Semantic Author Name Disambiguation with Word Embeddings

Mark-Christoph Müller

Heidelberg Institute for Theoretical Studies, Heidelberg, Germany
mark-christoph.mueller@h-its.org

Abstract. We present a supervised machine learning AND system which tackles semantic similarity between publication titles by means of word embeddings. Word embeddings are integrated as external components, which keeps the model small and efficient, while allowing for easy extensibility and domain adaptation. Initial experiments show that word embeddings can improve the Recall and F score of the binary classification sub-task of AND. Results for the clustering sub-task are less clear, but also promising and overall show the feasibility of the approach.

Keywords: Author name disambiguation · Semantic similarity · Word embeddings · Classification · Clustering · Machine learning · Deep Learning

1 Introduction

Author name ambiguity can be observed in collections of (scientific) publications when several authors bear, or publish under, the same name. It is caused by the natural limitation of available person names, and by the fact that some names are much more frequent than others. It is further aggravated by the common publishing practice of initializing authors' first names. Author name *disambiguation* (**AND**) is the task of deciding, for a given pair of publications with the same author name, whether that name refers to the same author individual [4, 18]. AND is a multi-facetted task, which comprises 1) content similarity, 2) co-author similarity, and 3) publication meta data similarity. In this paper, we present a supervised machine learning system which handles these three facets in a unified and extensible way. In particular, our system uses **word embeddings** (**WEs**) to deal with the content similarity facet of AND. WEs are employed to detect *semantic* content similarity or relatedness between pairs of publication titles, beyond surface-based string matching. To give just one illustrative example which exhibits neither title string nor co-author overlap, consider the following pair of publications from our data set.

A. Verma, **A. Kumar** (2004): Articulatory class based spectral envelope representation for voice fonts.
A. Karmakar, **A. Kumar**, R. K. Patney (2006): A Multiresolution Model of Auditory Excitation Pattern and Its Application to Objective Evaluation of Perceived Speech Quality.

Here, the author name *A. Kumar* does refer to the same person, but the only hint is in the semantic relatedness of the underlined tokens.

WEs have recently become popular in Deep Learning approaches to natural language processing (NLP). Full-blown Deep Learning models can take long to train and are technically demanding, often requiring specialized hardware. These requirements limit their practical applicability for digital libraries or online bibliographies. Our approach, in contrast, avoids these problems by 1) using only a simple machine learning model, which is fast and easy to train, and 2) keeping the WEs separate from the model. This way, the WEs are trained in a one-off effort, and they can easily be re-used and combined, even as the model architecture gets more complex. The rest of this paper is structured as follows: Section 2 provides a brief definition of AND and outlines how we cast AND as binary classification followed by clustering. Section 3 introduces the concept of WEs for the computation of semantic similarity and then provides a detailed description of our system. Section 4 describes and discusses our experiments, and Section 5 briefly reviews some related work. The paper concludes in Section 6.

2 Definition of AND

AND deals with *authorship records* [3], which consist of an author name and some representation of the publication content, co-authors, and other meta data. Publications with n authors yield as many authorship records, and for every author, the $n-1$ other records provide important information about the publication co-authors. Content and co-author similarity are two interrelated facets of AND, none of which is sufficient in isolation. *Content* similarity between two authorship records with the same author name is not necessary to establish author identity: The same author can produce publications on different topics, or even in completely distinct fields, which will not be very similar. High *co-author* similarity is normally a strong indicator for author identity [15, 17], and is the sole information source for some AND approaches. On the other hand, the absence of common co-authors does not indicate non-identical authors, as one author can collaborate with distinct groups of colleagues. Additional meta data like publication year distance can have a mediating function here, as it can capture changes in an author's interests over time [6].

2.1 AND as Binary Classification plus Clustering

We follow [6, 14], and others in separating the AND task into binary classification of pairs of authorship records and subsequent clustering. Commonly, the input for an AND system is a list of so-called *blocks*, i.e. a list of sets of authorship records with a shared identical, or highly similar, name, and the output is a partitioning of each block into sub sets for the individual authors. In the binary classification paradigm, a single data instance represents two authorship records (from the same block, but from different publications) and a binary label which is 1 if the publications are authored by the same person, and 0 otherwise. In our approach, each instance represents the following information for each of the two authorship records: ***Content information*** subsumes various textual information. We assume that minimally the publication title is available, but the same representation is easily applied to other textual artifacts, like abstracts or full texts. Title words of each of the two authorship records are lowercased, cleaned of stop words, and represented as one list of complete and one list of stemmed tokens (created using the PorterStemmer). Having these two lists at our disposal allows us to use pre-trained WE resources that expect either format. Each title is also split into one list each of character 3-, 4-, and 5-grams. In addition, we apply a two-word window on the stemmed token list for each publication to obtain a list of word bi-grams. ***Co-author information*** for each of the two authorship records is represented as a list of normalized co-author names, *excluding* the shared author name. Normalization includes initialization of the first name and lowercasing of the entire string. A list of three co-authors could thus look like "f.harary m.lim d.wunsch". ***Meta data*** includes relational, first-order attributes of the pair of authorship records: *Publication year distance* is the absolute difference between both publications' year attributes. If the year is missing for one or both publications, the value is -1. *Publication venue match* is 1 if the publication venues of the two publications match, 0 if they are available but do not match, and -1 if one or both is unavailable.[1] After obtaining binary classification results for a given block, the individual decisions have to be combined into clusters. This is commonly done as a graph partitioning task (cf. Section 5), where binary classification confidence scores are used as edge weights in an undirected graph, and the author partitions are obtained by some graph algorithm. Alternatively, we create the graph in such a way that its *connected components* can directly be interpreted as clusters. We do this by employing different minimum positive confidence thresholds during graph creation (cf. Section 4.2).

3 A Deep Learning Model for AND

3.1 Word Embeddings for Semantic Similarity

The basic idea behind WEs is that distributional (i.e. co-occurrence) information derived from a large text corpus is represented in low-dimensional vector space in such a way that proximity in this vector space can be interpreted as similarity or relatedness. The vector representation for a single word is commonly given as a list of n real-valued numbers, where n is the dimensionality of the embedding. Two popular algorithms for learning WEs from texts are GloVe [13] and word2vec [10]. Apart from the desired dimensionality, both algorithms accept, among others, one parameter for the window size, and one for the minimum vocabulary count. The first parameter controls the maximum distance

[1] We only consider venue *identity* rather than *similarity* because our data set only contains abstract, uninterpretable venue identifiers.

between words that are considered as co-occurrent, and the second parameter controls how often a word has to occur in the corpus in order to be considered at all. Depending on the choice of parameters and the size of the corpus, training WEs can be computationally expensive. However, since they are supposed to capture universal, task-independent semantic relations, they can be utilized in diverse settings without the need for re-training. Several studies have focussed on the evaluation of WEs. [16] perform extensive experiments with diverse WEs, and evaluate how well they reproduce *human* semantic relatedness judgements, and how much they contribute to tasks like e.g. sentiment classification. [5], in a similar vein, evaluate several WEs on what they call *NLP* (=extrinsic) and *linguistic* (=intrinsic) tasks. While the level of granularity of WEs is the individual word, computing the semantic similarity of arbitrarily long word sequences (e.g. sentences or publication titles) requires that those sequences are reduced to single vectors that somehow capture the semantics of the whole sequence. A common way to do this is to average over the embeddings for the individual words: Given a collection of n-dimensional WE vectors and a sequence of i words, we retrieve the j vectors for those words that are covered in the collection (with $j <= i$), sum over the j values for each of the n dimensions, and divide each of the values by j. This yields one n-dimensional embedding vector for each word sequence, which can be compared to similar representations of other sequences, e.g. by means of computing the cosine similarity. This simple and efficient method has been shown to work surprisingly well, and is often used as a baseline in more complex, training-intensive systems, e.g. [9][2]. We prefer this simple heuristic over more powerful Deep Learning devices (like e.g. RNNs or LSTM networks, which maintain a notion of *ordering* in the reduced sequence) because 1) our preliminary experiments showed that they dramatically increase the technical complexity and training time for our system, rendering it difficult to use in a practical setting, and 2) we think that, for publication titles, the subtle differences conveyed by word order are negligible.

3.2 System Architecture and Components

Figure 1 shows the architecture of the binary classifier employed in our system, which is implemented as one multi-layer neural network with Keras [2] on top of Theano [19], and which is trained using the Adam optimizer. Input to the network is provided by the two authorship records (depicted as documents). The network consists of three auxiliary models (horizontal boxes), each of which focusses on a particular facet of the classfication problem. Only the meta data attributes, due to their simplicity, do not have their own model. Each auxiliary model is a multi-layer neural network (cf. below for details) with sigmoid activation and a final softmax layer, which outputs the positive and negative class probability for each instance. The **simple co-author model** contains only two features, the cosine and the Jaccard similarity of the normalized co-author

[2] [7] report similar baseline results with *summing* instead of *averaging* over the embeddings of a sequence.

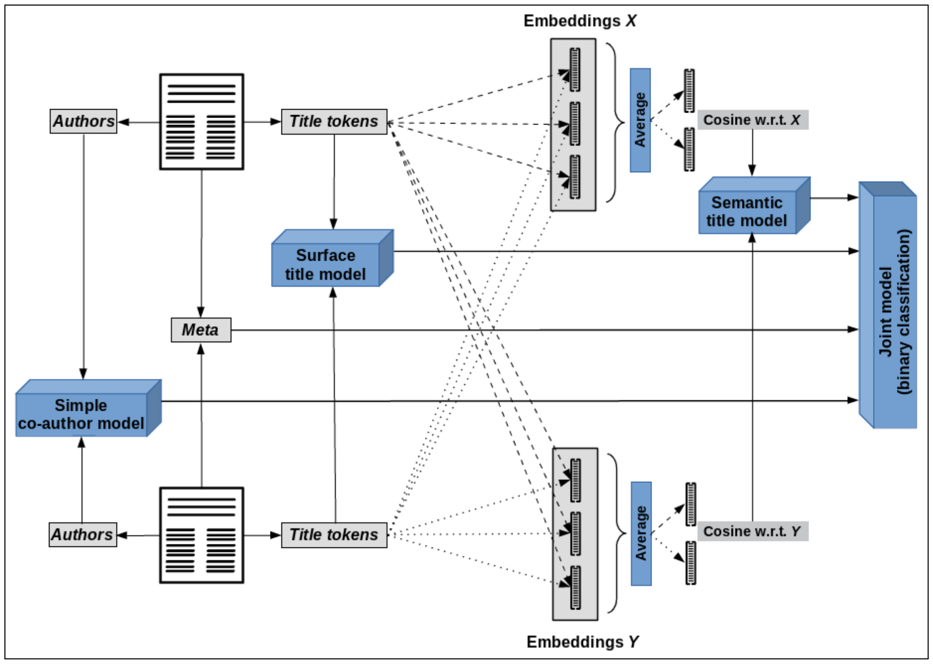

Fig. 1. System architecture: Binary Classifier

names (without the shared name). The model consists of one two-node hidden layer only. The simplicity of this model, which treats each name as one atomic token, is a result of the author name structure found in the KISTI data set (cf. Section 4.1), which, like most standard AND data sets [12], does not require any sophisticated string similarity measures due to the absence of name variability. As the focus of this paper lies on the semantic title model, the co-author model is intended as a high-precision baseline only. The **surface title model** covers the string-matching aspect of content similarity in terms of cosine and Jaccard similarity of stemmed tokens, character 3-, 4-, and 5-grams, and word bi-grams, resulting in a total of ten features. The model consists of one hidden layer with ten nodes. Here, stemmed tokens are used in order to increase the coverage. The features used in this model are more or less standard features found in many NLP and IR systems. Despite their simplicity, surface-based features can handle a lot of cases, and display a very reasonable performance (cf. the baseline results in Section 4.2). This can make it difficult to demonstrate the contribution of semantic features, because their effect is easily covered up by the effect of the surface-based features. The **semantic title model** (including its preprocessing) is the most interesting component of the binary classifier. The system can be supplied with c collections of WEs. For each collection, the preprocessing component computes the cosine similarity of the (stemmed or unstemmed) title tokens from the two authorship records, which are then passed to the semantic title model as input features. The model consists of three hidden layers with one node for each of the c input features. For reasons of clarity, Figure 1 shows only two collections of WEs (c=2), but the system can accept arbitrarily many, including none, in which case the semantic title model remains inactive. One important feature of the semantic title model is that tokens that appear in the publication titles of *both* authorship records are completely removed from its input prior to generating the averaged WEs. This is motivated by the idea that perfect string identity can and should be handled in the surface title model, and that the semantic title model should be allowed to ignore these cases in favor of cases that involve actual vector space *similarity* rather than *identity*. Finally, the **joint model** (vertical box) is also a multi-layer neural network with sigmoid activation and a final softmax layer. It simply integrates the outputs of the softmax layers of the auxiliary models, as well as the meta data attributes, and produces the final classification. The joint model consists of two hidden layers with eight (with semantic title model) or six (without semantic title model) nodes. At test time, it outputs for each instance the positive and negative class probability, and the higher of the two probabilities determines the final classification for the instance. During training, each auxiliary model is presented with its respective sub set of features, computed from the instance representation described in Section 2.1 above, and with the binary label. Likewise, the joint model is presented with the outputs of the auxiliary models and the same binary label. Being part of a single network, all four models are trained simultaneously. However, we decouple the training of the models by computing a training error and corresponding loss for each model individually. This allows the different parts of the system to train at different speeds: the simple co-author model and the surface title model, e.g., converge quickly, while the semantic title model and the joint model, depending on the parameters and the number of WE collections, converge more slowly.

In order to obtain author clusters, we then employ NetworkX[3] to create an undirected graph containing one node for every authorship record in the block, and to add edges between all node pairs that were classified with a minimum positive confidence above a given (variable) threshold.

4 Data, Experiments, and Results

4.1 Data Set and Word Embeddings

Our task-specific data set is KISTI [8], which is derived from dblp data and consists of 41.674 authorship records from 37.613 publications, and 6.921 different authors. The data is pre-structured into blocks which are identified by a first name initial and a full last name. Each block contains authorship records from 1 to max. 71 different authors. DEV-TRAIN, DEV-TEST, and EVAL data was generated from the KISTI data set as follows: We randomly distributed the individual authors in each block (i.e. *y.chen_1*, *y.chen_2*, ... *y.chen_n*) into

[3] https://networkx.github.io/

three sets of roughly the same size, ignoring blocks with less than six authors. Then, for each of the three sets, we paired all authorship records in the same block with each other, and created either a positive or a negative instance in the format described above in Section 2.1. Note that, of the various possible methods of creating data instances from the KISTI data set, this method makes it rather difficult for the system, because, at test time, all authors in DEV-TEST and EVAL are unseen. This yielded 190.009 DEV-TRAIN instances (41.8% pos, 58.2% neg.), 226.546 DEV-TEST (47.66% pos., 52.34% neg.), and 163.000 EVAL (42.95% pos., 57.05% neg.). Word level semantics is integrated into our system by means of pre-trained GloVe embeddings [13] and custom-built WEs trained on various text corpora. The GloVe embeddings[4] (**GloVe**) were trained on huge corpora (between 6 and 840 billion tokens) covering Wikipedia pages and other web data, as well as news wire texts. A second set of WEs (**dblp**) was trained on a text corpus of 3.5 million publication titles derived from a dblp XML dump[5]. We made sure to remove from the corpus the titles of all publications that are also contained in the KISTI data set. A third set of WEs (**MSAc**) was trained on a corpus of 6.5 million publication titles extracted from the Microsoft Academic Search dataset. Prior to training, for **dblp** and **MSAc**, special characters were removed, and the publication titles were lowercased, cleaned from stopwords, and stemmed with the PorterStemmer. A fourth set of WEs (**dblp+MSAc**) was trained on the concatenation of the dblp and the MSAc text corpora. All WEs were trained with the gensim[6] implementation of word2vec, using the CBOW variant. For all corpora, we employed several values for the parameters *dimensionality* (d=50,100,200,300), *minimum token count* (mc=5,20), and *window size* (w=5,10). There is one embedding file for each combination, resulting in 16 separate files per text corpus. Table 1 gives some statistics. Note that the total number of text tokens used for calculating the coverage of the embeddings is higher for the GloVe embeddings (10.734) because they contain unstemmed tokens, while all other embeddings were trained on stemmed tokens. Also, the glove.840B.300d embeddings are case-sensitive, which is why their coverage on our lowercased data set is smaller.

4.2 Experiments and Discussion

As mentioned in Section 2.1, our system consists of two parts, which we evaluate individually. The *binary classifier* is evaluated in terms of P, R, and F for retrieving positive instances. We train one binary classifier with each set of WEs individually, and with all sets of WEs (i.e. GloVe, dblp, and MSAc) at once. Note that we use *all* five (GloVe) resp. 16 (dblp, MSAc) embedding files per set *simultaneously.* Thus, the semantic title model has as many as 37 features when using all WEs at once. We do this in order to exploit potential complementarity of the WEs produced with different training parameters, and because our initial

[4] https://nlp.stanford.edu/projects/glove/
[5] http://dblp.uni-trier.de/xml/
[6] https://radimrehurek.com/gensim/

Table 1. Word Embedding Statistics

Name	**# Tokens**	**Coverage**
glove.6B.{100,200,300}d	400.000	9.151 / 10.734 (85%)
glove.42B.300d	1.917.494	9.944 / 10.734 (92%)
glove.840B.300d	2.196.017	9.511 / 10.734 (88%)
dblp.cbow.{50,100,200,300}d.mc5.{w5,w10}	56.081	6.522 / 7.263 (89%)
dblp.cbow.{50,100,200,300}d.mc20.{w5,w10}	23.738	6.081 / 7.263 (83%)
msac.cbow.{50,100,200,300}d.mc5.{w5,w10}	198.383	6.411 / 7.263 (88%)
msac.cbow.{50,100,200,300}d.mc20.{w5,w10}	74.255	5.924 / 7.263 (81%)
dblp+msac.cbow.{50,100,200,300}d.mc5.{w5,w10}	224.599	6.680 / 7.263 (91%)
dblp+msac.cbow.{50,100,200,300}d.mc20.{w5,w10}	84.104	6.383 / 7.263 (87%)

experiments gave no clear indication as to which parameters are optimal. The results for the binary classifier are given in Table 2, with the maximum P, R, and F values in bold. Each classifier is trained for 40 epochs, and we report the performance with the maximum F on DEV-TEST, along with the number of epochs that were required to reach that result (E).

Table 2. Binary Classification Results (max. F score reached after *E* epochs)

ID	**WEs**	**Files**	**P**	**R**	**F**	**E**
0	-	-	**82.48**	45.82	58.91	9
1	GloVe	5	81.21	47.22	59.72	9
2	dblp	16	76.96	67.18	71.74	20
3	MSAc	16	78.29	53.82	63.79	23
4	dblp+MSAc	16	76.38	65.29	70.40	23
5	GloVe, dblp, MSAc	37	75.67	**69.01**	**72.19**	13

As a baseline, we trained and tested the classifier with ID **0** without supplying any WEs, so that the semantic title model remains inactive. As expected, this classifier shows the worst performance (58.91 F), but not, however, by a large margin. The baseline binary classifier is clearly biased towards P, with the highest P and the lowest R of all binary classifiers. We see this as the result of the classifier's limitation to simple string matching, which prevents it from retrieving instances which are semantically related, but where this relatedness is not obvious on the surface. The worst non-baseline classifier is **1**, only slightly above baseline with 59.72 F. The other non-baseline classifiers are considerably better, with F scores of 63.79 (**3**), 70.40 (**4**), and 71.74 (**2**). Note that classifier **2** uses the WEs trained on the dblp corpus, which is in-domain in the sense that it bears the most similarity - although no publication overlap - to the KISTI data set. It yields by far the best binary classification performance of all WEs trained on individual corpora. Note also that classifier **4**, which is trained on the concatenation of the dblp and MSAc corpora, fails to improve, or even reach, the F score of the classifier trained on the dblp corpus alone (**2**). Thus, simply merging both corpora prior to WE training did not yield an improvement. As mentioned earlier, another way of integrating different sources of word level semantics into our system is by using *several* sets of WEs *simultaneously*, as in classifier **5**. And indeed, this configuration yields the best performance of all binary classifiers: although P drops to the lowest value of all, the associated gain in R is sufficient to also result in the best overall F of 72.19.

In the next step, we applied each classifier to the *clustering* task. Table 3 reports B^3 [1] results on DEV-TEST calculated by the CONLL scorer[7]. For each binary classifier from Table 2, we report one set of results for different minimum positive confidence threshold values (mpc=0.5, 0.75, 0.9, 0.95).[8] The maximum P, R, and F values for each of the four sets are given in bold. In addition, the best F value for each classifier (i.e. for each *row* in Table 3) is underlined. The intuition behind the mpc thresholds is to increase the precision (P) of the clustering by only allowing high-confidence binary classifications to cause the clustering together of two authorship records. At the same time, the recall (R) will decrease, as fewer clusters are created, but we expect this tradeoff to be less severe for 'better', more discriminative binary classifiers.

Table 3. Clustering Results

	mpc=0.5			**mpc=0.75**			**mpc=0.9**			**mpc=0.95**		
ID	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
0	**57.23**	97.05	**72.00**	**75.74**	90.77	**82.57**	**87.95**	82.95	85.37	93.61	**70.29**	80.29
1	56.48	97.36	71.49	71.76	92.43	80.80	85.97	85.24	85.60	93.65	70.20	80.25
2	53.50	98.99	69.46	65.19	95.50	77.49	83.98	**86.94**	85.44	**95.31**	68.45	79.68
3	53.86	98.46	69.63	69.87	92.92	79.76	85.42	85.40	85.41	94.49	69.89	80.35
4	52.67	99.35	68.85	65.28	94.97	77.37	85.00	86.25	85.62	94.50	69.81	80.30
5	52.70	**99.37**	68.87	64.67	**95.80**	77.21	85.41	85.68	**85.66**	95.06	69.63	**80.38**

It can be seen that increasing the mpc threshold has the intended effect, as the P values for all classifiers increase from left to right without exception. Also as expected, the R values decrease from left to right. Up to and including mpc=0.9, this decrease is compensated for by the associated increase in P, such that the F values do also increase. For mpc=0.95, however, there is a sharp decline in R for all classifiers, which also causes the F values to drop. Thus, in our experiments, all binary classifiers reach their best F values for mpc=0.9 (underlined). However, for all classifiers, including the baseline, these F values are extremely similar at around 85. Although the best binary classifier (**5**) performs also best in clustering with an F of 85.66, it does so by a negligible margin only. So, the observed differences in binary classification do not translate to similar differences in clustering, which is somewhat surprising. We also see, however, that all non-baseline classifiers (i.e. those with an active semantic title model)

[7] http://conll.github.io/reference-coreference-scorers/

[8] mpc=0.5 corresponds to *no threshold*, as 0.5 is the minimum confidence in a binary classification.

have a better R (at least 85.24 (**1**)) than the baseline classifier (82.95), which is the intended effect of the semantic title model.

5 Related Work

As a computational task, AND has a long history in both computer science and digital library science, and has been tackled with symbolic and heuristic approaches, as well as with supervised and unsupervised machine learning [4]. To our knowledge, WEs have not previously been used for the task. However, a similar approach for inventor disambiguation in patent data bases is described in [11]. [20] is the only work so far in which Deep Learning methods have been applied to AND. Their model consists of an ensemble of (an unreported number of) N multi-layer perceptrons (MLPs) with seven layers of 50 hidden units each. As data, [20] use 30.537 binary labeled pairs of authorship records (12.93% positive pairs, 87.07% negative) featuring a matching, or highly similar, Vietnamese author name. In total, the data set includes names and name variants of ten authors. Each data instance contains numerical scores representing the similarity of the two records' author names, co-author names, affiliations, paper keywords, and author interest keywords, respectively. Note that paper titles are apparently not used. The scores are calculated using the Jaccard, Levenshtein, Jaro, Jaro-Winkler, Smith-Waterman, and Mogne-Elkan measures. The model is trained by iteratively providing each of the N MLPs with a randomly selected sample from the training data set. In contrast to our work, every MLP is exposed to the full set of features of the selected instances during training, so that there is no 'division of labour' between the MLPs with respect to the individual facets of the AND task. At test time, a classification is obtained by simply averaging over the predictions of the individual MLPs, while we use a dedicated network (the **joint model**) to integrate the outputs of the three individual models. [20] report a binary classification accuracy of 99.31 on their 20% hold-out data set, which the authors claim significantly outperforms their earlier systems based on conventional machine learning. Given the strong negative bias in their data set (almost 90%), we argue that it would have been more appropriate to evaluate the binary classifier according to Precision, Recall, and F-measure for retrieving positive instances (like in the present work). No clustering of the binary decisions is performed, so a full comparison of [20] and our work is not possible. Even more importantly, the system comprises (in our terminology) only the simple co-author model and the surface title model, and does not address semantic similarity beyond the string level.

The NC system described in [15] marks the current state of the art on the KISTI data set. It relies on manually encoded, domain-specific expert heuristics, which operate on automatically extracted string-similarity scores (including cosine similarity and tf*idf). The weights of these scores are manually tuned, rendering the approach completely unsupervised. Supervised training can optionally be employed to further optimize parameters. The system uses (co-)author names as well as publication and venue titles but, like [20], does not go beyond the string level. The output of the system are clusters, which are evaluated with the K score, which is roughly equivalent to F. On the KISTI data set [15] report a total K score of 94.00, which is obtained in a supervised setup by doing ten runs of two-fold (50%-50%) cross-validation per block, averaging the results per block, and again averaging the results for all blocks. Although the data set and the evaluation measure are the same or similar, the results of [15] cannot directly be compared with our results. One reason is that, in the block-wise cross-validation of [15], authors in the test split have probably also been in the training split, while in our approach, authors in test are always unseen.

6 Conclusion and Future Work

We presented the first AND system which tackles semantic similarity between publication titles by means of WEs. The system, although using some Deep Learning technology, aims at being practically usable and efficient. We found that adding WE-based semantic similarity can make a significant contribution to the binary classification part of the AND task, which is the most important result of this paper, and that WEs trained on in-domain corpora perform better than those trained on other, less similar corpora. Pre-trained general-purpose WEs (GloVe), although of high quality, were not helpful. We also found that complementarity of different WEs can best be exploited by using many independent WEs simultaneously, while training single WEs on concatenated corpora was not successful. Improvements observed in binary classification, however, did not clearly translate to improvements in clustering. In future work, therefore, we will improve the way our system exploits the individual binary, weighted classifications, e.g. by employing more powerful, state-of-the-art graph clustering algorithms. In order to create a competitive AND system, we also plan to extend the simple co-author model to be a more powerful component. All extensions and improvements will be able, if required, to make use of state-of-the-art methods in the Deep Learning ecosystem, which our system is already part of.

Acknowledgments The research described in this paper was conducted in the project *SCAD – Scalable Author Name Disambiguation*, funded in part by the Leibniz Association (grant SAW-2015-LZI-2), and in part by the Klaus Tschira Foundation. We thank Florian Reitz (dblp) for data preparation and the anonymous TPDL reviewers for their useful suggestions.

References

1. A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566, 1998.
2. F. Chollet. Keras. https://github.com/fchollet/keras, 2015.
3. R. G. Cota, A. A. Ferreira, C. Nascimento, M. A. Gonçalves, and A. H. F. Laender. An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. *J. Am. Soc. Inf. Sci. Technol.*, 61(9):1853–1870, Sept. 2010.

4. A. A. Ferreira, M. A. Gonçalves, and A. H. Laender. A brief survey of automatic methods for author name disambiguation. *SIGMOD Record*, 41(2):15–26, 2012.
5. S. Ghannay, B. Favre, Y. Estève, and N. Camelin. Word embedding evaluation and combination. In *Proceedings of LREC 2016, Portorož, Slovenia, May 23-28, 2016*, 2016.
6. T. Gurney, E. Horlings, and P. van den Besselaar. Author disambiguation using multi-aspect similarity indicators. *Scientometrics*, 91(2):435–449, 2012.
7. B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050, 2014.
8. I.-S. Kang, P. Kim, S. Lee, H. Jung, and B.-J. You. Construction of a large-scale test set for author disambiguation. *Information Processing & Management*, 47(3):452–465, 2011.
9. T. Kenter and M. de Rijke. Short text similarity with word embeddings. In *Proceedings of CIKM 2015*, pages 1411–1420, New York, NY, USA, 2015.
10. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*, 2013.
11. N. Monath and A. McCallum. Discriminative hierarchical coreference for inventor disambiguation. Presentation at PatentsView Inventor Disambiguation Technical Workshop, September, 2015.
12. M.-C. Müller, F. Reitz, and N. Roy. Data sets for author name disambiguation: an empirical analysis and a new resource. *Scientometrics*, 111(3):1467–1500, 2017.
13. J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1532–1543, 2014.
14. Y. Qian, Q. Zheng, T. Sakai, J. Ye, and J. Liu. Dynamic author name disambiguation for growing digital libraries. *Information Retrieval Journal*, 18(5):379–412, 2015.
15. A. F. Santana, M. A. Gonçalves, A. H. F. Laender, and A. A. Ferreira. On the combination of domain-specific heuristics for author name disambiguation: the nearest cluster method. *International Journal on Digital Libraries*, 16(3-4):229–246, 2015.
16. T. Schnabel, I. Labutov, D. M. Mimno, and T. Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 298–307, 2015.
17. D. Shin, T. Kim, J. Choi, and J. Kim. Author name disambiguation using a graph model with node splitting and merging based on bibliographic information. *Scientometrics*, 100(1):15–50, 2014.
18. N. R. Smalheiser and V. I. Torvik. Author name disambiguation. *ARIST*, 43(1):1–43, 2009.
19. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.
20. H. N. Tran, T. Huynh, and T. Do. Author name disambiguation by using deep neural network. In *Intelligent Information and Database Systems: 6th Asian Conference, ACIIDS 2014, Bangkok, Thailand, April 7-9, 2014, Proceedings, Part I*, pages 123–132, Cham, 2014. Springer International Publishing.