term, it should increase the score [8]. For example, this might turn a 2 term weak proximity match captured by $3 \times 3$ kernel into a stronger 3-gram match. Thus we conclude that model is not biased towards long documents

**LNC2:** Concatenating a document with itself $i$ times will result in a $i$-fold increase in the matching patterns. This will result in signals from these matching patterns to make it past the $k$-max pooling step thus contributing more to the document score.

### TF-Length Constraint

As discussed above, adding more query terms will result in more matches being registered. Thus when $k$-max pooling is applied, signals from these matches will be preserved because they are the strongest and contribute to the score.

# References

[1] Donald Metzler and W. Bruce Croft. “A Markov Random Field Model for Term Dependencies”. In: *SIGIR* (2005).

[2] Tao Tao and ChengXiang Zhai. “An Exploration of Proximity Measures in Information Retrieval”. In: *SIGIR* (2007).

[3] Liang Pang et al. “A Study of MatchPyramid Models on Ad-hoc Retrieval”. In: *SIGIR Workshop on Neural Information Retrieval* (2016).

[4] Kai Hui et al. “PACRR: A Position-Aware Neural IR Model for Relevance Matching”. In: *WSDM* (2018).

[5] Zhuyun Dai et al. “Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search”. In: *WSDM* (2018).

[6] L. Pang et al. “Text matching as Image Recognition”. In: *Proceedings of the 13th AAAI Conference on Artificial Intelligence (AAAI)* (2017), pp. 2793–2799.

[7] C. Xiong et al. “End-to-End Neural Ad-hoc Ranking with Kernel Pooling”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research  Development in Information Retrieval* (2017).

[8] H. Fang and C. Zhai. “Semantic term matching in axiomatic approaches to information retrieval”. In: *SIGIR* (2006), pp. 115–122.