Unsupervised learning blocking keys technique for indexing Arabic entity resolution

Marwah Alian, Arafat Awajan & Bandan Ramadan

International Journal of Speech Technology

ISSN 1381-2416

Int J Speech Technol
DOI 10.1007/s10772-018-9489-6

Springer

International Journal of Speech Technology
https://doi.org/10.1007/s10772-018-9489-6

Unsupervised learning blocking keys technique for indexing Arabic entity resolution

Marwah Alian[1,2] · Arafat Awajan[2] · Bandan Ramadan[3]

Received: 12 October 2017 / Accepted: 3 January 2018

Abstract
Attribute values in textual datasets are subjects of different types of errors due to the data entry processes such as typographical errors, pronunciation errors or dialects alterations. These errors make the entity resolution process more challenging. The iterative blocking indexing technique can be used for correcting this type of errors mainly in query access where the records are stored into more than one block. Blocking indexing technique selects a subset of object pairs saved in the same block for later detailed computation for similarity discarding other pairs in other blocks considering them as irrelevant. This work aims to solving such problems for Arabic texts. It proposes to adapt a specific model for learning blocking keys and analyze its performance for Arabic datasets. The resulted blocking keys are passed as blocking keys to Dynamic Aware Inverted Index (DySimII) that worked efficiently with Arabic datasets. The model is tested against a telephone book dataset that contains duplicates and errors in attribute values according to phonetic and typing errors. The results reach a matching accuracy of 84% for using learned keys with small number of corrupted attributes while the performance is declined with the increase of the number of corrupted attributes.

Keywords Arabic entity resolution · Learning keys · Indexing · Arabic datasets

1 Introduction

The process of finding matches for a query record in a dataset or multiple datasets that represent the same real world entity is called Entity resolution (ER) (Ramadan and Christen 2015). ER techniques are important for organizations to improve their business operations as well as for a data warehouse which needs integration of data from multiple sources into one consistent center. ER techniques are very useful for cleaning and standardizing data (Ramadan 2016).

Indexing is an essential step in ER process mainly for large datasets since it has the advantage of reducing the number of candidate records that are going to pass to the detailed comparison step (Ramadan 2016). There are two main approaches for indexing in ER; the first approach is blocking in which records in a dataset are partitioned into different blocks according to a blocking key criterion and the records that stored in the same block are compared with each other. The second approach is sorting in which records are sorted with a sorting key and this will bring similar records to be closed to each other to compare only records that are close to each other (Ramadan 2016).

In Entity Resolution or Record Linkage, the number of similarity computation between pairs increase quadratically as the size of the dataset increase since the similarity function will be computed for all pairs in the dataset (Bilenko et al. 2006). However, a good indexing approach should store similar records in the same block or bring them close to each other based on the key criteria (Ramadan 2016). Thus, blocking techniques relieve the problem of comparing all records in the dataset by computing the similarity between pairs in the same block that are approximately similar (Bilenko et al. 2006).

In real world, data always contain errors and alterations (Hernandez and Stolfo 1998) due mainly to data entry errors.

✉ Marwah Alian
marwah2001@yahoo.com

Arafat Awajan
awajan@psut.edu.jo

Bandan Ramadan
bramadan@psu.edu.sa

[1] Hashemite University, Zarqa, Jordan

[2] Princess Sumaya University for Technology, Amman, Jordan

[3] Prince Sultan University, Riyadh, Saudi Arabia

Published online: 15 January 2018

Accordingly, same records related to same entity may be difficult to identify by an SQL query. For solving this problem, we need to implement a solution able to identify potential records related to the same entity even if these name entities were altered by different sources of errors.

Blocking indexing approach is one of the most efficient approaches able to help in solving this problem. It aims to gather similar attribute values in one block (Christen 2012). Two issues are taken in consideration when selecting attributes to be utilized as blocking keys. The first one is the sensitivity of candidate record pair quality which is affected by the quality of the attribute values. However, missing true matches could happen when records are stored in an incorrect block because of errors in the attribute value which is used to generate blocking keys. Thus, attribute values with fewer errors or missing values should be selected as blocking keys (Christen and Goiser 2007). The second issue that should be considered is to take care of the size of the generated blocks as the size is affected by the frequency distribution for the values used as blocking keys (Christen 2012). Therefore, the frequency distribution of the attribute values should be also considered when choosing an attribute as a blocking key.

Most work on this topic was realized for European languages (Bilenko et al. 2006; Ramadan et al. 2013; Christen and Hawking 2009) while other works focus on other languages such as Japanese. However, this problem did not receive appropriate previous effort in Arabic. Arabic is a challenging language because of its rich morphological features, its sparsity, and the limited availability of linguistics resources. The already realized work on Arabic ER is very limited to (Alian et al. 2017) which tackles the problem of entity resolution for structured Arabic datasets.

This research aims to utilize and adapt an unsupervised technique for learning blocking keys (Ramadan and Christen 2015) to work with Arabic dataset. The resulted optimal keys will be applied in an indexing technique called Dynamic Similarity Aware Inverted Indexing (DysimII) (Ramadan et al. 2013) that was experimented and had produced good performances.

This research is organized as follows; Sect. 2 provide an overview for related work, Sect. 3 illustrates problems related to processing Arabic datasets and Sect. 4 describes the methodology and the algorithms used for handling Entity Resolution for Arabic datasets. In Sect. 5, experiments and results are discussed and finally the conclusion is described in Sect. 6.

2 Related work

Standard blocking indexing technique store similar records in blocks based on a blocking key which depends on one or more attribute values (Fellegi and Sunter 1969). This technique is sensitive for dirty data since errors in attributes used as blocking keys will store the records into the incorrect block (Ramadan and Christen 2015).

In order to have a good blocking indexing approach, it should store similar records in the same block based on the key criterion that is used to distribute records among blocks (Ramadan 2016). To achieve this, automatic selection for optimal blocking key has been studied by many researchers where the studies are categorized into supervised and unsupervised learning for the blocking keys.

One of the recent supervised techniques is the work proposed by Vogel and Naumann (2012) to automatically select blocking keys where they use unigrams of attribute values to create blocking keys then reuse these keys for datasets from similar domains that do not have gold standard. The quality of the blocking keys was measured with regards to accuracy and efficiency of the blocks generated by these keys. They show by experiments that finding high quality blocking keys would be possible for similar domains datasets having no gold standard by selecting these keys from training dataset results. The disadvantage of this approach is that it required a labeled training data set (i.e. gold standard) which could not be available in all cases.

While Bilenko et al. (Bilenko et al. 2006) proposed an adaptive framework for automatic learning of the blocking functions. They provide formalization for the problem of blocking function adaptation to a specific domain using training data. In this work they introduce constructing blocking functions depending on a group of general blocking predicates that choose all pairs that assure a criterion of binary similarity. The key advantage of this approach is that it could be utilized in the adaptation of the blocking function in any domain and let experts add the domain-specific predicates.

Based on this work, Kejriwal and Miranker (2013) proposed an unsupervised learning for the blocking functions for tabular data sets. Their approach has two main phases. The first phase generates weakly labeled training datasets automatically while the second phase uses the generated labeled training data sets to learn the optimal blocking keys by a Fisher discrimination criterion where this Fisher score is utilized to rank the candidate blocking keys, then an optimal blocking key is selected according to the highest Fisher score. In their work, Key coverage is the only considered measure for calculating Fisher scores and selecting the optimal blocking keys. The results of their work presented

encouraging results when it is compared to a supervised algorithm.

Kejriwal's approach (2013) concentrates on the quality of the generated blocks when it learns the blocking key. Thus, their selected blocking keys are not necessary suitable for use with real-time ER. To overcome this issue, Ramadan and Christen (2015) proposed an unsupervised learning approach, that is based on Kejriwal (2013). However, Ramadan's approach (Ramadan and Christen 2015) proposes an automatic approach that selects optimal blocking keys taking into consideration the maximum size of the generated blocks, and other factors such as the distribution of the sizes of the generated block and the coverage of the selected blocking key. Their objective is to learn blocking keys that are suitable for use with real-time ER.

The work in the field of Arabic ER is very limited and depends on other techniques that works with other languages and try to apply it for Arabic. For example Alian et al. (Alian et al. 2017) investigate the Dynamic Aware Inverted Indexing technique (Ramadan et al. 2013) to work for Arabic ER with some adaptation to be suited for Arabic datasets.

In this research, one of the state of the art unsupervised techniques for learning blocking keys (Ramadan and Christen 2015) is adapted and applied on Arabic dataset, then the optimal blocking keys are passed to an indexing technique works for Arabic ER and compare its performance with the new learned blocking keys..

3 Specific problems for arabic

Arabic language is a Semitic language that has a complex linguistic structure with rich morphology (Attia 1999; Farghaly 1987). Processing Arabic language for NLP applications is challenging due to several features related to the nature and structure of the language. One of the problems is the lack of Arabic corpus which affects name entity recognition in rule based and statistical based systems (Farghaly 2009).

The main problems facing ER in Arabic are related to errors in typography. These errors are generated due to variations of huge amount of dialects among Arabic speakers; the different pronunciations of the same letters and words in addition to typing errors. Typing errors comes from mistakes in pressing keyboard letters that are in the same horizontal, vertical or diagonal position instead of the target letter.

Pronunciation errors are due to different dialects in Arabic region for example in Egypt some characters are pronounced in different way than they are pronounced in Jordan such as (ث، س), (ذ، ز), (ض،ظ), (ظ، ز), etc. other errors in typing comes from the similarity between letters in their shapes such as (ع، غ), (ش،س), (ث، ت), etc. Some examples for words with possible errors due to typing and dialects errors are shown in Table 1.

Table 1 Possible errors due to dialects and typing errors

Word	Possible errors due to dialects	Possible errors due to typing errors
قاسم qAsm	كاسم ، جاسم، اسم Asm, jAsm, kAsm	فاسم، قاشم qA$m, fAsm
ظاهر ZAhr	زاهر، تاهر، ضاهر DAhr, tAhr, zAhr	طاهر TAhr
هاشم hA$m	هاجم hAjm	هاسم، هاضم hADm, hAsm
ثامر vAmr	تامر، سامر sAmr, tAmr	ثامز، يامر yAmr, vAmz

These errors make real time Entity Resolution for Arabic data more challenging since it should retrieve same records related to the same entity regardless of these errors within part of seconds.

4 Model frame work

4.1 Basic models

In this research, a new model is developed with two phases for Arabic language. This model is based on the model proposed by Banda and Christen (Ramadan and Christen 2015) presented in Fig. 1.

Banda and Christen (2015) model consists of five steps; the first step produces a set of candidate blocking keys, positive and negative training datasets where the candidate blocking keys are produced depending on domain knowledge. The second step converts both positive and negative training sets into positive and negative blocking key vectors in order to be used in learning keys process. In the third step, the learning blocking keys algorithm is applied using the generated blocking key vectors (positive and negative) to select a set of optimal blocking keys (Ramadan 2016). The fourth and fifth steps are used for evaluating the blocking keys, where the fourth step uses the selected optimal keys produced from the previous step in blocking all records in a dataset by any real time indexing approach while in the fifth step, the index that is built in the previous step is utilized for matching in real-time query records with existing records in the index. For these two steps, they use real-time forest-based dynamic sorted neighborhood index (F-DySNI) (Ramadan and Christen 2014).

4.2 Proposed model

The proposed model that is presented in Fig. 2, uses the first three steps of Banda and Christen Model (Ramadan and

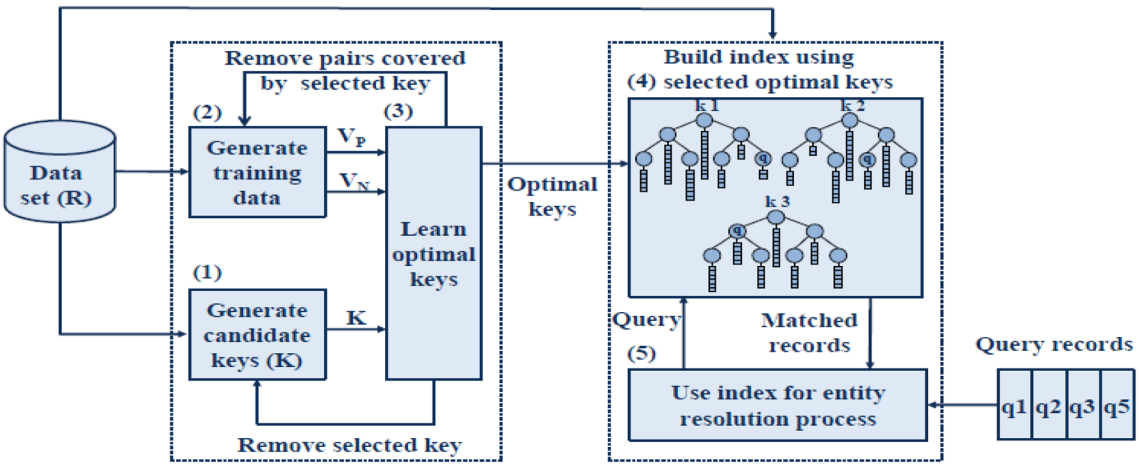

Fig. 1 Unsupervised selection for blocking keys (Ramadan and Christen 2015)

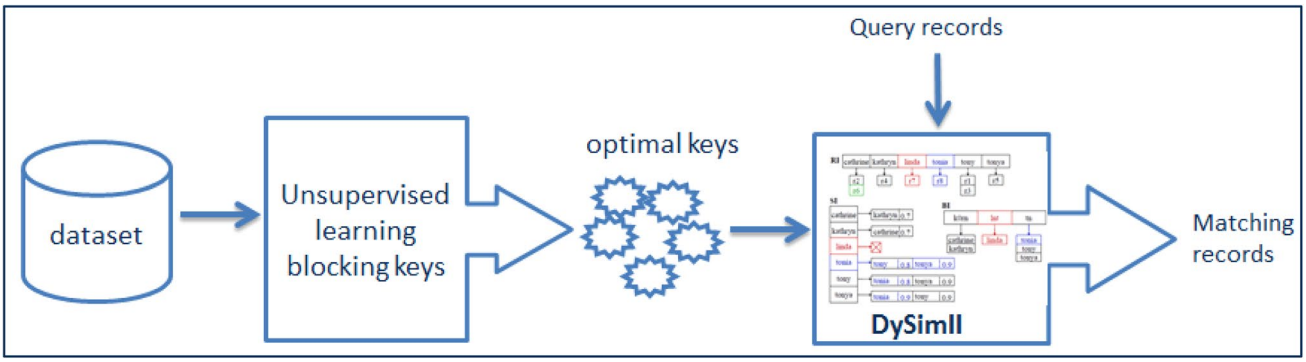

Fig. 2 Model framework

Christen 2015) to form its first phase. In the second phase of our model we use DySimII (Ramadan et al. 2013) instead of DySNI since it is proven by the results of our model that DySimII improves the results in the case of Arabic language.

In the first phase, the unsupervised learning blocking keys technique that is proposed by Ramadan and Christen (2015) is adapted to predict optimal blocking keys for Arabic datasets. In addition, we improve the system by adding stem as a candidate key and IsExactStem as a function key to their work for adapting this stage to work for Arabic datasets. However, the stem is generated for an attribute value as a candidate key by removing prefixes and affixes (i.e. light stem) (Hayder et al. 2005) then it is compared with other attribute values according to the function key IsExactStem which return true if the two values have the same stem and false otherwise.

The generated optimal blocking keys are passed to the second phase where DysimII (Ramadan et al. 2013) is used for blocking all records from the original (unlabeled) dataset as well as for matching query records in real-time with existing records in the built index.

4.3 Dynamic similarity aware inverted index

Dynamic Similarity Aware Inverted Index (DySimII) (Ramadan et al. 2013) technique added the query records to the index so it is updated after each query record making the index up to date. It is a solution based on memory since the full index has to fit in the available memory but this produce a challenge for large datasets.

This technique is based on the similarity-aware inverted index which is proposed by Christen et al. (2009) which is faster than standard blocking but it is static. Therefore DysimII is introduced to be dynamic where values can be added to the index while processing a new query record.

The DySimII consists of three indexes; Block Index, Similarity Index and Record Index. The Block Index (BI) is an inverted index which holds attribute values and their

associated Blocking Key Values which is provided using encoding functions such as Soundex, Phonex, or Double-Metaphone (Christen 2012). The keys of this index are the Blocking Key Values and each key point to a list of attribute values assigned to this Blocking Key Value. The Similarity Index (SI) holds the pre-calculated similarities using comparison functions for strings like Jaro-winkler, edit distance or Jaccard. In Similarity Index the keys are the unique attribute values and points to similarities between this key value and values that are in the same block. The similarity value is between 0 and 1. The Record Index (RI) holds the attribute values with their record identifiers where the keys in this index are the unique attribute values pointing to a list of record identifiers that have this attribute value (Ramadan 2016).

In the adapted version of DySimII (Alian et al. 2017) that works for Arabic the blocking keys in BI are the stems of the attribute values. It was tested using different stemmers such as ISRIstemmer and Asem stemmer with respect to different similarity functions such as Jaro-Winkler, N-gram, and Edit distance. The results represent that matching accuracy is improved using Asem stemmer when the number of corrupted attributes is increased, also using winkler similarity function provides better matching accuracy than other similarity functions. In this research we depend on these results to apply the adapted DysimII where Asem stemmer and Jaro-Winkler similarity function are used in the experiments.

Indexing is divided into two main phases; the first phase is the building phase, and the second is the query phase. In the first phase a data set is stored in memory using the three indexes BI, RI, SI. In the second phase, the attribute values of the query record are inserted into the indexes and a list is generated from records that have the same block as the query record then compared with the query record in detail in order to get matching records. (Ramadan et al. 2013).

The attribute values are added to the inverted indexes based on two cases; the first one when the attribute value is not in the inverted indexes and the second when the attribute value is already exist in the inverted indexes. If an attribute value for a record is new then it is inserted in RI with its record identifier and a computation for the blocking value for this attribute value is performed to determine to which block in BI index the attribute value will be added. If the block of this value is already exists then the attribute value is added to this block but if does not exist then a new block for the blocking value is created with the insertion of the attribute value in this new block (Ramadan et al. 2013).

When the attribute value is inserted into an existing block, the similarities between the inserted value and the other values in the block are calculated using an approximate similarity function and stored in the SI. However, if an attribute value is already indexed then only its record identifier is inserted in the RI and no need to update the BI or the SI because the value is indexed previously (Ramadan 2016).

In the query phase of DySimII, a new attribute value from a query record is inserted in the indexes. Thus, if the same attribute value is seen in other query records then the encoding and similarity calculation is already performed for this value (Ramadan et al. 2013).

5 Experiments and results

5.1 Dataset

This research is experimented using python2.7 environment and an Arabic dataset for phone book with 624,850 records consist of 50% duplicates used as query records, the maximum number of duplicates for a record is two duplicates. The attributes of the dataset are 7 attributes; record

Table 2 Adapted corruptors used in GeCo corruptor

Corruptor name	Function	Examples
OCR corruptor	Replaces characters by their similar shapes characters	(س،ش) (ر,ز),(ع,غ),(ت،ث),(ف،ق)
Keyboard corruptor	Depends on typing errors for keyboard characters by exchanging characters poisoned horizontally, vertically or diagonally on the keyboard	(ض,ص,ث,ق,ف,غ,ع,ه,خ,ح), (ش,س,ي,ب,ل,ا,ت,ن,م) (ى,ء,ؤ,ر, ,لا,ى,ة).
Pronunciation corruptor	Based on pronunciation errors by exchanging characters with similar character sound or phonetic.	(ش،ج),(ي,ج),(ت،د),(ذ،ز),(ظ،ض), (ط،ت),(ق،ك),(ث،س),(غ،خ),(ع،ح), (ي،ج) (ظ ،ز).
Missing value corruptor	An attribute value is replaced by none or ' ' to be considered as missing.	
Edit corruptor	Uses a number of modifications per string value (substitute, delete, insert, and transpose).	سالم --- اسلم transpose سالم --- الم delete سالم --- اشلم substitute سالم --- سالم insert

ID, first name, second name, third name, family name, city and phone number.

The duplicates are produced using GeCo corruptor (Tran et al. 2013) that is a free source code provided online used for generating data with duplicates and corrupting data for experiment purposes. It is downloaded and adapted to work for Arabic dataset by adding and modifying existing corruptors included in GeCo. The corruptors that we use are illustrated in Table 2.

GeCo corruptor produce duplicates with corruptions and add entity id for the dataset to be used later to get matching accuracy. In GeCo corruptor the user has to specify the number of corruptions per record and the percentage of each corruptor (OCR, Keyboard, phonetic, and missing value, and edit corruptor) for each attribute. The output datasets from GeCo are used in the experiments of this research.

5.2 Experiments and discussion

Three experiments were conducted, two of them are dedicated to investigate the validity of learning blocking keys for Arabic datasets using an unsupervised technique and its effect on the performance of the indexing technique that receive these learning keys.

We adopted the same overall score sck measure that is defined by Ramadan and Christen (2015) for evaluating which keys should be added to the optimal key list. The overall score is measured according to Eq. (1) (Ramadan and Christen 2015):

$$sc_k = \alpha \times \left(1 - c_k\right) + \beta \times s_{b(ave)_k} + (1 - \alpha - \beta) \times v_k \qquad (1)$$

where $sb_{(ave)k}$ is the average block size generated by evaluating keys in the set of valid keys, while c_k is the key coverage which is measured by fisher score Kejriwal and Miranker (2013) and v_k is the variance of the sizes of all blocks in the set of the generated blocks from applying a key on dataset. The variance v_k reflects how far the generated block sizes are spread. However, Kejriwal and Miranker (2013) depend only on key coverage to evaluate an optimal key.

The parameters α and β are used to control the weights of the three criteria based on the domain and application area. Each weight parameter is a value between 0 and 1 where the sum of all weights is equal to 1 regardless of the weighting parameters used (Ramadan and Christen 2015).

After score sck calculation for all keys in the set of valid keys, the scores are sorted in an ascending order because the lower overall score is better.

The same parameters that were used for measuring the overall score in the unsupervised learning keys algorithm (Kejriwal and Miranker 2013) are used in the three experiments in this research.

For example, α which is the weight that is used for the key coverage equals to 0.2 and β which is the weight that is used for the block size equals 0.4.

The first experiment applies the unsupervised learning blocking keys on a sample from the phone book dataset with 3000 records that have duplicates but without corruption, then taking the resulted optimal keys and uses them in the ER indexing technique DysimII for the complete phone book dataset. In this experiment the optimal keys were the concatenation between each attribute and Family Name as a blocking key (BK) for that attribute and the stem of Family name as a blocking key for Family name. For example Concatenated_FirstName_FamilyName is the optimal key for First Name attribute, Concatenated_ThirdName_FamilyName is the optimal key for Third Name attribute and concatenated also with FamilyName for Second Name and city attributes.

While in the second experiment, a sample from the Arabic phone book dataset is used as an input to generate positive and negative training sets. Then it is used with the unsupervised algorithm for learning blocking keys. The sample consists of 3000 records containing duplicates and corruptions in one, two, three, four and five attribute values. A sample is used since in the unsupervised learning key algorithm for datasets with short attribute values that has one word, the attribute value is converted into q-grams and those q-grams become the blocks that are used to partition records the dataset (Ramadan 2016).

Each one of the corrupted datasets resulted in optimal keys. The resulted optimal keys for corrupted data were mostly the same and they are passed to the ER indexing technique (DysimII) then the DysimII is applied on the phone book dataset. The execution runs for the phone book with one corrupted attribute, two, three, four, and five corrupted attributes. In this experiment, the optimal keys were the exact-Family name as a blocking key for Family name attribute and the concatenation between family name and each attribute as the blocking key for that attribute. Moreover, the optimal blocking key for first name is Concatenated_firstName_FamilyName, for second name the optimal key is concatenated_secondName_FamilyName and the concatenation with FamilyName is also the optimal for city and thirdName attributes.

In the third experiment, the stem of attribute values is used as the blocking key for the ER indexing technique DysimII. In this experiment Asem stemmer is used in this experiment which is a light stemmer (Chelli 2016). The stem or light stemmer removes the most frequent suffixes and prefixes from the word but do not produce the linguistic root (Hayder et al. 2005) and it is used frequently in indexing Arabic documents and in Information Retrieval. DysimII with stem as a blocking indexing key for all attributes is executed for datasets with duplicates and corruption in one, two, three, four, and five attributes.

Table 3 Recall (matching accuracy) results in the conducted experiments

No of corrupted attributes	Family name stem blocking key and all other concatenate with family name (%)	Exact family name blocking key and all other concatenate with family name (%)	Stem blocking key all attributes (%)
1. Attribute	83.80	84.07	83.49
2. Attributes	70.22	70.76	70.22
3. Attributes	35.81	32.89	40.19
4. Attributes	5.41	1.02	11.49
5. Attributes	1.00	0.72	3.24

Table 4 Query average time in seconds for the three experiments

Number of corrupted attributes	Family name stem blocking key and all other concatenate with family name	Family name exact blocking key and all other concatenate with family name	Stem blocking key all attributes
1. Attribute	0.113	0.091	0.115
2. Attributes	0.088	0.089	0.080
3. Attributes	0.068	0.066	0.063
4. Attributes	0.049	0.049	0.041
5. Attributes	0.027	0.029	0.040

The recall as a matching accuracy measure is computed for the three experiments with datasets include one, two, three, four, and five corrupted attribute values. The results are shown in Table 3.

As shown in Table 3, experiment one which uses exact family name as a blocking key and the concatenation between an attribute and family name as a blocking key provide better matching accuracy (recall) when the dataset contains small number of corruptions (one or two corruption per record) while it's matching accuracy decline with the increase in the number of corruptions per record.

While in experiment two, using the stem of family name as a blocking key and concatenated family name with an attribute value as a blocking key for other attributes provide small improvement in the matching accuracy over using the stem as a blocking key for all attributes only when the one attribute corrupted per record and has a similar matching accuracy to the use of stem when there is two corruption per record. However, when the dataset has three and more corruptions per record the matching accuracy decreased significantly.

Experiment three that apply the stem as a blocking key for all attributes outperform other choices for blocking keys with the increase in the corruption per record as shown in Table 3.

Table 4 represents the average query time in seconds for the three experiments. It is shown that with more corruptions per record the query time decreased where it don't find the query even it is for the same entity since it is saved in another blocking key because of errors. The average query time is improved with exact FamilyName and Concatenated attributes (First,Second,Third, and city attribute) with FamilyName only with one corruption per record while the average query time is improved by the use of the stem as a blocking key with two, three, four and five corruption per attribute.

6 Conclusion

In this research we investigate an unsupervised learning blocking key technique if it works with Arabic datasets. The technique is adapted to be executed on Arabic dataset then the predicted optimal keys from this technique are tested on an ER indexing technique. DySimII was selected as the indexing technique since it was tested previously on Arabic datasets and provides good matching accuracy (i.e. recall).

The experiments conducted according to the blocking keys used with DysimII based on the output blocking keys from the unsupervised learning blocking keys. The results show that the exact Family name and the concatenation between Family name and another attribute as Blocking keys outperform other choices for blocking keys when the corruption per record is small while the stem as a blocking key for all attributes provide better results with the increase in the number of corruptions per record.

References

Attia, M. (1999). A large scale computational processor of Arabic morphology and applications. Master's Dissertation, Computer Engineering. Egypt: Cairo University.

Ramadan, B. (2016). Indexing techniques for Real-time entity resolution on large dynamic databases, PhD Thesis, Ed.: Australian National University.

Ramadan, B., Christen, P. (2014). Forest-based dynamic sorted neighborhood indexing for real-time entity resolution, in Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, Shanghai, pp. 1787–1790.

Chelli, A. (2016). ASem Light Stemmer, available at: http://www.arabicstemmer.com/.

Christen, P. (2012). *Data matching: Concepts and techniques for record link-age, entity resolution, and duplicate detection*. New York: Springer.

Christen, P. (2012). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering, 24*(9), 1537–1555.

Christen, P., & Goiser, K. (2007). Quality and complexity measures for data linkage and deduplication, In Hamilton, F., Guillet, H. J. (Ed.) *Quality measures in data mining, ser. studies in computational intelligence*, New York: Springer, pp. 127–151.

Farghaly, A. F. (1987). Three level morphology for Arabic," in the Arabic Morphology Workshop (AMW'87), Italy.

Fellegi, I., & Sunter, A. (1969). A theory for record linkage. *Journal of the American Statistical Association, 64*(328), 1183–1210.

Al Ameed, H. K., Al Ketbi, S. O., Al Kaabi, A. A., Al Shebli, K. S. A., Al Shamsi, N. F., Al Nuaimi, N. H., Al Muhairi S. S., (2005). Arabic Light Stemmer: Anew Enhanced Approach," in The Second International Conference on Innovations in Information Technology (IIT'05),, pp. 1–9.

Hernandez, M. A., & Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery, 2*(1): 9–37.

Alian, M., Al-Naymat, G., Ramadan B. (2017). Using transliteration with entity resolution for Arabic datasets, in 14th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'2017), Hammamet.

Kejriwal M., Miranker, D. P. (2013). An unsupervised algorithm for learning blocking schemes, in 2013 IEEE 13th International Conference on Data Mining, pp. 340–349.

Bilenko, M., Kamath, B., Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage, in Sixth IEEE International Conference on Data Mining (ICDM-06), Hong Kong, pp. 87–96.

Christen, P., Gayler, R., & Hawking, D. (2009). Similarity-aware indexing for real-time entity resolution. in 18th ACM conference on Information and knowledge management, Hong Kong, China, pp. 1565–1568.

Ramadan, B., & Christen, P. (2015). *Unsupervised blocking key selection for real-time entity resolution in advances in knowledge discovery and data mining PAKDD 2015*. Lecture Notes in Computer Science, Cham, Vol. 9078, pp. 574–585.

Ramadan, B., Christen, P., Liang, H., & Gayler, R. W. (2013). Dynamic similarity-aware inverted indexing for real-time entity resolution in *The series of lecture notes in computer science,trends and applications in knowledge discovery and data mining*. New York: Springer, Vol. 7867, pp. 47–58.

Farghaly, A., Shaalan, K. (2009). Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing, 8*(4), 22.

Tran, K. N., Vatsalan, D., Christen, P., (2013). GeCo—An online personal data Generator and Corruptor, in ACM Conference on Information and Knowledge Management (ICIKM'13), San Francisco, CA, USA, pp. 2473–2475, http://dmm.anu.edu.au/geco.

Vogel, T., Naumann, F. (2012). Automatic blocking key selection for duplicate detection based on unigram combinations, in the international workshop on quality in databases (QDB).