Embedded Attributes for Cuneiform Sign Spotting

Eugen Rusakov[1] (✉)[0000−0002−3801−573X],
Turna Somel[2][0000−0003−2084−4967],
Gerfrid G.W. Müller[2][0000−0003−2466−367X],
and Gernot A. Fink[1][0000−0002−7446−7813]

[1] TU Dortmund University, Department of Computer Science,
44227 Dortmund, Germany
{eugen.rusakov, gernot.fink}@tu-dortmund.de
[2] Hittitology Archive, Academy of Science and Literature Mainz,
55131 Mainz, Germany
{turna.somel, gerfrid.mueller}@adwmainz.de

Abstract. In the document analysis community, intermediate representations based on binary attributes are used to perform retrieval tasks or recognize unseen categories. These visual attributes representing high-level semantics continually achieve state-of-the-art results, especially for the task of word spotting. While spotting tasks are mainly performed on Latin or Arabic scripts, the cuneiform writing system is still a less well-known domain for the document analysis community. In contrast to the Latin alphabet, the cuneiform writing system consists of many different signs written by pressing a wedge stylus into moist clay tablets. Cuneiform signs are defined by different constellations and relative positions of wedge impressions, which can be exploited to define sign representations based on visual attributes. A promising approach of representing cuneiform sign using visual attributes is based on the so-called Gottstein-System. Here, cuneiform signs are described by counting the wedge types from a holistic perspective without any spatial information for wedge positions within a sign. We extend this holistic representation by a spatial pyramid approach with a more fine-grained description of cuneiform signs. In this way, the proposed representation is capable of describing a single sign in a more detailed way and represent a more extensive set of sign categories.

Keywords: Cuneiform Script · Retrieval · Visual Attributes.

1 Introduction

Besides Egyptian hieroglyphs, the cuneiform script is one of the earliest attested writing systems in history. Developed in Mesopotamia at the end of the 4th millennium BCE, it was used until the 1st century CE across the Near East.

Cuneiform is a 3D script written primarily on clay tablets. It was written by pressing a stylus into moist clay surfaces in order to create signs composed of wedge-shaped impressions, henceforth to be called wedges. Figure 1 shows an

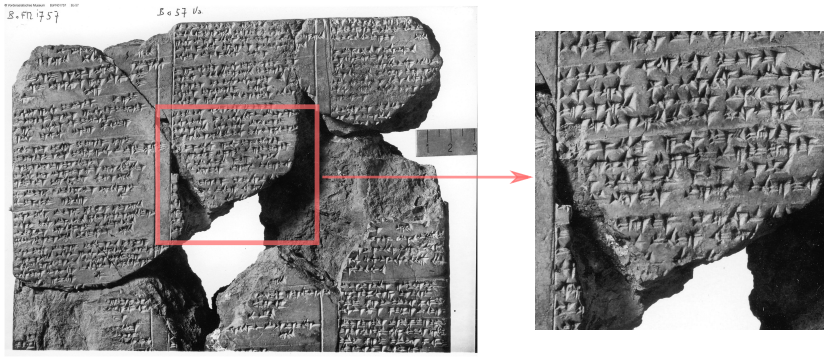

Fig. 1. The left side shows a whole tablet reassembled from several fragments. The right side shows an enlarged section from the tablet. This photograph has been provided [11].

example of a cuneiform tablet. The image on the left-hand side shows a typical tablet that has been reassembled by philologists by joining fragments in an attempt to restore heavy damage through the millennia. Displayed on the right side of Figure 1 is a section of the left image where one can see the wedge impressions on the clay surface. A sign can be identified by the reader according to wedge direction, number, and positioning within the sign. As cuneiform signs may have syllabic or logographic values, the writing system contains a significantly greater number of signs than in an alphabet, with an approximate maximum of 900 signs. As this writing system was used over three millennia, it was distributed across a wide geographical range, adapted to over 10 languages. Hence, the inventory of regular signs and the variations observed within a single sign class, henceforth to be called sign variants, used in different local areas are heterogeneous. A single variant may differ from other variants in a variety of sign characteristics, namely the direction, number, and positioning of wedges. As presented in [16], a system proposed by Gottstein [5] for representing cuneiform signs based on four different wedge types and their corresponding counts fit well for a novel retrieval scenario called *Query-by-eXpression (QbX)*. However, the binary attribute vectors derived from the Gottstein-System have a limited representational power in terms of inventory size. In this work, we derive a representation based on the Gottstein-System enriched with spatial information, which encodes the presence of wedge types and counts and their approximate position within a cuneiform sign. In this way, the signs are projected into a subspace capable of representing a more extensive inventory size. We evaluate the sign representations using annotated texts written in Hittite cuneiform script in a

segmentation-based *Query-by-Example (QbE)* and *Query-by-eXpression (QbX)* scenario and show the superiority of a Gottstein representation enriched with spatial information. This work is organized as follows. In the next section, related work is reviewed, followed by a brief introduction to the Gottstein-System in section 3.1. The resulting attribute representation based on a pyramidal decomposition of cuneiform signs is explained in section 3.2. The experiments are discussed in Section 4. Finally, a conclusion is drawn.

2 Related Work

In this section, related work in terms of cuneiform analysis and word embeddings is discussed. At first, we review the current publications in *cuneiform character recognition and retrieval*. Afterward, we discuss recently proposed *word embeddings* for word spotting.

2.1 Cuneiform Character Retrieval

Although the cuneiform writing system is still a less known domain in the document analysis community, several approaches based on graph representations and bag-of-features for cuneiform sign retrieval have been presented. In 2012, a method [12] for extracting 2D vector drawings from 3D cuneiform scans was proposed. Afterward, the resulting spline graphs were converted in order to compare the similarity between two cuneiform characters based on graph representations [3] or retrieve them using pattern matching on structural features [2]. With a similar goal, Massa et al. [13] described an approach to extract wedge-shaped impressions from hand-drawn cuneiform tablets. In [4] the *inkball model* [9] was adapted by Bogacz et al. to treat different wedge features as individual parts arranged in a tree structure in order to perform segmentation-free cuneiform character spotting. Next to the graph-based feature representations, Rothacker et al. [14] proposed an approach based on *Bag-of-Features* in combination with *Hidden Markov Models* integrated into a patch-based framework for segmentation-free cuneiform sign spotting.

2.2 Word Embeddings

All the approaches mentioned above require a visual example of a query and can only perform in a retrieval scenario called *Query-by-Example (QbE)*. Sudholt et al. [19] noted that this approach poses certain limitations in practical applications. A user has to identify a visual example of a query from a whole document collection which can be very tedious, especially for infrequent queries. The field of word spotting is dominated by mainly two types of spotting scenarios, namely *Query-by-Example (QbE)* and *Query-by-String (QbS)*. The *QbS* scenario is defined by queries given as word strings. Here, the word image representations and the textual representations are projected into a common subspace. The major

advantage of *QbS* is that the user is not required to search for a visual example of a word in the document collection to define a query. On the other hand, a word spotting system must learn a mapping from a textual to a visual representation first. Therefore, annotated training samples are required. Almazan et al. [1] proposed an approach for a word string embedding called *Pyramidal Histogram of Characters (PHOC)*. The authors presented a method to map a word string to a binary attribute vector representing the presence or absence of characters. Furthermore, information about character positions within a word is included by splitting the word string in a pyramidal way. Inspired by the PHOC representation, Sudholt et al. [19] proposed a CNN model based on the *VGGNet* [17] named *PHOCNet*. This model is capable of estimating a certain PHOC representation in an end-to-end fashion using *binary logistic regression* (also called *sigmoid logistic regression*) in combination with a *binary cross-entropy (BCE)* loss.

3 Cuneiform Sign Expressions

This section describes our approach of extending the representation of cuneiform signs following the Gottstein-System by including spatial information. After a brief introduction of the Gottstein-System in 3.1, the structure of spatial information encoded in our pyramidal representation is introduced in 3.2. Afterward, we explain how the sign expressions enriched with spatial information are modeled as binary attribute vectors.

3.1 Gottstein System

The Gottstein-System [5] was introduced following the idea of a unified cuneiform sign description based on commonly occur wedge types. This representation should be usable in databases without existing knowledge of cuneiform sign categories, especially in a retrieval scenario. In order to simplify the search process, Gottstein [5] proposed a system where an alphanumeric encoding represents a cuneiform sign. The Gottstein-System decompose a cuneiform sign into four

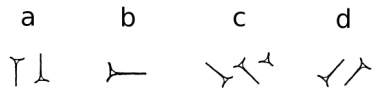

Fig. 2. Four hand drawn examples of wedge types from the Gottstein-System and their corresponding alphanumeric encodings $gs = \{a, b, c, d\}$

different wedge types $gs = \{a, b, c, d\}$, where gs denotes the set of Gottstein wedge types. Figure 2 shows four examples of hand-drawn wedge types from the Gottstein-System. For every item from gs the count of each wedge type is given.

The type *a* represents a vertical wedge tending from top to bottom and vice-versa. *b* represents a horizontal wedge type, tending from left to right. Here, the variant right to left does not exist. Type *c* represents three different variants of a wedge type, the so-called *Winkelhaken* and an oblique wedge tending diagonally from the upper left to the lower right as well as from lower right to the upper left. Finally, *d* represents a perpendicular wedge type to *c* tending from the lower left to the upper right direction and vice-versa. Figure 3 shows three cuneiform

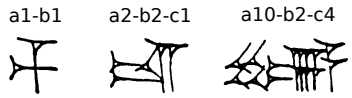

Fig. 3. Three examples of hand drawn cuneiform signs and their corresponding Gottstein representations, showing a simple sign on the left side and a more complex sign on the right side.

signs and their corresponding Gottstein representations. On the far left side, a simple sign consisting of 2 wedge types is shown. The sign in the middle shows three different wedge types and $2 + 2 + 1 = 5$ wedges in total. The far-right side shows a more complex sign with 3 different wedge types and $10 + 2 + 4 = 16$ wedges in total.

3.2 Gottstein with Spatial Information

Although the Gottstein system provides a holistic representation to express a particular cuneiform sign, this simplified description exhibits some shortcomings. For example, distinguishing between different sign categories consisting of the same wedge types, additional information for the relative position of a wedge is missing. Figure 4 shows the shortcoming of a holistic Gottstein representation.

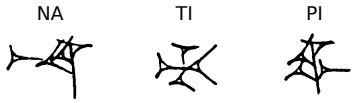

Fig. 4. Three examples of different cuneiform signs consist of the same wedge types described by the same Gottstein representation a1-b1-c2-d0.

Here, three different cuneiform signs are described by the same Gottstein representation. The three signs, from three different categories having different visual appearances, are mapped to the same alphanumeric encoding. In order to overcome this ambiguous mapping, we decompose the signs in a pyramidal fashion. This way, a higher diversity in the mapping between categories and encodings is achieved.

Pyramidal Partitioning Intuitively, writing systems are partitioned according to their reading directions like left to right in Latin scripts. The common reading direction for cuneiform texts tends from left to right and less common from top to bottom. Due to the physical process of writing cuneiform signs by pressing a wedge stylus into moist clay, a typical sequential arrangement of wedge types and positions, similar to Latin scripts, is missing. Therefore, we split the signs based on their main visual variant in a pyramidal fashion to get a Gottstein representation containing spatial information that represents cuneiform signs in a more detailed way. As the reading direction tends from left to right, horizontal partitioning is performed. We apply the splitting at level 2 and 3 which results in 2 left-right and 3 left-middle-right splits, respectively. Additional to the horizontal direction, vertical partitioning on levels 2 and 3 are applied. Figure 5

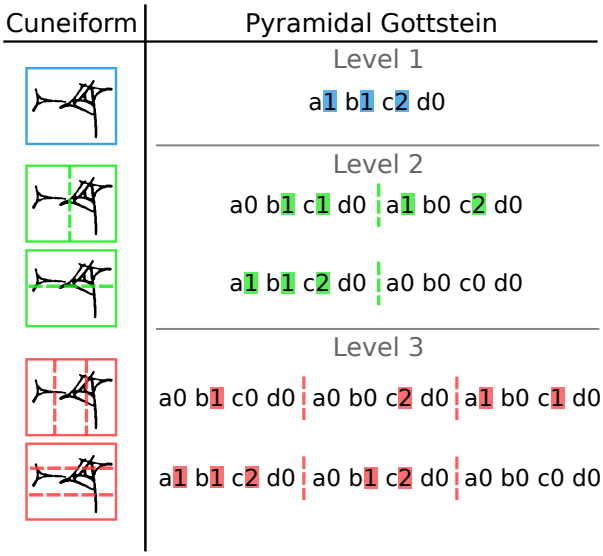

Fig. 5. Visualization of splitting the sign NA in the horizontal and the vertical direction for levels 2 and 3, respectively.

shows the proposed way of splitting the cuneiform sign *NA* in a pyramidal fashion. Assigning annotations to splits based on the visual appearance entails an issue. While most of the wedges can be assigned distinctively to a certain split, some wedge constellations lie between partitions without a visually clear assignment to a certain split. We relax the assignment between wedges and splits by defining the head of a wedge as the main orientation point. Here, a wedge-head intersecting the boundary between two splits is assigned to both splits.

Combining Horizontal and Vertical Splits Next to the pyramidal partitioning discussed before, a more fine-grained splitting can be achieved by combining the horizontal and vertical partitioning. Figure 6 shows three examples of possible combinations. For level 2 the horizontal and vertical partitioning can be combined into $2 \times 2 = 4$ splits. While combining the horizontal and vertical parts of level 3 results in $3 \times 3 = 9$ splits. Next to the combinations within a pyramid level, it is also possible to combine between different pyramid levels as shown in figure 6. In contrast to a manual partitioning process, as described in Sec. 3.2, where wedge types and counts are assigned to certain splits by a human expert, we achieve the combinations by automatically compare annotations from two overlapping splits and take the minimum count for corresponding wedge types.

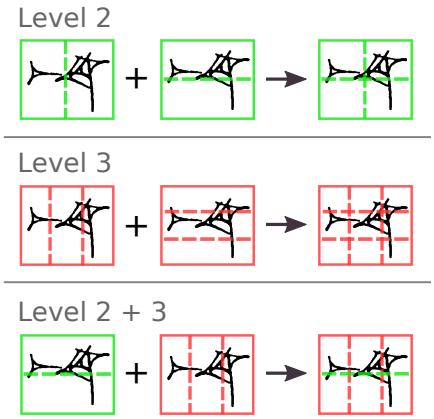

Fig. 6. Three examples of possible combinations between horizontal and vertical partitions, for level 2, 3, and a combination of 2 and 3.

Modeling Binary Attributes In order to transfer the Gottstein-System into a binary attribute vector, we first count the minimum and the maximum number of wedges for each type. For the database (see Sec4.1) we use in this work, the minimum count of each wedge type denoted by a, b, c, and d is 0. For the wedge type a and b the maximum count is 10. The wedge type c has a maximum count of 12 and d have a maximum count of 2. Except for the count of 0 all counts are modeled as single attributes, respectively. The absence of a wedge type is modeled as a zero-vector. In this way, the representation results in a $10 + 10 + 12 + 2 = 34$ dimensional binary attribute vector. In the end, all attribute vectors from each split are concatenated to form the final Gottstein representation as a $34 \times N_{splits}$ dimensional vector, where N_{splits} denotes the total number of splits. For example, the partitioning from Figure 5 will be concatenated to a $34 \times (1 + 2 + 2 + 3 + 3) = 374$ dimensional vector.

4 Evaluation

In this section, the evaluation of our approach is described. At first, our cuneiform database and the data partitioning into training and test sets are shown. Afterward, the model architecture and our decisions for different modifications are explained. In the Sections 4.3, 4.6, and 4.5 the training setup for the model, the similarity metric, the evaluation protocol, and the evaluation metric are described. Finally, the results are presented in Section 4.7.

4.1 Database

In this work, we use a set of annotated cuneiform tablets collected from a total of 232 photographs, with annotated 95 040 samples of cuneiform signs, provided by [11]. These samples represent an inventory of 300 different cuneiform signs effectively. The sign images have considerable variability in size, ranging from a minimum of 11 pixel height to a maximum of 520 pixel in width, as shown in Table 1. Figure 7 shows examples of cuneiform signs. On the far left, the sign class, and next to it, the Gottstein encoding is given. In the middle of Figure 7 hand-drawn examples and corresponding sign crops from the tablet photographs are shown. As one can see, the visual variability and quality in terms of wedge impression, image resolution, and surface damage can be tremendous.

Table 1. Statistics about image sizes given in pixel.

	max	*min*	*mean*	*std*
height	319	11	85.55	±31.50
width	520	16	106.09	±46.31

Data Partitions As the data presented above does not have an official partitioning into training and test sets, we split the data into four equal parts. This way, we can perform four-fold cross-validation similar to [16]. First, all samples are sorted according to their category. Afterward, every fourth sample from a category is assigned to one validation set in an interleaving fashion. This assignment results in four validation sets containing approximately the same number of samples with the same distribution over sign categories. Table 2 shows the number of samples for each of the four cross-validation splits.

4.2 Model Architecture

Similar to [16] we used the successful ResNet [8] architecture. From the different versions proposed for the ResNet model, we adapted the one with 34 layers and version B, as the preliminary experiments revealed the ResNet-34B to have

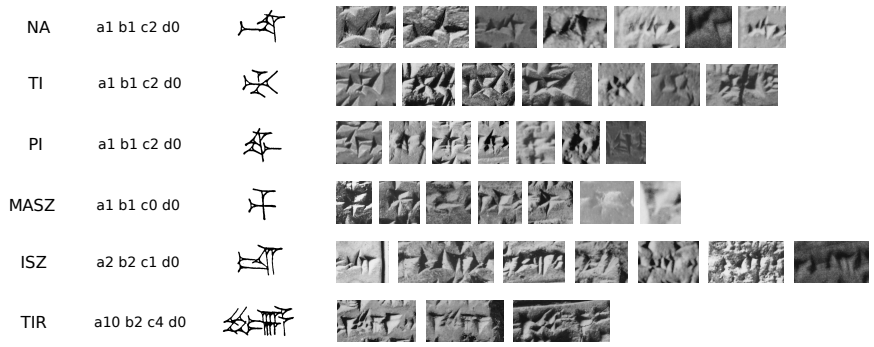

Fig. 7. Examples of six different cuneiform signs are shown. On the far right side, the sign names and the Gottstein representations are given. Next to them, the corresponding hand-drawn examples are shown. And on the right side, cropped snippets of cuneiform signs from tablet photographs are visualized.

Table 2. Number of samples for each cross-validation split.

	$Split_1$	$Split_2$	$Split_3$	$Split_4$	$Total$
# Samples	23 878	23 799	23 717	23 646	95 040

the best trade-off between retrieval performance, number of parameters, and computation time. We apply some modifications to our model in order to achieve higher retrieval performance. While the original ResNet architectures perform five times downsampling, we discard the last three downsamplings and only kept the first two. One in the first convolutional layer and the second one in the first Pooling layer. In this way, all Residual-Blocks are computing on equally sized feature maps, while spatial information is preserved. After the last convolutional layer, a Temporal Pyramid Pooling (TPP) [18], in analogy to the pyramid levels presented in Sec 3.2, with the levels $(1, 2, 3)$ is applied. We also experimented with the original Global Average Pooling (GAP) and the Spatial Average Pooling (SPP) introduced in [6], but the TPP achieved the best results in preliminary experiments. The output of the TPP layer effectively results in a $512\times(1+2+3) = 3072$ dimensional feature vector, which is mapped to the attribute representation using a fully connected layer. As in [16], we do not rescale the images to uniform image sizes.

4.3 Training Setup

The model is trained using the *binary cross-entropy (BCE)* loss and the *Adaptive Momentum Estimation (Adam)* [10] optimizer. For the momentum mean and variance values β_1 and β_2 are set to 0.9 and 0.999 respectively, while the variance flooring value is set to 10^{-8} as recommended in [10]. The batch size is technically

set to 1 while accumulating the gradients over 10 iterations, which results in an effective batch size of 10. We set the initial learning rate to 10^{-4} and divide the learning rate by 10 after 100 000 and another division by 10 after 150 000 iterations, respectively. All models are trained for 200 000 iterations in every training run. As parameter initialization, we use the strategy proposed in [7]. The weights are sampled from a zero-mean normal distribution with a variance of $\frac{2}{n_l}$, where n_l is the total number of trainable weights in layer l. In order to extend the number of training samples sampled from the database, we use some common augmentation techniques. At first, we balance the training data by sampling the sign categories from a uniform distribution, followed by augmentation techniques like perspective transformation, shearing, rotation, translation, scaling, lightness changing, and image noise generation.

4.4 Similarity Metric

For the segmentation-based scenario, retrieval is performed by computing a certain similarity between a given query and all test samples, excluding the query itself. In this work, we make use of a similarity measurement called *Probabilistic Retrieval Model* proposed in [15]. Here, the assumption is made that the binary attribute representation is a collection of d independent Bernoulli distributed variables, each having their probabilities p, which evaluates to 1. Equation 1 shows the definition of the probabilistic retrieval model.

$$p_{prm}(\mathbf{q}|\mathbf{a}) = \prod_{i=1}^{D} a_i^{q_i} \cdot (1 - a_i)^{1-q_i} \tag{1}$$

Here, $\mathbf{q}$ denotes the query vector, and $\mathbf{a}$ represents the estimated attribute representation. As the evaluation of the PRM from Eq. 1 requires to compute a product of many probabilities, we evaluate the PRM scores in the logarithmic domain in order to improve the numerical stability:

$$\log p_{prm}(\mathbf{q}|\mathbf{a}) = \sum_{i=1}^{D} q_i \log a_i + (1 - q_i) \log(1 - a_i) \tag{2}$$

4.5 Evaluation Protocol

We follow the evaluation protocol from [16] and evaluate our approach in a segmentation-based QbE and QbX scenario on the cuneiform database shown above (Sec. 4.1). As the given data is partitioned into four separate validation folds, three folds are used to train a single model, and the remaining fold is considered the test partition to perform the retrieval evaluation. For the QbE scenario, all images of cuneiform signs from a test fold, which occur at least twice, are considered as queries. Retrieval is performed by estimating the attribute representations for all sign images from a test fold given the estimated attributes from a certain image query. Afterward, the retrieval lists are obtained

by evaluating the PRM scores between the list items and a given query estimate and sort them in ascending order with respect to the highest probabilities. For QbX, retrieval is performed equivalently to QbE except that a sign category is only considered once as a query. It is important to note that the relevance category determines the relevance of an item within the retrieval list. We perform retrieval based on models either trained in a one-hot encoding fashion using the sign categories as targets or in a multi-class encoding using the Gottstein attribute representations as targets.

4.6 Evaluation Metrics

To evaluate the performance of our approach, we decided to use the interpolated Average Precision (AP) as this metric is commonly used in the spotting literature [18]. The AP is defined as follows:

$$AP = \frac{\sum_{n=1}^{N} P(n) \cdot R(n)}{t} \tag{3}$$

$P(n)$ denotes the precision if we cut off the retrieval list after the i-th element. $R(n)$ is the indicator function, which evaluates to 1 if the i-th position of the retrieval list is relevant with respect to the query and 0 otherwise. N represents the length of the retrieval list, and t is the total amount of relevant elements. Similar to segmentation-based word spotting tasks, the retrieval list contains all cuneiform sign images from the respective test set. Afterward, the overall performance is obtained by computing the mean Average Precision (mAP) overall queries, where Q denotes the total amount of queries:

$$mAP_{qry} = \frac{1}{Q} \sum_{q=1}^{Q} AP(q) \tag{4}$$

We denote the standard mean Average Precision with the subscript *qry*, as the mean is computed across all average precision values obtained from all queries. As proposed in [16], we additionally evaluate a separate mean Average Precision for every sign category by computing the mean of all mAP_{qry} values with respect to the category:

$$mAP_{cat} = \frac{1}{C} \sum_{c=1}^{C} mAP_{qry}(c) \tag{5}$$

Where C denotes the set of sign categories and $mAP_{qry}(c)$ evaluates to the mAP value of the sign category c. In this way, the mAP_{cat} ensures a more balanced performance evaluation across all sign categories contained in our database.

4.7 Results and Discussion

Table 3, 4 and 5 list the results for the QbX and QbE experiments, given in mAP percentage. All results are averaged over four cross-validations folds. The

Table 3. QbX and QbE experiments using a one-hot encoding as sign category representation in combination with the cross-entropy (CE) and binary cross-entropy (BCE) loss function, respectively. Sign categories are used as relevance categories. Results are shown in mAP [%].

Loss	N_{reps}	QbX	QbE	
		mAP_{cat}	mAP_{qry}	mAP_{cat}
CE	300	90.19	86.46	79.68
BCE	300	84.10	75.34	68.50

first results are shown in Table 3, evaluated for a sign encoding in a one-hot fashion. Here, all sign categories are represented by a 300 dimensional output vector where every element of this vector represents a certain category. The column N_{reps} denotes the number of unique representations if all 300 sign categories are mapped to a certain representation. Here, $N_{reps} = 300$ shows that 300 sign categories are mapped to 300 unique representations (i.e. one-hot encodings). We evaluate two different loss functions, namely cross-entropy (CE) and binary cross-entropy (BCE). For the CE configuration, the Softmax-Function is applied to the output of the model. For a model trained with the BCE loss, we apply a Sigmoid-Function after the model output. Table 3 shows distinct differences in retrieval performance depending on the loss and output function used. For QbX, the CE configuration outperforms the BCE by about 6%, while for QbE the performance difference is about 11%. Table 4 shows the results for a Gottstein representation combined with different partitioning strategies. The column sign representation denotes the configuration of an attribute representation. Here, GR means Gottstein-Representation, and GR+LE means Gottstein-Representation combined with the Last Element as shown in [16]. The configuration for pyramidal splits is described by H_l for horizontal and V_l for vertical splitting, while l denotes the pyramid level in the respective direction. For example, the notation GR+H_2+V_2 means a Gottstein Representation combined with a horizontal and vertical splitting on level 2. Retrieval is performed by evaluating the retrieved items with respect to the sign representation itself. The representation estimated by the model defines the relevance of retrieved items. As in Table 3, the number of unique representations is denoted by N_{reps}. For example, GR maps all 300 sign categories to 177 unique attribute representations, which means 1.69 signs are mapped to the same representation, on average. Table 4 shows that the model can learn an arbitrary representation based on binary attributes. All representations show approximately the same performance. In Table 5, the relevance of the items in a retrieval list is defined by the sign categories. Here, the retrieval performance is correlating with the number of unique representations. The mAP values increase for higher N_{reps} numbers, which is not surprising. On the one hand, a Gottstein Representation can distinguish between 177 out of 300 different categories. Thus, appropriate retrieval performance is not possible.

Table 4. QbX and QbE experiments using different attribute encodings as sign category representation with the binary cross-entropy (BCE) loss function. The relevance category is defined by the sign representation itself, respectively. Results are shown in mAP [%].

Representation	N_{reps}	QbX		QbE	
		mAP_{qry}	mAP_{cat}	mAP_{qry}	mAP_{cat}
GR	177	**90.32**	88.78	80.28	75.52
GR+LE	267	89.27	88.72	**81.66**	76.93
GR+H_2	263	90.02	**89.51**	81.51	77.27
GR+H_2+V_2	287	89.12	89.01	80.77	**77.30**
GR+H_2+H_3	291	88.94	88.90	80.84	76.78
GR+H_3+V_3	297	89.85	89.79	80.89	76.78
GR+H_2+V_2+H_3+V_3	300	88.76	88.76	80.62	76.85
GR+$H_2 \times V_2$	287	88.49	88.37	78.81	74.71
GR+$H_3 \times V_3$	297	86.95	86.89	77.25	72.30
GR+$H_2 \times V_2$+$H_3 \times V_3$	300	87.40	87.40	77.02	73.02

On the other hand, exploiting the position of wedges within a sign and defining a Gottstein Representation based on a spatial pyramid splitting increases the performance by 32.95% for QbX (comparing GR and GR-H_3-V_3) and 20.47% respectively 16.70% for QbE (comparing GR and GR-H_2-V_2-H_3-V_3).

5 Conclusion

In this work, we enhanced the Gottstein Representation by applying a pyramidal partitioning of cuneiform signs. This way, the signs can be expressed by wedge types and their respective counts, and the approximate positions of their wedges. We evaluate our approach on a cuneiform database consisting of 232 photographs in two segmentation-based scenarios: Query-by-Example and Query-by-eXpression. The experiments show that a Gottstein Representation enhanced with spatial information achieves better results than the Gottstein Representation proposed in [16] using item relevancy based on sign categories. Although a representation enhanced with spatial information can be superior compared to one without, this spatial information entails the problem of ambiguity, as shown in Sec 3.2. Hence, the users' representation should be intuitively interpretable and robust against "incomplete" or noisy query definitions. Therefore, further investigations need to be made to define a cuneiform sign representation with fewer (or even without) ambiguities and an intuitive interpretation for human experts. Nevertheless, we would like to derive the conclusion that an appropriate

Table 5. QbX and QbE experiments using different attribute encodings as sign category representation with the binary cross-entropy (BCE) loss function. The sign categories are used as the relevance categories. Results are shown in mAP [%].

Representation	N_{reps}	QbX		QbE	
		mAP_{qry}	mAP_{cat}	mAP_{qry}	mAP_{cat}
GR	177	55.87	55.87	60.15	60.15
GR-LE	267	80.54	80.54	77.15	71.20
GR-H_2	263	79.19	79.19	74.03	71.05
GR-H_2-V_2	287	85.93	85.93	78.34	75.00
GR-H_2-H_3	291	86.69	86.69	78.72	74.84
GR-H_3-V_3	297	**88.82**	**88.82**	80.31	75.91
GR-H_2-V_2-H_3-V_3	300	88.76	88.76	**80.62**	**76.85**
GR-$H_2 \times V_2$	287	85.31	85.31	76.51	72.53
GR-$H_3 \times V_3$	297	85.93	85.93	76.75	71.50
GR+$H_2 \times V_2$+$H_3 \times V_3$	300	87.40	87.40	77.72	73.02

representation based on binary attributes which encode spatial information fits well a cuneiform sign description in order to define retrieval queries.

Acknowledgements This work is supported by the German Research Foundation (DFG) within the scope of the project Computer-unterstützte Keilschriftanalyse (CuKa)

References

1. Almazán, J., Gordo, A., Fornés, A., Valveny, E.: Word spotting and recognition with embedded attributes. TPAMI **36**(12), 2552–2566 (2014)
2. Bogacz, B., Gertz, M., Mara, H.: Character retrieval of vectorized cuneiform script. In: Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR). p. 326–330. ICDAR '15, IEEE Computer Society, USA (2015). https://doi.org/10.1109/ICDAR.2015.7333777, https://doi.org/10.1109/ICDAR.2015.7333777
3. Bogacz, B., Gertz, M., Mara, H.: Cuneiform character similarity using graph representations (2015)
4. Bogacz, B., Howe, N., Mara, H.: Segmentation free spotting of cuneiform using part structured models. In: Proc. of the ICFHR. pp. 301 – 306 (2016)
5. Gottstein, N.: Ein stringentes identifikations- und suchsystem für keilschriftzeichen (2012)
6. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. European Conference on Computer Vision pp. 346–361 (2014)

7. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: Proc. of the Int. Conf. on Computer Vision. pp. 1026–1034 (2015)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
9. Howe, N.R.: Part-structured inkball models for one-shot handwritten word spotting. In: Proc. of the ICDAR. pp. 582–586 (Aug 2013). https://doi.org/10.1109/ICDAR.2013.121
10. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: Proc. of the Int. Conf. on Learning Representations (2015)
11. Akademie der Wissenschaften und Literatur, M., Julius-Maximilians-Universitat, W.: Hethitologie Portal Mainz (2000), http://www.hethiter.net/
12. Mara, H.: Multi-Scale Integral Invariants for Robust Character Extraction from Irregular Polygon Mesh Data. Ph.D. thesis (10 2012)
13. Massa, J., Bogacz, B., Krömker, S., Mara, H.: Cuneiform detection in vectorized raster images. In: Computer Vision Winter Workshop (CVWW) (2016)
14. Rothacker, L., Fink, G.A.: Segmentation-free query-by-string word spotting with bag-of-features HMMs. In: Proc. of the ICDAR. pp. 661–665 (Aug 2015). https://doi.org/10.1109/ICDAR.2015.7333844
15. Rusakov, E., Rothacker, L., Mo, H., Fink, G.A.: A Probabilistic Retrieval Model for Word Spotting Based on Direct Attribute Prediction. In: Proc. Int. Conf. on Frontiers in Handwriting Recognition. Niagara Falls, USA (2018), winner of the IAPR Best Student Paper Award
16. Rusakov, E., Somel, T., Fink, G.A., Müller, G.G.W.: Towards Query-by-eXpression Retrieval of Cuneiform Signs. In: Proc. Int. Conf. on Frontiers in Handwriting Recognition. Dortmund, NRW, Germany (2020)
17. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Proc. of the Int. Conf. on Learning Representations (2015)
18. Sudholt, S., Fink, G.: Evaluating word string embeddings and loss functions for CNN-based word spotting. In: Proc. of the ICDAR. pp. 493–498. Kyoto, Japan (2017)
19. Sudholt, S., Fink, G.A.: PHOCNet: A deep convolutional neural network for word spotting in handwritten documents. In: Proc. of the ICFHR. pp. 277 – 282. Shenzhen, China (2016). https://doi.org/10.1109/ICFHR.2016.55