A Tale of Two Scripts: Transliteration and Post-Correction for Judeo-Arabic

Juan Moreno Gonzalez,[1] Bashar Alhafni,[2] Nizar Habash[3]
[1]University of Cambridge
[2]Mohamed bin Zayed University of Artificial Intelligence
[3]New York University Abu Dhabi
jm2553@cam.ac.uk, bashar.alhafni@mbzuai.ac.ae, nizar.habash@nyu.edu

Abstract

Judeo-Arabic refers to Arabic variants historically spoken by Jewish communities across the Arab world, primarily during the Middle Ages. Unlike standard Arabic, it is written in Hebrew script by Jewish writers and for Jewish audiences. Transliterating Judeo-Arabic into Arabic script is challenging due to ambiguous letter mappings, inconsistent orthographic conventions, and frequent code-switching into Hebrew. In this paper, we introduce a two-step approach to automatically transliterate Judeo-Arabic into Arabic script: simple character-level mapping followed by post-correction to address grammatical and orthographic errors. We also present the first benchmark evaluation of LLMs on this task. Finally, we show that transliteration enables Arabic NLP tools to perform morphosyntactic tagging and machine translation, which would have not been feasible on the original texts. We make our code and data publicly available.[1]

JA	Arabic	Dotless T	Dotted T	English
קאל	قال *qAl*	قال *qAl*	قال *qAl*	*said*
אלכזרי	الخزري *Alxzry*	الكزري *Alkzry*	الخزري *Alxzry*	*al-Khazari*
,	: ،	، ،	، ،	,
וכיף	وكيف *wkyf*	وكيف *wkyf*	وكيف *wkyf*	*and how*
דלך	ذلك *θlk*	دلك *dlk*	ذلك *θlk*	*is that*
והברכות	"والتسبيحات" *w"AltsbyHAt"*	والبركوت *wAlbrkwt*	والبركوت *wAlbrkwt*	*and the blessings*
כלפה	كلفة *klfħ*	كلفه *klfh*	كلفه *klfh*	*burden*
ואידה	زائدة *zAŷdħ*	زايده *zAydh*	زايده *zAydh*	*extra*
.	. .	. .	. .	.

Table 1: A sentence in Judeo-Arabic (Hebrew script), aligned with Arabic script, Arabic transliterations from dotless (Dotless T) and dotted (Dotted T) Hebrew, and English glosses. The sentence is 'Al-Khazari said: And how is that when blessings are an extra burden?'

1 Introduction

Judeo-Arabic (JA) refers to Arabic varieties historically used by Jewish communities across the Arab world, primarily in the Middle Ages. Although closely related to regional Arabic dialects, JA is written in Hebrew script and incorporates elements from Hebrew. Thousands of JA texts are now available online, covering genres such as philosophy, biblical commentary, and Bible translations (Friedberg Geniza Project, 1999; Rustow and Koeser, 2022). However, because JA texts were intended for readers familiar with Hebrew script, they remain largely inaccessible to Arabic speakers and incompatible with modern Arabic NLP tools, which typically assume Arabic script input.

Transliterating JA into Arabic script is therefore a crucial step for making these texts accessible to Arabic readers and for ensuring compatibility with modern Arabic NLP tools. However, this task is challenging due to ambiguous character mappings between the scripts, inconsistent orthographic conventions, and frequent Hebrew borrowings (Table 1). To address this, we **propose a two-step approach**: an initial transliteration stage using a rule-based method, followed by a post-correction step to resolve orthographic and grammatical errors. We also present **the first benchmark evaluation of large language models (LLMs)** on this task. Notably, **our approach requires no training on JA data**, yet outperforms previous work. Finally, we demonstrate the **downstream impact of transliteration** by enabling morphosyntactic tagging and Arabic-to-English machine translation (MT) on JA texts. Our results demonstrate the viability of this approach: post-correction significantly improves transliteration quality, and the resulting outputs can be effectively processed by Arabic NLP tools for both morphosyntactic tagging and MT.

[1]https://github.com/CAMeL-Lab/jawhar

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2100–2113
March 24-29, 2026

2 Background and Related Work

2.1 Judeo-Arabic Linguistic Facts

Judeo-Arabic (JA) is the language of the Jews from the Arabo-Islamic world. Its history spans from pre-Islamic Arabia to the present day and went through several different periods (Blau, 1961, 1965). This paper focuses on Classical JA, which developed in parallel to Classical Arabic in the territories under Muslim rule around the Mediterranean and the Middle East during the Middle Ages. JA is a Jewish language. It was written by Jews and its intended audience were other Jews. This explains two of its most distinctive characteristics. First, JA is written in Hebrew characters. Second, it incorporates some vocabulary from Hebrew and, less commonly, Aramaic. One should be cautious not to describe JA just as Arabic written in Hebrew characters (Stillman, 2010). Most Jews from the medieval Mediterranean and Middle East lived under Muslim rule and spoke Arabic as their mother tongue. However, the most common writing system they used was Hebrew, not Arabic. From a very young age, Jewish children were taught Hebrew for religious purposes. Some of them would not be able to read or write Arabic, despite reading and writing in JA. Thus, for them JA was not a direct transliteration of Arabic, but rather the obvious way of registering their language in writing. For this same reason, some scholars have approached JA as an Arabic dialect (Mansour, 1991; Khan, 1997; Gębski, 2024).

Transliterating JA written in Hebrew to Arabic characters is as artificial as transliterating it to Latin script. Nevertheless, this approach is justified when the goal is to make JA accessible to Arabic readers who do not read Hebrew, or to enable the use of NLP tools designed for Arabic. This transliteration task entails many challenges.

The Arabic *abjad* has twenty-eight basic letters and some additional special characters (e.g., Hamzated Alifs and Teh Marbuta); the Hebrew *abjad* has twenty-two. While many Hebrew letters have one-to-one or many-to-one mappings to Arabic letters (Blau, 1961; Lanza, 2020), some Arabic letters without corresponding sounds in the Hebrew *abjad* are represented by a Hebrew letter with *an upper dot*, imitating the Arabic alphabet (see Table 2). Seven Judeo-Arabic Hebrew letters have an upper dot diacritic variant: גדהטכצת *jdhTkSt*.[2,3]

Unfortunately, the use of diacritical marks or the Hebrew-Arabic character mapping has never been standardized. Thus, the rules followed by different texts may vary (Lanza, 2020). Furthermore, common errors of omission or misplacement of the upper dot can lead to erroneous conversions, בלגה بلجة *bljħ*[3] instead of the correct בלגה بلغة *blγħ* 'in a language', or והדא وةدا *whdA* instead of the correct והדא وهذا *whðA* 'and this'. Furthermore, not every Arabic character is explicitly marked in JA. The Hamza at the end of the word is often dropped (גא جا *jA* instead of جاء *jA'* 'he came'), a common phenomenon in many Arabic dialects (Habash, 2010). Although sometimes it is represented by its support letter, e.g., רויא رويا *rwyA* for رؤيا *rwyA* 'vision' and סילת سيلت *sylt* for سئلت *sŷlt* 'be asked'.

Finally, transliteration is problematic for Hebrew words in JA. Labeling them as Arabic or Hebrew imposes an artificial distinction, since they function as part of JA and are treated like native Arabic words. For example, some Hebrew roots could be integrated into Arabic and then conjugated following its verbal system. This is the case of גיר غير *γyr* 'convert to Judaism' (Stillman, 2010). In Arabic texts derived from JA, such borrowings are typically translated rather than transliterated.

2.2 Transliteration into Arabic Script

Latin-based Arabic Transliteration Several studies have investigated the transliteration of Latin-based scripts into Arabic, with the most extensively studied case being *Arabizi*, a non-standard romanization of Arabic widely used in informal online communication (Chalabi and Gerges, 2012; Voss et al., 2014; Darwish, 2014; Al-Badrashiny et al., 2014; Eskander et al., 2014; Guellil et al., 2017; Younes et al., 2018; Shazal et al., 2020; Nagoudi et al., 2022). While Arabizi has received the most attention when it comes to transliterating into Arabic, other Latin-script languages have also been explored. Micallef et al. (2023) treated Maltese as a dialect of Arabic, and investigated the effect of transliterating Maltese into

[2]The upper dot is sometimes used for other purposes, such as indicating abbreviations (JA: וג *wγ*, Ar: وغير *wγyr*, 'etc') or indicating that a letter has a numeric value (מד, "44").

[3]Arabic transliteration is presented in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007):
ا ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي
A b t θ j H x d ð r z s š S D T Ď ς γ f q k l m n h w y
and additionally: ' ء, Â أ, Ǎ إ, Ā آ, ŵ ؤ, ŷ ئ, ħ ة, ý ى.

Arabic script on downstream NLP tasks. Previous work has also addressed the problem of diacritizing and transliterating foreign words, particularly English proper nouns, into fully diacritized Arabic (Mubarak et al., 2009; Darwish et al., 2017; Bondok et al., 2025).

Judeo-Arabic to Arabic Transliteration In the case of JA, Bar et al. (2015) proposed a three-step pipeline: identifying code-switched words to distinguish Hebrew words from Arabic, applying a character-level statistical MT model for transliteration, and using a recurrent neural network (RNN) for post-correcting a limited set of orthographic errors. Terner et al. (2020) proposed an RNN to perform end-to-end transliteration. More recently, Mitelman et al. (2024) adopted a two-step approach: first identifying code-switched words in Hebrew, and then transliterating JA to Arabic script. Both of their components leverage HeAr-BERT (Rom and Bar, 2024), a BERT-based model pretrained on Arabic and Hebrew.

While prior work on JA transliteration has made important contributions toward making JA accessible to Arabic readers, it suffers from several limitations in data preprocessing, evaluation metrics, and reproducibility. Data preprocessing pipelines vary considerably, particularly in how code-switched Hebrew words, orthographic markers (e.g., diacritics, hamzas, dots), and punctuation are handled. All previous studies rely on extracted parallel JA and Arabic texts aligned from published books; however, they adopt different alignment algorithms and uniformly remove Arabic translations corresponding to Hebrew citations (§2.1) from the target Arabic data. Orthographic preprocessing also differs: Bar et al. (2015) retain diacritics, Terner et al. (2020) and Mitelman et al. (2024) remove them; Mitelman et al. (2024) further remove hamzas and dots from the Hebrew input and retain Hebrew citations in the output, producing mixed-script transliterations. Punctuation is also inconsistently treated, with Bar et al. (2015) and Terner et al. (2020) preserving it, while Mitelman et al. (2024) discard it entirely. Evaluation metrics also vary: Bar et al. (2015) reported character-level accuracy, Terner et al. (2020) used letter error rate, while Mitelman et al. (2024) adopted character-level precision, recall, F_1, and accuracy. Finally, prior work uses different data sources for training and evaluation.

None of the previous studies make their models or data publicly available, with the exception of Mitelman et al. (2024), who released their pretrained models and data. However, the released data is provided only at the character level without sentence boundaries or punctuation, limiting its utility for full-text evaluation. We therefore benchmark their system under our setup and treat it as the only available baseline for comparison. Comparisons with earlier studies remain infeasible due to the absence of released models and outputs.

In our work, as in prior studies, the goal is to facilitate access to JA texts for Arabic readers. However, our approach differs in several key aspects. We preserve the full input text structure, retaining Hebrew tokens and punctuation during processing, and restrict evaluation to Arabic tokens. We systematically evaluate a range of models *without* any training on JA data and explore systems ranging from simple character mappings to LLMs. Inspired by Bar et al. (2015), we incorporate a post-correction stage to address orthographic errors; however, rather than relying on task-specific models targeting a limited set of orthographic errors, we leverage state-of-the-art pretrained Arabic grammatical error correction systems. Finally, we address an aspect overlooked in prior work by analyzing the impact of dotless versus dotted Hebrew orthography on transliteration performance, and demonstrate how transliteration benefits downstream Arabic NLP applications, such as morphosyntactic tagging and MT.

3 Data

3.1 Sources

We use the edition of *Al-Khazari* by Nabih Bashir (Ha-Levi, 2012). This is the only edition of the text in Arabic and follows closely the JA original. However, because it was created with an Arabic audience in mind, the text was edited for clarity in some places and some of the terms from the JA original were replaced with their modern Arabic equivalents. This happened especially with words that come from Hebrew or with Biblical quotations that appeared in Hebrew in the original text. For example, the Hebrew word עני ʿny 'poor' is translated into its Arabic equivalent فقير *fqyr*. Thus, our Arabic source was not an exact human transliteration of the JA original, and we had to deal with the mismatches between the JA and the Arabic versions in an alignment step (§3.3).

Nabih Bashir has produced other Arabic versions of JA texts: the "Introduction to the Mishnah" by

Maimonides and "The book and beliefs and opinions" by Saadia Gaon. Both were used by Mitelman et al. (2024) for training, although they only used Al-Khazari for evaluation. These two texts are unpublished and not publicly available, which is the primary reason we excluded them from our work and focused on Al-Khazari instead, though not the only one. A further motivation, however, relates to the use of the upper dot diacritic. As discussed above (§2.1), its usage is not consistent across all JA texts, but it is consistent within a single text. This consistency within Al-Khazari enables us to analyze the upper dot diacritic in a controlled manner and properly understand its function. In contrast, drawing conclusions from multiple texts without accounting for text-specific conventions can misleadingly suggest that the upper dot diacritic is mere noise, as assumed by Mitelman et al. (2024). Our findings show that the upper dot diacritic carries meaning, can aid disambiguation, and should therefore be preserved whenever it appears. Finally, our approach requires no training on JA data: it relies on deterministic transliteration followed by pretrained grammatical error correction models.

3.2 Basic Character Mapping

For our basic one-to-one character mapping, we follow common conventions used in the field of JA studies (Blau, 1961; Lanza, 2020). See our full mapping in Table 2. Our choices were later confirmed by the success rate of our transliteration. Our transliteration results using JA text with the upper dot diacritic (64.9%) outperformed those without it (53.0%) (§5.2). Examples where the upper dot diacritic made a difference include יכאטבה يخاطبه *yxATbh* 'he speaks to him', מרצי مرضي *mrDy* 'satisfactory', and תגיר تغير *tγyr* 'he changed', whose respective dotless versions are *incorrect*: יכאטבה يكاطبه *ykATbh*, מרצי مرصي *mrSy*, and תגיר تجير *tjyr*.

3.3 Preprocessing and Alignment

Both the JA source and the Arabic reference transliteration produced by Nabih Bashir were obtained from Sefaria, an online library of Jewish texts.[4] The files were divided into the same number of sections, which varied in length from a single sentence to several paragraphs. Footnotes in the Arabic reference were removed, after which we created two parallel text files with each section aligned to a line. We then reviewed sections with substantial word imbalances between the JA and Arabic texts and, after close reading, decided to drop 16 lines.[5]

We first produced an automatic Arabic transliteration of the JA source using the character mapping described in §3.2, accounting for the upper dot diacritic. Before alignment, Arabic diacritics were removed from the reference as commonly done in Arabic NLP (Elgamal et al., 2024; Inoue et al., 2026), and punctuation was separated in both the reference and the automatic transliteration.

We then aligned the Arabic reference and the automatic transliteration using word-level alignment (Khalifa et al., 2021). Words in the automatic transliteration with no match in the reference were mapped to an unknown token (*UNK*) to preserve

Undotted			Dotted		
Hebrew	**Arabic**		**Hebrew**	**Arabic**	
א	ا	*A*			
ב	ب	*b*			
ג	ج	*j*	גּ	غ	*γ*
ד	د	*d*	דּ	ذ	*ð*
ה	ه	*h*	הּ	ة	*ħ*
ו	و	*w*			
ז	ز	*z*			
ח	ح	*H*			
ט	ط	*T*	טּ	ظ	*Ď*
י	ي	*y*			
כ ך	ك	*k*	כּ ךּ	خ	*x*
ל	ل	*l*			
מ ם	م	*m*			
נ ן	ن	*n*			
ס	س	*s*			
ע	ع	*ʕ*			
פ ף	ف	*f*			
צ ץ	ص	*S*	צּ ץּ	ض	*D*
ק	ق	*q*			
ר	ر	*r*			
ש	ش	*š*			
ת	ت	*t*	תּ	ث	*θ*

Table 2: Arabic to Judeo-Arabic mapping used for transliteration. The dotted version takes into account the upper dot diacritic in the Judeo-Arabic text.[3]

[4] https://www.sefaria.org/Kuzari

[5] The lines dropped were: 130, 131, 203, 229, 230, 231, 232, 233, 234, 235, 242, 243, 291, 292, 303, 321.

alignment, while unmatched insertions from the reference were discarded. Because the transliteration maintains a one-to-one correspondence with the JA source, this procedure effectively maps JA words to their Arabic counterparts. Finally, after completing the three-way alignment between the JA source, automatic transliteration, and reference, we removed upper dot diacritics from the JA source to generate an alternative *dotless* Arabic transliteration.

For evaluation, each word in the JA source was labeled as JA, Hebrew, or punctuation. Hebrew words were identified primarily through the presence of biblical quotations; in the Arabic reference, such words are usually fully diacritized and translated, rather than transliterated (§3.1).

To validate the labeling, two independent annotators proficient in both Arabic and Hebrew evaluated a random subset of 1,000 words and reached full agreement, identifying only three labeling errors. In all cases, a word labeled as JA was in fact of Hebrew origin and had been translated into Arabic in the reference: והמשכילים *whmškylym* as والعقلاء *wAlςqlA'* 'and the wise ones'; יבינו, *ybynw* as يفهمونه *yfhmwnh* 'they understand it'; and the JA abbreviation תע *tς* as تعالى *tςAlý* 'Exalted'.

It is important to note that, both Hebrew words and punctuation were excluded from evaluation. Hebrew words were excluded because their Arabic counterparts are translations, not transliterations, and therefore not meaningful for transliteration evaluation. Punctuation was excluded because it is inconsistently used between the JA source and the Arabic reference; where it is often changed, added, or removed in the Arabic edition, making it unreliable as a point of comparison. See Appendix B Table 7 for an alignment example.

3.4 Statistics

Our dataset is made of 325 paragraphs, 1,286 sentences and 46,529 words. Of those words, 4,576 (9.8%) are labeled as punctuation, 916 (2%) as Hebrew and 41,362 (88.3%) as JA. Among the JA words, 856 were not aligned to anything in the Arabic reference, due to deletions or alignment errors. In total, these categories account for 6,348 words (13.6%), which are excluded from transliteration evaluation. The evaluation is thus performed on the remaining 40,506 words (86.5%).

4 Approach

Our goal is to transliterate JA texts written in Hebrew script into readable and fluent Arabic script. To this end, we propose a two-step approach. The first step starts with **transliteration** to convert the Hebrew script into Arabic. However, this step alone often results in Arabic text that is orthographically erroneous (§2.1). We address this by introducing a second step of **post-correction**, which treats the output of transliteration as noisy Arabic and applies grammatical error correction (GEC) models to improve fluency and correctness.

4.1 Transliteration

Character Mapping We implement a rule-based character mapping approach to convert JA from Hebrew to Arabic script using manually curated mappings. These mappings are informed by linguistic analyses of JA orthography, as outlined in §2.1 and §3.2. It is important to note that this type of direct transliteration often results in orthographic spelling errors and inconsistencies, such as missing hamzas, incorrect use of dots, and irregular morphological forms.

LLMs We evaluate OpenAI's GPT-3.5-turbo and GPT-4o (OpenAI et al., 2024), prompting them to directly transliterate JA from Hebrew to Arabic script. The models are accessed via the OpenAI API and evaluated in a zero-shot setting. The prompts we use are in Table 8 in Appendix C.

4.2 Post-Correction

While transliteration maps JA text from Hebrew to Arabic script, it does not guarantee orthographic or grammatical correctness. To address these challenges, we introduce a post-correction stage aimed at producing fluent, orthographically correct Arabic. We treat this as a Modern Standard Arabic (MSA) GEC task and explore both publicly available pretrained MSA GEC systems and LLMs.

For pretrained systems, we use the sequence-to-sequence models developed by Alhafni et al. (2023), including their best-performing vanilla model (Seq2Seq) as well as variants that incorporate additional signals such as morphological preprocessing and grammatical error detection (Seq2Seq+MG). We also evaluate the recently introduced text editing GEC system by Alhafni and Habash (2025), which reformulates GEC as a sequence tagging problem (SWEET).

In addition, we evaluate LLMs as general-purpose Arabic GEC systems. We experiment with GPT-3.5-turbo and GPT-4o, along with two open-source Arabic-centric models: Fanar (Team et al., 2025) and Jais-13B-Chat (Sengupta et al., 2023). GPT-3.5-turbo, GPT-4o, and Fanar are accessed through the OpenAI API, while Jais is prompted using Hugging Face Transformers (Wolf et al., 2020). Following the prompting strategy of Alhafni and Habash (2025), we use English prompts in a zero-shot setting (Table 8 Appendix C).

Details about all the models and the hyperparameters we use are provided in Appendix A.

5 Experimental Results

5.1 Evaluation Metrics

We use separate evaluation metrics for the transliteration and post-correction stages, aligned with the goals of each task.

For **transliteration**, We rely on the word-level alignment described in §3.3 and report exact match accuracy, which measures the percentage of transliterated JA words in Hebrew script that exactly match the gold Arabic reference. This metric reflects the system's ability to accurately map Hebrew words to Arabic script. Hebrew words and punctuation are excluded from the evaluation, as they are either fully translated or inconsistently used, and do not contribute meaningfully to the task (§3.3).

For **post-correction**, we use the MaxMatch (M^2) scorer (Dahlmeier and Ng, 2012), a standard metric for evaluating GEC systems. The M^2 scorer computes overlap between the system's edits and the gold-standard edits derived from the target Arabic, reporting precision (P), recall (R), F_1, and $F_{0.5}$ scores. The $F_{0.5}$ metric places twice the weight on precision relative to recall, emphasizing the correctness of the edits over their coverage. Gold edits are generated by aligning the automatic *dotless* transliterated Arabic (§3.3) with the Arabic reference using the algorithm proposed by Alhafni et al. (2023). Unlike in transliteration, Hebrew words and punctuation are not excluded in GEC. This is because preserving the full sentence structure ensures that the model's corrections are evaluated in context, accurately reflecting its performance without ignoring potential errors these elements may introduce or correct.

	Dotless	**Dotted**
Mitelman et al. (2024)	**70.0**	**70.1**
CharMapper	53.0	64.9
GPT-3.5-turbo	32.8	32.2
GPT-4o	52.9	38.6

Table 3: Transliteration results in terms of word-level exact match accuracy. Dotted and Dotless refer to whether the input Judeo-Arabic text includes dots or not.

5.2 Transliteration Results

Table 3 presents transliteration results measured by word-level exact match accuracy. The model by Mitelman et al. (2024) achieves the highest overall accuracy on both dotted and dotless input. This is expected, as it was trained directly on JA data. Interestingly, its performance on dotted and dotless text is nearly identical, despite the model being trained only on dotless input.

The rule-based character mapping approach, henceforth referred to as *CharMapper*, outperforms both GPT-3.5 and GPT-4o on dotted and dotless JA inputs, with particularly strong performance on dotted text. In contrast, the LLMs perform better on dotless input compared to dotted input, suggesting that they are more effective at handling dotless input.

5.3 Transliteration & Post-Correction Results

Table 4 presents transliteration and post-correction results on transliterated Arabic derived from dotless and dotted JA input, using the baseline model of Mitelman et al. (2024) and our rule-based system *CharMapper*.

Before post-correction, Mitelman et al. (2024) shows identical GEC performance on dotless and dotted input, mirroring its transliteration trend (§5.2). In contrast, *CharMapper* records an $F_{0.5}$ of 0 on dotless input, as evaluation compares its raw transliteration to the gold reference (§5.1). On dotted input, it outperforms Mitelman et al. (2024) in GEC (40.4 $F_{0.5}$) but lags in transliteration accuracy.

Post-correction substantially improves GEC and transliteration for both Mitelman et al. (2024) and *CharMapper*, except for Fanar and Jais on Mitelman et al. (2024) outputs and Jais on *CharMapper*. For Mitelman et al. (2024), dotless and dotted results remain similar, with Seq2Seq+MG giving the best GEC scores (55.5 dotless, 55.9 dotted $F_{0.5}$) and GPT-4o the best transliteration accuracy (86.1 dotless, 87.4 dotted). For *CharMapper*, dotted in-

	Dotless					Dotted				
	P	**R**	**F_1**	**$F_{0.5}$**	**Acc.**	**P**	**R**	**F_1**	**$F_{0.5}$**	**Acc.**
Mitelman et al. (2024)	58.5	14.8	23.6	36.7	70.0	58.4	14.7	23.5	36.7	70.1
⇒ Seq2Seq	75.0	26.6	39.3	55.0	82.9	74.9	26.7	39.3	55.0	82.8
⇒ Seq2Seq+MG	75.6	26.9	39.7	55.5	82.9	75.5	27.4	40.2	55.9	82.9
⇒ SWEET	76.9	23.5	36.0	52.9	83.9	77.0	23.5	36.0	52.9	84.1
⇒ GPT-3.5-turbo	74.0	20.8	32.5	49.0	78.1	74.1	21.6	33.4	49.8	79.3
⇒ GPT-4o	77.9	23.8	36.4	53.5	86.1	77.7	23.5	36.1	53.2	87.4
⇒ Fanar	66.4	9.7	17.0	30.7	61.0	66.8	11.5	19.6	34.0	61.7
⇒ Jais-13B-Chat	54.0	3.8	7.0	14.7	24.9	53.6	4.2	7.8	16.0	24.7
CharMapper	**100**	0	0	0	53.0	75.5	14.1	23.8	40.4	64.9
⇒ Seq2Seq	72.8	27.2	39.6	54.5	74.5	81.2	30.3	44.1	60.8	85.2
⇒ Seq2Seq+MG	73.4	**29.0**	41.6	56.2	76.3	82.5	30.8	44.9	61.8	86.9
⇒ SWEET	71.8	28.5	40.8	55.1	76.5	80.1	**32.9**	**46.7**	62.3	86.0
⇒ GPT-3.5-turbo	72.1	23.9	35.9	51.4	75.6	78.7	27.5	40.8	57.4	82.6
⇒ GPT-4o	82.0	28.4	**42.2**	**59.5**	**86.4**	**85.9**	30.7	45.2	**63.1**	**90.4**
⇒ Fanar	65.5	13.9	23.0	37.7	67.6	71.9	18.9	29.9	46.0	74.0
⇒ Jais-13B-Chat	48.0	3.4	6.4	13.3	31.4	53.2	4.2	7.8	16.0	31.1

Table 4: Post-correction results for various systems on transliterated Arabic text derived from either dotless or dotted Judeo-Arabic Hebrew script.

	Dotless						Dotted					
	Match	**Match'**	**POS**	**Diac**	**Lex**	**Tok**	**Match**	**Match'**	**POS**	**Diac**	**Lex**	**Tok**
GPT-3.5-turbo	32.8	35.8	46.3	33.4	38.0	35.7	32.2	35.6	46.7	33.1	37.7	35.5
GPT-4o	52.9	53.5	59.3	51.9	55.1	53.4	38.6	39.2	44.2	37.9	40.6	39.2
Mitelman et al. (2024)	70.0	83.9	85.2	80.0	83.4	83.6	70.0	84.0	85.2	80.0	83.5	83.7
⇒ GPT-4o	86.1	86.4	89.1	83.5	87.0	86.3	87.4	87.7	90.3	84.7	88.2	87.5
CharMapper	53.0	71.2	76.9	67.8	71.4	71.0	64.9	85.8	88.5	82.7	86.0	85.5
⇒ GPT-4o	**86.4**	**86.7**	**92.4**	**84.7**	**87.7**	**86.6**	**90.4**	**90.6**	**93.9**	**88.9**	**91.5**	**90.6**

Table 5: Morphological tagging results in terms of word-level accuracy for Arabic outputs produced by various transliteration systems, using either dotted or dotless Judeo-Arabic input.

put yields consistently larger gains compared to dotless, with GPT-4o leading on both GEC (59.5 dotless, 63.1 dotted $F_{0.5}$) and transliteration (86.4 dotless, 90.4 dotted).

These results highlight two key points: (1) post-correction consistently improves performance, and (2) despite its simplicity, *CharMapper* with the upper dot diacritic, when combined with GEC, surpasses the JA-specific Mitelman et al. (2024) model under the same post-correction setting.

5.4 Downstream Tasks Results

Most Arabic NLP tools assume input in Arabic script, making transliteration a necessary step for processing JA texts. Converting JA from Hebrew to Arabic script enables these texts to be analyzed and processed using existing Arabic NLP systems. To demonstrate the utility of transliteration, we evaluate its impact on two downstream tasks: morphosyntactic tagging and machine translation.

5.4.1 Morphological Tagging

We use the contextualized Arabic morphosyntactic tagger (Inoue et al., 2022) from CAMeL Tools (Obeid et al., 2020) to obtain morphological tags for both the transliterated JA text and its gold Arabic reference. Although no gold morphological annotations exist for our dataset, we treat the tags generated for the gold Arabic reference as *silver annotations*, allowing us to compare the tagger's output on the transliterated text against a high-quality baseline. We evaluate six systems: GPT-3.5 and GPT-4o used directly for transliteration, and four systems based on the model by Mitelman et al.

(2024) and our *CharMapper*. Each is applied either alone or followed by GPT-4o post-correction, the setup with the best transliteration results.

We report word-level accuracy across multiple morphological features, including exact match (**Match**), minimal spelling correction (**Match'**), part-of-speech (**POS**), fully diacritized form (**Diac**), lemma (**Lex**), and tokenization (**Tok**). It is worth noting that the **Match** score corresponds to the transliteration accuracy reported earlier (§5.2) and reflects whether the system's output exactly matches the gold Arabic word form.

Table 5 presents morphological tagging results over Arabic outputs produced from transliterated JA input. For both GPT-3.5 and GPT-4o, outputs derived from dotless input yield better tagging performance, with GPT-4o showing a particularly large improvement over its dotted counterpart. In contrast, both Mitelman et al. (2024) and *CharMapper* outperform the LLMs across all tagging dimensions, regardless of input type, with higher accuracy on dotted text, a difference especially pronounced for *CharMapper*.

Post-correction with GPT-4o yields substantial improvements across all features for both Mitelman et al. (2024) and *CharMapper*, on both dotless and dotted input. The strongest results are achieved with *CharMapper* on dotted text, where performance is consistently higher, highlighting the value of the upper-dot diacritic and the effectiveness of combining *CharMapper* with high-quality error correction to produce well-formed Arabic.

5.4.2 Machine Translation

To assess the impact of transliterating JA to Arabic on MT to English, we first generate *silver translations* by translating the Arabic gold references into English using GPT-4o. We then translate the Arabic outputs of six systems into English using GPT-4o: GPT-3.5, GPT-4o, Mitelman et al. (2024) (alone or with GPT-4o post-correction), and *CharMapper* (alone or with GPT-4o post-correction). All translations are generated in a zero-shot setting with English prompts (Table 8, Appendix C). Table 6 presents Arabic-to-English MT results. Across all systems, translations from dotted input outperform those from dotless input, except Mitelman et al. (2024). The highest BLEU scores are achieved with GPT-4o post-correction.

	Dotless	**Dotted**
GPT-3.5-turbo	5.1	3.7
GPT-4o	23.8	24.6
Mitelman et al. (2024)	**26.1**	24.6
⇒ GPT-4o	25.5	27.1
CharMapper	20.2	24.1
⇒ GPT-4o	24.1	**28.3**

Table 6: Arabic-to-English machine translation results in terms of BLEU for outputs generated by different transliteration systems, using either dotted or dotless Judeo-Arabic input.

6 Error Analysis

Our dataset contains 46,529 words organized into 325 paragraphs. We exclude 6,023 words from evaluation due to punctuation, Hebrew content, or alignment errors, leaving 40,506 words for analysis. We present four error analysis studies below.

6.1 Analysis of Dotting Effect

Of the 40,506 analyzed words, 83.4% contain no dotted letters, while 16.6% contain dotted letters. The *CharMapper* system correctly maps 52.9% of all words with no dots (63.5% relative) and 11.9% of all words with dots (71.8% relative). The 11.9% last mentioned is the difference between *CharMapper* baseline on Dotted vs. Dotless input. The vast majority of errors for both dotted and undotted words stem from missing additional dots or from Arabic-only letters in the reference, such as hamza forms. For example, סנה 'year' lacks a dot on the final letter, yielding سنه *snh* instead of the correct سنة *snħ*. Incorrect dot insertion is rare, occurring in only 0.04% of dotted words, and typically involves letters that do not permit dotting. For instance, the dot in הֵם 'they' is superfluous and is ignored, resulting in the correct output هم *hm*.

6.2 Recovery from Dotless Input in GPT-4o

Comparing the *CharMapper* ⇒ GPT-4o transliteration performance in Table 4 on Dotted (90.4%) vs Dotless (86.4%) input reveals a 4.0% absolute difference, which is substantially smaller than the 11.9% gap observed for the *CharMapper* baseline. This suggests that GPT-4o is largely able to recover from the absence of dots in the dotless experimental setting. To unpack this overall robustness, we examine performance separately on words with and without dotted letters (in the Dotted input setting)

and the effect of removing the dots in the Dotless input setting. We further analyze recovery behavior at the letter level.

Performance on words without dots (83.4% of all words) differs only marginally between Dotted and Dotless input settings (89.8% vs. 89.2%), whereas the gap for words containing dots (16.6%) is substantial (93.4% vs. 72.1%). This confirms that GPT-4o successfully recovers the majority of cases that would otherwise require dotted input, although recovery effectiveness varies considerably across letters. We group the letters into three categories:

- ד-*d* / ڈ-*ð* (93.1%), ה-*h* / ۃ-*ħ* (85.0%), and ת-*t* / ث-*θ* (81.4%), which resemble errors made by Arabic native speakers, arising from dialectal variation as well as typographic errors (Attia et al., 2012; Habash et al., 2018).
- ט-*T* / ظ-*Ď* (82.1%) and ص-*S* / ض-*D* (69.2%), which primarily resemble Arabic native speaker typographic errors only.
- ג-*j* / غ-*γ* (54.1%) and ך-*k* / خ-*x* (47.7%), which differ substantially from typical Arabic native speaker errors and are therefore more challenging for MSA GEC systems.

Words containing multiple dotted letters in the Dotted setting are the most difficult to recover in the Dotless setting, with an accuracy of only 31.5%.

6.3 Analysis of Best System Errors

The best-performing system (*Dotted* ⇒ *CharMapper* ⇒ GPT-4o) produces 3,896 errors (9.6%). We examined the first 100 errors, occurring within the first 2,690 words. We grouped errors into six categories, summarized in Table 9 (Appendix D).

Valid Paraphrase (36%) covers cases where the Arabic gold reference is not a direct transliteration of the JA source but a close substitution. For example, the system output قال *qAl* 'he said' is evaluated against the reference فقال *fqAl* 'then he said'. **Unnecessary Change** (21%) refers to cases in which the *CharMapper* output is changed by GPT-4o but the change was not necessary. **Alif-Hamza** (18%) refers to errors caused by a wrong transliteration and placing of the Hamza (e.g., الآب *AlĀb* instead of الأب *AlÂb* 'the father'). **Wrong Word** denotes cases where the system output diverges entirely from the gold reference. For example, the system produces وإنجاء *wĂnjA'* instead of the reference وتسليم *wtslym* 'salvation/deliverance'. **Source Error** (7%) refers to spelling mistakes in the JA source itself. For example, the source אנה انه *Anh* should have been אנא أنا *ÂnA* 'I'. Errors that are not covered by the previous cases were categorized as **Other** (6%).

6.4 Analysis of Hebrew Input

Although Hebrew input words (2% of the dataset) were excluded from transliteration evaluation, we manually analyzed a random sample of 100 cases to assess how our best-performing system handles them. Appendix E Table 10 presents the categories. In 84% of cases, the system did not match the reference: 46% were due to Hebrew words being translated into Arabic in the reference, 17% involved substitution with a different word, and 21% fell into other categories such as deletions or insertions. Notably, in 16% of cases the system produced the correct output, which is plausible given the shared cognates between Hebrew and Arabic.

7 Conclusions and Future Work

We presented a two-step approach for transliterating Judeo-Arabic texts from Hebrew to Arabic script, combining character-level mapping with post-correction to resolve orthographic and grammatical inconsistencies. Our results show that post-correction substantially improves transliteration quality and, despite requiring no training on Judeo-Arabic data, our approach outperforms previous work. We also benchmarked LLMs on this task for the first time and demonstrated the downstream benefits of transliteration on Arabic morphosyntactic tagging and Arabic-English machine translation. Our goal is to make Judeo-Arabic texts accessible to Arabic readers and compatible with modern NLP tools, and we will make our code and data publicly available to support future research.

In future work, we aim to address the challenge of code-switching by automatically detecting and processing Hebrew segments embedded in Judeo-Arabic text. Specifically, we plan to develop models that combine transliteration with selective translation, converting non-Arabic segments into Arabic script. Additionally, we intend to investigate alternative modeling strategies to improve transliteration accuracy and robustness across diverse Judeo-Arabic texts. We also plan to assess the utility of transliteration on other downstream Arabic NLP tasks, such as named entity recognition.

Acknowledgement

We acknowledge the support of the High Performance Computing Center at New York University Abu Dhabi. We thank the anonymous reviewers for their insightful and constructive comments.

Ethical Considerations

We use publicly available datasets and language models. We do not anticipate any potential risks associated with this work, as it does not involve the collection of personal data, sensitive content, or human subjects. We used AI writing assistance within the scope of "Assistance purely with the language of the paper" described in the ACL Policy on Publication Ethics.

Limitations

While our work demonstrates promising results, several limitations should be noted that may affect its broader applicability and reproducibility. First, our experiments rely on closed-source commercial LLMs, which are subject to periodic updates that are not publicly documented. This introduces some uncertainty and may affect the reproducibility of our results over time. Second, although our transliteration and post-correction pipeline handles Judeo-Arabic well at the sentence level, it does not explicitly address Hebrew segments. These code-switched elements are common in Judeo-Arabic texts and present unique challenges, particularly when they are morphologically integrated into the surrounding Arabic. Finally, our downstream evaluations rely on silver annotations: morphological tags from pretrained Arabic taggers and English translations generated by GPT-4o. While informative, these approximations are not substitutes for human-validated gold data and may introduce evaluation noise.

References

Mohamed Al-Badrashiny, Ramy Eskander, Nizar Habash, and Owen Rambow. 2014. Automatic transliteration of Romanized dialectal Arabic. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 30–38, Ann Arbor, Michigan. Association for Computational Linguistics.

Bashar Alhafni and Nizar Habash. 2025. Enhancing text editing for grammatical error correction: Arabic as a case study. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17892–17914, Vienna, Austria. 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. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Mohammed Attia, Pavel Pecina, Younes Samih, Khaled Shaalan, and Josef van Genabith. 2012. Improved spelling error detection and correction for Arabic. In *Proceedings of COLING 2012: Posters*, pages 103–112, Mumbai, India. The COLING 2012 Organizing Committee.

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. 1961. *A Grammar of Mediaeval Judaeo-Arabic*. Magnes Press, Jerusalem. [In Hebrew].

Joshua Blau. 1965. *The Emergence and Linguistic Background of Judaeo-Arabic: A Study in the Origins of Middle Arabic*. Oxford University Press.

Rawan Bondok, Mayar Nassar, Salam Khalifa, Kurt Micallef, and Nizar Habash. 2025. Proper noun diacritization for Arabic Wikipedia: A benchmark dataset. In *Proceedings of the 2nd Workshop on Advancing Natural Language Processing for Wikipedia (WikiNLP 2025)*, pages 31–44, Vienna, Austria. Association for Computational Linguistics.

Achraf Chalabi and Hany Gerges. 2012. Romanized Arabic transliteration. In *Proceedings of the Second Workshop on Advances in Text Input Methods*, pages 89–96, Mumbai, India. The COLING 2012 Organizing Committee.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Kareem Darwish. 2014. Arabizi detection and conversion to Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 217–224, Doha, Qatar. Association for Computational Linguistics.

Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali. 2017. Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the Third Arabic Natural*

Language Processing Workshop, pages 9–17, Valencia, Spain. Association for Computational Linguistics.

Salman Elgamal, Ossama Obeid, Mhd Kabbani, Go Inoue, and Nizar Habash. 2024. Arabic diacritics in the wild: Exploiting opportunities for improved diacritization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14815–14829, Bangkok, Thailand. Association for Computational Linguistics.

Ramy Eskander, Mohamed Al-Badrashiny, Nizar Habash, and Owen Rambow. 2014. Foreign words and the automatic processing of Arabic social media text written in Roman script. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 1–12, Doha, Qatar. Association for Computational Linguistics.

Friedberg Geniza Project. 1999. Friedberg Geniza Project.

Imane Guellil, Faiçal Azouaou, Mourad Abbas, and Sadat Fatiha. 2017. Arabizi transliteration of Algerian Arabic dialect into Modern Standard Arabic. In *Social MT 2017/ First workshop on Social Media and User Generated Content Machine Translation*, Prague, Czech Republic.

Wiktor Gębski. 2024. *A Grammar of the Jewish Arabic Dialect of Gabes*. Open Book Publishers, Cambridge, UK.

Judah Ha-Levi. 2012. *The Kuzari: The Book of Refutation and Proof on the Despised Faith*. Al-Kamel Verlag, Freiberg. Nabih Bashir, Transliterated and edited with the assistance of 'Abed al-Salam Muosa.

Nizar Habash, Salam Khalifa, Fadhl Eryani, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified Guidelines and Resources for Arabic Dialect Orthography. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer, Netherlands.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Go Inoue, Bashar Alhafni, Nizar Habash, and Timothy Baldwin. 2026. Do diacritics matter? evaluating the impact of arabic diacritics on tokenization and llm benchmarks. In *Findings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL 2026)*, Rabat, Morocco.

Go Inoue, Salam Khalifa, and Nizar Habash. 2022. Morphosyntactic tagging with pre-trained language models for Arabic and its dialects. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1708–1719, Dublin, Ireland. Association for Computational Linguistics.

Moussa Kamal Eddine, Nadi Tomeh, Nizar Habash, Joseph Le Roux, and Michalis Vazirgiannis. 2022. AraBART: a pretrained Arabic sequence-to-sequence model for abstractive summarization. In *Proceedings of the Seventh Arabic Natural Language Processing Workshop (WANLP)*, pages 31–42, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Salam Khalifa, Ossama Obeid, and Nizar Habash. 2021. Character edit distance based word alignment. `https://github.com/CAMeL-Lab/ced_word_alignment`.

Geoffrey Khan. 1997. The Arabic dialect of the Karaite Jews of Hit. *Zeitschrift für Arabische Linguistik*, 34:53–102.

Valentina Bella Lanza. 2020. Judeo-Arabic orthographies: Insights from a fifteenth-century šarḥ. *Eurasian Studies*, 18(1):134–148.

Jacob Mansour. 1991. *The Jewish Baghdadi Dialect: Studies and Texts in the Judeo-Arabic Dialect of Baghdad*. The Babylonian Jewry Heritage Center & The Institute for Research on Iraqi Jewry.

Kurt Micallef, Fadhl Eryani, Nizar Habash, Houda Bouamor, and Claudia Borg. 2023. Exploring the impact of transliteration on NLP performance: Treating Maltese as an Arabic dialect. In *Proceedings of the Workshop on Computation and Written Language (CAWL 2023)*, pages 22–32, Toronto, Canada. 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.

Hamdy Mubarak, Mohamed Al Sharqawy, and Esraa Al Masry. 2009. Diacritization and transliteration of proper nouns from arabic to english. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt. The MEDAR Consortium.

El Moatez Billah Nagoudi, AbdelRahim Elmadany, and Muhammad Abdul-Mageed. 2022. AraT5: Text-to-text transformers for Arabic language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 628–647, Dublin, Ireland. Association for Computational Linguistics.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Ale-

man, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Marina Rustow and Rebecca Sutton Koeser. 2022. Princeton Geniza Project. Center for Digital Humanities at Princeton.

Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, William Marshall, Gurpreet Gosal, Cynthia Liu, Zhiming Chen, Osama Mohammed Afzal, Samta Kamboj, Onkar Pandit, Rahul Pal, Lalit Pradhan, Zain Muhammad Mujahid, Massa Baali, Xudong Han, Sondos Mahmoud Bsharat, and 13 others. 2023. Jais and jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models. *Preprint*, arXiv:2308.16149.

Ali Shazal, Aiza Usman, and Nizar Habash. 2020. A unified model for Arabizi detection and transliteration using sequence-to-sequence models. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 167–177, Barcelona, Spain (Online). Association for Computational Linguistics.

Norman A. Stillman. 2010. Judeo-Arabic - history and linguistic description. In *Encyclopedia of Jews in the Islamic World Online*. Brill.

Fanar Team, Ummar Abbas, Mohammad Shahmeer Ahmad, Firoj Alam, Enes Altinisik, Ehsannedin Asgari, Yazan Boshmaf, Sabri Boughorbel, Sanjay Chawla, Shammur Chowdhury, Fahim Dalvi, Kareem Darwish, Nadir Durrani, Mohamed Elfeky, Ahmed Elmagarmid, Mohamed Eltabakh, Masoomali Fatehkia, Anastasios Fragkopoulos, Maram Hasanain, and 23 others. 2025. Fanar: An arabic-centric multimodal generative ai platform. *Preprint*, arXiv:2501.13944.

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.

Clare Voss, Stephen Tratz, Jamal Laoudi, and Douglas Briesch. 2014. Finding Romanized Arabic dialect in code-mixed tweets. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2249–2253, Reykjavik, Iceland. European Language Resources Association (ELRA).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jihene Younes, Emna Souissi, Hadhemi Achour, and Ahmed Ferchichi. 2018. A sequence-to-sequence based approach for the double transliteration of Tunisian dialect. *Procedia Computer Science*, 142:238–245. Arabic Computational Linguistics.

A Model Details

Pretrained GEC Systems The Seq2Seq and Seq2Seq+MG models introduced by Alhafni et al. (2023) are built on AraBART (Kamal Eddine et al., 2022) and publicly available at `https://github.com/CAMeL-Lab/arabic-gec` . The Seq2Seq+MG variant incorporates morphological features and grammatical error detection signals obtained from external models, as described in Alhafni et al. (2023). Seq2Seq and Seq2Seq+MG consists of 139M and 502M parameters, respectively. The text editing model SWEET (Alhafni and Habash, 2025), built on AraBERTv02 (Antoun et al., 2020), has 135M parameters and is available at `https://github.com/CAMeL-Lab/text-editing`. For all pretrained GEC models, we use their respective default hyperparameters. Inference was done on a single A100 GPU.

LLMs Jais-13B-Chat has 13B parameters, and Fanar has 8.7B. Architectural details for GPT-3.5-turbo and GPT-4o are unavailable, as they are not publicly released. GPT-3.5-turbo, GPT-4o, and Fanar were prompted using default hyperparameters, while Jais was run on an A100 GPU with temperature 0.3, nucleus sampling 0.9, max length 2048, and repetition penalty 1.2.

B Alignment Example

JA	Transliteration	Reference	Label	Gloss
פיציר	فيصير *fySyr*	فيصير *fySyr*	JA	'so he is'
ענדנא	عند نا ς*ndnA*	عند نا ς*ndnA*	JA	'to us'
מרה	مره *mrh*	مرة *mrẖ*	JA	'sometimes'
אֵל	ال *Al*	إله *Ălh*	Heb	'God'
רַחוּם	رحوم *rHwm*	رحيم *rHym*	Heb	'compassionate'
וְחַנּוּן	وحنون *wHnwn*	ورءوف *wr'wf*	Heb	'and graceful'
ומרה	ومره *wmrh*	ومرة *wmrẖ*	JA	'other times'
אֵל	ال *Al*	إله *Ălh*	Heb	'God'
קַנּא	قنا *qnA*	UNK	Heb	'jealous'
וְנוֹקֵם	ونوقم *wnwqm*	ومنتقم *wmntqm*	Heb	'and vengeful'
,	,	UNK	Punc	

Table 7: Alignment example. Green indicates successful automatic transliteration, red marks transliteration errors, and yellow highlights items excluded from evaluation (Hebrew and punctuation).

C Prompts

Task	Prompt
Transliteration	`You are a transliteration system that can transliterate Judeo-Arabic text to Arabic. Please transliterate the following Judeo-Arabic sentence to Arabic without providing any explanation. The output should be in Arabic script.`
GEC	`Please identify and correct any grammatical and spelling errors in the following sentence marked with the tag <input> SRC </input>. Make the minimal changes necessary to correct the sentence. Do not rephrase any parts of the sentence that are already grammatically correct, and avoid altering the meaning by adding or removing information. After making the corrections, output the revised sentence directly without providing any explanations. Remember to format the corrected output with the tag <output> Your Corrected Version </output>.`
Machine Translation	`You are a machine translation system that can translate Arabic to English. Please translate the following Arabic sentence to English without providing any explanation.`

Table 8: Prompts used for each task in our experiments: transliteration, grammatical error correction (GEC), and machine translation.

D Error Analysis

Error Type	%	JA	CharMapper	CharMapper ⇒ GPT-4o	Reference	Gloss
Valid Paraphrase	38	קאל	قال *qAl*	قال *qAl*	فقال *fqAl*	‘said’
Unnecessary Change	21	כלקה	خلقة *xlqħ*	خلق *xlq*	خلقة *xlqħ*	‘creation’
Alif-Hamza Error	18	אלאב	الاب *AlAb*	الآب *AlĀb*	الأب *AlÂb*	‘the father’
Wrong Word	10	ותסלים	وتسليم *wtslym*	وإنجاء *wĂnjA’*	وتسليم wtslym	‘salvation’
Source Error	7	אנה	انه *Anh*	إنه *Ănh*	أنا *ÂnA*	‘I’
Other	6	מאיה	مايه *mAyh*	خمسمائة *xmsmAŷħ*	ماية *mAyħ*	‘hundred’

Table 9: Error categories and examples.

E Analysis of Hebrew Input

Category	%	JA	Hebrew Gloss	CharMapper	CharMapper ⇒ GPT-4o	Arabic Gloss	Reference	Arabic Gloss
False Cognate	46	וַיֹּאמֶר	‘and he said’	و يامر *wyAmr*	و يأمر *wyÂmr*	‘and he orders’	وقال *wqAl*	‘and he said’
Transliteration	17	מִשְׁפָּחוֹת	‘families’	مشفحوت *mšfHwt*	مشفحوته *mšfHwth*	[not Arabic]	فقط *fqT*	‘only’
No Reference	21	אֲשֶׁר	‘which’	اشر *Ašr*	الذي *Alθy*	‘which’	ϕ	ϕ
Correct	16	הוא	‘he’	هوا *hwA*	هو *hw*	‘he’	هو *hw*	‘he’

Table 10: Analysis of different Hebrew input processing results.