Measuring similarity for technical product descriptions with a character-level siamese neural network

Simone Falzone, Tobias Münster, Gabriele Gühring

***Abstract*—As part of the inventory process or when two companies merge and the enterprise resource planning system is taken over, databases with descriptions of technical parts are compared to get an overview of available parts. However, not all entries are created automatically, as the parts are recorded on site by employees and entered manually, with varying part IDs and descriptions. Therefore, it is of interest for a company to use a suitable method that is able to read technical documents and perform an automated comparison between employee information and the inventory data of an Enterprise Resource Planning system. We use a Manhattan Long Short-Term Memory network in order to measure sentence similarity between two entries of an Enterprise Resource Planning systems. The model is evaluated on industry data consisting of technical product descriptions. It outperforms models employing methods of substring analysis, classical machine learning methods as well as deep learning models with pre-trained word-embeddings.**

***Index Terms*—Siamese deep neural networks, Sentence similarity, Character-level, Technical language processing, Technical product descriptions**

I. Introduction

In reconciliation of inventory data all parts must be linked to the physical records. This requires that the digitally recorded items and their characteristics are matched with the items listed in the existing Enterprise Resource Planning (ERP) systems as part of the inventory. During the initial inventory of fixed parts, items are often recorded that are not clearly identified but already have part numbers entered in the ERP system. In some cases even two systems are fused and different part IDs or numbers have to be matched. The challenge in matching is to assign the corresponding data records in the ERP system to the articles found. In addition to tracking inventory data, it is also necessary for a company to keep track of the condition of all parts. The problem, as shown in [1], is that an inventory management system sometimes is fed with new data based on the input of data by maintainers related to equipment inspections, diagnostics and corrective actions. As a result, the data in a Computerized Maintenance Management System (CMMS) may contain unstructured raw text or inconsistent data with, e.g. missing part IDs. Most data in a CMMS can not be used for Natural Language Processing (NLP) diagnostics and analysis, as most out-of-the-box language processing pipelines are designed for non-technical language. In Table I we present examples of technical language description as it can be found in product description databases.

Table I
Exemplary descriptions of technical text

No.	Product Description
1	EE80251S1-000U-A99 - Sunon Sleeve Bearing Fan
2	18B11029-TA-1764-1 Tubeaxial Exhaust Fan 1Hp
3	Insulated Ferrule Single Wire 18 Awg

Furthermore, as described in [2], product matching is a crucial aspect for e-commerce portals that combine offers from multiple merchants to allow the user to find the best price for a specific product or efficiently find matching products from a variety of merchants.

In order to analyse technical textual data, such as inventory descriptions, the scientific community has successfully applied Natural Language Processing to its text-based technical data such as an affinity propagation clustering [3], a combination of bidirectional Long Short-Term Memory (LSTM) and conditional random field (CRF) for the task of named entity recognition [4], extracting information from clinical text data using the UMLS MetaMap Transfer (MMTx) application and a negation detection algorithm called NegEx [5].
We present in this paper a deep learning model especially useful for technical data consisting of part numbers, part IDs, units etc., which circumvents common pitfalls by learning important features of the input data, as for example shown in Table I or in Table VI.

Simone Falzone, Tobias Münster, and Gabriele Gühring Gabriele.Guehring@hs-esslingen.de. Faculty Computer Science and Engineering, University of Applied Sciences Esslingen, 73732 Esslingen, Germany.

II. Related Work

In NLP there are several methods to determine the similarity of words or sentences. A string based approach is presented in [6]. The Basic Local Alignment Search Tool (BLAST) algorithm used there examines two protein or gene sequences for their similarity. This approach can be adapted to regular or technical sentences. It is a dynamic programming approach which compares sentences at the character level. Equality is defined by the identical sequence of local and global characters in two sentences, with unequal characters being deleted from the sequence.

The approach of [2] uses descriptive statistics for technical product matching from which they combine three metrics in one sum, which additionally can be weighted by the hyperparameters α, β and γ. The Term Frequency-Inverse Document Frequency (TFIDF) is used for describing how often a certain word occurs in a text. The second metric is the Term Partition Relevance (TPR) and describes in which text partition a word occurs most frequently. Finally, the value of the Term Type Partition Relevance (TTPR) is added, which describes the probability of occurrence e.g. of a certain product category in the different text partitions.

In [7] an unsupervised learning algorithm is proposed to solve the problem of matching product titles based on the morphological analysis of the titles of the products. They present an algorithm that operates in two phases. In the first phase combinations of the words of the titles are computed and several statistic measures are recorded. In the second phase we used the gathered statistical information to assign a score to each computed combination. The combination with the highest score is then declared as label of the cluster which contains each product.

The field of NLP has received much public attention due to the successes of language models such as BERT [8], T5 [9] or GPT-3 [10]. One area of NLP is *extractive question-answering*, which can be viewed as a special case of text classification. Given a interrogative sentence and multiple candidate answers, the task is to classify each candidate answer as correct or not [11]. Contrary to [7] our approach is a supervised approach since we need a training data set to train our character embedding. However, it is possible to use our model on different problems and so as in [8] to implement a pretrained character embedding.

Common approaches for computing a semantic embedding of words are GloVe [12] and Word2Vec [13], which create embeddings based on their context and frequency of co-occurence in a text corpus. The idea is to create a vector space in which the position of a word describes how (un-)similar it is to its surrounding words. In our proposed model we do not use a pretrained word embedding, instead the embedding process is done within the training phase.

As for comparing or matching two sentences with each other, given their word or character-level vector representation, siamese neural networks and their deep neural network variants are used [11]. This architecture is utilized in [14], where a similarity metric on variable length character sequences is learned. The model combines multiple character-level bidirectional LSTMs in a siamese architecture. It learns to project variable length sequences into a fixed dimensional embedding space by using information about the similarity between pairs of strings.

In our work we add several convolutional layers to a siamese Manhattan-LSTM (MLSTM) for measuring sentence similarity. Our temporal convolutional neural network is inspired by the architecture of [15], which learns abstract knowledge about words and sentences on the basis of individual characters and generates a word embedding from this. Furthermore, our architecture is inspired by the siamese MLSTM of [16], which reads in two sentences given their word embedding representation using a siamese LSTM and generates a context between the words. From this LSTM representation of the sentences a similarity score is calculated using a similarity layer employing the L1-Norm. We evaluate our model against the established BERT language model with a WordPiece Model (WPM) as presented in [17]. With a WPM, the vocabulary also contains partial words, frequent endings and single numbers and letters. This makes it possible to divide unknown words into smaller known words. Thus, word embeddings can be created for unseen words.

III. Similarity based on WordPiece Model

The comparison of sentences based on their similarity is a well-known problem from the field of Natural Language Processing for which several pre-trained models are already available. One of the breakthroughs in the field of NLP is published in [8] presenting a model called Bidirectional Encoder Representations from Transformers (BERT). We try to establish a baseline with the BERT model and check if, thanks to the WPM used, sentences consisting of non-natural language, i.e. the names and IDs of (electronic) parts in an inventory process, can also be successfully compared.

We use a German BERT-model which is published on Hugging Face[1] and presented in [18], because in our first dataset, presented in section VI-A, the names, IDs or descriptions contain, if at all, mostly German expressions. The model is trained on German Wikipedia entries, the OpenLegalData dump [19] and several German news articles.

Our main goal is comparing two IDs, numbers and names of inventory parts in order to see if they are identical. For the comparison of the two parts, the most important attributes of labeled machine parts are extracted from the data. These include the part number, which uniquely identifies a component, the type of the part which is described in more detail in section VI-A and a short description as well as the manufacturer name. Subsequently, an embedding is created for each attribute, this is done for both parts that are compared with each other. The similarity of two technically described parts is calculated using cosine similarity like in [20] by comparing a word

[1]https://huggingface.co/transformers/pretrained_models.html

Table II
ACCURACY WITH PRE-TRAINED GERMAN BERT-MODEL (DEPENDING ON TRESHOLD T)

T	**Accuracy**
85	43.912
90	70.220

embedding of each attribute of the first part with the corresponding word embedding of the second part. To obtain the dependency of all attributes, the results of the obtained similarity values are averaged over all attributes.

We have defined two different threshold values for the evaluation, denoted by T. If the calculated similarity of the two parts is greater than or equal to T, they are classified as equal, otherwise as unequal. The $accuracy$ is then calculated for the classified components as in equation (3) section VI-D. The selected values for T and the achieved $accuracy$ is shown in Table II.

The results in Table 2 show that the pre-trained BERT model does not perform well for our use case of the comparison of a technical description of parts. We therefore pursue in the following sections another approach based on generating an intrinsic word embedding with a Convolutional Neural Network (CNN) which leads to higher $accuracy$ on our training and test data set.

IV. Data Augmentation for Text Based Data Sets

We first use data augmentation for training a more robust model. Data augmentation, commonly used in image processing, encompasses multiple techniques that enhance the size and quality of training datasets such that better Deep Learning models can be built. It acts as a regularizer and helps reduce overfitting when training a machine learning model [21]. Further, we assume that by increasing the amount of text artifacts, such as missing word/letter or swapped words, the robustness and reliability of our model is improved.
To do this, we use aspects of the Easy Data Augmentation (EDA) suite as in [22], which demonstrates performance improvements of NLP models by using augmentation techniques that are loosely subjected to those used in computer vision. The following section describes the EDA suite briefly, when it is applied to real word non-technical sentences.

Each sentence in a database is given the opportunity to be selected for augmentation using one of the following operations, which are chosen at random. Further, synonyms are found in text by employing a dictionary. The synonyms are then processed differently according to the selected operation.

1) **Synonym Replacement (SR)**: Select N_1 words randomly from the sentence that are not stop words. Replace each of these words with a randomly chosen synonym.
2) **Random Insertion (RI)**: Find a random synonym for a non-stop word in the sentence. Insert that synonym into a random position in the sentence. Do this N_2 times.
3) **Random Swap (RS)**: Choose two words in the sentence at random and swap their positions. Do this N_3 times.
4) **Random Deletion (RD)**: Delete each word in the sentence with probability p_{RD}.

Additionally, an individual parameter α_i, for $i = 1 \dots 3$ is calculated for the operations SR, RI and RS, by $\alpha_i = \frac{N_i}{L}$, where L denotes the length of the sentence. Each α_i denotes the amount of words in percent which shall be subjugated to its respective operation.

N_i may vary, depending on the α_i value for its respective operation. This is due to the assumption, that longer sentences can cope easier with manipulation before losing their assigned label. Finally a parameter n_{aug} can be chosen by the user, which indicates how many sentences are to be generated from the original one.
We basically follow the EDA implementation, except that in our algorithm we no longer rely on an English dictionary to identify words. Instead, we split the string by the whitespace character and call each sequence of characters a word. Consequently, we can not and do not use the functionality SR or RI in our setup. As for the parameters α_3, which denotes the amount of characters subject to the RS operation in percent, and p_{RD}, which denotes the probability of an individual character being subject to the RD operation in percent, we select the value of 0.2 for both parameters and set n_{aug} to 9. These parameters are inspired by [22] as reasonable performance gains are to be expected.

V. Siamese CNN-MLSTM

In this section we introduce our architecture for measuring whether two inputs of technical product description data classify the same product. The architecture is based on the character-level convolutional neural network presented in [15] for learning important features of the input sentences and the siamese LSTM shown in [16] for finding sequential patterns and calculating a similarity value. Since we are evaluating whether two product descriptions match to the same product we measure the success of our model by calculating an accuracy or an F-$score$ as in [23] or [24], although we really are interested about a statement of how similar two product descriptions are.

A. Model Architecture

The architecture of the proposed siamese CNN-MLSTM is shown in Fig. 1. The model takes two

Table III
CONVOLUTION LAYERS USED IN OUR ARCHITECTURE. NO LAYER USES STRIDE AND ALL POOLING LAYERS ARE NON-OVERLAPPING.

Layer	**Conv1D Filters**	**Conv1D Kernel Size**	**MaxPooling1D Size**
1	256	7	3
2	256	7	3
3	256	3	N/A

input texts, here named *Sentence 1* and *Sentence 2*, and outputs the probability that both inputs describe the same part. Both sentences are passed into the model in the form of a character embedding matrix. Our siamese CNN-MLSTM consists of several sequential layers: (a) convolution stacked with max-pooling layers, (b) feed forward LSTM, (c) similarity layer and (d) a sigmoid classification layer. Given the input sentences, the convolutional layers learn a specific feature representation of the sentences, which are down-sampled for the maximum presence of a feature, which is then passed to the LSTM layer. The LSTM layer operates on the feature map of the last convolution layer and in turn outputs an internal feature representation from the last LSTM cell, which encodes the sequential patterns. We follow the assumption of [16] that the hidden-state values of the last LSTM cell encode its respective input sentence, all of which is therefore passed into a similarity layer. The similarity layer applies the L1-Norm to both LSTM cells hidden-state values from each input sentence. Its output is finally passed into a sigmoid layer for the classification of whether the input sentences are similar or not. More details about each of the layers are shared in the following subsections.

B. *Sentence Encoding*

At the beginning, each character of the input sentences of variable length L is converted to lower case and encoded. Here we denote by *sentence* the concatenation of the part description consisting of IDs, names, types, manufacturer, etc. The encoding is done with a fixed alphabet of length m for the language defined here. The sentences are then encoded by a 1-out-of-m encoding (1-hot encoding). The sentence is first divided into an array of individual tokens (characters). Then each token is replaced by its corresponding vector of length m. The result is an array of length L consisting of vectors of length m, which make up the character $m \times L$ embedding matrix mentioned in section V-A. If a sentence exceeds the length defined by $L = 70$, all characters exceeding the limit are ignored. If a sentence is shorter than L characters, it is padded to length L by adding the zero vector. In addition, an UNK (Unknown) token is introduced to avoid equating an out-of-vocabulary error with padding. Therefore our alphabet, minus the UNK token is given by the following characters:

abcdefghijklmnopqrstuvwxyz0123456789-,;.!?:"'/
\{ }|_@#$%^&*~'+-=<>()[]

C. *Learning Sentence Representation*

At the core of the model is a CNN structure, which operates on a character-level input, but outputs a word-vector representation. The idea of the temporal 1D-Convolution is that through its filters a character-n-gram is created. Through stacking convolution operations and clever use of max-pooling, an understanding of word structures/patterns is learned which is specific to the task. The architecture consists of a Conv1D-MaxPooling1D sequence which is described in Table III. The initial weights and biases of the Conv1D elements are normally distributed with the parameters (0.0, 0.05) for mean and standard deviation. The structure of the CNN follows loosely the one presented in [15], but deviating in size of the Conv1D-MaxPooling1D sequence.

D. *Sequential Patterns and Sentence Similarity*

Subsequently, the resulting feature map of the final convolution layer with the dimensions $d_{in} = 3 \times 256$ is passed to the LSTM layer, which now learns a mapping from dimension d_{in} to dimension $d_{rep} = 64$. With d_{rep} being the dimension of the final vector representation of the sentence, encoded by the last hidden state of the LSTM layer [16]. Finally the LSTM representations of the left and right input sentences are compared by a predefined similarity function $g : \mathbb{R}^{d_{rep}} \times \mathbb{R}^{d_{rep}} \rightarrow \mathbb{R}$ (based on the L1-Norm). Let h_{64}^{left} be the hidden-state representation of the last LSTM cell of the left input and h_{64}^{right} of the right input. With $||x||_1$ denoting the L1-Norm, the similarity is computed as follows

$$g(h_{64}^{left}, h_{64}^{right}) = exp(-||h_{64}^{left} - h_{64}^{right}||_1) \qquad (1)$$

The final layer of our architecture is a single sigmoid node, which learns the probability of two inputs being similar or not.

VI. EXPERIMENTAL ANALYSIS

Our model for mapping inventory datasets is trained on two datasets, both describing technical parts or products. First, we use a proprietary dataset provided by a company dealing with machine parts. The second is a publicly available dataset from Kaggle[2] that is similar to the first dataset but consists of descriptions of electrical devices such as smartphones or television. The data is also used in [7] for the matching of product titles in order to compare parts retrieved by a user or marketer by an online search to compare similar products. This is comparable to our use case where we try to compare parts from different data sources,

[2]https://www.kaggle.com/lakritidis/product-clustering-matching-classification

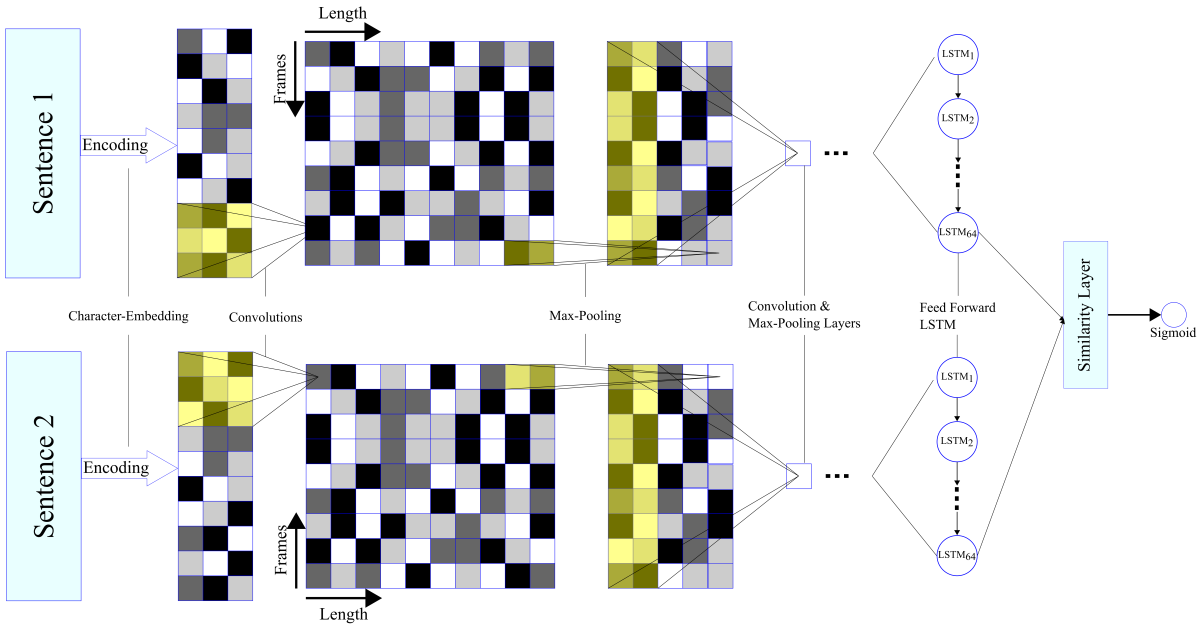

Figure 1. Illustration of our model architecture

Table IV
EXAMPLE OF TWO IDENTICAL PARTS WITH DIFFERENT DESCRIPTION

Sentence 1	Sentence 2
MLV12-54-**G32/124** Lichtschranke, Reflexion; RW	208637 **Pepperl + Fuchs** Lichtschranke MLV12-54-**G/32/124** 208637 10 - 30 1 St.

Table V
ATTRIBUTES OF BOTH SENTENCES 1 AND 2

Attribute	Description
Part number	Manufacturer part number
Type	Addition to the manufacturer part number to make the part number unique
Description	Contains the title and a description of the part

e.g. from different ERP systems of a company and determine whether both parts are the same or not. Both data sets contain approximately 80.000 records.

A. Proprietary dataset

Since our main goal is to match captured parts with existing parts in an ERP system, i. e. as part of an inventory, we have two parts of datasets. First the dataset consisting of the entries in the ERP system, which we consider as the single source of truth. In the following the description of these parts will be referenced as *Sentence 1*. Second there is a list of parts which are labeled differently either because they are manually labeled or because they belong to another system. They contain slight deviations in the description or part number, these parts are referred to as *Sentence 2*. It is for example quite often the case that when entering a component, the part number is entered with spaces instead of hyphens or the product name is prefixed e.g. with the manufacturer brand. Also it possible that in some cases the description is used to add a detailed product description in natural language, which leads to a significant fluctuation in the length of the sentences. An example of two matching parts with a slightly different descriptions is shown in Table IV

For the training dataset we label the Sentence 2 dataset with either 0 or 1, depending on whether the corresponding part matches the entry in the ERP system. Table V shows the attributes of one part and a short description of the attribute.

In order to use the data for our use case it is necessary to transform it into Sentences 1 and 2. Therefore the individual attributes from Table V are concatenated divided by blanks, starting with a part number and ending with a description. The maximum length of a sentence consists of 70 characters, if a sentence is longer it is truncated as described in section V-B. Table VI shows some example records of the resulting dataset.

The data exhibits a strong skewness as there are many more records of non matching parts than there are for matching parts. This means the target $y = 1$ is subrepresented, which could lead to wrong assumptions by the model. Therefore the data is augmented as described in section IV so both values for y are equally distributed.

Table VI
EXAMPLE RECORDS OF TRAINING DATASET FOR PROPRIETARY DATASET

No.	Sentence 1	Sentence 2	Label y
1	4923+14640 A40.891.00 Filter 1020x950x20mm frame	26-4923 Burner Cover - Southbend, SOU1182778	0
2	TIF15.AUXAAC-22596 Touch LED 15 TFT Display	IQ AUTOMATION Flatman TFT Display, Model: FS170A4GSDDH3	0
3	MLV12-54-G32/124 Lichtschranke, Reflexion; RW	208637 Pepperl + Fuchs Lichtschranke MLV12-54-G/32/124 208637 10 - 30 1 St.	1

B. Kaggle Product Dataset

The second dataset we use for training is a publicly provided dataset on Kaggle[3] titled "Product Clustering, Matching & Classification" [7], which we call in the following Kaggle Product Dataset. This dataset is very similar to the proprietary one. It consists of a product title, which already contains all attributes like a part number, the type and a short description of the part. The different parts are always assigned to a certain cluster, which we use here as a single source of truth and corresponds to the entries from the ERP system from section VI-A. Table VII shows a few example records of the Kaggle Product Dataset with the relevant attributes.

The training dataset is constructed by using the cluster label as Sentence 1 and the product title as Sentence 2. Table VII shows sample data records of the Kaggle Product Dataset. The rows of Table VII show an example data record of the Kaggle Product Dataset. Here the product title and the cluster label are describing the same part. Since all parts in the data set are assigned to the correct cluster label, the entries are labelled with $y = 1$. To obtain also non matching records the data is adjusted in a preprocessing step so that a random product is assigned a wrong cluster label and gets the target value $y = 0$, e.g. No. 2 in Table VIII. This gives us a very similar data set as listed in Table VI which compares well with the proprietary data.

C. Model Training

We train the parameters of the model with the objective of maximizing its prediction $accuracy$ given the target labels in a training dataset. For each of our two datasets an independent model is trained and evaluated with a 5-fold cross-validation. The hyperparameters are determined by a simple grid search, choosing the

[3]https://www.kaggle.com/lakritidis/product-clustering-matching-classification

Table VII
EXAMPLE RECORDS OF KAGGLE PRODUCT DATASET

No.	Product Title	Cluster Label
1	bosch serie 6 built under freezer in white	Bosch GUD15A50GB Integrated
2	lec cf61lw 60 litre chest freezer white a rated	Lec CF61LW White
3	canon ixus 185 digital camera silver	Canon IXUS 185

Table VIII
EXAMPLE RECORDS OF KAGGLE PRODUCT DATASET

No.	Sentence 1	Sentence 2	Label y
1	bosch serie 6 built under freezer in white	Bosch GUD15A50GB Integrated	1
2	lec cf61lw 60 litre chest freezer white a rated	Bosch GUD15A50GB Integrated	0

ones that give the best results. We used the Binary Cross-Entropy (BCE) as our loss function. Let $\hat{y}$ be the predicted label and y be the true label then the BCE is calculated as:

$$BCE(\hat{y}, y) = -(y \cdot log(\hat{y}) + (1-y) \cdot log(1-\hat{y})) \quad (2)$$

Finally, the algorithm used for training our model is stochastic gradient descent (SGD) with a minibatch size of 32 and using momentum as in [26], [27]. Our implementation is done using KERAS (TensorFlow 2) [28].

D. Experimental Analysis

Our model is evaluated with a 5-fold stratified cross-validation and $accuracy$ is used as metric. Let TP be the true positives and let TN be the true negatives [29], then the $accuracy$ is calculated as.

$$accuracy = \frac{TP + TN}{N}, \quad (3)$$

where N denotes the total count of all records in the dataset [30].

Furthermore, the recall and precision are calculated in order to calculate the $F\text{-}score$. Recall is the fraction of true events and precision is the fraction of detections reported by the model that are correct [30]. Let TN be the false negatives and let FP denote the false positives [29], then the recall and precision are computed as shown in equation (4) and (5).

$$Recall = \frac{TN}{TN + FP} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

Table IX
ACCURACY AND F-SCORE OF OUR SIAMESE CNN-MLSTM

Dataset	Optimizer	Learning-rate	Momentum	Avg. Loss	Avg. *accuracy*	Avg. *F-score*
Proprietary	SGD	0.1	0.9	0.172	**94.837**	**0.949**
Proprietary	ADAM	0.001	0.9	0.232	92.760	0.929
Kaggle	SGD	0.1	0.9	0.185	93.797	0.937
Kaggle	ADAM	0.001	0.9	0.249	90.944	0.909

Table X
F-SCORES FROM RELATED WORKS

Paper	Method	Dataset	*F-score*
[2]	Combining several methods from descriptive statistics	Not available	0.5 - 0.67
[25]	Uses tailored approaches for product matching based on a preprocessing of product offers to extract and clean new attributes usable for matching.	Not available	$\sim$ **0.4 - 0.69**
[7]	Morphological analysis of the titles of the products.	Kaggle Product Dataset	0.66

The F-$score$ is then finally calculated as [30]:

$$F\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{6}$$

The results of the 5-fold cross-validation are averaged and can be found in the Table IX.

In each of our model evaluations, we achieve a value above 90% for both Precision and Recall. Several papers deal with similar use cases as we do. As can be seen from Table X our model outperforms all F-$scores$ achieved by the others. The best comparison is between our results and those of [7], as they are working on the same dataset.

VII. CONCLUSION

In this paper we introduce a new neural network architecture for matching products or product parts with a technical description. Even today, matching product descriptions is a difficult problem in machine learning as also seen in [7] and [25]. The benefit of automated and reliable product matching is not only practically relevant for the matching of manually recorded parts with existing parts in an ERP system it also can be used in online shopping, since it allows the users to compare products from various suppliers. The traditional string similarity metrics do not perform well in product matching as shown in [2]. We also show that pre-trained WordPiece models are not suitable to process technical texts like product descriptions. The reason for this is that the models are trained on a text corpus consisting of natural language texts. In our experiment with a WordPiece model we reached an accuracy of about 70%.

However, it could be successfully shown that a neural network is able to build up its own understanding of word patterns without pre-computed statistics about word distributions. The architecture of the developed model, shown in section V-A, is a new approach for evaluating sentence similarities at a character level. Our model achieved an accuracy of up to 94.8% and an F-$score$ of up to 93% when working with a simple forward LSTM in the Siamese CNN-MLSTM.

REFERENCES

[1] M. P. Brundage, T. Sexton, M. Hodkiewicz, A. Dima, and S. Lukens, "Technical language processing: Unlocking maintenance knowledge," *Manufacturing Letters*, vol. 27, pp. 42–46, 2021.

[2] A. Thor, "Toward an adaptive string similarity measure for matching product offers," in *INFORMATIK 2010. Service Science ? Neue Perspektiven für die Informatik. Band 1*, K.-P. Fähnrich and B. Franczyk, Eds. Bonn: Gesellschaft für Informatik e.V, 2010, pp. 702–710.

[3] X. Chen, H. Xie, F. L. Wang, Z. Liu, J. Xu, and T. Hao, "A bibliometric analysis of natural language processing in medical research," *BMC medical informatics and decision making*, vol. 18, no. Suppl 1, p. 14, 2018.

[4] M. Gridach, "Character-level neural network for biomedical named entity recognition," *Journal of biomedical informatics*, vol. 70, pp. 85–91, 2017.

[5] S. Meystre and P. J. Haug, "Natural language processing to extract medical problems from electronic clinical documents: performance evaluation," *Journal of biomedical informatics*, vol. 39, no. 6, pp. 589–599, 2006.

[6] W. R. Pearson, "An introduction to sequence similarity homology searching," *Current protocols in bioinformatics*, vol. Chapter 3, 2013.

[7] L. Akritidis and P. Bozanis, "Effective Unsupervised Matching of Product Titles with k-Combinations and Permutations," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*. Thessaloniki: IEEE, Jul. 2018, pp. 1–10.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html.

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners." [Online]. Available: https://arxiv.org/pdf/2005.14165.

[11] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–40, 2021.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, "Glove: Global vectors for word representation," 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/pdf/1301.3781.

[14] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru, "Learning text similarity with siamese recurrent networks," *Proceedings of the 1st Workshop on Representation Learning for NLP*, pp. 148–157, 2016.

[15] X. Zhang and Y. LeCun, "Text understanding from scratch." [Online]. Available: https://arxiv.org/pdf/1502.01710.

[16] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI?16. AAAI Press, 2016, pp. 2786–2792.

[17] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Å. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation." [Online]. Available: https://arxiv.org/pdf/1609.08144.

[18] B. Chan, S. Schweter, and T. Möller, "German's Next Language Model," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 6788–6796.

[19] M. Ostendorff, T. Blume, and S. Ostendorff, "Towards an Open Platform for Legal Information," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*. Virtual Event China: ACM, Aug. 2020, pp. 385–388.

[20] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *arXiv:1908.10084 [cs]*, Aug. 2019.

[21] Connor Shorten and Taghi M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[22] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks." [Online]. Available: https://arxiv.org/pdf/1901.11196.

[23] D. Giampiccolo, H. T. Dang, B. Magnini, I. Dagan, E. Caprio, and B. Dolan, "The fourth pascal recognizing textual entailment challenge," in *TAC 2008*. Citeseer, 2008, p. 545.

[24] S. Sultana and I. Biskri, "Identifying similar senteces by using n-grams of characters," in *Recent Trends and Future Technology in Applied Intelligence*. Cham: Springer, 2018, pp. 833–843.

[25] H. Köpcke, A. Thor, S. Thomas, and E. Rahm, "Tailoring entity resolution for matching product offers," in *Proceedings of the 15th International Conference on Extending Database Technology - EDBT '12*. Berlin, Germany: ACM Press, 2012, p. 545.

[26] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[27] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning," *International Conference on Machine Learning*, pp. 1139–1147, 2013.

[28] F. Chollet *et al.*, "Keras," 2015.

[29] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning*. New York ; London: Springer, 2010.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press, 2016.

Simone Falzone received his B.Sc. degree in computer science from the Hochschule für Technik Stuttgart in 2020. Since 2020, he is pursuing his Master's degree at Hochschule Esslingen. He is currently working on projects using deep learning for text analytics.

Tobias Münster received his B.Eng. degree in technical computer science from the University Esslingen. In 2021 he obtained his M.Sc. degree in applied computer science from the University Esslingen. Since 2022 he is working in the field of automated driving research.

Gabriele Gühring studied mathematics and physics at the University of Tübingen. After completing her Ph.D. in mathematics she has advised banks and industrial companies throughout Germany in risk controlling, internal models and the valuation of financial derivatives and has supported the implementation of trading systems. She is a professor at Esslingen University of Applied Sciences since 2008 and gives lectures in the subjects of mathematics, statistics and data analytics. Her current publications in the field of machine learning deal with anomaly detection in time series and multimodal models of deep learning.