JABERT: A New State-of-the-Art BERT Model for Judeo-Arabic

Elisha Rosensweig,[1,2,‡] **Yitzchak Lindenbaum,**[1,2,‡] **Hillel Gershuni,**[1,2,‡]
Vered Raziel-Kretzmer,[2,§] **Daniel Caine,**[2,◇] **Avi Shmidman**[1,2,†]
[1]DICTA / Jerusalem, Israel [2]Bar-Ilan University / Ramat Gan, Israel
[†]avi.shmidman@biu.ac.il, [§]razielv@post.bgu.ac.il
[◇]daniel.caine@mail.huji.ac.il
[‡]{benshafat, gershuni, yitzilindenbaum}@gmail.com

Abstract

We present JABERT, the first BERT model pretrained specifically for historical Judeo-Arabic texts. We demonstrate that JABERT outperforms Arabic and multilingual models on the downstream task of Judeo-Arabic homograph disambiguation. Furthermore, in order to test the latter, we have curated and annotated the first Judeo-Arabic homograph test set. We release both JABERT and the Judeo-Arabic homograph test to the public for unrestricted use.

Keywords: Judeo-Arabic, BERT, WSD

1. Introduction

Judeo-Arabic (=JA) refers to a diverse family of Arabic dialects historically spoken and written by Jewish communities throughout the Middle East and North Africa, from the 8th century and onward through the middle ages. Judeo-Arabic is written in Hebrew script, incorporating Hebrew and Aramaic lexical elements while maintaining Arabic grammatical structures and core vocabulary. As a primary vehicle for Jewish intellectual, religious and everyday communication in Arabic-speaking lands, Judeo-Arabic has left behind a vast textual legacy spanning philosophy, biblical commentary, legal responsibility, scientific treatises, personal correspondence, and literary works. Notable examples include the writings of Saadia Gaon, Judah Halevi and Maimonides, whose influence extended far beyond their immediate communities.

Despite its historical and cultural significance, Judeo-Arabic remains severely underrepresented in NLP research. The language presents unique challenges as a low-resource variety: digital corpora are limited in size, texts exhibit substantial orthographic and dialectal variation across regions and time periods, and the Hebrew-script encoding creates additional barriers for standard Arabic NLP tools. While recent years have seen rapid advances in transformer-based language models for both Modern Standard Arabic and various Arabic dialects, as well as for historical Hebrew, no dedicated BERT model for Judeo-Arabic has been developed to date. This gap has hindered computational research on Judeo-Arabic texts and limited the application of modern NLP techniques to digital humanities scholarship in this domain.

In this paper, we present JABERT, the first pre-trained BERT model specifically designed for Judeo-Arabic. We test JABERT's ability to predict missing words within actual Judeo-Arabic historical documents, and we present both quantitative and qualitative evaluations. Furthermore, we compare the performance of JABERT to that of Arabic and multilingual BERT models when put to the downstream task of Judeo-Arabic homograph disambiguation. We demonstrate that JABERT outperforms these configurations.

In what follows we describe our training methodology, including corpus collection and preprocessing strategies for handling the unique characteristics of Judeo-Arabic texts, and we present an evaluation of the model's performance. We release JABERT on huggingface for unrestricted use.[1]

2. Related Work

As noted above, there does not currently exist a BERT model specifically trained for Judeo-Arabic. However, because Judeo-Arabic is effectively a mixture of Hebrew and Arabic, we survey the existing BERT models for Hebrew and for Arabic, as well as existing systems for transliterating between Hebrew and Arabic scripts.

2.1. BERT Models

2.1.1. Arabic Models

Multiple Arabic BERT models exist (e.g., AraBERT (Antoun et al.), CAMeLBERT (Inoue et al., 2021), MARBERT (Abdul-Mageed et al., 2021)). However, these are trained exclusively on Arabic script, while Judeo-Arabic uses Hebrew-script, thus preventing them from being directly applicable to Judeo-Arabic processing. Nevertheless, they can be applied via transliteration; see below, 2.2.

[1]Huggingface link: https://huggingface.co/MiDRASH-ERC/JABERT . The exact license can be found at https://creativecommons.org/licenses/by/4.0/

Proceedings of LT4HALA 2026 @ LREC 2026, pages 58–71.

2.1.2. HeArBERT: a Hebrew-Arabic Model

HeArBERT (Rom and Bar, 2024) is a bilingual Hebrew-Arabic BERT that uses transliteration to map Arabic text into Hebrew script, creating a shared character space for both Semitic languages. It was not trained on Judeo-Arabic texts, but since it operated in Hebrew script, it is suitable to use as a reference point to the performance of our Judeo-Arabic model JABERT. In Section 5.3 we evaluate the performance of JABERT vis-a-vis HeArBERT.

2.2. Transliteration Methods

With a lack of models directly applicable to Judeo-Arabic, an alternative approach is to transform the Judeo-Arabic text into Arabic script, and then apply Arabic BERT models to the transliterated text. In Section 5.3 we evaluate the performance of JABERT vis-a-vis this transliteration approach.

Transliterating Judeo-Arabic was initially addressed by Bar et al. (Bar et al., 2015), which introduced statistical machine translation for character-level Judeo-Arabic to Arabic transliteration, achieving 96.9% precision. This was extended by Terner et al. (Terner et al., 2020) using RNN with CTC loss, reducing character error rate from 9.5% to 2% through synthetic data augmentation.

In a more recent work, Gonzalez et al. (2025) used a transliteration-based approach to process Judeo-Arabic. Their transliteration was comprised of two steps. The first step involved a rule-based transliteration from Hebrew to Arabic script. However, because there are cases where a given Judeo-Arabic letter stands in more than one Arabic letter, a rule-based transliteration may induce a certain margin of error (see ibid., Section 2.1). Due to this factor, they applied a second stage in which the text was auto-corrected either by commercial LLMs (e.g. GPT4o) or using dedicated GEC (grammatical error correction) models (specifically, the Seq2Seq SWEET models proposed by Alhafni et al. (2023)). The resulting auto-corrected text was then processed in Arabic BERT models. They showed that transliteration combined with post-correction enables using existing Arabic NLP tools for downstream tasks such as machine translation, thus establishing a pathway to handle Judeo-Arabic processing in the absence of a dedicated Judeo-Arabic model.

The transliteration method has three notable drawbacks. First, as noted, transliterating Judeo-Arabic into Arabic script may produce an incorrect result due to the cases where a single Judeo-Arabic letter represents more than one Arabic letter; the aforementioned autocorrection methods do not always yield a perfect result. Second, the current SOTA method from Gonzalez et al. (2025) utilizes decoder methods which do not always produce a direct word-for-word conversion, and which may instead produce a paraphrase of the original text; this poses a major difficulty for word-based labeling tasks. Finally, the Arabic models can only correctly process the Arabic portions of the transliterated text. Yet, in practice, Judeo-Arabic texts integrate many Hebrew loan-words, Hebrew citations, Hebrew Biblical verses, and more. Due to this issue, performance in Gonzalez et al. (2025) is measured by applying a code-switching detector and isolating the analysis to the Arabic portions alone. JABERT is designed to handle an entire Judeo-Arabic text as-is, thus addressing all three drawbacks.

3. Model Training

3.1. Tokenizer

We use the Word-Piece tokenization method proposed by Song et al. (2021) with the default normalizers and preprocessors suggested by HuggingFace. Following the work of Gueta et al. (2023) demonstrating the substantial advantage of larger token vocabularies for morphologically-rich languages, the tokenizer was trained with a vocabulary size of 128K tokens.

Due to the fragmentary nature of many historical Judeo-Arabic texts, we introduce two special tokens into the vocabulary to handle textual lacunae:

- "[ONEGAP]" indicates a single word with at least one missing letter. If the transcription indicated several consecutive missing characters, up to 5 characters were assumed to constitute a single word.
- "[GAP]" indicates multiple missing words or a lacuna of indeterminate length.

3.2. Training Data

The Judeo-Arabic texts included in JABERT's training corpus were amassed from three different sources: the Princeton Genizah Project (PGP; Rustow and Koeser, 2025), the Friedberg Judeo-Arabic Project by the Friedberg Jewish Manuscript Society[2] (FJMS), and quotations in Blau's *Dictionary of Mediaeval Judaeo-Arabic Texts* (Blau, 2006). Since these sources would not have been sufficient to train a competent BERT model, we augmented them with medieval Arabic texts from the Open Islamicate Texts Initiative (OpenITI; Romanov et al., 2024) transliterated into Hebrew characters, approximating the way these texts would appear if they had been written in Judeo-Arabic. We included texts by authors whose listed *hijri* year of

[2]Resource can be accessed at https://github.com/princetongenizalab/pgp_transliteration/tree/main/resources/scrapes.

death is between 390 and 905 (corresponding to 1000–1500 C.E.)[3] and transliterated them using a simplified and slightly adjusted version of the table provided in Weisberg Mitelman et al. (2024). We employed weighted stochastic decision based on typical Judeo-Arabic usage to simulate regional and scribal variation.[4]

The texts from each source were preprocessed to handle punctuation, glosses/translations, and document segmentation as appropriate. The full sets of preprocessing steps applied to each corpus are detailed in Appendix B. To the extent possible, only the original medieval text was preserved, although where lacunae were filled in by editors, the suggestions were accepted. Remaining lacunae were handled using the `[ONEGAP]` and `[GAP]` tokens (see section 3.1).

Source	Word Count	Notes
PGP	~1.6 million	Transcribed manuscripts
FJMS	~3.1 million	Printed works
Blau	~42,000	Extracted quotes
OpenITI	~1.2 billion	Transliterated

Table 1: Training data sources

3.3. Training Details and Hyperparameters

We trained our model with the HuggingFace architecture wrapped with NVIDIA libraries,[5] which are highly optimized for training compute-heavy machine learning models on NVIDIA hardware. We pre-trained the model on four NVIDIA A40 48GB GPUs for a total of 86,010 iterations, with a total of 106.56B tokens.

The training was carried out in 3 phases:

(1) Sequences of 128 tokens with a learning rate of 6e-3 for 13,000 iterations.

(2) Sequences of 128 tokens with a learning rate of 1e-4 for 57,380 iterations.

(3) Sequences of 512 tokens with a learning rate of 5e-5 for 15,630 iterations.

The training was done with a batch size of 8,192 in phases 1 & 2 and 4,096 in phase 3. Warmup proportion of 0.2843 for phases 1 & 2 (jointly, as phase 2 is a continuation of phase 1), and 0.128 for phase 3.

4. Datasets

4.1. JABERT Cairo Genizah Dataset

In order to evaluate JABERT's ability to predict masked words within actual Judeo-Arabic historical documents, we curated a new dataset of lines of Judeo-Arabic text not included in JABERT's training data. In order to do so, we employed a Judeo-Arabic expert who transcribed lines of Judeo-Arabic from Cairo Genizah manuscripts stored in the Cambridge University Library.[6] In total, our expert transcribed 3862 lines of text. For the transcription protocol according to which the Judeo-Arabic expert transcribed these lines, see Appendix C.

4.2. JABERT Homograph Disambiguation Challenge Dataset

We also sought to test JABERT's ability to distinguish between different meanings of Judeo-Arabic words. To this end we curated a challenge dataset consisting of 16 Judeo-Arabic homographs. Two Judeo-Arabic experts examined dozens of occurrences of each of these homographs within Judeo-Arabic texts, and they annotated each of these occurrences for its context-specific meaning.[7] To the best of our knowledge, this is the first instance of a Judeo-Arabic homograph challenge set. Table 2 provides a quantitative overview of this dataset.

Category	# instances	median / word
words	16	1
meanings	37	2
samples	1119	71

Table 2: Homograph Dataset at a glance

Table 3 illustrates four representative homographs from the dataset, selected to highlight various disambiguation challenges: functional (e.g., distinguishing particles from other parts of speech), lexical and syntactic (e.g., noun vs. adjective or different verb forms), and root-level ambiguity in Judeo-Arabic orthography.[8]

[3] Documents tagged as "UNCORRECTED_OCR" were not used.

[4] Specifically, 80% of the documents (chosen randomly) were transliterated with jīm (ج) rendered as ג (without geresh) and ghayn (غ) as ג'. In the remaining 20%, jīm → ג' and ghayn → ג. Hamza (ء on the line) was randomly rendered as either א or omitted entirely. The complete transliteration table is presented in Appendix A.

[5] https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/BERT

[6] The documents were selected from the following four boxes of the Taylor-Schechter collection of Cairo Genizah fragments in the Cambridge University Library: T-S Misc. 24; T-S Arab. 31; T-S Arab. 43; and T-S 8F.

[7] Where there was a disagreement between the two experts, we discarded the sample.

[8] The full list of Judeo-Arabic homographs, with grammatical explication and translation, is presented in Appendix D.

#	Translit.	Arabic	Meaning
1. אלא (*'l'*)			
1	*alā*	أَلَا	“Lo”, “is it not?”
2	*illā*	إِلَّا	“except”, “other than”
12. עאלם (*'lm*)			
28	*'ālam*	عَالَم	“world”
29	*'ālim*	عَالِم	“knows”
13. כלק (*ǩlq*)			
30	*ǩalaqa*	خَلَقَ	“created”
31	*ǩalq*	خَلق	“creation”
16. אכד (*'kd*)			
36	*akkada*	أَكَّدَ	“confirmed”
37	*aǩaḏa*	أَخَذَ	“took”

Table 3: Sample homographs illustrating common ambiguities: functional (1), semantic and POS (12), and morphological or root-based (13, 16).

4.3. Release of our datasets to the public

We are pleased to release these two datasets to the public, both in the interests of reproducibility, and also so that these datasets can continue to be used as benchmarks to test and evaluate future Judeo-Arabic NLP models.[9] The datasets are released under a permissive Creative Commons CC-BY license.

5. JABERT Evaluation

To evaluate the utility of the model and the degree to which it captures the properties of Judeo-Arabic, we utilize two tasks:

- **Masked Word Prediction** - Given a Judeo-Arabic line from the JABERT Cairo Genizah Dataset (Section 4.1), we mask a word in the line and ask the model to predict the missing word. This task sets the bar quite high, as multiple words might fit reasonably in place of the mask (e.g., swapping "Sunday" for "Monday"). The task is even more challenging in our case since each line in the dataset consists of a single *physical* line from actual historical Judeo-Arabic manuscripts, wherein the lines generally begin and end mid-sentence. This reflects actual expected scholarly usage of the BERT model when working on historical manuscripts, and substantially increases the difficulty of the task. We evaluate the ability of JABERT to predict the correct word within the top-k candidates it proposes, for various values of k. Because there are no existing BERT models capable of predicting full Judeo-Arabic words as MLM predictions, this task is not a comparative one. Below we provide quantitative results followed by a qualitative analysis.
- **Homograph Disambiguation** - We evaluate the model's ability to distinguish between different senses of a single Judeo-Arabic word form. Success at this task would indicate that the model captured key semantic elements of Judeo-Arabic. For this evaluation task, we use the JABERT Homograph Disambiguation Challenge Set (Section 4.2). Furthermore, because this test relies on the contextual embedding rather than on the MLM prediction, it can be used to compare JABERT's abilities to other multilingual BERT models. Thus, for this experiment, we compare JABERT's performance with that of the industry-standard multilingual mBERT, side-by-side with the aforementioned HeArBERT model and the transliteration-based method proposed by Gonzalez et al. (2025). Here too, we provide quantitative results followed by a qualitative analysis.

Crucially, the test sets for these two evaluations consist entirely of authentic Judeo-Arabic text. This is important because, as described earlier (3.2), in order to amass a corpus large enough for pretraining JABERT, we relied heavily on transliterated medieval Arabic texts, and the reader may legitimately wonder whether the model truly captures Judeo-Arabic, or rather just Arabic transliterated into Hebrew characters. In order to address this concern, we ensured that the test sets contain only authentic Judeo-Arabic text. Thus, the results obtained on our evaluation tasks reflect the model's ability to handle actual Judeo-Arabic.

5.1. Masked Word Prediction - Quantitative Evaluation

For each single-token word[10] in each line of the JABERT Cairo Genizah Dataset, we mask the given word and run the line through JABERT's MLM head, retrieving predictions for the masked word position. In total, this yielded $16,129$ masked words on which we ran MLM inference and evaluated the model's predictive performance.

When predicting the word, we considered how far into the ranked candidates list we needed to proceed in order to find a prediction of the correct

[9] Link to download: https://github.com/ERC-Midrash/ja-models-eval

[10] Other than the first and last words on the line; masking either of these words would be an unfair challenge because BERT would not have access to the bi-directional context on the other side of the word.

word, for $k \in [1, 250]$. Figure 1 shows the percentage of samples where the model predicted the missing word correctly within the top k words as k increases. Table 4 provides some representative performance milestones for clarity. As can be seen here, despite the high difficulty level of the task, the model rapidly increases in performance until approximately $k = 25$, where it reaches 42% accuracy. From that value on we get steady but diminishing returns as k increases.

k	Percentage	k	Percentage
1	16.36%	25	42.01%
2	21.40%	50	48.38%
3	24.47%	100	54.89%
5	28.55%	150	58.48%
10	34.01%	250	63.31%

Table 4: Results for various values of k

5.2. Masked Word Prediction - Qualitative Evaluation

The quantitative analysis poses a high bar for success: to accurately predict the correct token out of 128K tokens. This is complicated further by the fact that certain tokens cannot be predicted in this fashion - for example, a date or number might be interchangeable in a given position with other, equally valid dates and numbers. As such, we turn to qualitative analysis of some examples, to demonstrate meaningful patterns that are not captured by the preceding quantitative analysis.

Consider the following examples where the correct word was not ranked first:

- **Ground Truth:** פיה (*fīhi*, "in it")
  Context: יכון אלאקראר ממא יגב עליה פיה חכם (*yakūn al-iqrār mimmā yajib ʿalayhi **fīhi** ḥukm*, "the acknowledgement is among the things for which a ruling is required **therein**")
  Model Rank: 8
  Top Candidates: לה (*lahu*, "to it"), ענדה (*ʿindahu*, "at/with it"), מעה (*maʿahu*, "with it"), פי (*fī*, "in"), עליה (*ʿalayhi*, "upon it")
  Analysis: All predictions belong to the correct semantic category of relational prepositions, demonstrating the model's ability to identify the appropriate syntactic slot despite the specific token variation.

- **Ground Truth:** מר (*mar*, honorific title)
  Context: ואלהלכה פי דלך כקול מר שמואל (*wa-l-halākha fī dhālik ka-qawl **mar** Shemūʾel*, "And the law in this matter follows **Mar** Samuel")
  Model Rank: 8
  Top Candidates: רבינו (*rabbēnū*), רבנו (*rabbānū*, both "our rabbi"), ר' (*rabbī*, "rabbi"), אלחבר (*al-ḥaver*, "the scholar"), רב (*rav*, "rabbi/master")
  Analysis: The model correctly identifies this as a slot for an honorific or rabbinic title.

- **Ground Truth:** תצח (*taṣiḥḥu*, "is valid", fem.)
  Context: הו אן אלדי ימלך בה אלאשיא אלתי תצח מלכהא בה (*huwa anna alladhī yamliku bihi al-ashyāʾ allatī **taṣiḥḥu** milkuhā bihi*, "it is that by which one possesses the things whose ownership **is valid** through it")
  Model Rank: 43
  Top Candidates: יצח (*yaṣiḥḥu*, "is valid"), יקע (*yaqaʿu*, "takes effect"), ימכן (*yumkinu*, "is possible"), יגוז (*yajūzu*, "is permissible"), יגב (*yajibu*, "is required"), all masc. forms.
  Analysis: While the manuscript uses a feminine verb, likely influenced by the preceding non-human plural *al-ashyāʾ* and the feminine pronominal suffix in *milkuhā*, the model correctly identifies the masculine agreement required by the subject *milk*. This demonstrates JABERT's grasp of normative syntax despite scribal variation.

These patterns hold broadly: predictions tend to cluster within the same part of speech and semantic field as the target word. In several instances, the model's "errors" actually reflect normative syntactic rules, such as correct gender agreement, even when the original manuscript exhibits scribal variation. This suggests that the top-1 accuracy underestimates the model's grasp of Judeo-Arabic. It captures the distributional structure of the language even when it does not recover the exact lexical item. The jump from 16.35% (top-1) to 34.01% (top-10) reflects not just additional guesses, but the model's ability to identify the correct semantic and syntactic category.

5.3. Homograph Disambiguation - Quantitative Evaluation

Next, we challenge the model to disambiguate Judeo-Arabic homographs. To succeed at this task, JABERT would need to have captured properties of Judeo-Arabic that allow it to determine the correct meaning of a word given the context.

For each homograph, we perform the following steps[11]:

- Take each sentence with the homograph, remove punctuation and mask the homograph.
- Run the sentence through the BERT model and compute the final embedding of the masked token.

[11] See our Homograph eval code here: https://github.com/ERC-Midrash/ja-models-eval

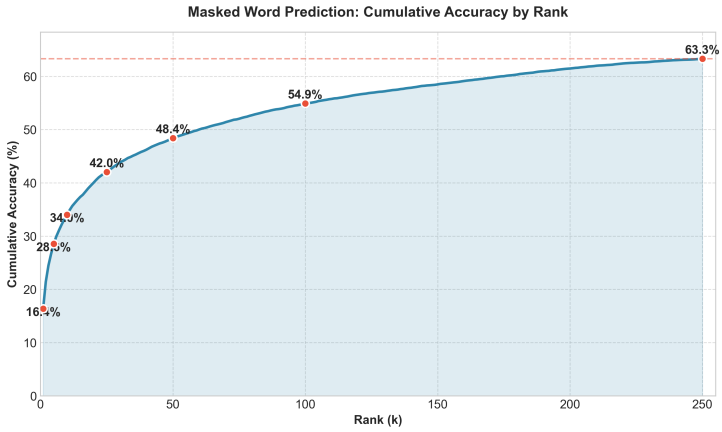

Figure 1: Cumulative accuracy for masked word prediction by rank k

- Run clustering algorithms on the computed embeddings. We attempt here both supervised and unsupervised clustering.
- Compute metrics measuring the quality of the clustering vis-a-vis the ground truth labels (= actual meaning of the homograph in each sentence).

Note that we compute the embedding on the *masked* homograph. This method solves several problems. First, it ensures the embedding is based only on the context, not the word itself, making the task both more challenging and increasing the applicability and stability of this approach across other words. Second, it prevents us from having to handle word-pieced words differently. Most importantly, it allows us to compare performance across models while remaining agnostic to their different vocabularies (i.e., tokenization). It also allowed us to compare our performance to that of the transliteration method discussed in Section 2.2.

We conducted two types of clustering on the embeddings: supervised (kNN-LOOCV, with k=5 neighbors) and unsupervised (k-means, where k is chosen to match the known number of meanings per homograph). For kNN-LOOCV the success metric is simply the number of correctly-assigned samples. For the unsupervised clustering task we considered two performance metrics to evaluate the quality of the proposed clustering *HYP* compared to the Ground Truth labels *GT*:

- **Accuracy (percentile)** - fraction of correctly-assigned words. Computing this requires mapping the *HYP* cluster labels to *GT* cluster labels, such that the accuracy measure is maximized.
- **Adjusted RAND Index** - Given *HYP*, for each pair of data-points (= embeddings) u, v, we check if they have the same relative status (same cluster / different cluster) in *HYP* as in *GT*. If they do, we call this *pairwise agreement*. The RAND index is then *(# pairwise-agreements) / (# total pairs)*. The *adjusted* RAND Index normalizes this value to reflect the degree to which the computed score deviates from the score we would get from a *random* clustering.

We compare the performance of our model to several competing models and flows:

- `mbert` - multilingual BERT.[12]
- `HeArBert` (above, 2.1.2).
- The transliteration approach by Gonzalez et al. (2025), as outlined in Section 2.2. Specifically, we transliterated the Judeo-Arabic homograph dataset into Arabic based on Fig. 1 in Appendix A in their paper, followed by running their top-performing "SWEET" GEC model on the transliterated output. Next, we applied the clustering process, using embeddings for the contextualized masked Arabic word from `CAMeLBERT-MSA` and `CAMeLBERT-CA`, models trained on Modern Standard Arabic (MSA) and Classic Arabic (CA), respectively[13]

For the transliteration process, Gonzales et. al. report that at times it causes structural and stylistic changes to the text, and we observed this phenomenon as well. Our initial homograph dataset consisted of 1118 entries (=sentences with homographs), but post-transliteration + GEC model the homograph word was recoverable perfectly only for a subset of the entries - 859 entries in total. In order to run clustering algorithms we require a minimal amount of datapoints, so we ended up here with 853 entries and 14 homographs to analyze.

[12] https://huggingface.co/google-bert/bert-base-multilingual-cased

[13] See our code here: https://github.com/ERC-Midrash/ja-models-eval.

Due to all of the above, when comparing performance, we ran both (a) full-dataset comparison without the transliteration flow (Fig. 2), and (b) limited dataset comparison, including transliteration flow (Fig. 3).

As can be seen in both Fig. 2 and 3, JABERT exceeds performance of all alternative methods in k-means, with the highest median accuracy and Adjusted RAND scores and competitive in all other quartiles, and in some cases reaching close-to-perfect alignment with Ground Truth. Also, in Fig. 4 we see similar improvements in JABERT over the alternatives when applying the supervised kNN clustering. Of all the alternative methods, the best ones were those based on transliteration followed by strong Arabic models, following Gonzales et. al., for the reduced dataset as discussed above.

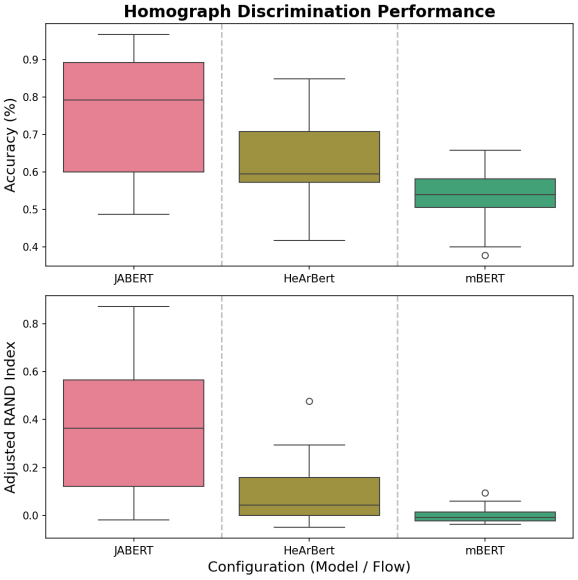

Figure 2: Performance for *k-means* clustering. Accuracy (%) and adjusted RAND scores for JABERT compared to two alternate models aggregated across the entire homograph dataset.

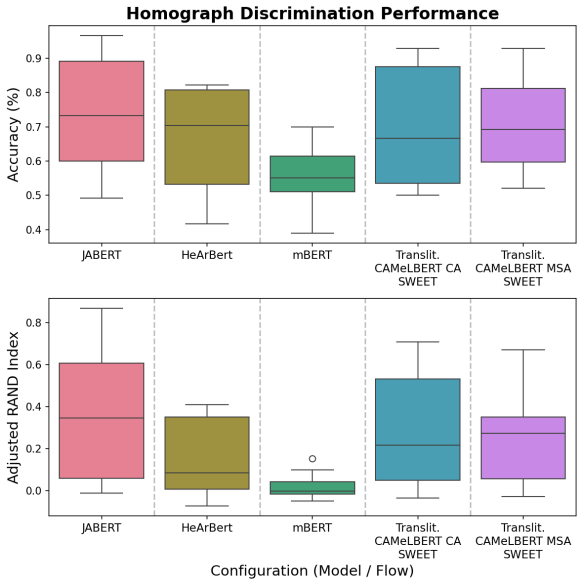

Figure 3: Performance for *k-means* clustering. Accuracy (%) and adjusted RAND scores for JABERT compared to two alternate models and the transliterate + GEC + arabic BERTs flow, aggregated across the subset of the homograph dataset where the transliteration + GEC process was stable.

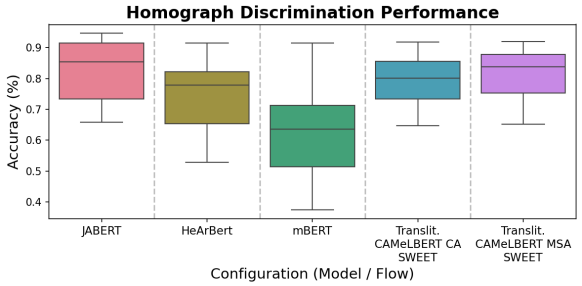

Figure 4: Performance for *kNN-LOOCV* clustering. Accuracy (%) scores for JABERT compared to two alternate models and the transliterate+arabic models flow, aggregated across the subset of the homograph dataset where the transliteration + GEC process was stable.

5.4. Homograph Disambiguation - Qualitative

We examined several cases where clustering results diverged from our annotations. Rather than straightforward errors, many of these cases point to linguistic subtleties.

Formulaic expressions. The homograph אלא (*'l'*) has two meanings: the interrogative *alā* ("Lo", "is it not?", meaning 1) and the particle *illā* ("except", meaning 2). Notably, meaning 1 appears frequently in the fixed phrase אלא תרי (*alā tarā*, "do you not see?"). As shown in Fig. 5, the model clusters these formulaic instances tightly, separating them from the few non-formulaic interrogative uses–a distinction our annotation scheme did not make.

Definiteness patterns. For the homograph עאלם (*''lm*)–representing *'ālam* ("world", meaning 28) vs. *'ālim* ("knowing", meaning 29)–one instance (701: קדר עאלם, *qadr 'ālam*, "the size of a world") clustered among the meaning 29 instances (in close proximity to instance 735, see Fig. 6). The reason is probably distributional: *'ālam* typically appears in the construct state (e.g., עאלם אלאפלאך, *'ālam al-*

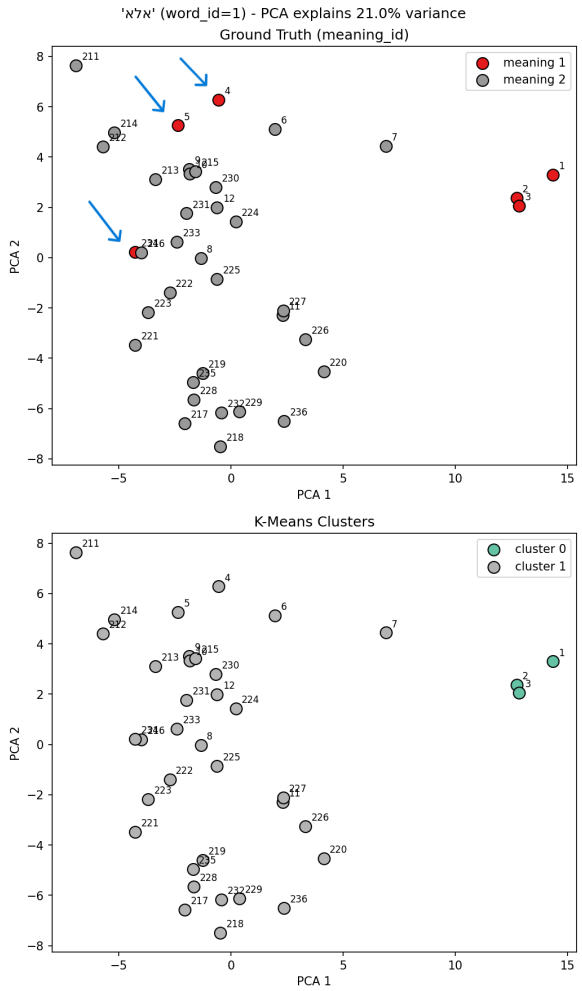

Figure 5: PCA visualization of JABERT embeddings for the homograph **אלא**. Top: Ground truth annotations for meaning 1 (*alā*, interrogative) and meaning 2 (*illā*, exceptive). Bottom: Unsupervised *k-means* clusters ($k = 2$). Note that the formulaic instances of *alā tarā* (points 1, 2, 3) are clustered together, while non-formulaic interrogative instances (e.g., 4, 5, 10, indicated by arrows) are closer to meaning 2. This suggests the model is highly sensitive to formulaic patterns that transcend our binary semantic categories.

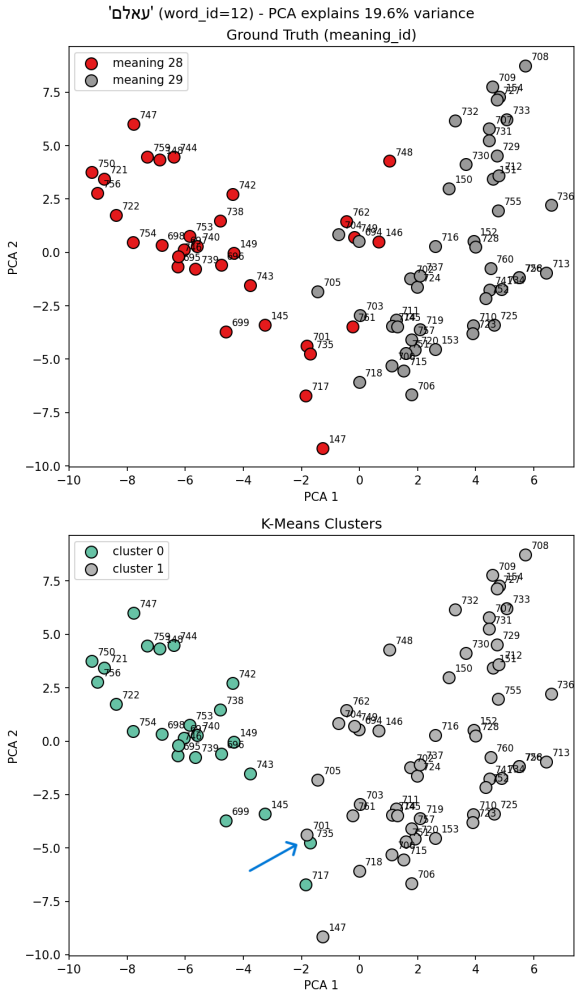

Figure 6: PCA visualization of JABERT embeddings for **עאלם**. Top: Ground Truth for meaning 28 (*ʿālam*, world) and 29 (*ʿālim*, knows). Bottom: k-means clusters ($k = 2$). The model separates senses based on syntactic distribution (definiteness and construct state); instance 701 (indicated by arrow) is an outlier due to its rare indefinite, non-annexed form.

aflāk, "the world of the [celestial] spheres") or with the definite article **אל** (*al-ʿālam*, "the world"). The indefinite, non-annexed form in 701 is rare in this meaning and much more common in meaning 29, and the model's embeddings reflect this.

Voice distinctions and annotation choices. The homograph **כלק** (*ḥlq*) was annotated as a verb (*ḥalaqa*/*ḥuliqa*, active/passive "created", meaning 30) vs. a noun (*ḥalq*, "creation/creatures", meaning 31). Instance 551 (**לדלך כלק**, *li-ḏālik ḥuliqa*, "for this purpose he was created") clusters with meaning 31, probably because its passive voice was marked as unique. Compare this to **יערף** (*yʿrf*), where we annotated active (*yaʿrifu*), passive (*yuʿrafu*), and causative (*yuʿarrif*) as three separate meanings (16–18). For *hlq*, we grouped active and passive together. The number of clusters defined ahead has thus a crucial impact on the clustering outcomes.

Orthographic identity vs. syntactic role. The homograph **פהם** (*fhm*) can be the noun *fahm* ("understanding", meaning 34) or the sequence *fa-hum* ("and they", meaning 35). Most instances cluster correctly. However, instance 872 is an outlier: the text has **פהם** where standard orthography would

have פאהם (*fāhim*, act. part., "a person who understands"). This spelling variation led to misclassification.

Collocational pull. The homograph אכד (*'kd/'ḫḍ*, when the diacritical mark is often omitted in manuscripts) represents *'akkada* ("confirmed/affirmed", meaning 36) vs. *'akhaḍa* ("took", meaning 37). instance 186 (אכד פי אלקסם, *'akkada fī al-qasam*, "affirmed by an oath") was correctly annotated as "confirmed/affirmed" but clustered with "took." The likely cause: the strong collocation between "oath" and "take" in Arabic (*'akhaḍa al-qasam*, "took an oath"). This lexical association apparently outweighed the contextual evidence for the Form II (*'akkada*) reading.

6. Future Directions

As noted, our evaluation datasets are comprised entirely of authentic Judeo-Arabic text, and thus they effectively demonstrate the ability of JABERT to handle actual cases of Judeo-Arabic, rather than the transliterated Arabic which formed much of the pretraining corpus. Nevertheless, it remains to be explored whether JABERT's capabilities derive from the quality of the genuine Judeo-Arabic training signal, the quality of the transliterated Arabic signal, or a combination of both. In future work, we plan to train a separate version of JABERT based entirely on the transliterated part of the corpus, and a further version based entirely on the authentic Judeo-Arabic part of the corpus, in order to address the question of the relative contribution and necessity of each part of JABERT's pretraining corpus.

7. Conclusion

In this paper, we introduced JABERT, the first pre-trained BERT model dedicated specifically to Judeo-Arabic. By leveraging a massive corpus of transliterated medieval Arabic alongside curated historical Judeo-Arabic manuscripts, we have addressed a significant gap in the NLP landscape for low-resource, historical Jewish languages. Our model not only sets a new SOTA for Judeo-Arabic NLP tasks, but also obviates the need for the cumbersome and brittle transliteration process which the previous SOTA methods required.

We hope that JABERT will lower barriers to computational analysis of Judeo-Arabic materials, and empower researchers to unlock the vast intellectual heritage contained within the corpus of Judeo-Arabic literature.

8. Acknowledgments

This work has been funded by the European Union (ERC, MiDRASH, Project No. 101071829; principal investigators: Avi Shmidman, Bar-Ilan University; Daniel Stökl, EPHE-PSL; Nachum Dershowitz, Tel Aviv University; and Judith Olszowy-Schlanger, EPHE-PSL), for which we are grateful. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

9. Bibliographical References

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2021. ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7088–7105, Online. Association for Computational Linguistics.

Bashar Alhafni, Go Inoue, Christian Khairallah, and Nizar Habash. 2023. Advancements in Arabic grammatical error detection and correction: An empirical investigation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6430–6448, Singapore. Association for Computational Linguistics.

Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, page 9.

Kfir Bar, Nachum Dershowitz, Lior Wolf, Yackov Lubarsky, and Yaacov Choueka. 2015. Processing judeo-arabic texts. In *2015 First International Conference on Arabic Computational Linguistics (ACLing)*, pages 138–144.

Joshua Blau. 2006. *A Dictionary of Mediaeval Judaeo-Arabic Texts*. The Academy of the Hebrew Language; The Israel Academy of Sciences and Humanities.

Juan Moreno Gonzalez, Bashar Alhafni, and Nizar Habash. 2025. A tale of two scripts: Transliteration and post-correction for judeo-arabic.

Eylon Gueta, Avi Shmidman, Shaltiel Shmidman, Cheyn Shmuel Shmidman, Joshua Guedalia, Moshe Koppel, Dan Bareket, Amit Seker, and Reut Tsarfaty. 2023. Large pre-trained models with extra-large vocabularies: A contrastive analysis of Hebrew BERT models and a new one to outperform them all.

Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, Kyiv, Ukraine (Online). Association for Computational Linguistics.

Princeton Geniza Project. 2021. Transcription history and conventions. Google Docs. Archived at https://web.archive.org/web/20260204094447/https://docs.google.com/document/d/1pyFajrGRYTzRsb2yFjfX0XmNqpz1kUO9mhlISGwfpxk/edit?tab=t.0. Accessed March 25, 2025.

Aviad Rom and Kfir Bar. 2024. Training a bilingual language model by mapping tokens onto a shared character space. *arXiv preprint arXiv:2402.16065*.

Maxim Romanov. 2023. OpenITI mARkdown. Accessed April 7, 2025.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. Fast wordpiece tokenization.

Ori Terner, Kfir Bar, and Nachum Dershowitz. 2020. Transliteration of Judeo-Arabic texts into Arabic script using recurrent neural networks. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 85–96, Barcelona, Spain (Online). Association for Computational Linguistics.

Daniel Weisberg Mitelman, Nachum Dershowitz, and Kfir Bar. 2024. Code-switching and back-transliteration using a bilingual model. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1501–1511, St. Julian's, Malta. Association for Computational Linguistics.

10. Language Resource References

Romanov, Maxim G. and Miller, Matthew Thomas and Savant, Sarah Bowen and Verkinderen, Peter. 2024. *OpenITI: a Machine-Readable Corpus of Islamicate Texts*. Zenodo, 2023.1.8. Accessed September 25, 2024.

Rustow, Marina and Koeser, Rebecca Sutton. 2025. *Princeton Geniza Project*. Center for Digital Humanities at Princeton, 4.28.0. Accessed March 25, 2025. Text data retrieved via https://github.com/princetongenizalab/pgp-text.

A. Appendix: Arabic to Hebrew Transliteration Table

As explained above in Section 3.2, this is a simplified and slightly adjusted version of the transliteration table provided in Weisberg Mitelman et al. (2024)

Arabic	Name	Hebrew	Notes
ا	alif	א	No change for hamza or madda
ب	bā’	ב	
ت	tā’	ת	
ث	thā’	ת’	
ج	jīm	ג’ / ג	See note 1
ح	ḥā’	ח	
خ	khā’	כ’	
د	dāl	ד	
ذ	dhāl	ד’	
ر	rā’	ר	
ز	zāy	ז	
س	sīn	ס	
ش	shīn	ש	
ص	ṣād	צ	
ض	ḍād	צ’	
ط	ṭā’	ט	
ظ	ẓā’	ט’	
ع	‘ayn	ע	
غ	ghayn	ג’ / ג	See note 1
ف	fā’	פ	
ق	qāf	ק	
ك	kāf	כ	
ل	lām	ל	
م	mīm	מ	
ن	nūn	נ	
ه	hā’	ה	
ة	tā’ marbūṭa	ה	
و	wāw	ו	
ي	yā’	י	
ى	alif maqṣūra	י	
ؤ	hamza on wāw	א	
ئ	hamza on yā’	י	
ء	hamza	א / ∅	See note 2

Table 5: Arabic to Judeo-Arabic (Hebrew Script) Transliteration Table

Notes:

1. **Jīm/Ghayn variation:** The algorithm models regional and scribal variation. 80% of the documents (chosen randomly) were transliterated with jīm (ج) rendered as ג (without geresh) and ghayn (غ) as ג’. In the remaining 20%, jīm → ג’ and ghayn → ג. This approximates the variation found in actual Judeo-Arabic manuscripts, as per our quantitative inspection of a sample of the corpus.

2. **Hamza:** Hamza (ء on the line) is randomly rendered as either א or omitted entirely, reflecting orthographic fluidity in medieval Judeo-Arabic texts.

3. **Final letter forms (*sofiot*):** Hebrew letters in word-final position are automatically converted to their final forms: מ→ם, נ→ן, צ→ץ, כ→ך, פ→ף. The algorithm correctly handles cases where the final letter carries a geresh modifier.

B. Appendix: JA Dataset Pre-process

B.1. PGP Corpus Preprocessing

The following algorithm was used to preprocess the PGP corpus for training.[14] In addition to the process in the pseudocode below, a small number of Hebrew translations inserted in between original JA transcriptions were removed. These can be found by searching for the word תרגום surrounded by word boundaries and inspecting manually.

Algorithm 1 PGP Corpus Preprocessing

Require: Raw transcription text T, word-length parameter N (default: 5)
Ensure: Cleaned text with gap markers

Phase 1: Bracket and Parenthesis Normalization
Replace empty bracket pairs `[ ]` or `( )` and adjacent non-whitespace with `[GAP]`
Replace unmatched `]` at line start or `[` at line end (along with adjacent text) with `[GAP]`
Remove content within curly braces `{...}` and rasura brackets ⟦...⟧
Remove remaining brackets, parentheses, and symbols: `[ ] ( ) / \ | ! ? _`

Phase 2: Dot Sequence Handling
for all sequences of one or more periods with surrounding (non-whitespace) text **do**
    if single period immediately following text and preceding whitespace **then**
        Keep as sentence-final punctuation
    else
        $c \leftarrow$ count of non-space characters in matched sequence
        if $c \leq N$ **then**
            Replace with `[ONEGAP]`
        else
            Replace with `[GAP]`
        end if
    end if
end for

Phase 3: Gap Consolidation
Replace consecutive gap markers with single `[GAP]`

Phase 4: Character Filtering
Remove lines containing Arabic script
Remove Latin alphanumeric characters (preserving gap markers)
Truncate documents at editorial markers ("footnotes", "הערות")

[14] The algorithm was developed based on the transcription conventions described at Princeton Geniza Project (2021) as well as some manual inspection of the corpus.

B.2. FJMS Corpus Preprocessing

The following algorithm was used to preprocess the FJMS corpus for training.[15]

Algorithm 2 FJMS Corpus Preprocessing

Require: Raw transcription text T, word-length parameter N (default: 5)
Ensure: Cleaned text with gap markers

Phase 1: Editorial Annotation Removal
Remove content within parentheses `(...)`

Phase 2: Character Filtering
Remove all characters except: Hebrew letters, period, apostrophe, quotation marks, geresh (`U+05F3`), gershayim (`U+04F4`), horizontal ellipsis (`U+2026`), and whitespace

Phase 3: Multi-Period Sequence Handling
for all runs of ≥ 2 consecutive periods (allowing internal spaces) with surrounding (non-whitespace) text **do**
    $c \leftarrow$ count of non-space characters in matched sequence
    if $c \leq N$ **then**
        Replace with `[ONEGAP]`
    else
        Replace with `[GAP]`
    end if
end for

Phase 4: Horizontal Ellipsis Handling
▷ The Unicode character `U+2026` (...), distinct from three periods
for all sequences containing horizontal ellipsis characters with surrounding text **do**
    if multiple horizontal ellipsis characters in sequence **then**
        Replace with `[GAP]`
    else
        Replace with `[ONEGAP]`
    end if
end for

Phase 5: Normalization
Replace consecutive gap markers with single `[GAP]`
Collapse multiple spaces to single space
Remove spaces immediately preceding periods

[15] The algorithm was developed based on manual inspection of the corpus.

B.3. OpenITI Corpus Preprocessing

The following algorithm was used to preprocess the OpenITI corpus for training.[16] The OpenITI corpus contains Arabic texts in Arabic script. Prior to preprocessing, these texts were transliterated into Hebrew characters using the table in Appendix A.

Algorithm 3 OpenITI Corpus Preprocessing (Post-Transliteration)

Require: Transliterated text T
Ensure: Cleaned text suitable for training

Phase 1: Metadata Removal
Delete all lines beginning with `#META#`

Phase 2: Line Continuation Handling
Replace newline followed by `~~` with a single space

Phase 3: Editorial Section Removal
for all markers $m \in \{$`|EDITOR|`, `|APPENDIX|`, `|PARATEXT|`$\}$ **do**
    Delete all text from m until the next line beginning with `###`
end for

Phase 4: Character Filtering
Remove all characters except: Hebrew letters, period, apostrophe, and geresh (`U+05F3`)

Phase 5: Whitespace Normalization
Strip leading and trailing whitespace from each line
Collapse multiple consecutive spaces to single space

B.4. Blau Dictionary Preprocessing

After extracting the Judeo-Arabic quotes from Blau (2006), which have a distinct font, a few minor adjustments were made: ellipses were replaced with "`[GAP]`", dots above letters (`U+0307`) were normalized to apostrophe, and all other diacritics were removed.

B.5. Post-preprocess For All Corpora

In each corpus, after its respective preprocess, blocks of text with fewer than three words were removed to make sure that at least some minimal context was provided for each word during training. Finally, geresh (`U+05F3`) was normalized to apostrophe (`U+0027`) and gershayim (`U+04F4`) to quotes (`U+0022`).

[16] The algorithm was developed based on the transcription conventions described at Romanov (2023) as well as some manual inspection of the corpus.

C. Appendix: Annotation Protocol

For the `JABERT` Cairo Genizah Dataset (Section 4.1), which contains transcriptions of 3862 lines of naturally-occurring Judeo-Arabic text in Cairo Genizah documents, we instructed the Judeo-Arabic expert with the following transcription protocol:

- Overall, the objective is to capture the letters of the manuscripts exactly as they are, without normalizing any orthographic oddities or mistakes.
- Diacritic vocalization marks (that is, diacritics which indicate vowels) shall not be transcribed.
- Consonantal diacritic dots (that is, dots placed above letters to indicate a variant of the consonant) shall be transcribed; however, if the consonantal diacritic dot is absent from the manuscript, it shall not be filled in, even if the word appears to require it, because Judeo-Arabic scribes often choose to omit these diacritic dots.
- If the manuscript employs a single ligature in place of the two-letter sequence *aleph* and *lamed*, then it shall be transcribed as a single character, using the Unicode point designated for this ligature (U+FB4F).
- If the manuscript employs a *rafe* mark (a horizontal line above a letter), then it shall be transcribed using the corresponding Unicode point (U+05BF).
- Finally, because Cairo Genizah manuscripts can be difficult to read in places, and because scribal strokes can be ambiguous at times, and because only one Judeo-Arabic expert was employed to do the transcriptions for this first dataset, we instructed our Judeo-Arabic expert to completely skip any line which posed an ambiguity or reading difficulty. Indeed, because the aim of this dataset is to evaluate a model's ability to predict a word based on the context of the individual line, the elision of lines which happen to be difficult to decipher does affect the effectiveness of the dataset. Nevertheless, in future work, we plan to have the lines transcribed independently by a second annotator, so that we may properly evaluate inter-annotator agreement and indicate cases of disagreement.

D. Appendix: Homograph Dataset

Table 6 presents the complete list of homographs used in our evaluation dataset. Each homograph is a word spelled identically in unvocalized Judeo-Arabic text but carrying distinct meanings depending on context. The "Arabic" column shows the vocalized Arabic form that disambiguates pronunciation and meaning. Arabic verb forms are indicated: Form I (basic stem), Form II (intensive/causative), act. = active, pass. = passive. The 16 homographs yield 37 distinct meanings across 1,119 annotated samples.

#	Translit.	Arabic	Meaning
1. אלא (*'l'*)			
1	*alā*	أَلَا	Interrog.: "is it not?"
2	*illā*	إِلَّا	"except", "other than"
2. מן (*mn*)			
3	*min*	مِن	Prep.: "from"
4	*man*	مَن	Rel. pron.: "whoever"
3. מא (*mā*)			
5	*mā*	مَا	Int./Neg.: "what?"/"not"
6	*mā*	مَا	Rel. pron.: "that which"
4. צרב (*ṣrb*)			
7	*ḍarb*	ضَرب	Noun: "type", "manner"
8	*ḍaraba*	ضَرَبَ	Verb: "struck"
9	*ḍaraba*	ضَرَبَ	Verb: "gave a parable"
5. עלי (*'ly*)			
10	*'alā*	عَلَى	Prep.: "on", "upon"
11	*'alayya*	عَلَيَّ	Prep.+pron.: "on me"
6. קסם (*qsm*)			
12	*qasama*	قَسَمَ	Verb: "divided"
13	*qasam*	قَسَم	Noun: "oath"
14	*qism*	قِسم	Noun: "portion"
15	*qasama*	قَسَمَ	Verb: "swore"
7. יערף (*y'rf*)			
16	*ya'rifu*	يَعرِفُ	Verb (I act.): "knows"
17	*yu'rafu*	يُعرَفُ	Verb (I pass.): "is known"
18	*yu'arrifu*	يُعَرِّفُ	Verb (II): "informs"
8. ראי (*r'y*)			
19	*ra'ā*	رَأَى	Verb: "saw", "thought"
20	*ra'y*	رَأي	Noun: "opinion"
9. קבל (*qbl*)			
21	*qabila*	قَبِلَ	Verb: "received"
22	*qabla*	قَبلَ	Prep.: "before"
10. דכר (*ḏkr*)			
23	*ḏakara*	ذَكَرَ	Verb: "mentioned"
24	*ḏikr*	ذِكر	Noun: "memory"
25	*ḏakar*	ذَكَر	Noun: "male"
11. פרק (*frq*)			
26	*faraqa*	فَرَقَ	Verb: "separated"
27	*farq*	فَرق	Noun: "difference"
12. עאלם (*'lm*)			
28	*'ālam*	عَالَم	Noun: "world"
29	*'ālim*	عَالِم	Adj.: "knows"
13. כלק (*ẖlq*)			
30	*ẖalaqa*	خَلَقَ	Verb: "created"
31	*ẖalq*	خَلق	Noun: "creation"
14. גיר (*ġyr*)			
32	*ġayr*	غَير	Neg.: "not", "non-"
33	*ġayr*	غَير	Prep.: "other than"
15. פהם (*fhm*)			
34	*fahm*	فَهم	Noun: "understanding"
35	*fa-hum*	فَهُم	Conj.+pron.: "and they"
16. אכד (*'kd*)			
36	*akkada*	أَكَّدَ	Verb (II): "confirmed/affirmed"
37	*aẖaḏa*	أَخَذَ	Verb (I): "took"

Table 6: Judeo-Arabic homographs: word forms and meanings