Automatic Error Type Annotation for Arabic

Riadh Belkebir and **Nizar Habash**
Computational Approaches to Modeling Language (CAMeL) Lab
New York University Abu Dhabi
{rb4822,nizar.habash}@nyu.edu

Abstract

We present ARETA, an automatic error type annotation system for Modern Standard Arabic. We design ARETA to address Arabic's morphological richness and orthographic ambiguity. We base our error taxonomy on the Arabic Learner Corpus (ALC) Error Tagset with some modifications. ARETA achieves a performance of 85.8% (micro average F1 score) on a manually annotated blind test portion of ALC. We also demonstrate ARETA's usability by applying it to a number of submissions from the QALB 2014 shared task for Arabic grammatical error correction. The resulting analyses give helpful insights on the strengths and weaknesses of different submissions, which is more useful than the opaque M^2 scoring metrics used in the shared task. ARETA employs a large Arabic morphological analyzer, but is completely unsupervised otherwise. We make ARETA publicly available.

1 Introduction

There has been a lot of interest recently in Automatic Error Evaluation for many languages. Many specialized shared tasks in grammatical error correction (GEC) and text normalization have used tools like M^2 Scorer (Dahlmeier and Ng, 2012) and ERRANT (Bryant et al., 2017). In contrast with the opaque results of M^2 Scorer based on extracted edits, ERRANT, designed primarily for English, allows for deep interpretation of GEC error types since it gives more detailed explanations. Error type explainability is helpful for many NLP applications, including second language learning.

Arabic is a morphologically rich and complex language with a high degree of ambiguity at the orthographic, morphological, syntactic, lexical and semantic levels (Habash, 2010). Figure 1 presents a motivating example for the complexity of Arabic error type annotation (discussion in Section 2). Previous Arabic text correction shared tasks like QALB 2014 (Mohit et al., 2014) and QALB 2015

Figure 1: An example of aligned erroneous source and reference phrases with three different error types.

(Rozovskaya et al., 2015) evaluated submissions using the M^2 Scorer. Alfaifi (2015) proposed a taxonomy for Arabic error types and annotated the Arabic Learner Corpus (ALC) using it; but, he does not provide an error classification tool.

In this paper, we present ARETA, a system for the extraction and annotation of error types in Arabic. ARETA is inspired by ERRANT, but addresses the unique and complex challenges of Arabic. We base our error taxonomy on the ALC Error Tagset (Alfaifi and Atwell, 2014; Alfaifi, 2015) with some modifications. ARETA reaches a micro average F1 score of 85.8% on an ALC blind test. We also demonstrate ARETA's usability on a number of submissions from QALB 2014 (Mohit et al., 2014) shared task. While ARETA employs a large Arabic morphological analyzer, it is completely unsupervised otherwise. To our knowledge, this is the first system of its kind for Arabic. We make ARETA publicly available.[1]

The remainder of this paper is organized as follows. Sections 2 and 3 present a motivating example, and related work, respectively. Section 4 describes our approach. Section 5 presents experimental results and discussions.

---

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

2 A Motivating Example

Figure 1 shows an example of an erroneous Arabic source phrase بالسياره أحمر *bAlsyArh ÂHmr*[2] and its correct reference في السيارة الحمراء *fy AlsyArħ AlHmrA'*. The phrase, meaning 'in the red car', includes three error types.

- (+ب *b+* → في *fy*)
  Semantic error: the preposition proclitic +ب *b+* 'by/with' is used instead of the free preposition في *fy* 'in'.

- (السياره *AlsyArh* → السيارة *AlsyArħ*)
  Orthographic error: the Ta-Marbuta feminine marker, ة *ħ*, is misspelled as ه *h*.

- (أحمر *ÂHmr* → الحمراء *AlHmrA'*)
  Morphological errors: (a) masculine gender is used instead of feminine, and (b) the definite article proclitic +ال *Al+* 'the' is dropped.

A simple Levenshtein edit distance (Levenshtein, 1966) between the source and reference phrases suggests the reference is modified through two word substitutions and one word deletion, or three character substitutions and five character deletions. In contrast, a linguistically motivated error type classification is more insightful.

From the point of view of the source phrase, there are two words, and they each get two error tags according to the ALC error taxonomy (Table 1). The first word بالسياره *bAlsyArh* has an attachable proclitic and as such includes both semantic and orthographic errors. And the second word أحمر *ÂHmr* includes two morphological errors (gender and definiteness). A system to identify the exact error types needs to be aware of not only the complexity of Arabic morphology but also the possibility of multiple co-occurring error types. We address these issues in ARETA's design.

3 Related Work

While M[2] Scorer (Dahlmeier and Ng, 2012) has been used for automatic evaluation of GEC shared tasks in different languages (Ng et al., 2014, 2013; Mohit et al., 2014; Rozovskaya et al., 2015), a lot of attention has been paid to annotating and evaluating the output of English text correction systems, e.g., ERRANT (Bryant et al., 2017). There is still

---

a lack of tools that allow such utility for other languages, including Arabic. In the rest of this section, we present the main tools for evaluating and annotating error types, some of the challenges of Arabic processing, and the Arabic error taxonomy which we modify.

3.1 M[2] Scorer, ERRANT, and SERRANT

The M[2] Scorer (Dahlmeier and Ng, 2012) is a tool used for evaluating GEC systems based on F1 or F0.5 scores. It uses a method called $MaxMatch$ (M[2]) to compute the sequence of phrase-level edits that achieve the highest overlap with the gold (reference) annotation. Although the M[2] Scorer evaluates GEC systems based on extracted edits, it does not provide error types based on the linguistic features of the language.

ERRANT (Bryant et al., 2017) is a grammatical ERRor ANnotation Toolkit that automatically extracts edits from parallel original and corrected sentences and classifies them using a rule-based framework. It was first applied to the CoNLL-2014 shared task (Ng et al., 2014) to carry out detailed error type analyses. Most current GEC systems use ERRANT to annotate extracted edits and evaluate system outputs. The ERRANT taxonomy has 25 main error categories.

SERRANT (Choshen et al., 2021) is a system for automatic classification of English grammatical errors that combines ERRANT (Bryant et al., 2017) with SERCL, a taxonomy of Syntactic Errors and an automatic Classification (Choshen et al., 2020). SERRANT uses ERRANT's annotations when they are informative and those provided by SERCL otherwise.

While the M[2] Scorer is generic and can be applied to many languages to extract edits and evaluate GEC system quality, ERRANT and SERRANT focus more on linguistic aspects and give better explainability of error types. However, these frameworks require knowledge about the targeted language and are expensive to build. Furthermore, the ambiguity challenges that are part of the Arabic language make the task even more challenging since the error types can be interpreted differently for many words.

3.2 Arabic Language Processing Challenges

Arabic poses a number of challenges for natural language processing in general and the task of grammatical error correction and error type annotation in particular (Habash, 2010). First, Arabic

---

[2]All Arabic script examples are paired with a strict 1-to-1 transliteration in the HSB scheme (Habash et al., 2007).

Class	Err Tag	Description	Arabic Example	Transliteration
Orthography	OA	Confusion in Alif, Ya and Alif-Maqsura	علي ← على	ςly → ςlý
	OC	Wrong order of word characters	تبرينا ← تربينا	tbrynA → trbynA
	OD	Additional character(s)	يعدوم ← يدوم	yςdwm → ydwm
	OG	Lengthening short vowels	نقيمو ← نقيم	nqymw → nqym
	OH	Hamza errors	اكثر ← أكثر	Akθr → Âkθr
	OM	Missing character(s)	سالين ← سائلين	sAlyn → sAŷlyn
	ON	Confusion between Nun and Tanwin	ثوبن ← ثوبٌ	θwbn → θwbũ
	OR	Replacement in word character(s)	مصلنا ← وصلنا	mSlnA → wSlnA
	OS	Shortening long vowels	أوقت ← أوقات	Âwqt → ÂwqAt
	OT	Confusion in Ha, Ta and Ta-Marbuta	مشاركه ← مشاركة	mšArkh → mšArkħ
	OW	Confusion in Alif Fariqa	وكانو ← وكانوا	wkAnw → wkAnwA
	OO	*Other orthographic errors*	-	-
Morphology	MI	Word inflection	معروف ← عارف	mςrwf → ςArf
	MT	Verb tense	تفرحني ← أفرحتني	tfrHny → ÂfrHtny
	MO	*Other morphological errors*	-	-
Syntax	XC	Case	رائع ← رائعاً	rAŷς → rAŷςAã
	XF	Definiteness	السن ← سن	Alsn → sn
	XG	Gender	الغربي ← الغربية	Alγrby → Alγrbyħ
	XM	Missing word	على ← Null	*Null → ςlý*
	XN	Number	فكرتي ← أفكاري	fkrty → ÂfkAry
	XT	Unnecessary word	Null ← على	ςlý → *Null*
	XO	*Other syntactic errors*	-	-
Semantics	SF	Conjunction error	سبحان ← فسبحان	sbHAn → fsbHAn
	SW	Word selection error	من ← عن	mn → ςn
	SO	*Other semantic errors*	-	-
Punctuation	PC	Punctuation confusion	المتوسط. ← المتوسط،	AlmtwsT. → AlmtwsT,
	PM	Missing punctuation	العظيم ← العظيم،	AlςĎym → AlςĎym,
	PT	Unnecessary punctuation	العام، ← العام	AlςAm, → AlςAm
	PO	*Other errors in punctuation*	-	-
Merge	MG	Words are merged	ذهبتالبارحة ← ذهبت البارحة	ðhbtAlbArHħ → ðhbt AlbArHħ
Split	SP	Words are split	المحا دثات ← المحادثات	AlmHA dθAt → AlmHAdθAt

Table 1: The ALC error type taxonomy extended with merge and split classes. The error tags are listed alphabetically, except for the highlighted *Other tags, which we do not support in ARETA.

is morphologically rich. Words in Arabic inflect for person (per), gender (gen), number (num), aspect (asp), voice (vox), mood (mod), state (stt) and case (cas). Furthermore, Arabic uses a number of attachable proclitics (prc0-2) and enclitics (enc0). Second, Arabic is orthographically very ambiguous due to the use of optional diacritics, which are almost always absent. Figure 2 demonstrates the various analyzes associated with two Arabic words. In some cases the analyses differ in part of speech (POS). To address these challenges, ARETA uses CAMeL Tools (Obeid et al., 2020), an open source Python toolkit for Arabic language processing. CAMeL Tools uses the CALIMA-Star Arabic morphological analyzer (Taji et al., 2018) and provides morphological disambiguation functionality over it.

In developing ARETA, we took inspiration from

AMEANA (El Kholy and Habash, 2011), which also relies on morphological analyzers to provide morphological error analysis in the context of machine translation evaluation. However, ARETA addresses a wider range of error types, and is intended to be more general.

3.3 The Arabic Learner Corpus Error Taxonomy

Alfaifi and Atwell (2014) proposed a taxonomy of 29 error tags for Arabic (See Table 1). They annotated a portion of Arabic Learner Corpus (ALC) dataset such that for each erroneous word, one of the classes of error is given along with the word's correction. In Alfaifi and Atwell (2015) they presented a tool that facilitated *semi-automatic error tagging*. The tagging feature worked as translation memory, where annotated words are saved in

a database, and recalled when seen again.

We base the error taxonomy we use in ARETA on Alfaifi and Atwell (2014)'s comprehensive taxonomy with two modifications. First, we add two error classes - merge (MG) and split (SP) to allow handling man-to-many word corrections. And secondly we drop all of the *Other* error tags - OO, MO, XO, SO and PO, corresponding to *other* orthographic, morphological, syntactic, semantic and punctuation errors, respectively. These errors tags collectively accounted for 0.7% of all error tags in the ALC, and would have added a lot of complexity to our system. As such, ARETA's full taxonomy has 7 classes and 26 error tags. When we evaluate ARETA against the ALC annotations, we penalize ARETA for missing all the *Other* tags.

4 Approach

In this section, we present our approach to developing ARETA, an automatic error annotation framework for Arabic. We organize this section in three parts: basic word alignment, automatic error annotation, and error-type-based evaluation.

Given a raw input sequence (S_{raw}), a system output sequence (hypothesis) (S_{hyp}), and a reference sequence (S_{ref}), we want to be able to annotate and evaluate the quality of the system output (hypothesis) (S_{hyp}).

4.1 Basic Word Alignment

Before we can annotate a word's error type, we need to align said word to its correction. The first step is thus to word-align the two sequences (source and target) whose differences we want to annotate. These may be the pair of S_{raw} and S_{ref}, S_{raw} and S_{hyp}, or S_{hyp} and S_{ref}. Since this task assumes the source and target to be of the same language with some differences in spelling, it is a simpler task than general word alignment (Och and Ney, 2003). We start with character-level edit-based alignment to align the characters, and then we group them in words such that the source is aligned to target as 1-to-many words (where many include zero). We make extensive use of the CED word alignment tool by Khalifa et al. (2021). We refer in the next step to the generated word alignment using the nomenclature of $Align_{source,target}$, e.g., $Align_{raw,ref}$.

One limitation of the current implementation of this step is that it cannot handle many-to-many alignments; and as such split errors cannot be

modeled in ARETA unless they are independently provided. The ALC data does not annotate split nor merge errors, so this limitation has no effect on it. However, when working with the QALB 2014 data, we exploited the shared task's `.m2` file information which provided some of the merge alignment for raw and reference. These files were created off the QAWI interface (Obeid et al., 2013) which was used in the QALB project annotation (Zaghouani et al., 2014).

We evaluate our word alignment against the manual alignment in the ALC corpus' raw and reference sequences. Our automatic word alignment matches the manual alignment for 99.24% of the words. The failed alignment cases include minor word reordering such as لم أكن *lm Âkn* 'not-past I-be' aligning with كنت لا *knt lA* 'I-was not', where the negation particles appear at either side of the verb. Other failed alignment cases include inserted words that could be paired with their left or the right neighbors. For example, in the raw sequence وكان كل *wkAn kl* 'and-it-was all' which is paired with reference وكنت في كل *wknt fy kl* 'and-I-was in all', the word في *fy* is paired as وكنت *wknt* in the manual alignment, but the automatic aligner pairs it with كل *kl*. Both alignments are plausible.

4.2 Automatic Error Type Annotation

The input to the automatic error annotation process is an alignment, e.g., $Align_{raw,ref}$. Then for each aligned pair of words, the system tries to extract the error type. The system is divided into four components to allow modeling combinations of error tags, but with some constraints driven by efficiency and control over the error tag search space.

First, the **Punctuation component** detects all punctuation error tags (PC, PM, and PT) using regular expressions. The punctuation error tags can be used with tags detected by other components.

Second, the **Regex component** uses regular expressions to detect all merges (MG) and splits (SP), word insertions (XT) and deletions (XM), as well as some orthographic errors (OC, OG, ON, OS, and OW). In the current implementation, this component is terminal if an error tag is identified. Otherwise, we move to the third component if we have a 1-to-1 word pairing, or to the fourth component if we have a 1-to-many word pairing. Many-to-many word pairings are not supported in this version.

Error: Analyses for أحمر *ÂHmr*															
diac	lex	pos	gloss	prc2	prc1	prc0	per	num	gen	asp	vox	mod	stt	cas	enc0
أَحْمَرَ	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	c	a	0
أَحْمَرُ	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	c	n	0
أَحْمَر	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	c	g	0
أَحْمَرَ	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	d	a	0
أَحْمَرُ	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	d	n	0
أَحْمَر	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	d	g	0
أَحْمَرَ	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	i	a	0
أَحْمَرُ	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	i	n	0
أَحْمَر	أَحْمَر	adj	red	0	0	0	na	s	m	na	na	na	i	g	0
أَحْمَّرَ	حَمَّر	verb	redden;roast	0	0	0	1	s	m	i	a	s	na	na	0
أَحْمَّرَ	حَمَّر	verb	redden;roast	0	0	0	1	s	m	i	a	i	na	na	0
أَحْمَّرَ	حَمَّر	verb	redden;roast	0	0	0	1	s	m	i	a	j	na	na	0
أَحْمَرَ	أَحْمَر	verb	turn_red;blush	0	0	0	1	s	m	i	a	j	na	na	0
أَحْمَرَ	أَحْمَر	verb	turn_red;blush	0	0	0	1	s	m	i	a	s	na	na	0
أَحْمَرَ	أَحْمَر	verb	turn_red;blush	0	0	0	1	s	m	i	a	i	na	na	0
إِحْمَرَ	أَحْمَر	verb	turn_red;blush	0	0	0	3	s	m	p	a	i	na	na	0

Reference: Analyses for الحمراء *AlHmrA'*															
diac	lex	pos	gloss	prc2	prc1	prc0	per	num	gen	asp	vox	mod	stt	cas	enc0
الحَمْراءَ	أَحْمَر	noun_prop	Alhambra	0	0	Al_det	na	s	f	na	na	na	c	a	0
الحَمْراءُ	أَحْمَر	noun_prop	Alhambra	0	0	Al_det	na	s	f	na	na	na	c	n	0
الحَمْراء	أَحْمَر	noun_prop	Alhambra	0	0	Al_det	na	s	f	na	na	na	c	g	0
الحَمْراءَ	أَحْمَر	noun_prop	Alhambra	0	0	Al_det	na	s	f	na	na	na	d	a	0
الحَمْراءُ	أَحْمَر	noun_prop	Alhambra	0	0	Al_det	na	s	f	na	na	na	d	n	0
الحَمْراء	أَحْمَر	noun_prop	Alhambra	0	0	Al_det	na	s	f	na	na	na	d	g	0
الحَمْراءَ	أَحْمَر	adj	red	0	0	Al_det	na	s	f	na	na	na	c	a	0
الحَمْراءُ	أَحْمَر	adj	red	0	0	Al_det	na	s	f	na	na	na	c	n	0
الحَمْراء	أَحْمَر	adj	red	0	0	Al_det	na	s	f	na	na	na	c	g	0
الحَمْراءَ	أَحْمَر	adj	red	0	0	Al_det	na	s	f	na	na	na	d	a	0
الحَمْراءُ	أَحْمَر	adj	red	0	0	Al_det	na	s	f	na	na	na	d	n	0
الحَمْراء	أَحْمَر	adj	red	0	0	Al_det	na	s	f	na	na	na	d	g	0

Figure 2: Morphological changes between the words أحمر *ÂHmr* 'red$_{ms}$' and الحمراء *AlHmrA'* 'the red$_{fs}$'

Third, the **Ortho-Morph component** handles the more challenging orthographic and morphological error types and their combination. This component relies heavily on the CAMeL Tools' morphological analyzer to handle Arabic's rich morphology and ambiguous orthography (Obeid et al., 2020). The algorithm is as follows:

For each $Pair_i = <raw\_word_i, ref\_word_i>$ in $Align_{raw,ref}$:

1. Generate the list of possible orthographic edits ($edit\_list$) that transforms raw_word_i to ref_word_i.

2. Generate the possible subsets of elements of $edit\_list$ including the empty set. We call this list p_edits.

3. For each subset p_ed_k of p_edits, calculate the morphological feature differences ($morph\_edits$) between raw_word_i and ref_word_i after applying p_ed_k to raw_word_i.[3] This generates a path of edits $path = [orth\_edits + morph\_edits]$.

4. Add $path$ to the list of possible paths ($paths$).

5. Return the shortest path from the list of possible paths ($paths$).

Figure 2 demonstrates the step of identifying $morph\_edits$ between raw_word أحمر *ÂHmr* 'red$_{ms}$' and ref_word الحمراء *AlHmrA'* 'the red$_{fs}$'. The green-shaded analyses represent the subset of all analyses sharing the same lemma and POS. The six linked pairs of analyses all have the same smallest number of $morph\_edits$ (2): $prc0 : 0 \rightarrow Al\_det$ and $gen : m \rightarrow f$.

---

[3] $morph\_edits$ are calculated over analyses of raw_word_i and ref_word_i that share the same lemma (lex) and pos tags.

The final step in the third component uses rules to map the set of edits in the shortest edit path to corresponding error tags. For example, any $morph\_edit$ involving gen will result in the XG error tag, and any $orth\_edit$ involving a Ta-Marbuta change results in the OT error tag. As such, the example in Figure 2 receives the complex error tag $XF+XG$ (definiteness and gender). In cases with only $orth\_edit$, we map to additional/missing/replaced character error tags (OD, OM, OR) when the percentage of affected raw word characters is below 50%. In cases above that heuristic threshold, we assign the word selection error tag (SW).

Fourth, the **Multi-Word component** handles 1-to-many word pairings by applying Arabic Treebank (ATB) tokenization to both sides (Maamouri and Bies, 2004; Habash, 2010). ATB tokenization splits all clitics except for the definite article. We generate the unique ATB tokenizations for all the possible analyses using the CAMeL Tools morphological analyzer (Obeid et al., 2020). At this point, for each tokenized sequence pair (e.g., raw and reference), we apply the basic word alignment step (Section 4.1) locally, and pass the aligned ATB tokens through components one, two and three. The resulting error tags for the various ATB tokens are joined and assigned to the word that produced them. In the example in Figure 1, the 1-to-many pairing of بالسياره $bAlsyArh$ and في السيارة fy $AlsyAr\hbar$ is handled by this component, and receives the complex error tag $SW+OT$ (word selection and Ta-Marbuta).

4.3 Error-Type-based Evaluation

ARETA can be used to conduct error-type-based evaluations in a number of configurations.

First, given reference error tags, as in triplets of $(S_{raw}, S_{ref}, ErrorTag)$, we can evaluate how well ARETA performs in automatic error-type annotation in terms of F1-score (Micro Avg, Macro Avg, and Weighted Avg) of the different error tags. See Section 5.2.

Second, in the case of no reference error tags, we use our system to identify the *reference* error tags using the pair (S_{raw}, S_{ref}) and compare them using F1 score to the predicted error tags using the pair (S_{raw}, S_{hyp}). See Section 5.3

Finally, ARETA can be also used to diagnose a system's output given the reference directly (S_{hyp}, S_{ref}) to identify *remaining error types*.

5 Experiments

We present next two sets of experiments. First, we evaluate the quality of error type annotation in comparison to the ALC. Second, we calculate the correlation between an evaluation metric based on our error type prediction system and the M^2 Scorer.

5.1 Datasets

To perform the experiments, we used two datasets. First is the **Arabic Learner Corpus** (Alfaifi and Atwell, 2014), which contains 10K words annotated for error type and corrections. The number of changed raw words is 1,688 (∼16.8%), of which 75% appear only one time and 12% appear two times. Because of this sharp-tailed Zipfian distribution and limited training instances, we expect it to be hard to learn from this data using machine learning systems. We split the data into two parts: Dev and Test by randomly selecting complete documents from the corpus (10 for Dev and 10 for Test).[4] Dev was used while building ARETA.

The second dataset is the **QALB 2014 Shared Task** test set and some of the submitted systems' outputs (Mohit et al., 2014). We use it to evaluate the correlation between the M^2 Scorer and our metrics: F1-score (Micro Avg, Macro Avg, and Weighted Avg) of the different error tags.

5.2 ALC Automatic Error Annotation

Table 2 presents the results of evaluating ARETA's performance in identifying error tags on the Dev and Test portions of the ALC data set.

We compare four settings that vary in terms of the number of analyses from the CAMeL Tools morphological analyzer (Obeid et al., 2020): using the top 1, 2, or 3 analyses from the MLE disambiguator, or using all analyses (16 analyses/word). On average, using the top 1, 2 or 3 analyses took about the same time (∼16 secs to run Dev), while the full analyzer took 44% more time (∼23 secs to run on Dev).[5] Consistently, in both Dev and Test, using more analyses improves the performance of ARETA across all metrics. ARETA's best setting (All) matches the ALC annotation with 89.2% (F1 Micro Avg) on Dev, and 85.8% (F1 Micro Avg) on

---

Dev

Err Tag	Top 1	Top 2	Top 3	All	Support %
OA	80.0	80.0	80.0	80.0	0.3
OC	100.0	100.0	100.0	100.0	0.3
OD	50.8	61.2	66.7	72.7	1.8
OG	0.0	0.0	0.0	0.0	0.0
OH	88.4	90.4	91.4	93.8	14.2
OM	54.7	58.6	61.1	76.1	3.7
ON	0.0	0.0	0.0	0.0	0.0
OR	63.8	65.7	66.7	72.1	3.1
OS	0.0	0.0	0.0	0.0	0.0
OT	97.4	98.3	98.3	98.3	6.5
OW	66.7	66.7	66.7	66.7	0.2
OO	0.0	0.0	0.0	0.0	0.1
MI	19.0	19.0	22.7	38.5	1.3
MT	85.7	93.3	93.3	93.3	0.9
MO	0.0	0.0	0.0	0.0	0.0
XC	47.6	54.8	59.5	85.4	4.2
XF	72.5	87.2	96.5	97.7	4.8
XG	70.2	71.2	71.2	84.1	4.0
XM	79.5	81.3	81.3	87.8	5.5
XN	71.4	75.0	75.0	58.3	1.0
XT	89.2	87.1	89.2	89.2	4.8
XO	0.0	0.0	0.0	0.0	1.0
SW	69.7	71.3	71.3	79.6	9.3
SF	50.0	50.0	50.0	91.7	1.3
SO	0.0	0.0	0.0	0.0	0.0
PC	92.3	92.3	92.3	92.3	3.3
PT	60.0	60.0	60.0	60.0	0.6
PM	99.4	99.4	99.4	99.4	27.9
PO	0.0	0.0	0.0	0.0	0.0
Micro Avg	80.5	82.6	83.9	89.2	100.0
Macro Avg	52.0	53.9	54.9	59.2	100.0
Weighted Avg	81.3	83.4	84.5	89.1	100.0

Test

Err Tag	Top 1	Top 2	Top 3	All	Support %
OA	83.3	83.3	83.3	72.7	0.6
OC	88.9	88.9	88.9	88.9	1.6
OD	52.2	49.2	49.2	52.8	2.3
OG	33.3	40.0	40.0	50.0	0.2
OH	94.1	94.3	94.7	96.1	24.1
OM	62.9	65.1	67.5	71.2	3.7
ON	66.7	66.7	66.7	66.7	0.1
OR	62.5	65.9	66.7	62.9	4.4
OS	0.0	0.0	0.0	0.0	0.0
OT	93.3	93.3	91.5	93.1	3.3
OW	0.0	0.0	0.0	0.0	0.0
OO	0.0	0.0	0.0	0.0	0.1
MI	5.7	10.5	10.5	16.0	1.3
MT	66.7	66.7	80.0	90.9	0.6
MO	0.0	0.0	0.0	0.0	0.0
XC	64.0	67.9	70.2	82.1	3.1
XF	77.9	82.5	90.7	93.5	4.9
XG	74.3	72.2	74.3	70.6	2.0
XM	79.0	80.5	81.0	82.4	4.6
XN	50.0	44.4	40.0	25.0	0.5
XT	86.2	86.2	87.3	91.2	6.0
XO	0.0	0.0	0.0	0.0	0.2
SW	69.4	70.7	72.2	77.6	7.6
SF	40.9	40.9	40.9	80.0	3.7
SO	0.0	0.0	0.0	0.0	0.0
PC	80.0	80.0	80.0	80.0	1.9
PT	52.2	52.2	52.2	52.2	0.6
PM	98.3	98.3	98.3	98.3	22.3
PO	0.0	0.0	0.0	0.0	0.0
Micro Avg	81.1	81.8	82.7	85.8	100.0
Macro Avg	51.1	51.7	52.6	55.0	100.0
Weighted Avg	81.6	82.4	83.3	86.4	100.0

Table 2: Comparing the F1 score results for error tag annotation on ALC **Dev** and **Test** sets using different number of analyses from the morphological analyzer.

Test. For some specific error tags the performance drops with more analyses, due to the larger search space introduced by the analyzer.

We observe that among the top five tags in terms of frequency in Dev, ARETA detects, with high accuracy, errors involving punctuation (PM), Hamzas (OH), and Ha/Ta/Ta-Marbuta (OT); however the performance on word selection (SW) and missing words (XM) are lower. The distribution of error tags varies between Dev and Test sets: there is a 91.3% correlation between the support of the tags in the two sets, but the top three error tags are the same in both (PM, OH, and SW). The top five tags in terms of frequency in Test also include unnecessary words XT and definiteness XF, both of which perform relatively well. The F1 Macro Average of the top five error tags (in terms of support) is 91.8% for Dev, and 91.3% for Test.

If we group the error tags by the components that handle them, the F1 weighted averages for Punctuation (PC, PM, PT), Regex (OC, OG, ON, OS, OW, XM, XT), and Ortho-Morph components (MI, MT, OA, OD, OH, OM, OR, OT, SF, SW, XC, XF, XG, XN), are 98.0, 88.3 and 85.9, for Dev, and 95.7, 86.7 and 83.0, for Test, respectively.

Table 2 does not include split and merge error tags as they are not present in the ALC corpus. We include all the *O *Other* error tags even though ARETA does not handle them, for completeness.

System	F1 Score							
	Micro Avg	*Rank*	Macro Avg	*Rank*	Weighted Avg	*Rank*	M2 Scorer	*Rank*
CLMB-1	66.9	*3*	38.7	*2*	**66.0**	*1*	**67.9**	*1*
CLMB-2	64.6	*7*	37.5	*3*	64.3	*3*	66.3	*2*
CLMB-3	65.1	*6*	35.1	*4*	64.4	*2*	65.2	*5*
CMUQ-1	67.4	*2*	32.4	*5*	64.2	*4*	65.4	*4*
CP13-1	54.3	*9*	23.0	*9*	51.5	*9*	51.8	*9*
CP13-2	62.0	*8*	26.7	*8*	57.4	*8*	58.6	*8*
CUFE-1	**67.9**	*1*	**46.8**	*1*	60.4	*7*	65.7	*3*
GLTW-1	34.9	*10*	17.3	*10*	29.4	*10*	35.4	*10*
GLTW-2	19.7	*11*	11.7	*11*	15.2	*11*	21.0	*11*
GWU-1	65.4	*4*	28.3	*7*	62.6	*6*	62.3	*6*
GWU-2	65.4	*5*	28.4	*6*	62.6	*5*	62.2	*7*

Table 3: Comparing submitted systems from QALB 2014 shared task using M^2 Scorer and F1-based metrics of ARETA's tags. Systems rankings are presented in italics.

	Micro Avg	Macro Avg	Weighted Avg
Avg ranking difference	1.45	0.72	0.91
F1 correlation	99.30%	87.30%	99.36%
Ranking correlation	79.09%	95.45%	86.36%

Table 4: Correlation between the M^2 Scorer and ARETA's F1-based metrics and rankings in Table 3.

5.3 Revisiting the QALB 2014 Shared Task Submissions

Our System vs M^2 Scorer   We compare the M^2 Scorer results on the QALB 2014 (Mohit et al., 2014) shared task submissions with F1-based metrics over the error tags produced by ARETA. We calculate the *reference* error tags using the (S_{raw}, S_{ref}) pairs, and compare them to the predicted error tags using the pair (S_{raw}, S_{hyp}). The systems outputs we have access to are: CLMB (Rozovskaya et al., 2014), CMUQ (Jeblee et al., 2014), CP13 (Tomeh et al., 2014), CUFE (Nawar and Ragheb, 2014), GLTW (Zerrouki et al., 2014) and GWU (Attia et al., 2014).

Table 3 presents these results and their associated system rankings. Table 4 compares the F1-based metrics with the M^2 Scorer results presented in Table 3 across all of the system outputs using Pearson correlation over F1 scores and rankings, and the average absolute ranking difference. We observe a high correlation between the Weighted Avg and M^2 Scorer as well as the Micro Avg and M^2 Scorer. In terms of ranking, the Macro Avg has the highest correlation and lowest average ranking difference with M^2 Scorer.

According to the F1 Weighted Avg, the best performing system is CLMB-1. This matches with the M^2 Scorer ranking. But according to the F1 Macro Avg and Micro Avg, CUFE-1 is the best system. We investigate the differences between these and other systems next using ARETA's rich error tag set.

Error Type Analysis   To understand the error patterns of the QALB 2014 shared task submitted systems, we perform a detailed error analysis comparing these systems over all of the produced ARETA error tags. Table 5 presents these results ordered by support. Only 24 of ARETA's 26 error tags are generated for this data set. The two missing tags are for Nun/Tanwin confusion (ON) and long-vowel shortening (OS).

We observe that the best performing system on most error types is CUFE-1 (Nawar and Ragheb, 2014). Interestingly, the best performing system in terms of F1 Weighted Avg is CLMB-1, but it is the best on only one minor error tag (OG). GWU-1 and GWU-2 are the best systems in correcting missing punctuation (PM), but on average they are mid-ranked.

The PM and OH are the classes representing most errors from QALB 2014 test set. The results also show that most systems struggle to correct morpho-syntactic errors like gender change (XG) and definiteness change (XF). On the other hand,

Err Tag	F1 Score											Support %
	CLMB-1	**CLMB-2**	**CLMB-3**	**CMUQ-1**	**CP13-1**	**CP13-2**	**CUFE-1**	**GLTW-1**	**GLTW-2**	**GWU-1**	**GWU-2**	**Support %**
PM	51.7	50.6	51.2	53.2	46.4	46.6	23.4	0.0	0.0	54.8	54.8	33.0
OH	95.1	93.1	94.3	95.2	83.1	94.1	**95.2**	59.3	19.1	92.3	92.3	29.3
SP	91.4	88.9	88.4	90.5	40.4	40.2	**96.8**	86.9	85.0	91.6	91.6	4.7
OT	90.8	88.7	90.0	88.1	52.4	79.8	**96.4**	64.2	42.5	85.0	85.0	4.2
PC	40.5	40.3	40.3	53.1	52.3	53.0	**56.5**	12.7	12.7	54.4	54.4	4.2
XC	23.4	23.1	24.1	26.2	12.8	23.6	**35.8**	13.4	13.4	16.9	16.9	3.2
OR	58.4	53.0	50.9	47.6	24.6	38.8	**74.5**	13.0	3.9	43.7	43.6	3.2
PT	38.0	37.8	37.5	14.5	12.1	11.6	18.1	**38.7**	38.1	6.7	6.7	2.8
MG	74.5	71.6	74.4	45.9	28.7	35.6	**87.3**	37.0	18.1	67.0	67.2	2.6
OM	42.6	38.4	26.2	30.0	21.6	21.6	**57.0**	4.9	0.0	22.3	22.4	2.3
OD	46.6	50.8	44.7	35.2	20.9	25.7	**69.0**	16.4	12.7	34.0	33.8	2.0
OA	80.8	74.4	77.5	82.3	64.0	74.3	74.2	55.5	35.3	63.8	63.8	1.6
SW	42.6	32.1	37.1	38.0	37.1	38.9	**55.7**	2.1	2.1	24.7	25.1	1.6
XM	31.6	29.8	25.4	18.0	12.4	12.6	**66.2**	7.0	2.2	16.1	16.8	1.4
XN	24.3	23.7	21.8	17.8	15.6	18.7	**29.3**	0.0	0.0	13.8	13.7	0.9
OW	68.5	79.4	62.5	58.5	31.7	32.6	**84.0**	0.0	0.0	0.0	0.0	0.6
XT	32.3	29.2	21.6	14.5	6.2	7.2	**57.8**	12.2	10.7	22.5	22.2	0.6
MI	40.0	34.4	34.4	33.1	10.3	28.4	**41.5**	32.1	20.2	30.3	30.1	0.5
XG	3.0	3.4	3.1	0.0	0.0	0.0	**18.8**	0.0	0.0	2.6	2.6	0.3
XF	7.8	**11.1**	0.0	0.0	0.0	0.0	7.8	0.0	0.0	0.0	0.0	0.3
OG	46.7	44.1	41.4	33.3	34.0	24.0	43.8	4.4	0.0	14.1	14.1	0.3
OC	14.3	14.3	0.0	0.0	14.3	14.3	**63.8**	8.0	0.0	8.3	8.3	0.1
SF	0.0	0.0	0.0	0.0	0.0	0.0	**11.8**	0.0	0.0	0.0	0.0	0.1
MT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	~0.0
Micro Avg	66.9	64.6	65.1	67.4	54.3	62.0	**67.9**	34.9	19.7	65.4	65.4	100.0
Macro Avg	38.7	37.5	35.1	32.4	23.0	26.7	**46.8**	17.3	11.7	28.3	28.4	100.0
Weighted Avg	66.0	64.3	64.4	64.2	51.5	57.4	60.4	29.4	15.2	62.6	62.6	100.0
M2 Ranking	**1**	2	5	4	9	8	3	10	11	6	7	

Table 5: Comparing different system submissions on the QALB 2014 test set (classes are ordered by support).

most systems are good at correcting orthographic errors such as Hamza (OH) and Ta-Marbuta (OT) with high accuracy.

This analysis demonstrate how ARETA can be helpful to evaluate and diagnose errors when building Arabic GEC systems. It also motivates interesting possibilities of system combination to reach a higher performance.

6 Conclusion and Future Work

In this paper, we presented ARETA, a publicly available automatic error type annotation system for Modern Standard Arabic targeting a modified error taxonomy based on the ALC error tagset. We validated ARETA's performance using a manually annotated blind test, where it achieved 85.8% (Micro Avg F1 score). We also demonstrated ARETA's usability in providing insightful error analyses over the submissions of the QALB 2014 shared task on Arabic text correction.

In the future, we plan to develop a new taxonomy that resolves overlapping and ambiguous error types in the ALC error tagset, and that includes more error types such as syntactic agreement and reordering operations. We also plan to use syntactic parsers, such as (Shahrour et al., 2016), to model long distance dependency errors. Naturally, we will continue to improve the various components of ARETA, e.g., extending the handling of many-to-many word pairs, and improving specific error types.

Acknowledgments

References

Abdullah Alfaifi and Eric Atwell. 2014. An evaluation of the Arabic error tagset v2. In *Proceedings of the AACL 2014-The American Association for Corpus Linguistics conference*. The American Association for Corpus Linguistics.

Abdullah Alfaifi and Eric Atwell. 2015. Computer-aided error annotation: a new tool for annotating

Arabic error. In *8th Saudi Students Conference*, Queen Elizabeth II Conference Centre, London.

Abdullah Yahya G Alfaifi. 2015. *Building the Arabic Learner Corpus and a System for Arabic Error Annotation*. Ph.D. thesis, University of Leeds.

Mohammed Attia, Mohamed Al-Badrashiny, and Mona Diab. 2014. GWU-HASP: Hybrid Arabic spelling and punctuation corrector. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 148–154, Doha, Qatar. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Leshem Choshen, Dmitry Nikolaev, Yevgeni Berzak, and Omri Abend. 2020. Classifying syntactic errors in learner language. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 97–107, Online. Association for Computational Linguistics.

Leshem Choshen, Matanel Oren, Dmitry Nikolaev, and Omri Abend. 2021. SERRANT: a syntactic classifier for English Grammatical Error Types. *arXiv preprint arXiv:2104.02310*.

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.

Ahmed El Kholy and Nizar Habash. 2011. Automatic Error Analysis for Morphologically Rich Languages. In *Proceedings of the Machine Translation Summit (MT Summit)*.

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.

Serena Jeblee, Houda Bouamor, Wajdi Zaghouani, and Kemal Oflazer. 2014. CMUQ@QALB-2014: An SMT-based system for automatic Arabic error correction. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 137–142, Doha, Qatar. 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.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Mohamed Maamouri and Ann Bies. 2004. Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages (CAASL)*, pages 2–9, Geneva, Switzerland.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The first QALB shared task on automatic text correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47, Doha, Qatar. Association for Computational Linguistics.

Michael Nawar and Moheb Ragheb. 2014. Fast and robust Arabic error correction system. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 143–147, Doha, Qatar. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Ossama Obeid, Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Kemal Oflazer, and Nadi Tomeh. 2013. A Web-based Annotation Framework For Large-Scale Text Correction. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1–4, Nagoya, Japan.

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 12th language resources and evaluation conference*, pages 7022–7032.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–52.

Alla Rozovskaya, Houda Bouamor, Nizar Habash, Wajdi Zaghouani, Ossama Obeid, and Behrang Mohit. 2015. The second QALB shared task on automatic text correction for Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 26–35, Beijing, China. Association for Computational Linguistics.

Alla Rozovskaya, Nizar Habash, Ramy Eskander, Noura Farra, and Wael Salloum. 2014. The Columbia system in the QALB-2014 shared task on Arabic error correction. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 160–164, Doha, Qatar. Association for Computational Linguistics.

Anas Shahrour, Salam Khalifa, Dima Taji, and Nizar Habash. 2016. CamelParser: A system for Arabic syntactic analysis and morphological disambiguation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 228–232.

Dima Taji, Salam Khalifa, Ossama Obeid, Fadhl Eryani, and Nizar Habash. 2018. An Arabic morphological analyzer and generator with copious features. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 140–150, Brussels, Belgium. Association for Computational Linguistics.

Nadi Tomeh, Nizar Habash, Ramy Eskander, and Joseph Le Roux. 2014. A pipeline approach to supervised error correction for the QALB-2014 shared task. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 114–120, Doha, Qatar. Association for Computational Linguistics.

Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. 2014. Large Scale Arabic Error Annotation: Guidelines and Framework. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.

Taha Zerrouki, Khaled Alhawiti, and Amar Balla. 2014. Autocorrection of Arabic common errors for large text corpus. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 127–131.