Linking IT Product Records

Katsiaryna Mirylenka[1], Paolo Scotton[1], Christoph Miksovic[1], and Salah-Eddine Bariol Alaoui[1,2]

[1] IBM Research – Zurich, Switzerland
{kmi,psc,cmi,bar}@zurich.ibm.com
[2] Polytech' Nice - Sophia, France
salaheeddine.bariol-alaoui@etu.univ-cotedazur.fr

Abstract. Today's enterprise decision making relies heavily on insights derived from vast amounts of data from different sources. To acquire these insights, the available data must be cleaned, integrated and linked. In this work, we focus on the problem of linking records that contain textual descriptions of IT products.
Following the insights of domain experts about the importance of alphanumeric substrings for IT product descriptions, we propose a trainable similarity measure that assigns higher weight to alpha-numeric tokens, is invariant to token order and handles typographical errors. The measure is based on Levenshtein distance with trainable parameters that assign more weight to the most discriminative tokens. Not being frequency-based, the parameters capture the semantic specificities of IT product descriptions.
For our task we assess the performance of the most promising lightweight similarity measures, such as (a) edit measure (Levenshtein), (b) frequency-weighted token-based (WHIRL) similarity measure, and (c) the measure based on BERT embeddings after unsupervised retraining. We compare them with the proposed spelling-error-tolerant and order-indifferent hybrid similarity measure that we call the Levenshtein tokenized measure. Using a real-world dataset, we show experimentally that the Levenshtein tokenized measure achieves the best performance for our task.

Keywords: Record linkage · Similarity measure · IT products.

1 Introduction

Data have become a precious resource for enterprise decision making. In the IT industry, a company's strategical marketing decisions are often made by considering information about products installed at a customer's sites and products that were already sold by the company to that particular customer. Such information is available through internal and commercial datasets which have heterogeneous representations of items. A fundamental and necessary step to gain insights from such datasets is the ability to link items in the various sets.

In this paper, we focus on the task of linking IT product records, which is crucial for company modeling and future product recommendations [17],[18],[20].

Linking product records is also a building block of similarity searches ([5],[6],[16], [15]) and streaming data analysis ([19],[14],[13]) for the data series of IT products. A given product can be represented in various more or less "similar" ways in different data sources. The differences across these representations may include formats, synonyms, abbreviations, acronyms and even typographical errors. An example of records to be linked is shown in Figure 1. The challenge is to detect

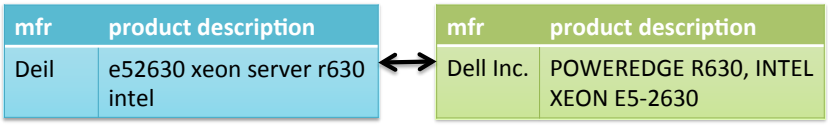


Fig. 1: Example of linking records.

whether all these representations correspond to the same unique product entity.

We consider a complete dataset of all products of interest, which we call the *master* dataset[3]. The task we wish to accomplish is to match records of a given *query* dataset against the master dataset. The objective is to find the "best" matching catalog entry for each of the items from the query dataset. Both query and master datasets are results of human input. Their vocabulary is not standardized, meaning that product descriptions may contain typos, omissions, and spelling varieties. To find the best matches, we need a quantitative similarity measure to deal with such inconsistencies.

As only a limited amount of ground-truth data is available, it is not feasible to apply supervised machine learning and probabilistic record-linking techniques. In this work, we consider two types of techniques:

1. Record linkage based on advanced contextual word embeddings called BERT (bidirectional encoder representations from transformers), where all the tokens in the product descriptions are correlated. In this case, a BERT network is retrained in an unsupervised manner with the product descriptions available in our master dataset. Then, a similarity search for a query product is performed in the space of retrained BERT product embeddings.
2. The second type of technique are the usual rule-based approaches, where product descriptions are regarded as a string or an arbitrary set of words. The main benefit of these approaches is their limited number of parameters (sometimes no parameters at all) that can be successfully trained, given small ground-truth datasets.

The main contributions of this work are the following:

- We analyze and compare lightweight similarity matching techniques from different families to address the record linkage problem for IT products.
- We assess the applicability of context-based word embeddings (BERT) for the task of linking IT product records.

[3] Sometimes also called a reference dataset.

- We propose a hybrid similarity measure called the *Levenshtein tokenized* measure that features trainable weights for alphanumeric[4] tokens. For convenience, we refer to this as the LT measure.
- We demonstrate experimentally that the LT measure outperforms both the Levenshtein measure, a frequency-weighted, token-based measure and the BERT-based measure using real data in our deployment.

2 Related Work

State-of-the-art methods of record linkage include fuzzy or probabilistic record linkage based on supervised machine learning and deep learning models [27,10]. In the context of our application, the amount of training data is limited, making these models infeasible. Thus, we consider either unsupervised machine learning methods or lightweight[5] supervised methods that nevertheless allow for certain statistical inference and parameter tuning.

In 2018, BERT improved the state-of-the-art performance of various NLP tasks such as sentiment analysis or question answering [7]. The principle of BERT is to apply a deep bidirectional transformer architecture to encode long sentences. Essentially, BERT leverages two previously proposed models. The first one is ELMo [22], an LSTM cell architecture that allows contextual word representation. The second is the generative pre-trained transformer (OpenAI GPT) model [23], which uses a left-to-right architecture, where every token can be expected only in the self-attention layers of the transformer. These two models do not allow a word to have context both to its left and its right, thus limiting their performance in some tasks where bi-directional context is important. The major problem when considering a bi-directional context is that a word would itself be taken into account by a bi-directional encoder. BERT uses the "masked language modeling" training objective to predict the missing words, given their bidirectional context.

We use the BERT model to place IT products into the space of BERT embeddings and, then, to make a similarity search within that space. Although we still suspect that the amount of training data might be too small ito retrain such a big network, it is worth comparing this model with much easier hybrid rule-based and machine-learning methods.

Rule-based methods for record linkage are mainly focused on optimal similarity measure searches. There are numerous different algorithms that measure the distance between strings for approximate matching. They implement a similarity function that maps two input strings to a number (a similarity score) such that higher numeric values indicate higher similarity. According to [3], string similarity metrics can be largely classified into edit-distance-based metrics and token-based metrics.

Edit-based measures express similarity by counting the number of primitive operations (insertion, deletion, substitution and transposition) required to con-

[4] Alphanumeric tokens must contain digits and may contain letters.
[5] Lightweight methods are those with a small number of trainable parameters.

vert one string into another. Techniques belonging to this class consider different subsets of these operations. Here are some examples of edit-based measures:

- The *Jaro similarity measure* [11] is designed for short strings such as people's names. It uses the number of matching characters and necessary transpositions to compute the string distance. The *Jaro–Winkler distance* [28] is a variation of the former, which assigns more weight to common prefixes.
- The *Levenshtein similarity* [12] counts the number of insertion, deletion, and substitution operations. Usually a unit cost is assigned to a single operation, and the sum of all costs is returned as the distance between strings. A variant of this is the *Damerau–Levenshtein distance*, which also allows transposition of two characters. Different cost values can be assigned to individual operations, leading to the weighted Levenshtein distance. By sacrificing the metric's properties, the Levenshtein distance measure can be turned into a ratio ($0 \leq r \leq 1$) such that higher ratio values indicate greater similarity.

According to the comparison studies, the Levenshtein similarity measure outperforms other edit-based methods in most cases [2], [3]. Therefore it is widely used in many different application scenarios that require the computation of approximate string similarity measures ranging from plagiarism [24] to iris detection [26]. An ensemble approach that uses weighted compositions of Levenshtein and Jaccard similarity measures for company record linkage is described in [9].

We choose the Levenshtein similarity measure as the underlying method for the approximate matching of IT product descriptions because it allows for typos and small uncertainties of hand-written text. There are also fast implementations of the Levenshtein similarity algorithm. It is claimed in [1] that an approximation of Levenshtein similarity can be computed in near-linear time.

Token-based distance measures consider strings as multisets of characters:

- The *Jaccard coefficient* originally comes from biology and is used to compare finite sets. It is simply the quotient of the cardinalities of the intersection and the union of all characters or tokens in two strings.
- The *Cosine similarity* [25] for strings is usually computed on vocabulary vector encodings of a string.
- The *WHIRL* similarity [4], [8] measures the distance of two strings in terms of cosine similarity of weighted tf–idf vectors of words. This introduces statistical weighting for the importance of terms in a set of documents.
- *Q-grams with tf–idf* [8] divide a string into q-grams instead of words and computes the weight of each word according to its tf–idf. The distance between two strings is computed as the cosine similarity of the weighted words.

For our task of linking the records of IT products, where the strings that describe the products can contain words (tokens) in arbitrary order, we also assess the performance of WHIRL, the frequency-weighted token-based distance measure. We do not assess the performance of more advanced token-based approaches, where tokens are smaller than words, as the word semantics in a product description are very strong, and we do not want to lose this by splitting the words into q-grams. Instead, to capture typos and small inconsistencies, we exploit the Levenshtein similarity for a token.

3 Hybrid Similarity Measure

In our case of matching IT product descriptions, we need a similarity measure that is independent of a token[6] order, resilient to minor typos and text inconsistencies, and assigns more weight to matching scores of discriminative tokens. On the one hand, the discriminative tokens can be defined in terms of tf–idf weighting captured by the WHIRL similarity, the efficiency of which is assessed in the experimental section (Section 4). On the other hand, with our customized similarity measure, we check the hypotheses of the domain experts in IT products that almost all tokens are important in the product descriptions from the query dataset and that the alphanumeric tokens should have more weight.

In this regard we propose a hybrid similarity measure (LT measure) based on the Levenshtein measure that is applied to tokenized product descriptions. Before applying the similarity measure, product descriptions from a query dataset are preprocessed by removing unnecessary punctuation, spaces and upper case, and short tokens are merged with consecutive numeric tokens, such as "DL 360" $\rightarrow$ "DL360". Vendor names are preprocessed further by eliminating uninformative stop-words such as "inc." or "corp.", and by using special mapping dictionaries for brand names and acronyms such as "hp" $\rightarrow$ "hewlett packard".

Next, a record from the query dataset q is split into tokens t_i, $i = 1, 2, ..., n$ which are compared with the tokenized records from the master dataset. For each token in the query record, we search for the closest token r_k, $k = 1, ..., m$ in the master record μ and obtain a similarity score of s_{t_i}.

$$s_{t_i} = \max_{r_k \in \mu} LevenshteinScore(t_i, r_k). \tag{1}$$

The query token scores are aggregated to yield the similarity score of the record pair. Let us note that $\mathbb{A}$ is a set of alphabetic[7] and $\mathbb{N}$ a set of alphanumeric tokens. The indicator function $\mathbb{1}_{t_j \in \mathbb{X}}$ outputs 1, if $t_j \in \mathbb{X}$, and 0 otherwise. The LT similarity score can be written as

$$LT(q, \mu) = \frac{\sum_{i=1}^{n} \alpha \cdot s_{t_i} \cdot \mathbb{1}_{t_i \in \mathbb{A}} + s_{t_i} \cdot \mathbb{1}_{t_i \in \mathbb{N}}}{\sum_{i=1}^{n} \alpha \mathbb{1}_{t_i \in \mathbb{A}} + \mathbb{1}_{t_i \in \mathbb{N}}}. \tag{2}$$

According to the assumption of the importance of alphanumeric tokens, we assign them a weight of $= 1$, whereas the alphabetic tokens are assigned a weight of $\alpha \in (0, 1]$. Thus, we ensure that alphabetic tokens are always assigned a smaller or equal weight. In the experimental section (Section 4) we verify the hypothesis regarding the importance of the alphanumeric tokens. We also evaluate the influence of $\alpha \geq 1$, when alphabetic tokens are assigned more weight. A pair with the highest LT similarity score is considered to be the best match.

As there are certain product records that should not be matched, there is one more parameter β that depends on the similarity score, $\beta \in (0, 1]$. If the closest similar record has a similarity score greater than β, we consider a product q

[6] We refer to tokens as words.
[7] Alphabetic tokens contain only letters.

from a query dataset to be matched to a product μ from the master dataset, otherwise q is considered to be unmatched:

$$q \sim \mu \iff sim_{score}(q, \mu) \geq \beta, \tag{3}$$

where sim_{score} is the score of a similarity function.

In this work, we consider only one best match with the similarity score greater than β. For other applications, top-k matches might be considered using a similar evaluation process. Parameters α and β are to be trained for the optimal LT similarity measure. For comparison approaches, namely Levenshtein and WHIRL similarity measures, only β is trained.

The proposed LT measure is similar to the Mongue–Elkan method [21] in that it also combines edit-based and token-based similarities. As discussed above, the LT measure additionally allows more impact for discriminative tokens.

4 Experimental Evaluation

In order to assess the performance of the promising similarity-matching techniques for our task of linking records, we use a labeled dataset containing 3570 records of IT products based on real examples from the datasets in our deployment. The manual labeling process is quite complicated because the master table contains 21 k products. A total of 544 records from the labeled dataset should be matched to particular records in the master dataset of IT products, whereas 3026 records should not be matched because they correspond to missing product entities in the master dataset. The query records contain certain variabilities of the product descriptions, which comes partially from human error and partially from variations of the product descriptions. One of the real examples combining two types of variance is shown in Figure 1. We treat matched and unmatched products as two classes that should be correctly labeled: *class 1* stands for matched products and *class 2* for unmatched.

The similarity-matching algorithms we compare were chosen from the best performers in their classes and do not require extensive supervised training. These are the Levenshtein similarity measure as a representative of edit distance, WHIRL as a representative of token-based similarity measures and BERT-based similarity measure as a representative of contextual unsupervised neural network models. We compare these algorithms with the new customized similarity measure (LT measure) introduced in Section 3. Its accuracy is measured in terms of precision, recall and $F1$ score for each class. Precision means the ratio of correctly classified products among the retrieved products of a certain class. Recall means the ratio of correctly classified products among all the products of a certain class. F1 score is a combined accuracy measure that is the harmonic mean of precision and recall. We measure the $F1$ score for each class separately and the aggregated average $F1$ score for both classes.

The labeled dataset is split into training (60%) and test (40%) datasets. First, we tune the parameters of the similarity measure on the training dataset. The records from the training dataset are matched against the master dataset.

For our application it is often more important to retrieve all the objects from class 1 (matched products) correctly. Thus, we choose the best similarity measure and its parameters, such as β and α, for the LT measure (only β for other comparison measures) by maximizing the recall of the matched class in the first place, and then by maximizing the average $F1$ score. A grid search serves to optimize the parameters.

First, for the LT measure we choose the best α and β values in the training set to maximize the recall of the matched class. The plot of the $F1$ score together with the corresponding precision and recall values for various α levels is shown in Figure 2. The accuracy values correspond to β with maximum recall.

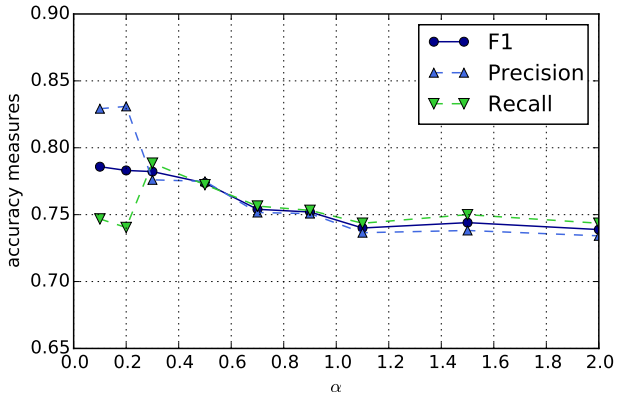
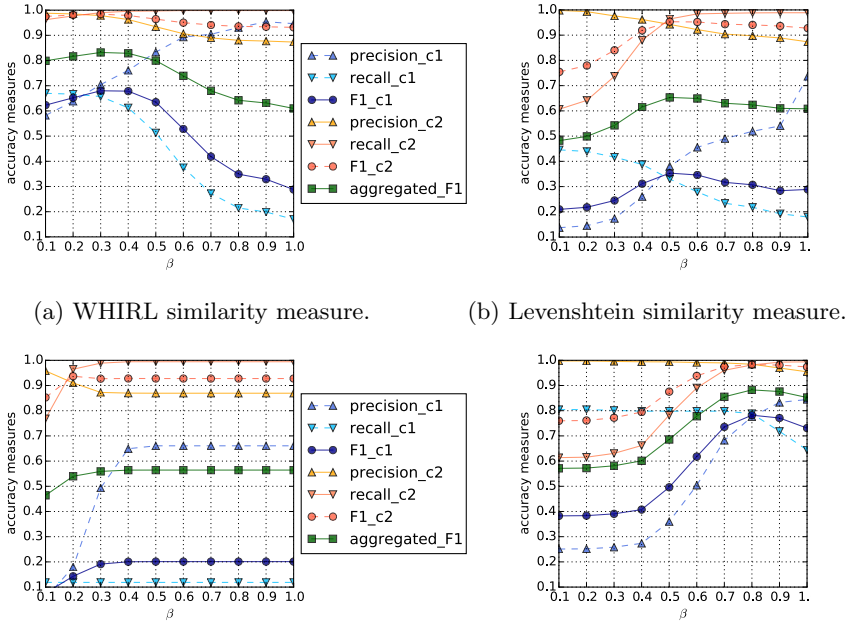


Fig. 2: Precision, recall and $F1$ score (training set) for class 1 as a function of the weight of alphanumeric tokens.

According to these results, $\alpha = 0.3$ provides the best retrieval of IT products from the matched class. This means that alphanumeric tokens should have a weight that is 3.3 times higher than that of alphabetical tokens. This supports the initial hypothesis that matching the alphanumeric tokens for IT product descriptions, such as "DL380" in "HPE ProLiant DL380" is much more important than matching alphabetical tokens, such as "HPE" or "Server".

Having chosen $\alpha = 0.3$, we compare the performance of LT with Levenshtein and WHIRL similarity measures for different values of β. The plot of performance measure variations for each similarity technique is shown in Figure 3.

The maximum F1 scores for class 1, and overall, are reached at different threshold levels β for each similarity measure. For the LT measure, the optimal β (β^*) is 0.8, for the Levenshtein measure it is $\beta^* = 0.5$, for WHIRL it is $\beta^* = 0.3$ and for the BERT-based measure it is $\beta^* \geq 0.3$. This shows that all the distance measures indeed capture different characteristics of similarity between the IT product descriptions. Other statistics on the performance of these three methods as a function of β can be found in Table 1.



(a) WHIRL similarity measure.

(b) Levenshtein similarity measure.

(c) BERT-based similarity measure.

(d) LT similarity measure.

Fig. 3: Performance of three similarity measures. Label $c1$ corresponds to the records that should be matched and label $c2$ to those that should be unmatched.

Note that the maximum precision and recall values reported in Table 1 do not necessarily correspond to maximum $F1$ scores. For example, maximum precision for class 1 of the WHIRL algorithm comes with quite low recall and $F1\_1$ values, namely 0.19 and 0.31, respectively, which can also be verified in Figure 3a. Thus, having a high precision for WHIRL means that, if WHIRL identifies a product as belonging to class 1, this is indeed a product of class 1in 92% of cases. At the same time, owing to the low recall of 19%, WHIRL is able to identify only 19% of class 1 products among all the products belonging to class 1.

After using trained α and β parameters for these methods, we report the accuracy values on the test set for class 1 in Figure 4. According to the results, the customized hybrid LT measure outperforms the best edit distance as well as BERT-based, token and tf–idf-based WHIRL distances.

A similar evaluation is performed to match vendor names of the products in query and master datasets. After vendors have been matched, product descriptions are matched within a vendor. In the case of vendor matching, the LT similarity measure also performs the best with $\beta^* = 0.85$. For lack of space, we do not describe the vendor matching process here. Vendor names are used

Table 1: Maximum performance values of matching algorithms for various β values on the training set. Labels c1 and c2 correspond to class 1 and class 2, respectively.

Acc. measure	**WHIRL**	**Levenshtein**	**BERT-based**	**LT**
max precision_c1	**0.95**	0.74	0.66	0.84
max recall_c1	0.67	0.45	0.12	**0.80**
max F1_c1	0.68	0.35	0.20	**0.78**
max precision_c2	0.99	**1.00**	0.96	**1.00**
max recall_c2	**1.00**	0.99	0.99	0.99
max F1_c2	**0.98**	0.95	0.94	**0.98**
max aggregated F1	0.83	0.65	0.56	**0.88**
optimal trained β	0.3	0.5	0.4	0.8
optimal trained α	–	–	–	0.3

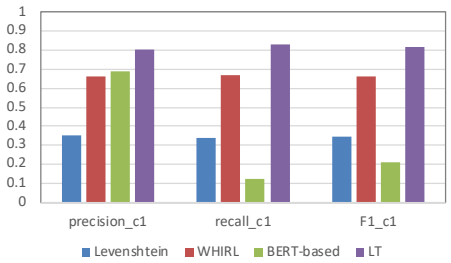


Fig. 4: Precision, recall and $F1$ score for test set.

as blocks within which the product records are matched in order to reduce the number of comparisons.

Although the BERT-based similarity measure did not outperform others, mainly having low recall for the products that should have been matched, we believe it can be adapted for our task in a better way. Its low recall is mainly associated with the fact that the large network (with many parameters) was solely retrained in the unsupervised manner with a limited amount of data. In addition, the task of linking IT product records is quite different from standard NLP language modeling tasks because the tokens in product descriptions are quite specific and therefore it is difficult to see them in usual texts. Moreover, typos and inconsistencies of encoding models for IT products create additional challenges for standard NLP modeling problems where vocabularies are fixed and typos are omitted.

5 Conclusions and Future Work

We introduced a customized similarity measure called the Levenshtein tokenized measure with the purpose to link the records of IT products. This similarity measure combines the benefits of edit and token-based measures and does not require extensive training. It also assigns higher weights to the alphanumeric tokens in product descriptions. This leads to higher matching accuracy for the tasks of record linkage and duplicate detection. We also evaluated the similarity measure based on contextual word embeddings. Although the LT measure did not outperform the proposed hybrid measure, we believe that neural unsupervised training of similar but easier neural network architectures combined with rule-based approaches such as our proposed LT measure might improve the accuracy of record linkage for IT products. We will investigate this in future work.

6 Acknowledgments

We thank the reviewers of this paper for their valuable comments, which greatly improved the paper.

References

1. Andoni, A., Krauthgamer, R., Onak, K.: Polylogarithmic Approximation for Edit Distance and the Asymmetric Query Complexity, pp. 244–252. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
2. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name matching in information integration. IEEE Intelligent Systems **18**(5), 16–23 (2003)
3. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string metrics for matching names and records. In: Kdd workshop on data cleaning and object consolidation. vol. 3, pp. 73–78 (2003)
4. Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. In: ACM SIGMOD Record. vol. 27, pp. 201–212. ACM (1998)
5. Dallachiesa, M., Nushi, B., Mirylenka, K., Palpanas, T.: Similarity matching for uncertain time series: analytical and experimental comparison. In: Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Querying and Mining Uncertain Spatio-Temporal Data. pp. 8–15. ACM (2011)
6. Dallachiesa, M., Nushi, B., Mirylenka, K., Palpanas, T.: Uncertain time-series similarity: Return to the basics. Proceedings of the VLDB Endowment **5**(11), 1662–1673 (2012)
7. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), `http://arxiv.org/abs/1810.04805`
8. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on knowledge and data engineering **19**(1), 1–16 (2007)
9. Gschwind, T., Miksovic, C., Mirylenka, K., Scotton, P.: Fast record linkage for company entities (2019), `http://arxiv.org/abs/1907.08667`

10. Hettiarachchi, G.P., Hettiarachchi, N.N., Hettiarachchi, D.S., Ebisuya, A.: Next generation data classification and linkage: Role of probabilistic models and artificial intelligence. In: IEEE Global Humanitarian Technology Conference (GHTC 2014). pp. 569–576 (2014)
11. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association **84**(406), 414–420 (1989)
12. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady **10**(8), 707–710 (Feb 1966)
13. Mirylenka, K.: Mining and Learning in Sequential Data Streams: Interesting Correlations and Classification in Noisy Settings. Ph.D. thesis, University of Trento (2015)
14. Mirylenka, K., Cormode, G., Palpanas, T., Srivastava, D.: Conditional heavy hitters: detecting interesting correlations in data streams. The VLDB Journal?The International Journal on Very Large Data Bases **24**(3), 395–414 (2015)
15. Mirylenka, K., Dallachiesa, M., Palpanas, T.: Correlation-aware distance measures for data series. In: EDBT. pp. 502–505 (2017)
16. Mirylenka, K., Dallachiesa, M., Palpanas, T.: Data series similarity using correlation-aware measures. pp. 11:1–11:12. SSDBM '17 (2017)
17. Mirylenka, K., Miksovic, C., Scotton, P.: Applicability of latent dirichlet allocation for company modeling. In: Industrial Conference on Data Mining (ICDM'2016) (2016)
18. Mirylenka, K., Miksovic, C., Scotton, P.: Recurrent neural networks for modeling company-product time series. Proceedings of AALTD pp. 29–36 (2016)
19. Mirylenka, K., Palpanas, T., Cormode, G., Srivastava, D.: Finding interesting correlations with conditional heavy hitters. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE). pp. 1069–1080. IEEE (2013)
20. Mirylenka, K., Scotton, P., Miksovic, C., Dillon, J.: Hidden layer models for company representations and product recommendations. In: EDBT. pp. 468–476 (2019)
21. Monge, A.E., Elkan, C., et al.: The field matching problem: Algorithms and applications. In: KDD. vol. 2, pp. 267–270 (1996)
22. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. CoRR **abs/1802.05365** (2018), `http://arxiv.org/abs/1802.05365`
23. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language under-standing with unsupervised learning. Technical report, OpenAI (2018)
24. Su, Z., Ahn, B.R., Eom, K.Y., Kang, M.K., Kim, J.P., Kim, M.K.: Plagiarism detection using the levenshtein distance and smith-waterman algorithm. In: ICICIC'08. pp. 569–569. IEEE (2008)
25. Tata, S., Patel, J.M.: Estimating the selectivity of tf-idf based cosine similarity predicates. SIGMOD Rec. **36**(2), 7–12 (Jun 2007)
26. Uhl, A., Wild, P.: Enhancing iris matching using levenshtein distance with alignment constraints. In: International Symposium on Visual Computing. pp. 469–478. Springer (2010)
27. Wilson, D.R.: Beyond probabilistic record linkage: Using neural networks and complex features to improve genealogical record linkage. In: The 2011 International Joint Conference on Neural Networks. pp. 9–14 (2011)
28. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proceedings of the Section on Survey Research. pp. 354–359 (1990)