"Shakespeare in the Vectorian Age" – An evaluation of different word embeddings and NLP parameters for the detection of Shakespeare quotes

Bernhard Liebl & Manuel Burghardt
Computational Humanities Group
Leipzig University, Germany
{liebl, burghardt}@informatik.uni-leipzig.de

Abstract

In this paper we describe an approach for the computer-aided identification of Shakespearean intertextuality in a corpus of contemporary fiction. We present the Vectorian, which is a framework that implements different word embeddings and various NLP parameters. The Vectorian works like a search engine, i.e. a Shakespearean phrase can be entered as a query, the underlying collection of fiction books is then searched for the phrase and the passages that are likely to contain the phrase, either verbatim or as a paraphrase, are presented in a ranked results list. While the Vectorian can be used via a GUI, in which many different parameters can be set and combined manually, in this paper we present an ablation study that automatically evaluates different embedding and NLP parameter combinations against a ground truth. We investigate the behavior of different parameters during the evaluation and discuss how our results may be used for future studies on the detection of Shakespearean intertextuality.

1 Introduction

Shakespeare is everywhere. Intertextual references to the works of the eternal bard can be found across all temporal and medial boundaries, making him not only the most cited and most performed author of all time, but also the most studied author in the world (Garber, 2005; Maxwell and Rumbold, 2018). But even though countless studies on Shakespearean intertextuality have examined individual aspects of his work by means of close reading, there is still no overview, no big picture, no systematic map of intertextual Shakespeare references for larger text corpora. It is also striking that up to now hardly any computational approaches have been used to detect Shakespeare references on a larger scale. This is all the more surprising as there are many methods in the fields of computer science and natural language processing for determining the *similarity between texts* (Bär et al., 2012), which actually can be seen as a formal definition of intertextuality.

We acknowledge that the full range of intertextual phenomena cannot be covered by mere means of text similarity determination. For our understanding of intertextuality we therefore refer to the definition of Gérard Genette, who defines it as "the effective presence of one text in another text"[1] (Genette, 1993), where we understand the *effective presence* of one text in another to be a more or less objectively recognizable, explicit reference on the surface of the text. Thus, our approach will not be able to detect highly implicit and indirect references that require a lot of domain knowledge and context. The following variant of a well-known quotation from Macbeth (Shakespeare's original variant is given in square brackets) would, however, be objectively recognizable from the text and clearly classified as an intertextual reference:

> By the stinking [pricking] of my nose [thumbs], something evil [wicked] this way goes [comes]. (Terry Pratchett: "I Shall Wear Midnight").

In order to identify such objectively recognizable references in an automated way, we present an approach that investigates the potential of word embeddings (Mikolov et al., 2013) in combination with

[1] Original quote: "la présence effective d'un text dans un autre".

Proceedings of LaTeCH-CLfL 2020, pages 58–68
Barcelona, Spain (Online), December 12, 2020.

other related parameters (e.g., weighting based on POS types, order of POS types, etc.). As we are aware that different word embeddings and NLP parameters will influence the results in very specific ways, we present an ablation study in which we systematically explore the effects of different parameter combinations. We hope that our evaluation will shed some more light on the role of different embeddings and NLP parameters for the detection of intertextuality in the sense of Molnar's desideratum of "interpretable machine learning" (Molnar, 2020).

2 Related work

While text reuse detection (Agirre et al., 2016; Bär et al., 2012) mainly finds application in the context of plagiarism detection and the identification of duplicate websites, there are also productive applications in the digital humanities. One example can be found in the project Digital Breadcrumbs of Brothers Grimm (Franzini et al., 2017), where computational text reuse methods are used to detect motifs of fairy tales across different languages and versions. Labbé and Labbé (2005) present a tool in the intersection of stylometry and text reuse, as they use intertextual distance to classify texts from French literature. Ganascia et al. (2014) describe an approach for the automatic detection of textual reuses in different works of Balzac and his contemporaries . Apart from these example studies, the majority of existing research on text reuse in the Digital Humanities can be located in the field of historical languages and classic studies (Bamman and Crane, 2008; Büchler et al., 2013; Coffee et al., 2012a; Coffee et al., 2012b; Forstall et al., 2015; Scheirer et al., 2014)

While clearly there has been interesting work on the problem of text reuse and intertextuality detection in various areas of the digital humanities, there are only very few studies that use computational methods to detect Shakespeare quotes (Burghardt et al., 2019; Hohl-Trillini, 2019; Molz, 2019). With this paper we contribute to computational intertextuality detection in Shakespeare studies by exploring a set of parameters that can enhance approaches to searching references based on word embeddings.

3 System design: Introducing "The Vectorian"

The Vectorian[2] is a high-performance sentence alignment search engine[3] designed around a number of explicit parameters that model different approaches to scoring sentence similarities. The search engine is also accessible via an internal batch interface for doing large hyperparameter searches, as is the case for the study presented in this paper. Throughout the rest of the section we describe the various parameters of the Vectorian step by step. An overview of the architecture is shown in Figure 1. In a preprocessing step (top half of Figure 1), we first split the search corpus into sentences and detect POS tags for each token of the sentences. Since the corpus in which we search for Shakespeare quotations consists entirely of contemporary literature, the use of *spaCy* 2.3.2 and the *en_core_web_lg-2.3.1* is unproblematic. When running a query – i.e. specific Shakespeare quotes – we run the same preprocessing on the query text but skip sentence splitting and assume a single sentence. We are aware that the selected spaCy model is not optimal for Shakespeare quotes because it does not reproduce details of Early Modern English. However, having looked at a number of samples and the assigned POS tags, we think it works good enough for this first pilot study. We plan to implement a language model that is more specific to Shakespeare as a future step. Next, the Vectorian computes similarities between tokens from the query and the search corpus based on precomputed contemporary word embeddings like *fasttext*. At this point, only word tokens are used and punctuation is ignored completely. For the Shakespearean text in the query, contemporary word embeddings pose an obvious challenge due to shifts in word meaning. We currently do not leverage historical word embeddings like *HistWords* (Hamilton et al., 2016), but plan to incorporate these in future extensions of this work. After preprocessing and storing the data in an efficient in-memory format suitable for high-performance realtime searches over a large corpus (see bottom right in Figure 1), we compute alignments based on similarity scores between the tokens (see bottom left in Figure 1). Ultimately, scores are derived from the embeddings and controlled by various parameters – the details are contained in a pipeline we refer to as *SIM_FULL* (see big box in Figure 1) and which will be described

[2] `https://github.com/poke1024/vectorian/tree/v4`

[3] For a similar approach see Manjavacas et al. (2019).

in more detail in the next sections (see the right half of Figure 2). Given the token similarity scores, we find optimal alignments on the sentence level using the Waterman-Smith-Beyer algorithm (Waterman et al., 1976), which we leverage through an optimized and highly customizable implementation[4].

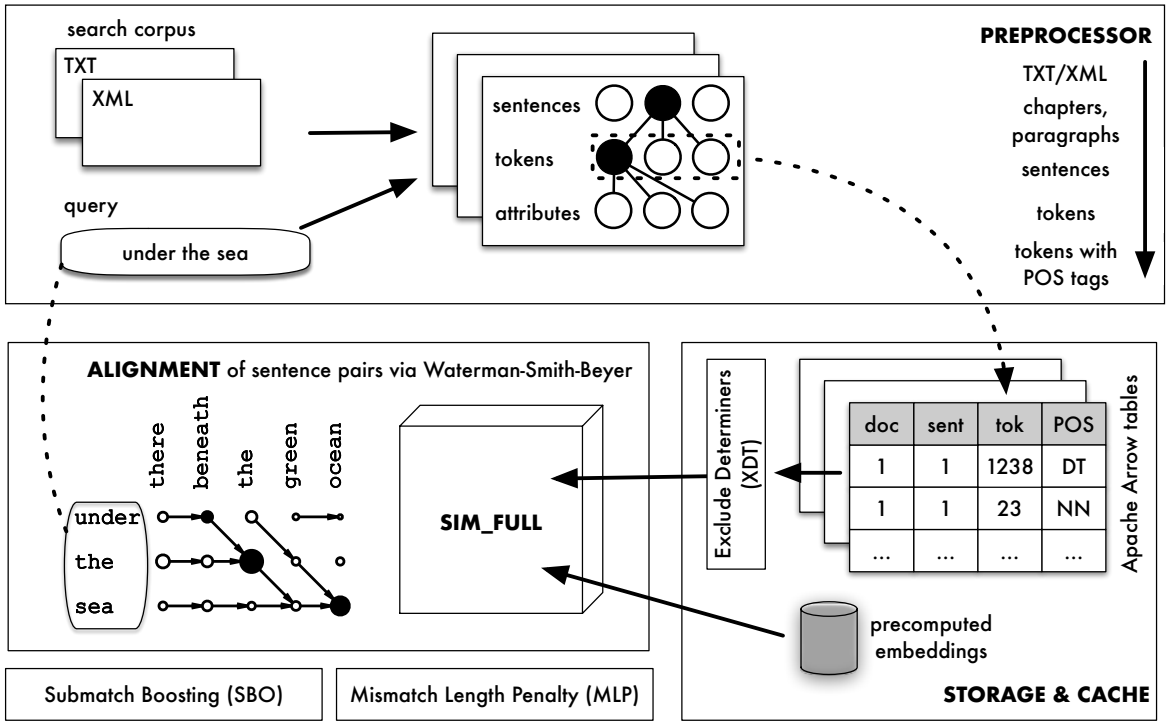

Figure 1: Simplified overview of overall architecture.

In the following we provide details on the ten parameters that are shown in Figures 1 and 2. These parameters tackle three different areas of similarity measures we found worth considering. (1) Three parameters (*XDT*, *SPW*, *PMP*) are concerned with how exactly part of speech (POS) tags contribute to the similarity computation. (2) Five parameters (*EMI*, *ESM*, *IFS*, *SIF*, *SIT*) are concerned with how to exactly compute a scalar similarity score from word embeddings. (3) Finally, two parameters (*MLP*, *SBO*) control details of how alignments are scored.

3.1 Parameters for POS Tag Influence

Exclude Determiners (XDT). A Boolean parameter. If enabled, it will perform a search as if all tokens in query and corpus that have been tagged with the universal POS tag[5] *DET* have been removed.

Semantic POS Weighting (SPW). A numeric parameter between 0 and 1. If set to 0, the similarity between two tokens is directly computed from the configured embedding metrics. If set to 1, token pair scores are weighted with the corpus token's PennTree POS tag (Taylor et al., 2003) using the weights given by Batanović and Bojić (Batanović and Bojić, 2015). As a result, and following the argumentation of Batanović and Bojić, some tokens (e.g. *VBP*) will have a greater influence on the final similarity scores than others (e.g. *NN*). If w is a token's weight according to Batanović and Bojić, we compute an overall token weight as $(1 - SPW) + (SPW \times w)$. Therefore, reducing *SPW* will gradually equalize these weights.

POS Mismatch Penalty (PMP). A numeric parameter between 0 and 1 that penalizes the similarity scores of token pairs if their universal POS tags do not match – i.e. giving tokens a lower score, even if the embedding considers them very similar. If set to 1, any POS mismatch will reduce the similarity score to 0, regardless of the embedding score. A value of 0 completely ignores POS mismatches.

[4] https://github.com/poke1024/simileco

[5] https://universaldependencies.org/u/pos/

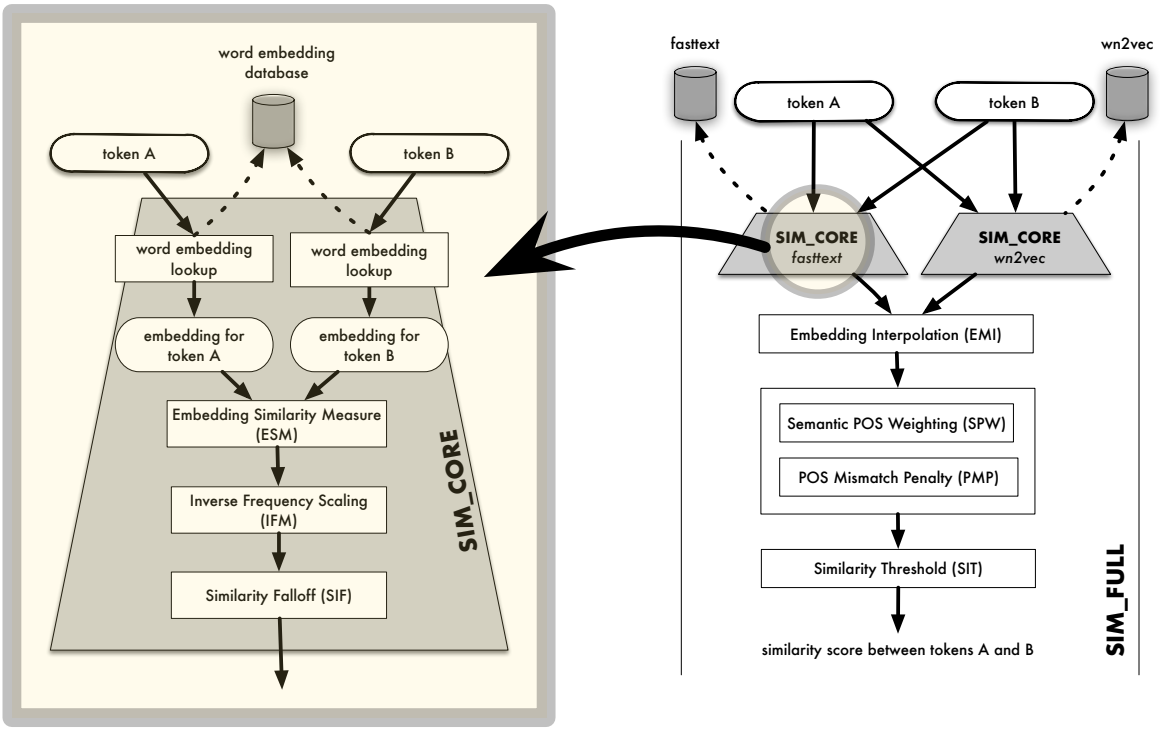

Figure 2: Steps and parameters involved in computing the similarity of two tokens. Details of the *SIM_FULL* module from Figure 1 are shown on the right side. The *SIM_CORE* sub module is shown on the left side. Solid lines show data flow, dotted lines show lookups.

3.2 Parameters for Embedding Similarity Computation

Embedding Interpolation (EMI). A numeric parameter between 0 and 1 that specifies a mixing of two embeddings. For our experiments, we use the official pretrained *fasttext* embeddings (Mikolov et al., 2018) and *wnet2vec* (Saedi et al., 2018) embeddings[6]. These two candidates were chosen as typical proponents of very different kinds of precomputed word embeddings: whereas *fasttext* is an established iteration of the *word2vec* school that are trained on unstructured corpora, *wnet2vec* is based on "ontological graphs" (Saedi et al., 2018), namely WordNet. By combining these embeddings, we hope to investigate if combining very different approaches can yield a benefit.

Our mixing computes a maximum similarity: if t is the value for *EMI*, we compute the mixed similarity s' from two original similarities s_1 and s_2 as $s' = max(2s_1(1-t), 2s_2t)$. Therefore, a value of 0 indicates that only *fasttext* scores are used, whereas a value of 0.5 indicates that for each token pair, the maximum similarity found in either embedding is used.

Embedding Similarity Measure (ESM). Specifies how two word vectors from an embedding are turned into a scalar similarity score. The Vectorian supports three strategies: cosine similarity, the *nonit-erative contextual dissimilarity measure* by Jegou et al. (Jegou et al., 2010) with a neighborhood size of 100 elements and finally the rank-based similarity metric by Santus et al. (Santus et al., 2018). We refer to these strategies as *cosine*, *nicdm* and *apsynp* respectively.

Inverse Frequency Scaling (IFS). A numeric parameter between 0 and 1 that weights similarity scores with a token's inverse probability of occurence in a typical corpus. If set to to 0, no such weighting takes place, if set to 1, the similarity score for rarer words will get boosted. Specifically, if p is a token's negative log probability and s is the similarity, a new similarity score s' is computed as

[6]These were computed by running `https://github.com/nlx-group/WordNetEmbeddings` on 58,492 unique words from 66 novels that were part of the search corpus. As not all of these words were present in WordNet, the resulting embedding covers only 27,718 words.

$$s' = s * (-p)^{IFS} \tag{1}$$

This is a rather simplistic approach, as we currently do not model approaches such as tf-idf (Leskovec et al., 2020).

Similarity Falloff (SIF). A numeric parameter between 0 and 1 that rescales similarity scores before POS weighting. This can help to increase the distance between high and low scores. Each similarity score s is rescaled to s^{SIF}. A value of 1 obviously disables rescaling.

Similarity Threshold (SIT). A numeric threshold between 0 and 1 that is applied to similarity scores after POS weighting. Any score below this value will be set to 0 for further processing. This has the effect of reducing noise from unwanted low similarities.

3.3 Parameters for Alignment Scoring

Mismatch Length Penalty (MLP). An integer value indicating that length of a mismatch – in number of tokens – that will reduce the similarity score by 0.5 – the maximum possible score being 1. Low values will enforce no or only short mismatches, whereas higher values allow longer runs of mismatching tokens. The score penalty is modelled as an exponential function. For a mismatch of length n we compute the penalty as

$$1 - 2^{-(\frac{n}{MLP})} \tag{2}$$

Submatch Boosting (SBO). A numeric parameter that models the score if only parts of the query get matched. Specifically, if a query contains n tokens of which m have been matched, and each individual token has a maximum score of 1 and the sum of matched token scores is s, then we compute an overall score s' using a discount factor α as follows[7]:

$$\alpha = \left(\frac{n-m}{n}\right)^{SBO} \tag{3}$$

$$s' = \frac{s}{m + \alpha(n-m)} \tag{4}$$

A value of 0 therefore indicates that no special submatch weighting takes place. Values larger than 0 decrease the impact of non-matched tokens in the overall scores, thereby making partial matches obtain higher scores.

4 Evaluation design

In this section we present an ablation study in which we automatically test different combinations of the parameters that were described in the previous section, in order to investigate how they influence the results. The ground truth required to carry out such an evaluation was derived from Molz (2019), who conducted a comprehensive study to identify references to Shakespeare in a corpus of postmodern fiction using a mixture of close and distant reading. We took a subsample of this work, which contains 73 quotes from one of Shakespeare's most popular plays: *Hamlet*. The rationale for taking only a subsample is that the ground truth cannot be considered 100% comprehensive, as was shown in related studies with the dataset (Bryan et al., 2020). We kept the sample size small in order to simplify the task of recognizing new true positives identified by the Vectorian. The 73 quotes are distributed among 31 novels that have a total size of 4,2 million tokens. Sticking to the search engine metaphor introduced in Section 2, we will treat each of the 73 Hamlet quotes as a query that is searched for in the collection of novels. In the evaluation study, each query is assigned an unranked set of expected results (as documented in the ground truth) in the corpus of novels. Each result refers to one specific sentence in a novel. Some queries have multiple expected results, e.g. for "There are more things in heaven and earth ..." our ground truth records 8 occurences in different novels. However, most queries have only 1 or 2 matches in our ground

[7]For simplicity, we give the unweighted case, though our implementation includes POS weighting here.

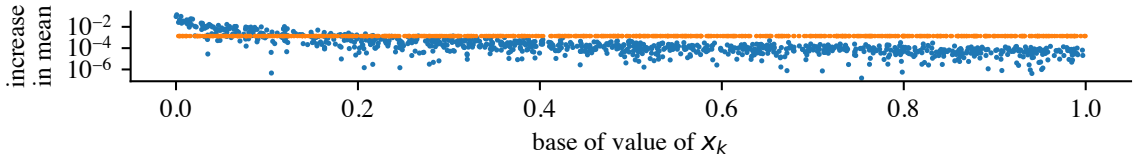

Figure 3: Scatter plot of absolute change in harmonic mean (blue) and arithmetic mean (orange) over random scores $x_1, ..., x_n$ when updating a single input score x_k by an improvement ϵ of 0.1. Y axis is logarithmic. The arithmetic mean always increases by a constant $\frac{\epsilon}{n}$ regardless of x_k's value, so in an optimizer it encourages increasing low and high scores similarly. The harmonic mean on the other hand tends to weigh the same improvement in a low input (left) considerably higher than in a high input (right) and therefore encourages increasing low inputs.

truth[8]. In total, our ground truth contains 149 result sentences for 73 unique queries. We measured the performance of each query by computing the *Discounted Cumulative Gain* (DCG) (Järvelin and Kekäläinen, 2002) against the unranked[9] ground truth for the first 50 results retrieved. DCG is a standard measure to rank the quality of a result set in information retrieval, where documents from higher ranks contribute more to the overall *gain* and documents at lower ranks contribute less, i.e. they are *discounted*.

Since a full grid search was not feasible for our parameter space, we ran an *Optuna* (Akiba et al., 2019) optimizer with the objective of maximizing a total performance score. This total performance score is computed as a mean over the DCGs of all queries. However, since some queries expect more results than others – and therefore the ideally obtainable DCGs for different queries vary – such a composite score only makes sense if all contributing DCGs have been scaled to a fixed range. Therefore we employed the commonly used formulation of Normalized DCGs (nDCGs) (Manning et al., 2008) to normalize the score for each query into the range between 0 and 1. To summarize these considerations: we used nDCG@50 for each query and then computed a mean to obtain a total performance score. We ran a first optimization with the objective of maximizing the commonly used *arithmetic* mean over all query nDCGs as the maximizing objective, and a second independent optimization with the objective of maximizing a *harmonic* mean of query nDCGs. The *harmonic* mean may seem like an unusual choice here, as its use in information retrieval is typically limited to the F-score (Manning et al., 2008). The reason we use it in this scenario, is the distribution of query difficulty in our ground truth and the specific characteristics of the *harmonic* mean. We found a very strong negative skew due to a high number of rather simplistic queries in our data. There are about 50% of queries that relate to verbatim or near-verbatim quotes of text from Shakespeare, which means that these are rather easy to detect from a text reuse perspective. About 10% of the queries on the other hand rely on implicit knowledge and are probably not easily found by the Vectorian, which relies entirely on explicit language features. The remaining 40% of the queries are neither trivial nor out of reach for the Vectorian. We therefore put them into the category *hard but feasible*. Optimizing on the *arithmetic* mean carries the risk of micro-optimizing the bulk of easy queries to an nDCG of 100%, but finding no good nDCGs for the few but more interesting queries. In order to encourage good nDCGs for *hard but feasible* queries, the *harmonic mean* seems to be a reasonable alternative. As Figure 3 illustrates, it tends to improve by higher values when low inputs get increased.

In our evaluation study, we ultimately ran optimizations for both types of means with 1,500 trials using a default configuration with *tree-structured Parzen estimators* (Akiba et al., 2019). As a caveat, it must be noted that due to the small size of our ground truth sample, our evaluation did not have a dedicated validation set. Since the parameters in our system are few and quite restricted, however, we believe the risk of overfitting is rather low. We interpret our results as a simplistic model that represents deeper characteristics of the query-result relationships given in our ground truth.

[8]The exact distribution is $44:1$ (i.e. 44 queries with one result) , $13:2, 5:3, 4:4, 2:5, 2:6, 2:8, 1:10$.

[9]I.e. a full score is obtained if the specified ground truth results are retrieved first, regardless of their internal order.

5 Results and Discussion

The best configuration found with *Optuna* produced nDCGs of 77.6% and 75.2% for the *arithmetic* (A) and *harmonic* (H) mean optimization objectives respectively. Since the decision what constitutes a quotation in some cases cannot be made on the language level alone and thus is highly subjective (Molz, 2019), we believe that these scores can be interpreted as a fairly good performance. The system also produced a small number of new true positive matches, which could be confirmed to be valid[10]. The specific parameter values for the best configurations are given in Table 1, together with the parameter domains that were searched. Note that *Optuna* does not use an initial starting or seeding configuration.

Parameter	Distribution	Domain	*A*	*H*
Exclude Determiners	categorical	*false*, *true*	*false*	*true*
Embedding Interpolation	uniform	$0 \leq x \leq 1$	1.0	0.41
Embedding Similarity Measure	categorical	*cosine*, *nicdm*, *apsynp*	*cosine*	*nicdm*
Inverse Frequency Scaling	uniform	$0 \leq x \leq 1$	0.0	0.96
Similarity Falloff	uniform	$0 \leq x \leq 1$	0.93	0.39
Semantic POS Weighting	uniform	$0 \leq x \leq 1$	0.46	0.09
POS Mismatch Penality	uniform	$0 \leq x \leq 1$	0.77	0.43
Similarity Threshold	uniform	$0 \leq x \leq 1$	0.73	0.83
Mismatch Length Penalty	int	$0 \leq x \leq 10$	1	1
Submatch Boosting	uniform	$0 \leq x \leq 5$	0.24	0.14

Table 1: Investigated parameter domains (first three columns) and best configurations found through *Optuna* search for *arithmetic* mean and *harmonic* mean (last two columns). Numerical values are rounded to nearest multiple of 0.01.

Unfortunately, due to the skewed nature of the query problem in our ground truth, the mean value of the nDCGs tells us only little about the performance of the two variants with respect to different types of queries. Figure 4 therefore gives a detailed histogram of the query nDCGs. As argued in the evaluation design, the distribution of the *hard but feasible* queries with low scores turned out differently indeed. Most notably for *H*, we see a salient group of (orange) queries scoring between 0.4 and 0.7, while the queries scoring at nDCG 0 and 1 both have been slightly diminished. In general, this is what we hoped for. The downside however is that the beneficial *arithmetic* (blue) peak between 0.7 and 0.8 is now gone, i.e. we have lost this score for some queries.

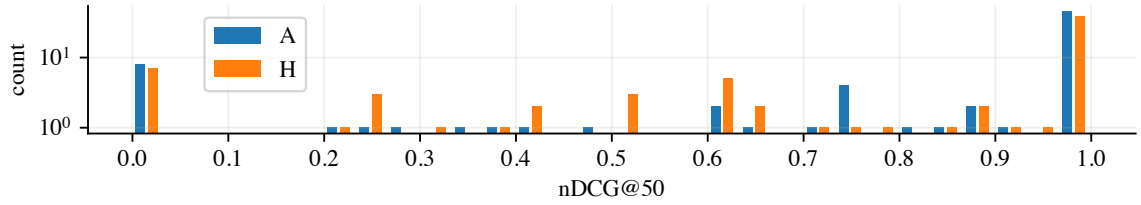

Figure 4: Histogram of nDCGs for variants *A* and *H*. Y axis is logarithmic.

Figure 5 shows the performance of both variants in terms of quantiles. Both variants operate optimally on the maximum 100% nDCG level for easy queries that are located at the quantiles above 0.5. Between the 0.15 and 0.5 quantiles however, the *arithmetic* mean variant performs better. On the other hand, the *H* variant does not show the dip below the 0.15 quantile, which seems to give it slightly better performance for some difficult queries.

[10] After marking these as correct, the nDCGs changed to 81.2% (for *A*)) and 78.9% (for *H*).

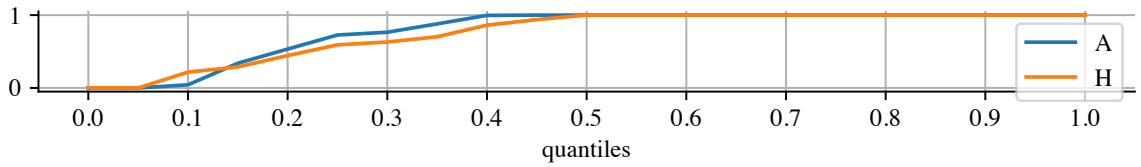

Figure 5: Quantiles of nDCGs for variants *A* and *H*.

We now discuss the parameters' importance by performing an ablation study on each of them, starting with the *harmonic* variant (see Figure 6), which shows a surprising combination[11]. *EMI* and *PMP* have no effect at all – any value produces the same optimal results – and nearly the same is true for *ESM*. In other words, the whole embedding pipeline seems to be irrelevant for the search results. Furthermore, *SPW*, *SIF*, and *IFS* are all basically *no-ops*. The only three salient choices are a *SIT* above 0.8, a *MLP* of 1 – that shows an interesting option for extending it up to 4 – and a slightly elevated *SBO* value. In summary, large parts of the Vectorian engine have been turned off in this case in order to facilitate a specific kind of search, namely looking for alignments of tokens without using any POS or embedding information.

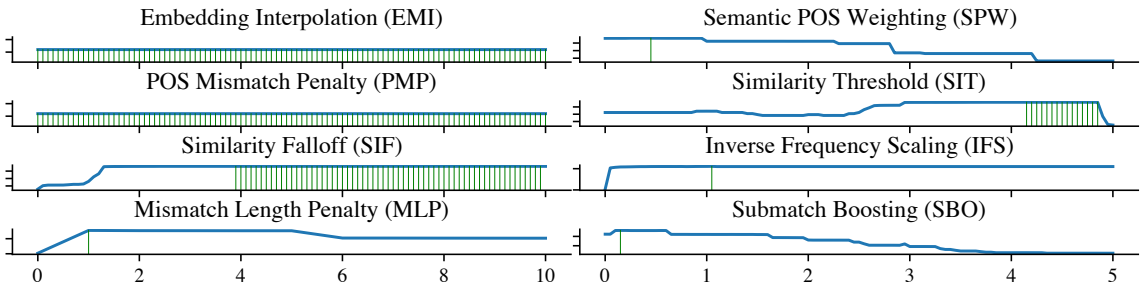

Figure 6: Ablations for various parameters of best configuration found through *harmonic* means of nDCGs. The *x* axis shows parameter values, the *y* axis shows achieved *harmonic* mean nDCG@50. Different plots expose different *y* ranges. Maximum values obtained per parameter are shaded green.

In contrast to the results for *H*, an ablation on the *A* variant shows more intertwined settings (see Figure 7). We only see one *no-op* with *SIF*, all other parameters are meaningful. *SIT*, *MLP* and *SBO* are somewhat similar to the *H* variant. The embedding and POS parameters are quite different. *SPW* has its maximum benefit between 0.4 and 0.5, meaning it should neither be turned fully on nor off[12]. The plot for *PMP* suggests that a POS mismatch should *always* override the computed similarity from an embedding and count that token pair as *not similar*. For *EMI*, we observe that the best value is not 1, as inferred in the *Optuna* search, but 0.55. This seems to confirm our assumption that mixing very different types of embedding can be beneficial. Without mixing, *wnet2vec* at 1 outperforms *fasttext* at 0. For *ESM*, *cosine* performs slightly better than *nicdm*. Both measures perform considerably better than *apsynp*.

The overall results are rather counterintuitive: If our distinction into easy and hard queries is correct, then the result would mean that the retrieval of easy queries benefits from embeddings and syntactic markers, whereas the retrieval of hard queries does not. To shed some light on what is really happening here, we looked at Recall at K (Manning et al., 2008) for both cases (see Figure 8). As expected when optimizing for nDCG, *A* excels in bringing many correct results to the very front of the result list (see $K < 3$). *H* on the other hand indeed focuses on queries with low scores – especially results that are not or hardly found at all. Starting at roughly $K = 20$ this clearly shows: *H* starts to recall more correct

[11]Note that the green areas indicate those parameter values that produced the best reproduction of the given ground truth.

[12]The plot suggests that applying the weights from Batanović and Bojić (2015) fully – through a parameter value of 1 – would harm the search performance considerably.

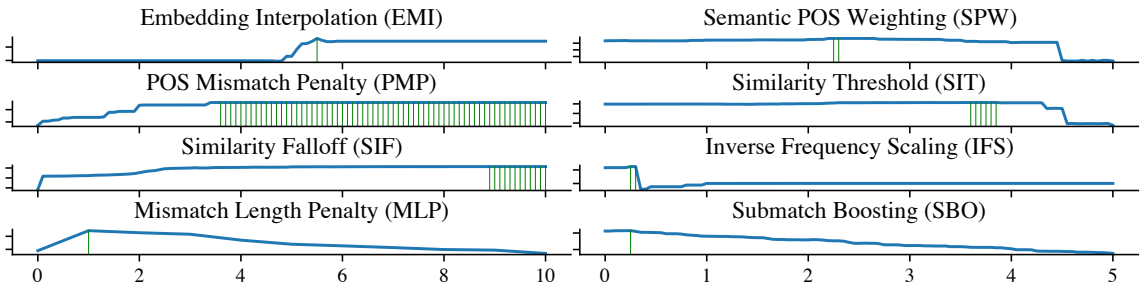

Figure 7: Ablations for various parameters of best configuration found through *arithmetic* mean of nDCGs. The *x* axis shows parameter values, the *y* axis shows achieved *arithmetic* mean nDCG@50. Different plots expose different *y* ranges. Maximum values obtained per parameter are shaded green.

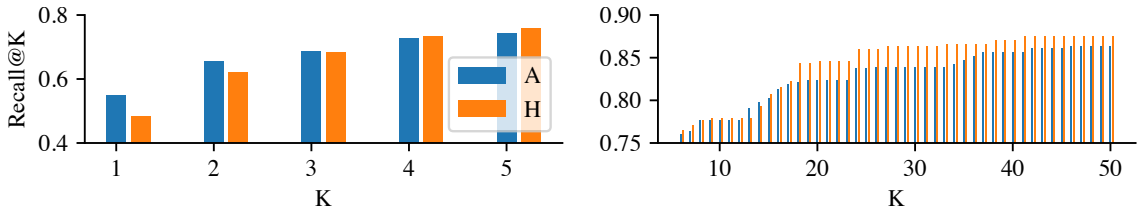

Figure 8: Recall@K for *A* and *H* variants. The *y* range differs on the left and on the right.

results than *A*. Closer inspection of the results shows that many of these *hard* queries contain only one or two tokens that exhibit any form of semantic alignment to the sentences in the ground truth. In other words, large parts of the query are ignored by the alignment engine when trying to find a match[13]. At the same time, the few tokens that do match are usually verbatim words from Shakespeare's texts. This observation explains *H*'s strategy to disable large parts of the Vectorian pipeline and basically build a verbatim single word matcher to cope with these queries.

6 Conclusion

We have investigated the optimal configuration of various explicit parameters in a text reuse detection pipeline and showed that it is able to achieve nDCGs of roughly 80% on a rather difficult test set. While for some queries focusing on the interplay of word embeddings, POS tags and alignments is optimal, other queries seem to benefit from turning off these features. We have demonstrated, how these choices are generated naturally by maximizing *arithmetic* and *harmonic* means of nDCG scores. Our analysis uncovered important ideas of what makes queries *hard* for our current architecture and indicates the need for a ground truth that is classified and balanced in terms of difficulty. As Molz (2019) described considerably more Shakespeare references in his study, we plan to enhance the ground truth accordingly and classify the quotes according to different categories (e.g. verbatim quote, semantic paraphrase, changed word order, etc.). We hypothesize that different types of quotes will result in different optimal parameter combinations. This will be investigated in more detail in a follow-up study, where we will also look into other types of embeddings and explore single parameters in more detail.

[13] A similar observation is true for any human expert, who, however, would have the advantage of knowing the quote's broader context from neighboring sentences.

References

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, Anchorage AK USA, July. ACM.

David Bamman and Gregory Crane. 2008. The logic and discovery of textual allusion. In *In Proceedings of the 2008 LREC Workshop on Language Technology for Cultural Heritage Data*.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2012. Text reuse detection using a composition of text similarity measures. In *Proceedings of COLING 2012*, pages 167–184, Mumbai, India, December. The COLING 2012 Organizing Committee.

Vuk Batanović and Dragan Bojić. 2015. Using Part-of-Speech Tags as Deep Syntax Indicators in Determining Short Text Semantic Similarity. *Computer Science and Information Systems*, 12(1):1–31, January.

Maximilian Bryan, Manuel Burghardt, and Johannes Molz. 2020. A computational expedition into the undiscovered country - evaluating neural networks for the identification of hamlet text reuse. *Proceedings of the 1st Workshop on Computational Humanities Research (CHR)*.

Marco Büchler, Annette Geßner, Monica Berti, and Thomas Eckart. 2013. Measuring the influence of a work by text re-use. *Bulletin of the Institute of Classical Studies. Supplement*, pages 63–79.

Manuel Burghardt, Selina Meyer, Stephanie Schmidtbauer, and Johannes Molz. 2019. *"The Bard meets the Doctor" – Computergestützte Identifikation intertextueller Shakespearebezüge in der Science Fiction-Serie Dr. Who.* Book of Abstracts, DHd.

Neil Coffee, Jean-Pierre Koenig, Shakthi Poornima, Christopher Forstall, Roelant Ossewaarde, and Sarah Jacobson. 2012a. The Tesserae Project: intertextual analysis of Latin poetry. *Literary and Linguistic Computing*, 28(2):221–228, 07.

Neil Coffee, Jean-Pierre Koenig, Shakthi Poornima, Roelant Ossewaarde, Christopher Forstall, and Sarah Jacobson. 2012b. Intertextuality in the digital age. *Transactions of the American Philological Association (1974-)*, pages 383–422.

Christopher Forstall, Neil Coffee, Thomas Buck, Katherine Roache, and Sarah Jacobson. 2015. Modeling the scholars: Detecting intertextuality through enhanced word-level n-gram matching. *Digital Scholarship in the Humanities*, 30(4):503–515.

Greta Franzini, Emily Franzini, Gabriela Rotari, Franziska Pannach, Mahdi Solhdoust, and Marco Büchler. 2017. The digital breadcrumb trail of brothers grimm. *Poster at the DATECH conference, Göttingen*.

Jean-Gabriel Ganascia, Peirre Glaudes, and Andrea Del Lungo. 2014. Automatic detection of reuses and citations in literary texts. *Literary and Linguistic Computing*, 29(3):412–421, 06.

Marjorie Garber. 2005. *Shakespeare After All*. Anchor Books.

Gérard Genette. 1993. *Palimpseste. Die Literatur auf zweiter Stufe.* Suhrkamp.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

Regula Hohl-Trillini. 2019. *'Look thee, I speak play scraps': Digitally Mapping Intertextuality in Early Modern Drama*. Oxford University, Bodleian and Folger Libraries, July.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October.

Herve Jegou, Cordelia Schmid, Hedi Harzallah, and Jakob Verbeek. 2010. Accurate Image Search Using the Contextual Dissimilarity Measure. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 32(1):10.

Cyril Labbé and Dominique Labbé. 2005. A Tool for Literary Studies: Intertextual Distance and Tree Classification. *Literary and Linguistic Computing*, 21(3):311–326, 10.

Jurij Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2020. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, third edition edition.

Enrique Manjavacas, Brian Long, and Mike Kestemont. 2019. On the Feasibility of Automated Detection of Allusive Text Reuse. *arXiv:1905.02973 [cs]*, May.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

Julie Maxwell and Kate Rumbold. 2018. *Shakespeare and Quotation*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Christoph Molnar. 2020. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/.

Johannes Molz. 2019. *A close and distant reading of Shakespearean intertextuality*. Ludwig-Maximilians-Universität München, Juli.

Chakaveh Saedi, António Branco, João António Rodrigues, and João Silva. 2018. WordNet Embeddings. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 122–131, Melbourne, Australia. Association for Computational Linguistics.

Enrico Santus, Hongmin Wang, Emmanuele Chersoni, and Yue Zhang. 2018. A Rank-Based Similarity Metric for Word Embeddings. *arXiv:1805.01923 [cs]*, May.

Walter Scheirer, Christopher Forstall, and Neil Coffee. 2014. The sense of a connection: Automatic tracing of intertextuality by meaning. *Digital Scholarship in the Humanities*, 31(1):204–217, 10.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The Penn Treebank: An Overview. In Nancy Ide, Jean Véronis, and Anne Abeillé, editors, *Treebanks*, volume 20, pages 5–22. Springer Netherlands, Dordrecht.

M.S Waterman, T.F Smith, and W.A Beyer. 1976. Some biological sequence metrics. *Advances in Mathematics*, 20(3):367–387, June.