WueDevils at SemEval-2022 Task 8: Multilingual News Article Similarity via Pair-Wise Sentence Similarity Matrices

Dirk Wangsadirdja and **Felix Heinickel** and **Simon Trapp**
{name}.{surname}@stud-mail.uni-wuerzburg.de
Albin Zehe and **Konstantin Kobs** and **Andreas Hotho**
{surname}@informatik.uni-wuerzburg.de
University of Würzburg

Abstract

We present a system that creates pair-wise cosine and arccosine sentence similarity matrices using multilingual sentence embeddings obtained from pre-trained SBERT and Universal Sentence Encoder models respectively. For each news article sentence, it searches the most similar sentence from the other article and computes an average score. Further, a convolutional neural network calculates a total similarity score for the article pairs on these matrices. Finally, a random forest regressor merges the previous results to a final score that can optionally be extended with a publishing date score.

1 Introduction

The goal of the *Multilingual News Article Similarity* task (Chen et al., 2022) is to check pairs of multilingual news articles against each other in terms of similarity of their information content. The challenge focuses on *what* is talked about (time, geolocation, shared entities), not *how* the information is expressed (writing style, emotional tone, etc.), which is essential for applications such as analyzing the news coverage between different regions. The participants' objective is the creation of a model that rates the similarity of article pairs on a 4-point scale from most (1) to least (4) similar and achieves the highest possible Pearson correlation score compared to the gold standard. The languages covered by this competition are English (en), German (de), Spanish (es), Turkish (tr), Polish (pl), French (fr) and Arabic (ar) in the training, additionally Italian (it), Russian (ru) and Chinese (zh) in the evaluation.

Our system uses an ensemble approach to score pair-wise sentence similarity matrices of the article pairs, which are created with SBERT and Universal Sentence Encoder sentence embeddings. Scores are obtained through simple matrix operations and our convolutional neural network *SimCNN* based on the TextCNN by Kim (2014). Finally, a random forest regressor (Breiman, 2001) consolidates the individual scores into a final result, which we extend with a publishing date score.

The key challenge of this task is the usage of multilingual text pairs, which requires the application of less precise multilingual language models. Also the splitting of sentences and named entity recognition becomes hard to accomplish, since models for these tasks are still monolingual in most cases and we found the few multilingual ones to be unreliable. The article scraping tool provided by the authors also did not reveal meaningful features to work with besides the article title and text, since the scraped keywords, tags and publishing dates were not always available and in a utilizable format.

In the competition, we took 9th place with a Pearson correlation coefficient of 0.759 on the evaluation set, which differs from the observed performance on our own validation sets. This is due to a shift from Latin languages to more complex languages like Chinese in the evaluation data, where the performance of the multilingual models and sentence tokenizing algorithm decreases. Our code is publicly available[1].

2 Background

The starting point of the task is the training data — a CSV file — which contains a list of article pairs[2]. Each article pair consists of the languages, IDs and URLs for both articles and the gold standard similarity scores for multiple aspects of the articles (Geography, Entities, Time, Narrative, Style, Tone, Overall), of which only the overall score is relevant for our evaluation.

To get the content and metadata of the articles,

[1] https://github.com/simontrapp/semeval-22-task-8

[2] We used the most recent version 0.2 with 4,964 labeled article pairs.

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 1235 - 1243

the task authors provide a scraping tool[3] utilizing the *newspaper3k* python package that downloads the HTML page and creates a JSON file that contains the title, text, keywords, labels, date, and more properties of each news article. For the training of the prediction system on the JSON files, the results can be compared against the overall scores in the initial CSV file. The final evaluation data CSV, for which the results are submitted, does not provide scores to compare against.

3 Related Work

For the scoring of our similarity matrices in subsection 4.4, we use an idea from Ginzburg et al. (2021), who introduce Self-Supervised Document Similarity Ranking (SDR), an unsupervised approach built upon the RoBERTa language model to rank the semantic similarity of a collection of documents to a source (query) document. SDR captures the intuitive fact that for each sentence or paragraph in one document, there should be at least one similar one *anywhere* (obtained by a *max* operation on the rows and columns of the similarity matrix) in the other document if both deal with the same topic. Since SDR is monolingual and is only trained on English texts, we combine its scoring approach with multilingual embeddings obtained from SBERT (Reimers and Gurevych, 2019) and Universal Sentence Encoder (Cer et al., 2018) for this challenge.

4 System Overview

Our system (see Figure 1) first splits the article text into a list of sentences including the title. Then we compute embeddings with SBERT and Universal Sentence Encoder for all sentences and create the respective similarity matrices (similarity of all sentences of one article to all sentences of the other) of all article pairs. In the next step, we apply two different scoring approaches to these matrices: First, we apply simple maximum and average operations to the two matrices for four similarity scores, as proposed by Ginzburg et al. (2021). Second, we feed the matrices into our *SimCNN* to increase score accuracy (see Appendix C). Finally, we combine the five scores into a final score with a random forest regressor to get a stable prediction. Optionally, an additional score for the publishing date distance of both articles can be computed and merged with the random forest result to refine the prediction.

[3] https://github.com/euagendas/semeval_8_2022_ia_downloader

4.1 Preparation of Article Data

We decided that only the title and the text of the article should be relevant for our model, since other features such as keywords, tags or publishing date of the articles are not always available or feasibly retrievable. We use the `sent_tokenize` function of the `nltk` python package to split the text of both articles into a list of sentences and append their title strings to the respective list. This allows us to feed text of arbitrary length into the embedding models.

4.2 Creation of Embeddings and Similarity Matrices

For all sentences in the lists, we create two separate sets of sentence embeddings:

The Universal Sentence Encoder (Yang et al., 2019) always uses the same pre-trained model (see subsection 5.2) to create the embeddings, for which we calculate the recommended *arccos*-based text similarity (Yang et al., 2018), that converts the cosine values into angular distances in $[0, \pi]$ pair-wise between all sentences of both articles.

For SBERT, the model that is selected to create the embeddings depends on the languages of both articles: If both are the same and a model with better performance than the multilingual one is known (see subsection 5.2), this model computes the embeddings for both articles, otherwise the default multilingual model is used. The pair-wise similarity matrix between article sentences is created analogously to the Universal Sentence Encoder, but with cosine similarity instead of *arccos*, because SBERT was optimized for it.

4.3 Architecture of SimCNN

The architecture of our CNN is based on the architecture by Yoon Kim for CNNs for text processing (Kim, 2014) (detailed layer information in Appendix A). As input, we use both precomputed sentence embeddings from the SBERT and the Universal Sentence Encoder model of two articles with lengths x and y. Further, we calculate one similarity matrix from both the SBERT and the Universal Sentence Encoder embeddings since tests have indicated an increased performance using this input. Finally, the input of the CNN is an $x \times y \times 2$ matrix, generated by concatenating the SBERT and the Universal Sentence Encoder similarity matrices. Due to this input, we named our network *SimCNN*. However, the CNN requires

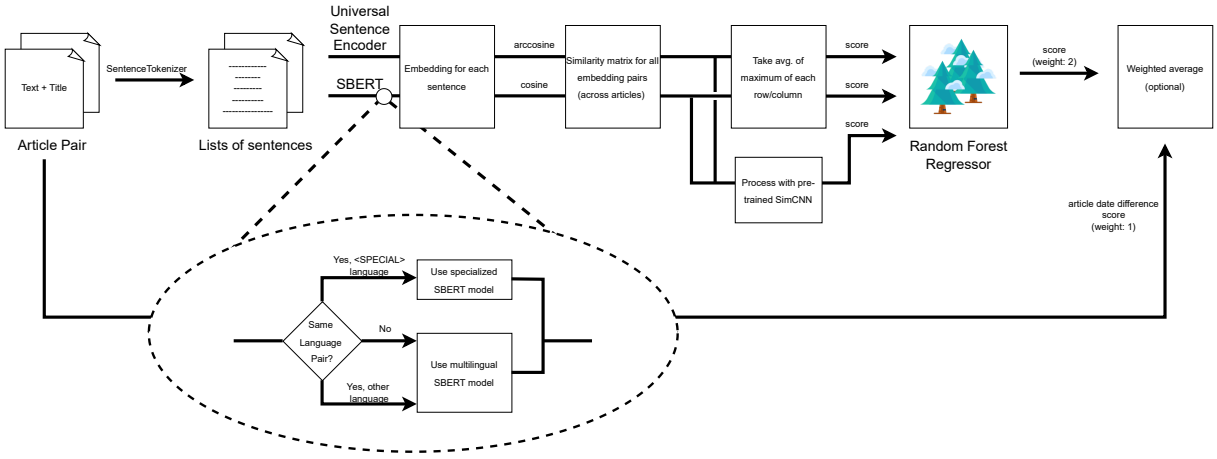


Figure 1: The data flow of the article pairs from raw text to the final score. If both articles are written in the same language, we use an SBERT model that is better than the default multilingual one, if one exists (see subsection 5.2).

a fixed input size within the y-dimension, so we set y to 100 (using zero-padding if $y < 100$ and cropping if $y > 100$). We chose $y = 100$ based on analysis that indicates only a few articles have longer sentence lists and do not set it to the maximum sentence list length since broken lists with up to 1500 sentences can occur.

Adapted from the TextCNN of Yoon Kim, the network consists of seven different convolutions with kernel size $w \times 100$ with $w \in \{2, 3, \ldots, 9\}$, and 128 filters, named sliding window. For extended feature extraction, we added two convolutional blocks before each sliding window SW_w, consisting of five convolutions. Thereby, each convolution of a block SW_k uses a kernel size of $w \times w$, the same padding, and 32 filters within the first block and 64 within the second. Furthermore, each convolution follows a ReLU6 activation and a batch normalization layer and, additionally, a dropout layer ($probability = 0.25$) after the 2nd and 4th convolution. After each sliding window convolution, a max over time pooling extracts the best feature of each filter, so we get a output vector of size 128. Afterwards, a separate linear layer is applied to each vector(mapping to 128 features), followed by another dropout ($probability = 0.5$) and a ReLU6 layer.

For the final prediction, we concatenate all vectors of the different sliding windows to one vector of size 1024 that is fed into five consecutive linear layers. Thereby, the output size for each linear layer is half the input size and every layer uses a ReLU6 activation function and, additionally, a dropout layer ($probability = 0.5$) every second time. Finally, a linear layer with an input size of 32 and output size of 1 predicts a score $s \in [0, 1]$ using a sigmoid activation function. This score is scaled to our target values $s \in [1, 4]$ subsequently.

4.4 Ensemble Scoring of Article Similarity

In addition to being fed into the SimCNN, the *arccos* and *cosine* similarity matrices of the article pairs are processed by taking the average over the maximum value of each row/column of the matrix, similar to the approach by Ginzburg et al. (2021) in section 3. The maximum values yield the most similar sentence in the other article for each sentence and the average operation acts as a proportion of how many sentences of one article share statements with the other article. By doing this for both rows and columns, we obtain two scores for both articles respectively. Figure 2 visualizes this process.

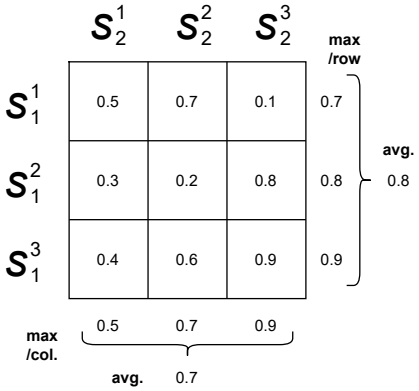


Figure 2: Example of the scoring operation on the *arccos* and *cosine* sentence similarity matrices. The superscript number denotes the number of the sentence in the document, the subscript number marks the document number of the sentence.

Finally, the four matrix scores (article similarity SBERT 1-to-2 and 2-to-1, Universal Sentence Encoder 1-to-2 and 2-to-1) and the SimCNN score are fed into a random forest (Breiman, 2001) regressor from the `scikit-learn` Python package[4]. The forest uses the results of 100 different trained decision trees and takes the average of their predictions, which is the final score reported back by the model.

4.5 Integration of a Publishing Date Score

We also compute scores based on the publishing date distances of each article pair (if available) for the reason that the further the publishing dates are apart, the more likely the articles are about different topics. In accordance with the 4-point score used for the text scoring, we use 10, 20, and 50 days as the boundaries, meaning the score would be ignored if the difference is less than 10 days, between 2 and 3 if the difference is between 10 and 20 days, between 3 and 4 if the difference is between 20 and 50 days, and a hard 4 for a difference of more than 50 days.

If a date score can be calculated and is not ignored, a weighted average of the random forest score (weight 2) and the date score (weight 1) is returned as a final result, otherwise the random forest score is returned. The 10, 20, and 50 day boundaries, as well as the weights, were determined based on experiments on the training data.

5 Experimental Setup

After outlining the model components and their interactions in the previous section, here we cover the data sets, experiments and training processes used to configure the separate parts of the system. The sole evaluation metric for this competition is the Pearson correlation coefficient r (Pearson, 1896; Lee Rodgers and Nicewander, 1988), which describes the linear association between two related variables X and Y. A score close to -1 or 1 implies that a linear equation can express the relationship between the two variables almost perfectly, while a score of 0 indicates no correlation.

5.1 Validation Data Sets

The only labeled data available for training are the 4,964 article pairs provided by the task authors. We reserve a static 10% subset of the training data consisting of 470 pairs of diverse language combinations for our model validation in Table 2, Appendix D and the following experiments. The other 90% of the data are used for the actual training.

5.2 Selection of Pre-Trained Models

The pre-trained multilingual SBERT model *paraphrase-multilingual-mpnet-base-v2* performed best overall on the validation set, so it is used as the default model. We could improve the accuracy for some same-language article pairs by using specialized pre-trained models (**en-en**: *all-mpnet-base-v2*, **es-es**: *distiluse-base-multilingual-cased-v1*, and **fr-fr**: *sentence-transformers/LaBSE*). For Universal Sentence Encoder we used version 3 of the model *multilingual-large*.

5.3 SimCNN Pre-Training

The presented model was implemented in python using `PyTorch` (Paszke et al., 2019) and trained on a consumer graphics card. For updating the parameters in the network, we employed the Stochastic Gradient Decent optimization algorithm using a learning rate of 0.05 with a batch size of 8. Mean squared error was used as the loss function, and additionally, we monitored the mean average error and the Pearson correlation coefficient for performance evaluation. We used early stopping with a patience of 20 epochs to prevent overfitting, so the network was saved when no longer improving in terms of the Person correlation coefficient. Eventually, our network was trained around 30 epochs before overfitting.

5.4 Random Forest Regressor Pre-Training

After the training of the SimCNN model, five similarity scores per article pair of the training data are available through our model pipeline: Two scores for both the SBERT and Universal Sentence Encoder matrices and the SimCNN score. These scores are fed into a random forest (Breiman, 2001) regressor[5] with the provided *Overall* scores of the training data as labels. The random forest is populated with 100 decision trees and uses the squared error as the optimization criterion. Appendix C shows how the performance of the model increases as we provide more data.

[4] `https://scikit-learn.org/stable/modules/ensemble.html#forest`

[5] RandomForestRegressor of the python package `scikit-learn` with version 1.0.2.

Data Set	Language Combination								
	pl-pl	de-de	de-en	fr-fr	ar-ar	en-en	tr-tr	es-es	zh-zh
Validation	0.80	0.75	0.80	0.91	0.76	0.79	0.89	0.81	-
Evaluation	0.63	0.67	0.77	0.74	0.59	0.85	0.66	0.74	0.64

Table 1: Performance of the random forest with date score on selected language combinations. The full table with all pairs is in Appendix D.

Model	Pearson r	
	Validation	Evaluation
SimCNN	0.800	0.699
RF	0.797	0.702
RF + Date	0.800	0.715

Table 2: Performance of our models on our validation set and the final evaluation data. The random forest (RF) is able to improve over just the SimCNN on the evaluation data when combined with the date score.

6 Results

Our system ranked 9th place in the competition with a Pearson score of 0.759.

To find the best models for submission, we tested the three last stages of our system on the aforementioned validation set separately: Just the SimCNN score, the prediction of the random forest regressor and the weighted average of the random forest score and the publishing date score.

The results in Table 2 indicate that the random forest regressor replicates the results of the SimCNN and barely considers the additional information provided. The combination of the random forest with the publishing date score on the other hand improves the results. Further, our models generally predict perceptibly worse scores on the evaluation data than on the training data split.

The language distribution in Appendix B shows a shift from Latin languages and article pairs to vastly different and more complex languages: Chinese-Chinese article pairs account for over 15% of article pairs in the evaluation data. Russian, Polish and Arabic articles also occur often, frequently in combination with other article languages such as English, French and German. This leads to problems in the preparation of the article texts for our system because our sentence tokenizer only supports English or similar texts and many of the new languages have a different structure and alphabet.

When taking a look at the performance per language pair of our model in Table 1, another reason for the score drop-off between training split and evaluation data becomes apparent: The commonly occurring Chinese-Chinese article pairs perform bad with a Pearson score of 0.64. Also, other combinations which previously did well on the split of the training data gave significantly worse results on the evaluation set, indicating that maybe the quality or structure of the new data differs from the earlier samples. Interestingly, our system improved on the English-English evaluation pairs.

All things considered, we achieved satisfying results with just slightly modified publicly available models to create sentence embeddings and a CNN to work with them. The system only depends on an article's text, title and sometimes the publishing date, if it is available, and is still able to achieve a Pearson correlation score of about 0.6 even for the most difficult examined language.

7 Conclusion

Pair-wise sentence comparison is a simple way to calculate the similarity of texts of arbitrary length if suitable multilingual models for sentence embeddings are available. With simple matrix operations like taking the maximum or average and a random forest regression algorithm, good results can be achieved. After introducing the more complex SimCNN and combining it with a score of the publishing dates of the article pairs, our model surpassed a Pearson correlation coefficient of 0.8 in some conditions.

Nevertheless, the current state of the system leaves many things to be improved: The sentence tokenizing currently only reliably works for English and similar languages. With a sentence splitting algorithm, that is capable of differently structured languages like Chinese or Japanese, results on such articles could be greatly improved. Further, our pair-wise sentence similarity matrix approach could be extended to named entities like locations and persons, which we think would also greatly improve accuracy, but for that, a more sophisticated multilingual algorithm for named entity recognition would be needed.

References

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Xi Chen, Ali Zeynali, Chico Q. Camargo, Fabian Flöck, Devin Gaffney, Przemyslaw A. Grabowicz, Scott A. Hale, David Jurgens, and Mattia Samory. 2022. SemEval-2022 Task 8: Multilingual news article similarity. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

Dvir Ginzburg, Itzik Malkiel, Oren Barkan, Avi Caciularu, and Noam Koenigstein. 2021. Self-supervised document similarity ranking via contextualized language models and hierarchical inference. *arXiv preprint arXiv:2106.01186*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Joseph Lee Rodgers and W Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

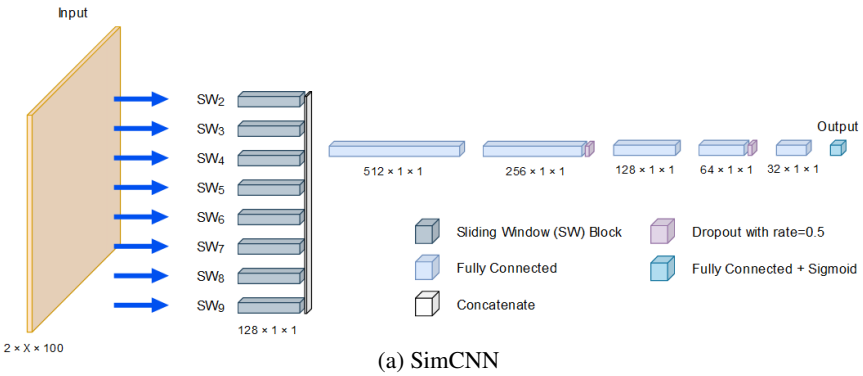
Karl Pearson. 1896. Vii. mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 187:253–318.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

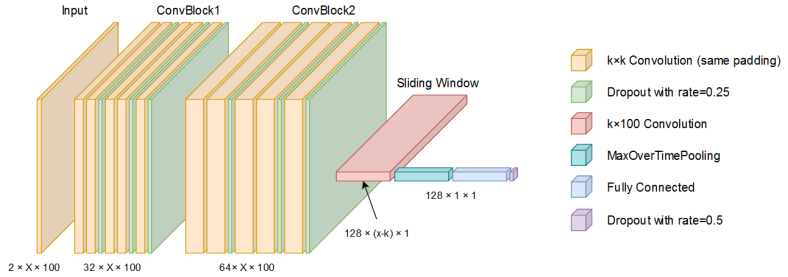
Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernández Ábrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Multilingual universal sentence encoder for semantic retrieval. *CoRR*, abs/1907.04307.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia. Association for Computational Linguistics.

A Architecture of SimCNN



(a) SimCNN



(b) Sliding Window Block with size k

Figure 3: Architecture and methods of the SimCNN based on the TextCNN described by Kim (2014).

Figure 3a shows the architecture of the SimCNN. The SimCNN is based on the architecture by Yoon Kim for CNNs for text processing (TextCNN) (Kim, 2014). It consists of seven sliding window blocks SW_w with window size $w \in \{2, \ldots, 9\}$. Each block SW_w receives the input of size $2 \times x \times 100$ and is structured as illustrated in graphic 3b. First, two convolution blocks, with five convolutions each, are applied to the input. Each convolution of SW_w has a kernel size of $w \times w$ and uses same padding. The convolutions of the first block have 32 filters and the ones of the second block 64. After a convolution, first, a ReLU6 activation layer is applied and, subsequently, a batch normalization layer. Furthermore, after the second, fourth and last convolution, a dropout is executed with a probability of 0.25. Afterwards, the sliding window convolution is applied, using a kernel size of $w \times 100$ and 128 filters, followed by a ReLU6 and a batch normalization layer. Next, a MaxOverTime pooling layer extracts the best feature of each filter. Last of the sliding window block, a fully connected layer using a Relu6 activation function, followed by a dropout with a probability of 0.5, maps to a feature vector of 128 features.

After all sliding window blocks $SW_2, \ldots, SW_9$, the outputs are concatenated to a feature vector of 1024 features. Five fully connected layers with output sizes $512, 256, 128, 64$ and 32 are applied to this vector. A Relu6 is used as activation function after these layers, and a dropout is performed after the second and fourth layer with a probability of 0.5. Finally, a fully connected layer maps the feature vector to one number, that is processed in a sigmoid function, to get the score $s \in [0, 1]$

B Language Distributions of Data Sets

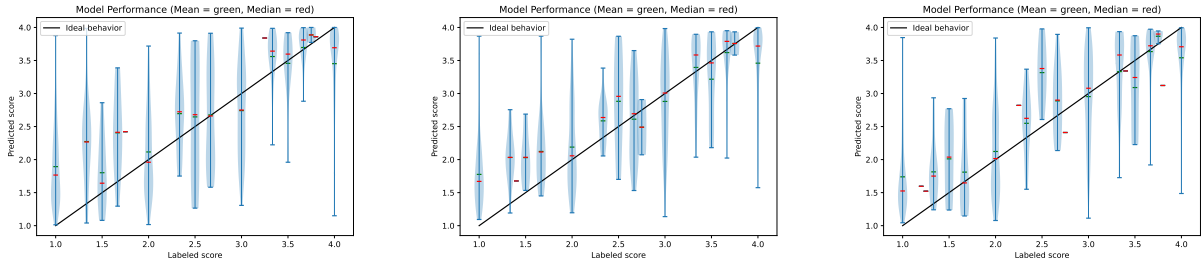
	en-en	de-de	de-en	es-es	tr-tr	pl-pl	ar-ar	fr-fr
Absolute Count	1800	857	577	570	465	349	274	72
Percentage [%]	36.26	17.26	11.62	11.48	9.37	7.03	5.52	1.45

Table 3: In the distribution of training data language pairs, English-to-English is the prevalent combination, with other similar European languages following. With our specialized English-to-English SBERT model, we therefore achieve very good results.

	zh-zh	de-de	es-en	it-it	es-it	ar-ar	ru-ru	tr-tr	es-es
Absolute Count	769	608	496	411	320	298	287	275	243
Percentage [%]	15.69	12.4	10.12	8.38	6.53	6.08	5.85	5.61	4.96
	en-en	pl-pl	zh-en	de-en	de-fr	fr-fr	pl-en	de-pl	fr-pl
Absolute Count	236	224	213	185	116	111	64	35	11
Percentage [%]	4.81	4.57	4.35	3.77	2.37	2.26	1.31	0.71	0.22

Table 4: In the evaluation set, the use of languages is vastly different: Previously unseen combinations (gray), often with complex languages like Chinese, make up a large part of the article pairs the models are scored upon.

C Random Forest Performance with Increasing Amount of Training Data



(a) SBERT (cosine) scores only ($r = 0.703$)

(b) SBERT + Universal Sentence Encoder scores ($r = 0.748$)

(c) SBERT + Universal Sentence Encoder + SimCNN scores ($r = 0.77$)

Figure 4: Performance of the random forest regressor with different inputs on a random train-test-split with 80% training and 20% test data. The more data is provided, the better our Pearson correlation score r gets.

D Model Performance per Language

Model	Language Combination								
	pl-pl	de-de	de-en	fr-fr	ar-ar	en-en	tr-tr	es-es	es-it
V SimCNN	0.89	0.75	0.82	0.83	0.77	0.80	0.81	0.80	-
V Random Forest	0.90	0.75	0.81	0.88	0.76	0.79	0.83	0.80	-
V Publish Date	0.80	0.75	0.80	0.91	0.76	0.79	0.89	0.81	-
E SimCNN	0.61	0.67	0.76	0.69	0.56	0.86	0.65	0.73	0.73
E Random Forest	0.61	0.66	0.76	0.72	0.58	0.85	0.67	0.73	0.73
E Publish Date	0.63	0.67	0.77	0.74	0.59	0.85	0.66	0.74	0.73
	fr-pl	pl-en	de-pl	zh-en	it-it	ru-ru	de-fr	zh-zh	es-en
E SimCNN	0.71	0.77	0.62	0.76	0.76	0.73	0.60	0.64	0.77
E Random Forest	0.70	0.77	0.62	0.75	0.76	0.73	0.58	0.63	0.77
E Publish Date	0.70	0.78	0.62	0.78	0.79	0.74	0.59	0.64	0.80

Table 5: Pearson correlation score of our model configurations on different language pairs. They often do perform significantly better on the validation data (V) than on the evaluation data (E) and seldom vice versa. The additional languages in the evaluation data do not perform noticeably worse than some of the languages already seen in the training set.