LLM-driven Text Normalization for Improved Meta-Learning in Arabic Sentiment Analysis

SHADI BALLOUL, YASSER RAHAL, KADAN ALJOUMAA
Department of Computer Science,
Higher Institute for Applied Science and Technology,
Damascus,
SYRIA

Abstract: - Arabic texts posted online are written in different dialects and styles. This makes the Arabic content very sparse, especially when the texts come from different domains such as reviews of hotels, books and others. These characteristics cause low accuracy when a sentiment classifier is trained on one domain and used on another. Few-shot classification, as implemented in meta-learning, is a promising solution. The problem with these algorithms is that they require a minimum level of similarity between domains, and to our knowledge no Arabic multi-domain dataset meets this condition. We propose a new approach that uses an LLM (Gemini 2.5 Pro) to apply normalization, preprocessing and conversion to MSA to existing Arabic datasets, which reduces the sparsity and variety of Arabic multi-domain data. We measured this reduction directly, as a drop in the average symmetric KL divergence between domains, and we tested the approach on three datasets that we composed: classification accuracy in the few-shot setting rose by up to 7 points for MAML and 5 points for Prototypical Networks. As the LLM processing is applied only once, before the training phase, its cost is low, and the approach can be applied with limited resources and budget.

Key-Words: - Arabic NLP, Dialect Normalization, Large Language Models (LLMs), Meta-Learning, Sentiment Analysis.

Received: March 17, 2026. Revised: June 14, 2026. Accepted: July 16, 2026. Available online: November 11, 2026.

1 Introduction

Writing in Arabic faces a problem that most other languages do not have: the same meaning can be written with different words depending on the region, and the main regions are Egypt, the Levant, North Africa and the Gulf. For example, "I want" is written بدي (*biddi*) in the Levant, عايز (*'ayez*) in Egypt and أبي (*abi*) in the Gulf. This problem affects sentiment analysis on multi-domain datasets, and the approaches most affected by it are the meta-learning algorithms, because these algorithms try to find the features shared between tasks or domains. It is very difficult for these algorithms to reach high performance when the same meaning is expressed with various words.

Large language models look like the natural tool here, and so far the field has used them in two main ways: as zero- or few-shot classifiers, which gets expensive when every prediction touches a large model, and as automatic annotators, which works until the labeling task becomes subtle. A third option has drawn much less attention — run the LLM once, offline, to improve data that humans have already labeled, and then never call it again during training.

That third option is what this paper develops. We convert a collection of multi-dialect Arabic datasets into Modern Standard Arabic (MSA) with an LLM — a procedure we call **LLM-based dialect normalization** — and ask whether the resulting, more homogeneous corpus makes a better training environment for meta-learning. Our working hypothesis is that shrinking inter-domain variance at the data level should pay off in both the performance and the stability of the learning algorithms, and the experiments bear this out.

The paper contributes:

1. a methodology for building unified multi-domain Arabic datasets, with LLM-based dialect normalization as its central step;
2. an empirical evaluation showing that normalization raises the few-shot accuracy of both MAML and Prototypical Networks on all three of our benchmarks;
3. a practical division of labor in which the LLM does the linguistics once, offline, while small and fast models (AraBERT with a compact classifier head) handle the actual classification.

2 Related Work

Two lines of work meet in this paper: meta-learning for few-shot adaptation, and the use of large language models in sentiment analysis. We review each briefly, with an eye on the gap between them that our method occupies.

2.1 Few-Shot Learning and Meta-Learning

Few-shot learning asks a model to generalize from a handful of labeled examples [1]. For Arabic sentiment analysis this is not an academic constraint — for most domains, large annotated corpora simply do not exist. The dominant answer is meta-learning, or "learning to learn" [2]: instead of fitting one task, the model is trained across many small tasks so that what it acquires is an adaptable learning strategy rather than a fixed decision rule.

Two families dominate:

- **Metric-Based Meta-Learning:** Learn an embedding in which same-class examples sit close together, then classify by distance. The canonical instance is **Prototypical Networks** [3], which represents each class by the mean of its support embeddings and assigns a query to the nearest such prototype.
- **Optimization-Based Meta-Learning:** Learn an initialization from which a few gradient steps suffice on any new task. The standard algorithm is **Model-Agnostic Meta-Learning (MAML)** [4], whose nested optimization tunes the starting weights so that adaptation is fast across a whole range of tasks.

These techniques were, however, developed largely on computer vision benchmarks, and language brings problems that images do not [5]. In NLP a "task" is usually a text domain — product reviews, tweets, news — and text domains differ in vocabulary and style far more than two image classification tasks differ from each other. Meta-learners are sensitive to this: when the training domains diverge too much, the meta-training process itself becomes unstable and the learned initialization generalizes poorly [6].

There are algorithmic remedies — grouping tasks by similarity, learning subspace structure [7] — but they add model complexity and compute, which is an awkward trade in a low-resource setting. And for Arabic there is a further, more basic obstacle: standardized multi-domain benchmarks barely exist, so every study builds its own ad-hoc collection and domain-shift effects are hard to compare across papers. Both observations point the same way: it would be useful to reduce inter-domain variance in the data itself, before any algorithm sees it. That is what our normalization does.

2.2 The Role of Large Language Models in Sentiment Analysis

In sentiment analysis, LLMs have so far been used mainly in one of two roles: as classifiers, or as annotators. Each role trades something away.

As classifiers, LLMs used through in-context learning reach zero- and few-shot accuracy that rivals fine-tuned models in many settings [8], [9]; the cost is that every prediction requires a call to a large model, which is hard to justify at corpus scale [10]. As annotators, they promise a way out of data scarcity [11]; the cost is reliability, since automated labels degrade noticeably on complex or fine-grained tasks [12].

A third role sits between these two and has been left mostly unexplored: improving data that humans have already labeled. We use the LLM neither to classify nor to label, but to normalize — to rewrite dialectal text into a standard form while the human-assigned label stays in place. For Arabic this targets the root of the problem, since dialectal variation is what makes the data sparse and the training domains divergent in the first place. The economics also work out: normalization runs once, offline, and at test time only the few support examples of a new task ever need it.

3 Methodology

3.1 Dataset Preparation

We built our benchmarks in stages. The raw material came from a survey of published Arabic sentiment analysis research, from which we collected the publicly available datasets summarized in Table 1 (Appendix). Their annotation schemes disagree — some use binary labels, others a five-point star scale — so we mapped everything to binary: ratings of 1 or 2 became `negative`, 4 or 5 became `positive`.

The raw texts then went through a light cleaning pass (URLs, hashtags and repeated characters removed), after which we capped every domain at 3,000 texts and kept the smaller domains whole. The result is an unbalanced collection of eleven domains that we call **Set1**. For a more controlled setting we then dropped the three small and heavily skewed domains (`ATT`, `MOV`, and `PROD`) and merged the `RES1` and `RES2` domains. This produced **Set2**: seven

domains of 3,000 texts each, balanced across classes as well.

The third collection has a different purpose. We wanted domains as linguistically close to one another as we could make them, so that the effect of inter-domain similarity could be studied under some control; that is **Set3**. We started from **Set2** and carved finer-grained domains out of three of its sources, using whatever metadata they happened to carry:

- **From the ARASAS dataset**, which is annotated for speech acts, we created two new domains: `ARASAS_Expr`, consisting of positive and negative samples of the 'Expression' type, and `ARASAS_ASSERT`, containing negative and neutral samples of the 'Assertion' type.
- **From the TEAD dataset** of tweets, we selected 200,000 samples and performed K-Means clustering (K=8). We then selected the three most distinct clusters to form three new domains: `TEAD1`, `TEAD2`, and `TEAD3`.
- **From the HARD dataset** of hotel reviews, we utilized user metadata to create three domains. Two domains, `HARD_FAMILY1` and `HARD_FAMILY2`, were built from users identified as 'Family', with the two sets of hotels kept disjoint. A third domain, `HARD_SINGLE`, came from 'Single' users.

The point of this finer partition is a set of domains that are related by construction, so that whatever divergence remains between them is small and, we hoped, mostly linguistic.

3.2 LLM-based Dialect Normalization

Dialect is the obstacle we set out to remove. When the same meaning is written in five regional ways, each spelling becomes a separate and rarer event for the model, and domains that happen to favor different regions drift apart for reasons that have nothing to do with sentiment. Meta-learning, which leans on task consistency, suffers most. Our remedy is to push all of the text into one register, Modern Standard Arabic (MSA), with an LLM, before any of it reaches a classifier.

We used Google's Gemini 2.5 Pro. Its main practical advantage for us is that it handles the major Arabic dialects without being told which one it is looking at, which matters because a single tweet file mixes Gulf, Levantine, Egyptian and Maghrebi writers. The model is driven by a long system prompt that casts it as an Arabic linguist and states one overriding rule: the sentiment of the text must not change. Everything else in the prompt is a list of cleaning and standardization operations:

- **Dialect-to-MSA Conversion:** Conversion of all dialect-specific words and phrases into their most common MSA equivalents, e.g., Syrian بدي to MSA أريد (both mean "I want").
- **Text Cleaning:** Correction of common typographical errors and removal of non-semantic, expressive letter repetitions, e.g., جمييييل to جميل ("beautiful").
- **Emoji and Foreign Language Processing:** Replacement of emojis with their textual descriptions in MSA, e.g., □ to أعجبني ("I like it"), together with translation of any embedded English words into Arabic.
- **Negation Handling:** A strict rule protects negation patterns (لا، ما، ليس، لم، لن، غير، مش، مو): the negator is never deleted unless the negated expression is replaced, as a whole, by a confirmed antonym of the same polarity, e.g., لا أنصح به ("I do not recommend it") may become أحذر منه ("I warn against it"), both negative. If no certain antonym exists, the negator is left untouched.
- **Removal of Artifacts:** Deletion of all numbers, URLs, email addresses, and other non-textual content.
- **Hashtag Segmentation:** Parsing of hashtags into their constituent words, e.g., #الخدمة_ممتازة to "الخدمة ممتازة", which were then subjected to the same normalization rules.

The datasets went in and came out as tab-separated files, the human label carried along unchanged next to each text. What comes out is a normalized twin of every dataset: same labels, fewer distinct word forms, and, as the next sections measure, a set of domains that sit much closer to one another.

3.3 Worked Examples and the Lexical Effect of Normalization

The transformations listed above are easiest to judge on real corpus material. Table 2 (Appendix) traces four texts drawn from the three source domains of **Set3** — restaurant reviews, hotel reviews, and tweets — through the pipeline. Each row shows the original text as it appears in the corpus, the output of Gemini 2.5 Pro, and an English gloss with the individual edits identified.

Two properties of these edits matter for the divergence analysis that follows. First, normalization is a many-to-one map on the lexicon. The dialectal variants بدي, عايز, and أبي all collapse to the single MSA form أريد; شفتها and its regional

siblings collapse to رأيتها; expressive spellings and missing hamzas collapse to one canonical orthography. Every such collapse removes word types from the joint vocabulary and moves their probability mass onto forms that all domains share, which raises cross-domain token overlap. This is the microscopic mechanism behind the drop in average symmetric KL divergence reported in Figure 1. Second, the edits are conservative exactly where sentiment is concerned. Across the four examples, 12 of 19 surface tokens are rewritten, yet the rewrites are confined to dialectal function words and auxiliaries, orthography, and non-textual artifacts; the only edit that touches an opinion-bearing expression (row 3) is governed by the negation rule and preserves polarity by construction. The normalized corpus is therefore lexically denser than the original but carries the same label information, which is precisely the combination that the meta-learning analysis of the next sections requires.

The same mechanism can be quantified at corpus level. Table 3 (Appendix) reports an alignment-free comparison of the original and normalized corpora for all ten Set3 domains, computed with a deliberately conservative tokenizer: URLs are removed, diacritics stripped, every punctuation mark treated as a delimiter, and pure numerals excluded, so that punctuation spacing, number deletion, and diacritic changes cannot contribute to the figures. Because the normalizer returned valid output for 73–98% of the inputs (unparseable outputs were discarded), vocabulary sizes are compared within the same number of running tokens N for both versions (texts shuffled with a fixed seed, averaged over five orderings), which removes the corpus-size confound. Under these conditions the vocabulary shrinks by 16.5% (HTL) to 29.4% (HARD_SINGLE): most in the dialect-heavy hotel reviews, least in the longer and more formal HTL reviews, with the tweet clusters in between (16.7–20.7%); the type–token ratio falls accordingly. Non-lexical artifacts all but vanish: emojis fall from as many as 317 per 10,000 tokens to at most 1.4, elongated spellings from 73 to at most 3.1, Latin-script tokens from 227 to at most 5.7, and URLs to at most 0.05. (The TEAD clusters reach us already stripped of URLs, emojis and repeated characters, hence their zero artifact rates before normalization.) Most important for the argument of the following sections, the domains move closer together. Averaged over the 45 domain pairs, the Jaccard similarity of domain vocabularies rises from 0.156 to 0.201, that of the 1,000 most frequent types from 0.250 to 0.283, and the share of a domain's running tokens whose type also occurs in the other domain from 0.670 to 0.732. The gain is largest where the domains were furthest apart: among the three TEAD clusters, which were chosen to be maximally distinct, the shared-token share rises from 0.61–0.66 to 0.71–0.77, and between the ARASAS tweets and the HARD hotel reviews from 0.56–0.59 to 0.61–0.63. This is the lexical counterpart of the divergence reduction reported in Figure 1.

3.4 Measuring Inter-Domain Similarity

To put a number on how far apart the domains are, before and after normalization, we use the Kullback-Leibler (KL) divergence. Each domain is treated as a probability distribution over a learned representation space; the smaller the divergence between two domains, the more alike they are.

Formally, KL divergence measures how far one distribution [13], P, strays from a reference distribution Q, or equivalently how much information is lost when Q stands in for P. It is not symmetric: in general $D_{KL}(P \| Q) \neq D_{KL}(Q \| P)$. The directional form is

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \quad (1)$$

We have no reason to privilege one direction, so we symmetrize by averaging the two directional divergences between a pair of domains D_1 and D_2:

$$D_{Sym}(D_1, D_2) = \frac{1}{2}(D_{KL}(D_1 \parallel D_2) + D_{KL}(D_2 \parallel D_1)) \quad (2)$$

One number per domain is more convenient than a matrix. For a target domain D_t and a collection of others $S_n = \{D_1, D_2, \ldots, D_n\}$, we therefore average the symmetric divergence between the target and every other member of the collection:

$$D_{Avg}(D_t, S_n) = \frac{1}{n-1} \sum_{D_i \in S_n, i \neq t} D_{Sym}(D_t, D_i) \quad (3)$$

To compute these divergences in practice, each domain must first be given a density. Every text in a domain is mapped to the 768-dimensional embedding of its `[CLS]` token under the pre-trained AraBERT model, and the domain's density over this representation space is estimated by Kernel Density Estimation [14] with a Gaussian kernel. We deliberately use a single fixed bandwidth, $h = 0.25$, shared by all domains and by both the baseline and normalized conditions, rather than a per-domain data-driven rule such as Silverman's. The reason is comparability: the bandwidth sets the smoothing scale of the estimator, and if it were re-selected per

domain, an observed drop in divergence after normalization could reflect a change in the estimator rather than a change in the data. With h held fixed, the before/after comparison isolates the effect of normalization itself. The KL integral is approximated by a plug-in scheme in the spirit of [13]: both fitted densities are evaluated at a common set of 1,800 points spanning the joint support of the two embedding samples, and the divergence is computed from the normalized density values, with a small constant (10^{-9}) added for numerical stability. We verified that repeating the analysis at $h = 0.2$ leaves the ordering of domains and all qualitative conclusions unchanged. We note that KDE is a consistent estimator as the sample size grows and the bandwidth shrinks at a suitable rate [14]; here it serves as a comparative instrument, and only relative divergences between conditions are interpreted.

3.5 Theoretical Motivation: Domain Divergence and Few-Shot Generalization

The KL measurements of the previous section are not merely descriptive. In this section we make precise the claim that reducing inter-domain divergence should improve few-shot generalization, by casting dialect normalization as a distribution alignment problem and connecting our D_{Avg} score to a known generalization bound.

Let □ denote the representation space (in our case, the space of AraBERT embeddings) and let each domain D_i induce a distribution P_i over □. The normalization procedure of Section 3.2 defines an operator T that maps a raw dialectal text to its MSA form. Because T is constrained by the prompt to preserve the sentiment label, it acts only on the marginal distributions: each P_i is pushed forward to $\tilde{P}_i = T_{\#}P_i$, while the conditional label distribution is left (approximately) unchanged. Our hypothesis is that the family $\{\tilde{P}_1, \ldots, \tilde{P}_n\}$ is more concentrated than $\{P_1, \ldots, P_n\}$, which is exactly what Figure 1 verifies empirically.

Why should concentration help? For a hypothesis h and domain D_i, write $\varepsilon_i(h)$ for the expected classification error of h on D_i. A classical result of [15] states that for any two domains,

$$\varepsilon_j(h) \le \varepsilon_i(h) + d_1(P_i, P_j) + \lambda_{ij} \qquad (4)$$

where d_1 is the total variation distance and λ_{ij} is the error of the best single hypothesis on both domains jointly. The middle term is the price of domain shift; the last term is irreducible and, crucially, is not increased by a label-preserving transformation such as ours. Pinsker's inequality [16] bounds total variation by KL divergence, $d_1(P, Q) \le [2D_{KL}(P \| Q)]^{1/2}$, and since $\min\{D_{KL}(P \| Q), D_{KL}(Q \| P)\} \le D_{Sym}(P, Q)$, we obtain the following statement in terms of the symmetric divergence of Equation (2).

Proposition 1. *For any hypothesis h and any pair of domains D_i, D_j,*

$$\varepsilon_j(h) \le \varepsilon_i(h) + \sqrt{2D_{Sym}(D_i, D_j)} + \lambda_{ij} \qquad (5)$$

Averaging over the $n-1$ source domains of a collection S_n and applying Jensen's inequality to the concave square root, the error on a held-out target domain D_t satisfies

$$\varepsilon_t(h) \le \frac{1}{n-1}\sum_{i \ne t} \varepsilon_i(h) + \sqrt{2D_{Avg}(D_t, S_n)} + \bar{\lambda} \qquad (6)$$

where $\bar{\lambda}$ is the mean of the λ_{it} terms, and the first term on the right is the average training-domain error.

The bound in Equation (6) is monotone in precisely the quantity D_{Avg} that we measure in our experiments. It therefore predicts that any label-preserving transformation which lowers D_{Avg} — as our normalization does for every domain in Figure 1 — tightens the worst-case gap between training-domain error and unseen-domain error. We do not claim the bound is numerically tight for neural hypotheses; its role here is the same as in the domain adaptation literature: it identifies which measurable quantity governs transfer, and justifies attacking that quantity directly at the data level.

The same conclusion follows from the meta-learning literature specifically. In the environment-level PAC-Bayes analysis of [17], the generalization error of a meta-learner decomposes into a within-task term and an environment-level term that grows with the dispersion of the task distribution; a more concentrated task environment yields a strictly smaller environment complexity term. [6] prove, for MAML in particular, that the benefit of the learned initialization degrades as tasks become more dissimilar, which is why their remedy is to cluster similar tasks before meta-training. Our approach reaches the same destination by the opposite route: instead of adapting the algorithm to a dispersed task environment, we contract the environment itself. For Prototypical Networks the mechanism is concrete: the class prototype c_k of Equation (7) is a mean of $|S_k|$ embeddings drawn, at test time, from a domain the model never saw. When the domains are aligned, the class-conditional distributions agree across domains, so a prototype estimated from a handful of support points is a lower-variance

estimate of the true class centroid, and the softmax over distances in Equation (8) is correspondingly more reliable.

3.6 Meta-Learning Algorithms

We evaluated our approach using two meta-learning algorithms: Model-Agnostic Meta-Learning (MAML), an optimization-based method, and Prototypical Networks, a metric-based method.

3.6.1 Model-Agnostic Meta-Learning (MAML)

The Model-Agnostic Meta-Learning (MAML) algorithm [4] does not learn a classifier so much as a starting point: parameters chosen so that a few gradient steps on a handful of examples from a new task already yield a usable model, which is only possible if training has seen many related tasks rather than one. Two nested loops do the work, as shown in Algorithm 1.

In the inner loop the model adapts to one task, taking one or more gradient steps on a few examples (the *support set*); the result is a task-specific copy of the parameters.

The outer loop then judges each adapted copy on a separate batch (the *query set*) and moves the shared initialization in whatever direction would have made those adaptations work better. Over many episodes the initialization drifts toward a point from which every task is cheap to reach.

Algorithm 1 MAML (Model-Agnostic Meta-Learning)

Require: $p(T)$: distribution over tasks (domains); α: inner-loop learning rate; β: meta learning rate; Episodes_number: number of outer-loop iterations
1: Randomly initialize model parameters θ
2: **for** j = 1 to Episodes_number **do** ▷ Outer loop
  3: Sample a batch of tasks $\{T_i\} \sim p(T)$
  4: **for** each task T_i in the batch **do** ▷ Inner loop
    5: Evaluate $\nabla_\theta L_{Ti}(f_\theta)$ using the support set of T_i
    6: Compute the adapted parameters via gradient descent: $\theta'_i \leftarrow \theta - \alpha \nabla_\theta L_{Ti}(f_\theta)$
  7: **end for**
  8: Update the meta-parameters θ using the loss on the query sets: $\theta \leftarrow \theta - \beta \nabla_\theta \Sigma_{Ti} L_{Ti}(f_{\theta'i})$
9: **end for**

3.6.2 Prototypical Networks

In contrast to the optimization-based approach of MAML, Prototypical Networks [3] take the metric route: learn an embedding in which the examples of a class gather around one representative point, its prototype, and let distance do the classifying.

Two steps. First, each class in the support set gets a prototype: its support examples are passed through the embedding network (with parameters φ) and their embeddings averaged. Equation (7) defines this process, where S_k is the set of support examples for class k.

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\varphi(x_i) \tag{7}$$

Second, a query example x is embedded with the same network f_φ, and class probabilities come from a softmax over the negative distances to the prototypes (Euclidean distance in our case); the nearest prototype wins, as in Equation (8).

$$p(y = k|x) = \frac{exp(-d(f_\varphi(x), c_k))}{\sum_{k'} exp(-d(f_\varphi(x), c_{k'}))} \tag{8}$$

Training is episodic, as for MAML: in each episode the embedding parameters φ are updated to lower the cross-entropy on the query examples, which pulls same-class embeddings together and pushes classes apart. One episode is written out in Algorithm 2.

Algorithm 2 Prototypical Networks Training Episode

Require: D: the training set; N_C: classes per episode; N_S: support examples per class; N_Q: query examples per class
1: Initialize loss $J \leftarrow 0$
2: Select a subset of N_C classes randomly from D
3: **for** each class k in the selected subset **do**
  4: Sample a support set S_k of N_S examples from class k
  5: Sample a query set Q_k of N_Q examples from the remainder of class k
6: **end for**
7: **for** each class k in the selected subset **do**
  8: Compute prototype $c_k \leftarrow (1/N_S) \Sigma f_\varphi(x_i)$ over $(x_i, y_i) \in S_k$
9: **end for**
10: **for** each class k in the selected subset **do**
  11: **for** each query example (x, y) in Q_k **do**
    12: Compute loss for the query point using Equation (8)
    13: $J \leftarrow J - \log p(y = k|x)$
  14: **end for**
15: **end for**
16: Update network parameters φ by taking a gradient step on loss J

4 Experiments and Results

4.1 Experimental Setup

Everything below asks one question: does normalizing the dialect away before training make few-shot sentiment classification with MAML and Prototypical Networks better?

Datasets and Task Structure The three collections of Section 3.1 serve as benchmarks: **Set1**, an

imbalanced collection of 11 domains; **Set2**, a balanced version with 7 domains; and **Set3**, a refined and more homogeneous set of 10 domains. Each dataset was split into distinct sets of domains for training, validation, and testing, as detailed in Table 4 (Appendix). A test domain is therefore a data source the model has never seen, which is the situation a few-shot system actually faces. Every collection exists in two forms: the original mixed-dialect text (*Baseline*) and its normalized twin (*Normalized*).

Data Representation and Model Architecture Texts are represented by AraBERTv2 [18]: we take the 768-dimensional embedding of the `[CLS]` token and feed it to a small feed-forward classifier: three ReLU layers, 768 to 512 to 256 to 2 outputs (positive/negative). The classifier is deliberately small; the linguistic work has already been done upstream.

Meta-Learning Framework and Evaluation Both meta-learners are trained in the usual *N*-way *K*-shot episodic fashion. We report classification accuracy on the held-out test domains of Table 4 (Appendix). Alongside accuracy we report the average symmetric KL divergence between training domains for the baseline and normalized versions, so that the effect on the data and the effect on the classifier can be read side by side.

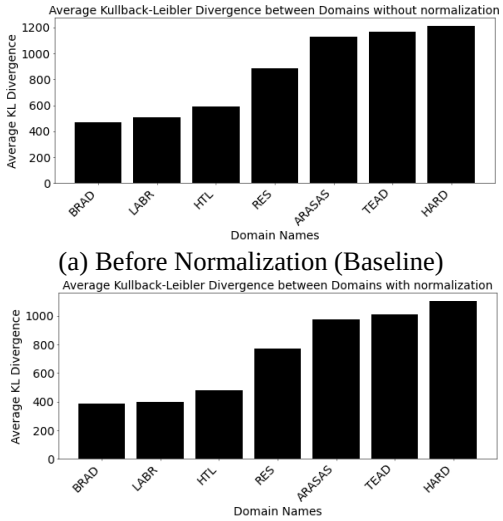
4.2 Results

We report the results in two steps: first what normalization does to the data, measured by KL divergence; then what it does to few-shot accuracy, baseline against normalized in each configuration.

Impact of Normalization on Inter-Domain Similarity Figure 1 shows the average KL divergence of each domain against the rest, before and after normalization. In the baseline (Figure 1a) the spread is wide — roughly 500 for BRAD up to more than 1200 for HARD. After normalization (Figure 1b) every domain sits lower, and the scores bunch together. The normalized collection is, by this measure, a more homogeneous task environment — which is what Section 3.5 says a meta-learner should want.

Normalization helps everywhere. On the imbalanced **Set1** and the more diverse **Set2**, the MSA versions gain between 2 and 5 percentage points over their baselines, in every model and shot setting — so the effect does not depend on having a carefully curated collection to begin with.

The largest gains, though, appear on **Set3**, the most refined collection: up to 7 points for MAML at 5 shots and 5 points for Prototypical Networks. This is the pattern our hypothesis predicts. Curate the domains first, so that what separates them is mostly language rather than topic; then let the LLM remove the linguistic part; what remains is a training environment in which the meta-learner can concentrate on sentiment itself instead of on dialect.



(a) Before Normalization (Baseline)

(b) After Normalization

Fig. 1: Average Symmetric KL Divergence between domains before and after LLM-based normalization. Lower scores indicate greater homogeneity.
Source: created by the authors

Few-Shot Classification Performance Table 5 collects the test accuracies of both algorithms on all three collections, baseline against normalized, with the difference in parentheses.

Table 5. Few-shot classification accuracy (%) on test domains. The table contrasts performance on the original Baseline datasets with the LLM-Normalized versions, with improvements shown in parentheses

Dataset	Alg.	Shot	Base %	Norm % (Δ)
Set1	MAML	1	61.0	64.0 (+3.0)
		5	62.0	67.0 (+5.0)
	ProtoNet	1	59.0	64.0 (+5.0)
		5	62.0	66.0 (+4.0)
Set2	MAML	1	64.0	66.0 (+2.0)
		5	66.0	70.0 (+4.0)
	ProtoNet	1	63.0	67.0 (+4.0)
		5	67.0	69.0 (+2.0)
Set3	MAML	1	71.0	75.0 (+4.0)
		5	73.0	**80.0 (+7.0)**
	ProtoNet	1	72.0	77.0 (+5.0)
		5	76.0	**81.0 (+5.0)**

Source: created by the authors

5 Analysis and Discussion

The experimental results support our main proposal: text normalization and dialect-to-MSA conversion improve meta-learning algorithms for Arabic sentiment analysis in few-shot settings. The drop in inter-domain divergence plays the main role in the accuracy gain, and the results show this in practice.

Normalization Consistently Improves Performance Table 5 shows enhancements in all settings: an increase in the range of 2.0 to 5.0 points on the Set1 and Set2 datasets, and improvements of up to 7 points in the Set3 experiments. These enhancements are correlated with the decrease in divergence shown in Figure 1: LLM normalization made the domains closer, and this improved the accuracy. We interpret this to mean that the baseline models were spending part of their capacity on dialect, for example on learning that بدي (*biddi*) and أريد (*urīd*), both meaning "I want", are the same concept. LLM normalization provides this information for free, leaving the sentiment signal as the primary feature left to learn.

The Synergistic Effect of Curation and Normalization The escalation from Set1 to Set3 is itself informative. Normalization is not a cure-all: a product review and a tweet do not become the same kind of text just because both are now written in MSA. In Set1 and Set2 much of the remaining inter-domain difference is topical, and no amount of linguistic standardization touches it. Set3 was deliberately built from thematically related domains, so there the residual variance really was dialectal — and once normalization removed it, the meta-learners had little left to fight. In hindsight this suggests a two-step recipe for dialect-rich languages: curate topically related domains first, normalize the language second.

Implications for the Meta-Learning Framework The two algorithms benefit for different reasons. MAML meta-optimizes an initialization θ across tasks; when the tasks agree more, their gradients conflict less, and the resulting initialization is a genuinely good starting point for all of them rather than a compromise between factions. Prototypical Networks live or die by the geometry of the embedding space, and unifying dialectal synonyms into single MSA terms tightens exactly the right thing: the class prototypes c_k are computed as means over few examples, and means over a less dispersed distribution are simply better estimates.

A Practical and Efficient Alternative in the LLM Landscape Finally, a word on cost, since that is where this approach differs most from the alternatives. Direct in-context classification pays for the LLM on every prediction [10]; automated annotation pays in label quality [11]. Here the LLM appears exactly once, at preprocessing time, and the model that actually serves predictions is a small fine-tuned stack (AraBERT + FFN). Even in a live few-shot deployment, only the k support examples of a new task would ever pass through the normalizer. The human labels are kept, the classifier stays small, and the LLM's linguistic competence is spent where it matters — on the text, not on the decision.

6 Conclusion

Dialectal divergence makes Arabic texts more dissimilar, and meta-learning algorithms pay the price of this dissimilarity in performance and stability. This research investigated whether the problem can be solved at the data level, by having an LLM rewrite the texts of the available multi-dialect Arabic datasets into Modern Standard Arabic. We found that it is possible.

Using three different compositions of datasets, the proposed normalization improved few-shot classification accuracy for both MAML and Prototypical Networks, and the improvement reached 7 points on the most refined dataset (Set3). The Kullback–Leibler divergence analysis, together with the lexical analysis, ties this improvement to a measured increase in domain similarity. The resulting workflow is simple: an LLM is used to enhance Arabic data that was annotated by humans; the operation is done offline and only once; the classification model stays small, and no extra resources are needed at inference time. Nothing in the proposal is specific to Arabic, so it can be generalized to other languages, and we expect comparable gains in languages that are rich in dialects and poor in resources.

Acknowledgment:
The authors would like to acknowledge the support provided by the Higher Institute for Applied Science and Technology, Damascus, Syria.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the accuracy, integrity, and final content of the publication.

References:

[1] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3), June 2020.

[2] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2022.

[3] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.

[5] Mengde Yang. A survey on few-shot learning in natural language processing. In *2021 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*, pages 294–297, 2021.

[6] Pan Zhou, Yingtian Zou, Xiao-Tong Yuan, Jiashi Feng, Caiming Xiong, and Steven Hoi. Task similarity aware meta learning: theory-inspired improvement on MAML. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 23–33. PMLR, 2021.

[7] Weisen Jiang, James Kwok, and Yu Zhang. Subspace learning for effective meta-learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10177–10194. PMLR, 2022.

[8] Zengzhi Wang, Qiming Xie, Yi Feng, Zixiang Ding, Zinong Yang, and Rui Xia. Is ChatGPT a good sentiment analyzer? A preliminary study. *arXiv preprint arXiv:2304.04339*, 2023.

[9] Yazhou Zhang, Mengyao Wang, Qiuchi Li, Prayag Tiwari, and Jing Qin. Pushing the limit of LLM capacity for text classification. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pages 1524–1528, New York, NY, USA, 2025. Association for Computing Machinery.

[10] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6), April 2024.

[11] Shiqi Liu, Sannyuya Liu, Lele Sha, Zijie Zeng, Dragan Gašević, and Zhi Liu. Annotation guideline-based knowledge augmentation: Toward enhancing large language models for educational text classification. *IEEE Transactions on Learning Technologies*, 18:619–634, 2025.

[12] Zhenyi Lu, Jie Tian, Wei Wei, Xiaoye Qu, Yu Cheng, Wenfeng Xie, and Dangyang Chen. Mitigating boundary ambiguity and inherent bias for text classification in the era of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7841–7864, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[13] Fernando Perez-Cruz. Kullback-Leibler divergence estimation of continuous distributions. In *2008 IEEE International Symposium on Information Theory*, pages 1666–1670, 2008.

[14] Alan Julian Izenman. Review papers: Recent developments in nonparametric density estimation. *Journal of the American Statistical Association*, 86(413):205–224, 1991.

[15] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, 2010.

[16] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2006.

[17] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *Proceedings of the 35th*

International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 205–214. PMLR, 2018.

[18] Wissam Antoun, Fady Baly, and Hazem Hajj. 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, May 2020. European Language Resource Association.

[19] Hady ElSahar and Samhaa R. El-Beltagy. Building large Arabic multi-domain resources for sentiment analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 23–34, Cham, 2015. Springer International Publishing.

[20] Houssem Abdellaoui and Mounir Zrigui. Using tweets and emojis to build TEAD: an Arabic dataset for sentiment analysis. *Computación y Sistemas*, 22(3):777–786, 2018.

[21] AbdelRahim A. Elmadany, Hamdy Mubarak, and Walid Magdy. An Arabic speech-act and sentiment corpus of tweets. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), May 2018. The 3rd Workshop on Open-Source Arabic Corpora and Processing Tools, OSACT3.

[22] Ashraf Elnagar and Omar Einea. BRAD 1.0: Book reviews in Arabic dataset. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8, 2016.

[23] Mohamed Aly and Amir Atiya. LABR: A large scale Arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 494–498, 2013.

[24] Ashraf Elnagar, Yasmin S. Khalifa, and Anas Einea. *Hotel Arabic-Reviews Dataset Construction for Sentiment Analysis Applications*, pages 35–52. Springer International Publishing, Cham, 2018.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

Shadi Balloul organized and executed the experiments, carried out the simulation and the optimization, and drafted the initial manuscript.
Yasser Rahal formulated the methodology, provided supervision, and reviewed the experimental design.
Kadan Aljoumaa contributed to the conceptualization of the research, provided supervision, and reviewed the final manuscript.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflicts of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US

APPENDIX

Table 1. A summary of the Arabic sentiment analysis datasets used in this study. For datasets based on 1-5 star ratings, the individual sentiment counts were not provided in the source and are marked as N/A; these were converted to binary labels during our preprocessing stage.

Data	Description	Total	Pos	Neg	Neu	Year	Ref.
ATT	Tourist attraction reviews (TripAdvisor)	2,153	2,072	81	0	2015	[19]
HTL	Hotel reviews (TripAdvisor)	15,572	10,775	2,647	2,150		
MOV	Movie reviews (elcinema.com)	1,523	968	384	171		
PROD	Product reviews (souq.com)	4,271	3,100	863	308		
RES1	Restaurant reviews (qaym.com)	8,363	5,946	2,417	0		
RES2	Restaurant reviews (tripadvisor.com)	2,641	2,108	268	265		
TEAD	Tweets from various Arabic dialects	5.2M	3.1M	2.1M	0	2018	[20]
ARASAS	Tweets expressing speech acts	18,678	4,400	7,384	6,894	2018	[21]
BRAD	Book reviews (goodreads.com, 1-5 stars)	510K	N/A			2016	[22]
LABR	Book reviews (goodreads.com, 1-5 stars)	63K	N/A			2013	[23]
HARD	Hotel reviews (booking.com, 1-5 stars)	105K	N/A			2018	[24]

Source: created by the authors, from the datasets cited in the Ref. column

Table 2. Worked examples of LLM-based normalization on real corpus texts from three Set3 source domains. Arabic is quoted verbatim; the gloss identifies each edit.

#	Domain	Original (dialectal)	Normalized (MSA)	Gloss and edits
1	RES	احلى تجربة للمطاعم شفتها بحياتي	أحلى تجربة للمطاعم رأيتها في حياتي	"The best restaurant experience I have seen in my life." Dialectal verb شفتها → رأيتها; initial hamza restored (احلى → أحلى); dialectal contraction بحياتي → في حياتي. 3 of 5 tokens rewritten.
2	RES	لكن الحين صار الموضوع تسخين في تسخين	لكن الآن أصبح الموضوع تسخين في تسخين	"…but now it has become just reheating upon reheating." Gulf temporal الحين → الآن; dialectal verb صار → أصبح. The five topical tokens are untouched; 2 of 7 tokens rewritten.
3	HARD	لا انصح فيه	أحذر منه	"I do not recommend it" → "I warn against it." The negation rule in action: negator and verb are replaced together by a confirmed antonym, so the negative label is preserved by construction; dialectal preposition فيه → منه.
4	ARASAS	□□ #Mohammadsaad مستنيكم	أنتظركم قلب عاشق محمد سعد	"Waiting for you □□ #Mohammadsaad" → "I await you, a loving heart, Mohammad Saad." Egyptian مستنيكم → أنتظركم; emojis replaced by their MSA descriptions; the Latin-script hashtag is segmented and rendered in Arabic.

Source: created by the authors

Table 3. Corpus-level lexical effect of normalization on the ten Set3 domains. Rows: texts before → after normalization (coverage in parentheses). Types@N: number of distinct word types within the same number of running tokens N in both versions; Δ: relative reduction. TTR@N: type–token ratio at N. Artifact columns give occurrences per 10,000 tokens, before → after.

Domain	Rows (cov.)	Types@N	Δ	TTR@N	Emoji	Latin	Elong.
ARASAS_ASSERT	6,330 → 6,134 (96.9%)	28,934 → 23,670	18.2%	0.234 → 0.191	148.3 → 0.3	227.1 → 1.0	13.6 → 0.8
ARASAS_Expr	6,284 → 5,484 (87.3%)	30,164 → 24,411	19.1%	0.254 → 0.205	317.2 → 0.1	136.9 → 0.2	24.9 → 1.4
TEAD1	6,000 → 5,100 (85.0%)	17,630 → 14,695	16.7%	0.264 → 0.220	0.0 → 0.0	0.0 → 0.0	0.0 → 1.5
TEAD2	6,000 → 5,600 (93.3%)	23,223 → 18,413	20.7%	0.303 → 0.240	0.0 → 0.0	0.0 → 0.0	0.0 → 3.1
TEAD3	6,000 → 5,700 (95.0%)	22,999 → 18,390	20.0%	0.291 → 0.233	0.0 → 0.0	0.0 → 0.0	0.0 → 2.9
HARD_FAMILY1	6,000 → 4,600 (76.7%)	19,926 → 14,317	28.2%	0.178 → 0.128	7.5 → 0.6	57.8 → 0.1	17.3 → 0.0
HARD_FAMILY2	6,000 → 4,700 (78.3%)	20,902 → 14,792	29.2%	0.177 → 0.125	7.9 → 1.4	92.0 → 0.3	19.8 → 0.3
HARD_SINGLE	6,000 → 5,907 (98.5%)	20,784 → 14,674	29.4%	0.169 → 0.120	5.1 → 0.0	96.9 → 0.3	19.6 → 0.2
RES	5,686 → 5,081 (89.4%)	35,582 → 26,239	26.3%	0.175 → 0.129	15.1 → 0.1	159.2 → 1.3	72.6 → 1.5
HTL	5,647 → 4,104 (72.7%)	47,944 → 40,012	16.5%	0.120 → 0.100	0.2 → 0.1	36.1 → 5.7	6.5 → 0.5

Mean cross-domain overlap over the 45 domain pairs, before → after: Jaccard similarity of full vocabularies 0.156 → 0.201; Jaccard similarity of the 1,000 most frequent types 0.250 → 0.283; shared-token share 0.670 → 0.732.

Source: created by the authors

Table 4. Domain splits for each dataset used in the meta-learning experiments.

Dataset	Role	Domains
Set1	Training	ATT, HTL, MOV, RES2, TEAD, Arasas, BRAD, LABR, HARD
	Validation	RES1
	Test	PROD
Set2	Training	TEAD, Arasas, BRAD, LABR, HARD
	Validation	RES
	Test	HTL
Set3	Training	ARASAS_Exp, ARASAS_ASSERT, TEAD1, TEAD2, TEAD3, HARD_FAMILY1, HARD_FAMILY2, HARD_SINGLE
	Validation	RES
	Test	HTL

Source: created by the authors