Supervised Rhyme Detection with Siamese Recurrent Networks

Thomas Haider
MPI for Empirical Aesthetics, Frankfurt
IMS, University of Stuttgart
thomas.haider@ae.mpg.de

Jonas Kuhn
IMS
University of Stuttgart
jonas.kuhn@ims-stuttgart.de

Abstract

We present the first supervised approach to rhyme detection with Siamese Recurrent Networks (SRN) that offer near perfect performance (97% accuracy) with a single model on rhyme pairs for German, English and French, allowing future large scale analyses. SRNs learn a similarity metric on variable length character sequences that can be used as judgement on the distance of imperfect rhyme pairs and for binary classification. For training, we construct a diachronically balanced rhyme goldstandard of New High German (NHG) poetry. For further testing, we sample a second collection of NHG poetry and set of contemporary Hip-Hop lyrics, annotated for rhyme and assonance. We train several high-performing SRN models and evaluate them qualitatively on selected sonnetts.

1 Introduction

Rhyme is a pervasive style device in historical poetry and Hip-Hop, but previous research has relied on small, handcrafted datasets. A reliable system for the detection of rhyme would allow large scale analyses, opening several directions for research. Given word pronunciations and a definition of rhyme, the problem is fairly easy. However, for domain specific or historical data, obtaining precise pronunciation information is a challenge (Katz, 2015). Also, a narrow definition of *perfect* rhyme[1] disregards frequently used and accepted deviations, as in imperfect rhyme (Primus, 2002) (Berg, 1990) or the related sonic devices assonance, consonance and alliteration. Information on the phonological similarity of two rhyme words can be used e.g. for the reconstruction of historical pronunciation (List et al., 2017) or the analysis of sonic pattern (McCurdy et al., 2015). A rhyme detection on grapheme strings is a step in this direction.

Previous research on the detection of rhyme is scarce. Reddy and Knight (2011) employed Expectation Maximization (EM) to predict (generate) the most probable scheme (e.g. *'abba'*) of a stanza. We use a supervised approach to rhyme detection to model the properties of rhyme itself. An accurate similarity measure between words would benefit an EM. Siamese Recurrent Networks are adept for rhyme, as they learn a (non-linear) similarity metric on variable length character sequences. This metric can be used to gauge the degree of imperfection in a rhyme, and by threshold, for a binary classification. We describe our architecture in section 3.2, followed by experiments and a qualitative error analysis to find the best detection system across languages (German, English, French) and domains (poetry, Hip-Hop).

We compiled three corpora with (gold) rhyme annotation, including (i) 116 German Hip-Hop lyrics, (ii) a diachronically balanced sample of 1.948 New High German poems, and (iii) 156 poems of school canon, covering the same period. Their sampling and annotation process will be discussed in section 2.

2 Rhyming Corpora

We describe the creation and annotation efforts of three corpora of lyric text 1.) DTR, a diachronically balanced sample of poems from the German text archive, containing 1,948 poems, 2.) ANTI-K, a very

[1] i.e. *identical* pronunciation of word segments starting from the last accented vowel (Fabb, 1997)

Proceedings of Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, pages 81–86
Santa Fe, New Mexico, USA, August 25, 2018.

	#Poems	Stanzas	Verses	Words	Rhyme Pairs	Authors	Poems/Author	
Corpus							avg.	median
HIPHOP	116	789	6,390	52,759	2,489	41	2.8	3
ANTI-K	156	731	3,603	20,412	1,440	50	3.1	1
DTR	1,948	8,147	40,523	251,730	13,785	57	33.5	31
DTL	28,275	74,523	507,663	3,557,632	?	75	377	139

Table 1: Size statistics of German poetry corpora

clean (and small) set of 156 poems of school canon, and 3.) HIPHOP, 116 Hip-Hop lyrics. All corpora were annotated for rhyme schema on stanza level. We extract rhyme pairs and non-rhyme pairs based on schema indices. Given a schema, matching indices ('aa') yield a rhyme pair, while non-matching indices ('ab') yield a non-rhyming pair. For a schema 'abba' we retrieve two rhymes and four non-rhymes. See table 1 for size statistics of the corpora. We will publish our datsets on github.[2]

2.1 DTR: German Text Archive Rhyme Gold

The German text archive (Deutsches Textarchiv[3], or **DTA**) is a corpus of New High German (NHG) texts. It is appealing because the editions are well curated, it contains gold POS (STTS) annotation, orthographic corrections and faksimiles, and the selection of authors and editions represents a small Canon for NHG poetry. DTA is balanced over four broad genres and includes 131 documents of *lyric poetry* (DTL). These documents span over 75 authors, where each document has only one author, typically in form of an anthology or collection. The DTL corpus contains in total 28,275 poems, stretching over 3,557,632 word token.

To create a rhyme schema goldstandard **DTR** we draw a diachronically balanced sample from DTL. We divide our timeline by 20-year wide slots (1630 - 1650, ..., 1790 - 1810, ..., 1890 - 1913), aiming at 500 stanzas per slot (allowing $\pm$ 10%). We left the original poems intact, sampling until the desired number of stanzas was fulfilled with complete poems. Additionally, an author needs to contribute enough poems within a std. deviation from the mean. No stanzas longer than one standard deviation over 12 lines (24) were allowed. DTR eventually contains 1,948 poems over 8,147 stanzas..

Students then annotated rhyme schema on stanza level (e.g. 'abba'). Annotators were instructed on rhyme, enumerating the most frequent imperfections (e.g. vowel rounding). For training and to inter-annotator agreement, each student annotated Georg Trakl's 1913 'Gedichte', 51 poems over 251 stanzas. Schemas were annotated directly in inline XML TEI P5, with the Oxygen XML editor to validate against RELAXng. Three annotators achieve .95–.97 Cohen κ among each other, measured on stanza label overlap. Reading stanzas vertically and annotating (typing) stanzas horizontally is prone to errors, generating false positives and negatives.

DTR yields 16,440 rhyme pairs. The list of these pairs was revisited. Out of 1.500 pairs, 244 did not actually rhyme (16% noise) and 17 instances were assonances. The remaining 1,200 pairs were split to a dev set and a untouched test set (both with neg. pairs from inverse schema indices). For further experiments, we clean the entire set which reduces the total set to 13,785 positive pairs.

2.2 ANTI-K: Lyrik.Antikoerperchen

The website `lyrik.antikoerperchen.de` provides a platform for students to upload essays about famous poems. These are neatly formatted in HTML, with clean line/stanza segmentation and reliable metadata, i.a. author, year, number of sentences, literary period, genre. The 156 poems (731 stanzas) are dispersed over 50 authors and over the NHG timeline. ANTI-K was was annotated by a competent student and re-checked. ANTI-K yields 1,440 rhyme pairs.

2.3 HIPHOP-R: Hip-Hop Rhyme

We collected 116 German Hip-Hop song texts and annotated them on rhyme and assonance (repetition of vowels). We retrieved the documents in plain text from `hiphoplyrics.de`, mainly covering the

[2] `github.com/thomasnikolaushaider`

[3] `deutschestextarchiv.de`

90's and 2000's, with 1–4 texts per author. Hip-Hop differs from lyric poetry in the regard that it makes heavy use of internal rhyme and assonance. As the annotation of internal rhyme is very time consuming, we confine our analysis to end-rhyme. Yet, assonances and rhymes often form a complex schema, so we decided to mark assonances with capital letters in the stanza level rhyme schema to extract them separately. We retrieve 2,489 rhyme pairs and 1,032 assonance pairs.

2.4 English and French Rhyme Gold

In order to analyze the similarity of rhymes within a graph structure, Sonderegger (2011) compiled a synchronous goldstandard of rhyme schema and their corresponding words. This dataset was modified to include diachronous variation and was used by Reddy and Knight (2011). The dataset includes 12,000 stanzas, yielding 54,000 rhyme pair token. The French rhyme standard includes 2,814 stanzas that yield 18,834 rhyme pair token.

3 Experiments

We introduce Siamese Recurrent Networks for supervised rhyme detection and it test it for English, French and for three German datasets. We ensure that no rhyme pair in the test sets occurs in the training set. We test on held out data (dev and test) from DTR (*dta*), on ANTI-K (*anti*), on Hip-Hop rhymes (only rhymes) (*hip*) plus assonances (*hipa*). We train several well performing models that then undergo a qualitative error analysis. We also include results from an EM baseline, although stanza and pair accuracy cannot be directly compared.

3.1 Schema inference with EM

We train Expectation maximization (EM) as described in Reddy and Knight (2011) on our two poetry corpora with the code provided by Sravana Reddy.[4] For certain experiments on English, they report accuracy up to .88 F1. Table 2 lists the results for two German corpora. When initialized orthographically, we achieve over .70 in accuracy. EM is initialized either uniformly (uni) or with orthographic overlap (orth) of potential rhyme words.

train	test	uni	orth
anti	anti	.63	.77
dta	anti	.37	.71

Table 2: Accuracy of EM on German stanzas

3.2 Siamese Networks

Siamese recurrent networks (SRN) have been successfully applied in NLP applications to measure the distance of texts, both on the level of characters and words. Neculoiu et al. (2016) use it for job title normalization, Mueller and Thyagarajan (2016) on sentence similarity and Das et al. (2016) for *Quora* question pair retrieval.

A SRN consists of two identical recurrent sub-networks that learn a vector representation from input pairs. The sub-networks each receive a rhyme word as character embedding vector and encode it through several layers of bidirectional Long Short Term Memory (biLSTM) Networks. The activations at each timestep of the final biLSTM layer are averaged to produce a fixed-dimensional output. This output is projected through a single densely connected feedforward layer. The respective dense layers are then connected at their outputs through an energy function. The energy of the model is then the similarity between the embeddings of x1 and x2. Our architecture, as that of Neculoiu et al. (2016) uses a cosine distance and a pair of three (later four) stacked biLSTM layers. The model learns through positive and negative examples and optimizes against accuracy and contrastive loss. Binary classification is carried out by setting a threshold of .5 for the cosine value (0-1). See figure (2) for an illustration of our architecture, as proposed by Neculoiu et al. (2016).

[4] https://github.com/sravanareddy/rhymediscovery

Our architecture is implemented in tensorflow. The initial codebase was provided by the github user *dhwajraj*.[5] Our parameter are as follows: *character embedding* is set to 100 dimensions. We experiment with the number of hidden units between the LSTM layers (20 – 100), settling on 50. We train 100 epochs, use a batch size of 64, leave dropout at 1.0 and L2 reg. at 0.0. Lastly, the system uses a random 80/20 train/dev split. Maximum document length is set to 30.

3.2.1 Binary Classification

In our initial setup we created only positive rhyme pairs and shuffled the words to generate negative examples at a ratio of 2:3 positive to negative examples.We train a model on the full DTR dataset of rhyme pair token (`token23`), i.e. pairs occur multiple times, and then removed redundant pairs and trained only on singular types (`type23`). We find that 5000 pairs already offer sufficient accuracy on DTA and ANTI (.96 on dta, .94 on anti). Both models are trained on three LSTM layers with 50 hidden units. We use this configuration to train models on 10,000 and 30,000 rhyme pairs from our English data and achieve .96 and .97 points accuracy on 5000 held out pairs (plus 5000 negative). We also train a 1:1 model on 12,000 French rhymes and test 3000 held out pairs (plus 3000 negative). Here, we also achieve .97 acc..

Subsequently, for German, we train on a 1:1 and a 1:4 ratio. (Neculoiu et al. (2016) report an ideal ratio of 1:4 for job title normalization). At 1:4, despite the large amount of data, the model only achieves .93 accuracy. For the 1:1 ratio, we first select 10.000 pairs (both positive and negative) to evaluate the number of LSTM layers, where four layers performed better overall than three or five layers. We train a model on four layers (`type11`) over the full dataset with 30 hidden units and achieve .98 acc. on the dta dev set, followed by the slightly noisier anti set (.96). We then gradually increase the training set to plot a learning curve against our four test sets. See figure (1) for the learning curve.

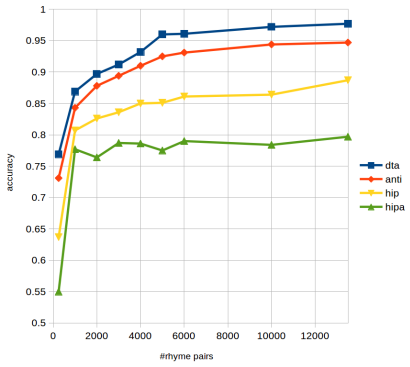

Figure 1: Learning rate of model `type11`

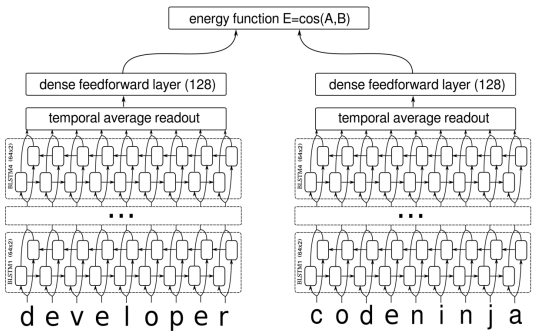

Figure 2: Siamese architecture, from (Neculoiu et al., 2016)

The reason why *anti* performs slightly worse is most likely owed to the noise introduced by wrong schema annotation. Training on dta and testing on Hip-Hop (hip) only achieves .88 accuracy. Generally, for a given test-set all three models exibit very similar performance. Hip-Hop rhyme is not detected that well from a model trained on historical poetry, as the language variety (slang, code switching) somewhat differs. Also, the bottom curve (hipa) indicates that assonances were not learned by this model, noticeably lacking in recall. Since the pairs retrieved from DTR included 16% noise, we decided to again manually correct the rhyme pair training set of DTR. We train another model (`clean11`) with the same parameters (but 50 hidden units), getting .98 accuracy.

3.2.2 Language & Domain Independence

We train two models with three and four layers and 50 epochs on all three languages (German, English, French). Each language contributes 20,000 pairs, evenly split for positive and negative examples. We report the 4-layer model in table 3 (`4indep`). Both models are on par with all previous approaches, in some cases even outperforming models trained on a single language. To further minimize the error

[5] `github.com/dhwajraj/deep-siamese-text-similarity`

on Hip-Hop, we include 4,000 (pos.+neg. 1:1) pairs from Hip-Hop into the lang. independent dataset and set aside the remaining 978 pairs for testing. We train on the same parameters (`4indeph`), and see improvement for Hip-Hop (.92 F1, .96 prec, .89 recall) while all other testsets remain stable and even slightly improve.

	dta_DE		anti_DE	hip_DE	ENG	FR
	dev	test				
Acc.	.976	.960	.961	.892	.965	.963
F1	.976	.960	.960	.885	.965	.963
Prec	.970	.954	.965	.948	.959	.960
Rec	.981	.967	.956	.830	.972	.966

Table 3: Lang. Independent Model `4indep`

Schlegel		Gryphius		Kappus	
a	wieder	a	verheret	a	Klage
b	Reihen	b	Posaun	b	weh
b	zweien	b	Carthaun	b	Blütenschnee
a	nieder	a	gezehret	a	Tage
a	glieder	a	vmgekehret	a	Frage
b	dreien	b	zerhawn	b	geh
b	gedeihen	b	schawn	b	see
a	Lieder	a	durchfehret	a	wage
c	kränzen	c	blutt	c	trübe
d	dünket	c	flutt	d	Sommernächte
e	Gesetze	d	fortgedrungen	e	wann
d	winket	e	todt	c	Liebe
c	Gränzen	e	noth	d	möchte
e	Gegensätze	d	abgewzungen	e	kann

Table 4: Rhyme words and schema of three sonnetts

3.2.3 Qualitative Error Analysis

We conduct a small qualitative error analysis on three german sonnetts from different literary periods. They include a variety of imperfect rhymes and orthographic deviations. The first poem is by Schlegel, the second by Andreas Gryphius, and the third is by Franz Xaver Kappus. Table 4 lists the end words of these poems and the associated rhyme scheme. We generate pairs on all permutations of end words and evaluate our four models `type23`, `token23` and `token11`, `clean11` and the two independent models `indep` with and without Hip-Hop. When not mentioned for a poem, a model delivers perfect performance.

Schlegel `clean11` and the `indep` models wrongly detect the mapping 'ei' $\rightarrow$ 'ä', therefore detecting (reihen, [g|k]ränzen), (zweien, [g|k]ränzen), etc..

Kappus `type23` wrongly identifies (weh, Frage), (blütenschnee, Frage/wage), so it is probably matching on the final 'e'. `type11` massively overgenerates on this poem with a precision .34 and perfect recall. This questions the overall sanity of this otherwise well performing model. The `indephip` model wrongly assigns very low cosine distance (<.1) to pairs that end with 'e', 'eh' or 'ee' such as (weh/geh, trübe/liebe), (Blütenschnee/See, trübe/liebe), e.g. (weh, trübe).

Gryphius The model `type23` does not detect the historical spelling variation of the diphtong [aw] instead of [au] and consequently does not detect the following pair combinations: (posaun/carthaun, zerhawn/schawn). After manually correcting the words, the system does detect them. `token23` additionally does not detect (schawn, zerhawn). Both `indep` models wrongly identify the combinations (blutt/flutt, todt/noth), probably matching on the 't'.

4 Conclusion

We have introduced three new poetry corpora for German and discussed their annotation with rhyme schema. Furthermore, we introduced a Siamese Recurrent Networks architecture to the detection of rhyme pairs and find that it learns this task with near perfect accuracy across languages. When switching domains from poetry to Hip-Hop we lose 10 points, and assonances are not that well detected. It is notable that SRNs can apparently compensate a noise level of 16% in the training set. Future research should determine the acceptable noise level. Finally, we have shown that a SRN can be trained on a dataset containing rhyme pairs of three languages and also a different domain without losing performance. But even though we achieve over 96% accuracy on pairs, each model exhibits individual errors in a qualitative error analysis, making it hard to determine an ideal model. While the independent models work well, they show some problems on particular character mappings.

References

Thomas Berg. 1990. Unreine reime als evidenz für die organisation phonologischer merkmale. *Zeitschrift für Sprachwissenschaft*, 9(1-2):3–27.

Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 378–387.

Nigel Fabb. 1997. *Linguistics and literature: Language in the verbal arts of the world*. Blackwell.

Jonah Katz. 2015. Hip-hop rhymes reiterate phonological typology. *Lingua*, 160:54–73.

Johann-Mattis List, Jananan Sylvestre Pathmanathan, Nathan W Hill, Eric Bapteste, and Philippe Lopez. 2017. Vowel purity and rhyme evidence in old chinese reconstruction. *Lingua Sinica*, 3(1):5.

Nina McCurdy, Vivek Srikumar, and Miriah Meyer. 2015. Rhymedesign: A tool for analyzing sonic devices in poetry. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 12–22.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792.

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

Beatrice Primus. 2002. Unreine reime und phonologische theorie. *Sounds and Systems. Studies in Structure and Change. A Festschrift for Theo Vennemann. Berlin/New York*, pages 269–298.

Sravana Reddy and Kevin Knight. 2011. Unsupervised discovery of rhyme schemes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 77–82. Association for Computational Linguistics.

Morgan Sonderegger. 2011. Applications of graph theory to an english rhyming corpus. *Computer Speech & Language*, 25(3):655–678.