Abstract

The paper describes the Egyptian Arabic-to-English statistical machine translation (SMT) system of the QCN team for the NIST OpenMT’2015 evaluation campaign. The QCN team included the Qatar Computing Research Institute, Columbia University, and New York University in Abu Dhabi. The OpenMT 2015 translation task asked participants to build systems that can translate Egyptian Arabic from three different genres (SMS, chat, and speech) into English. The challenges presented by this multigenre task were many, ranging from scarcity of parallel data for training, to noisy source text and heterogeneity of the references. For example, large portions of the provided data consisted of romanized Egyptian Arabic (aka Arabizi) rather than using the Arabic script. Therefore, several preprocessing steps were needed in order to clean the data before building our SMT system. First, we converted all Arabizi to Arabic script; we then normalized it, trying to make it more like MSA. We also morphologically segmented long Arabic words using segmentation schemes that are standard for MSA but harder to do for dialectal Arabic. We used a statistical phrase-based MT system – Moses (Koehn et al., 2007). We experimented with different data processing schemes and different SMT system settings to achieve better translation quality. Here are the major settings: Egyptian Arabic segmentation (ATB, S2, D3), Egyptian Arabic to MSA conversion, sparse features, class-based models, neural network joint model, hierarchically-interpolated language model, unsupervised transliteration mining, domain adaptation, and data selection for tuning.

It is worth mentioning that given the above-mentioned challenges, preprocessing by itself yielded the largest gains. The Egyptian Arabic segmentation gave us an improvement of up to 3 BLEU points. The hierarchically-interpolated language model added 1 extra BLEU point. The sparse features, class-based models, and neural network joint language models further improved translation quality by 0.66, 0.70 and 0.43 BLEU points absolute, respectively. In the next sections, we discuss in detail the different settings and decisions we made when preparing our submission.

The remainder of the paper is organized as follows: Section ?? explains the data preprocessing techniques and tools we tried, Section ?? explains in detail the non-standard features and components we tried in our SMT experiments, and Section ?? describes the actual system we submitted for the competition. Finally, Section ?? concludes with a summary of the main points.
2 Data Preprocessing

The NIST dataset contained text from three different genres: short text messages (SMS), chat (CHT), and transcribed conversational speech (CTS). We tackled each of the three genres separately, i.e., we built genre-specific systems. Furthermore, we split the provided training data for each genre into separate training and development sets by reserving approximately 3,000 sentences (the number of sentences is approximate as we did the splitting at the document level) from each set for development, and we used the rest for training.

For evaluation, the organizers provided two additional datasets: (i) an official devtest dataset (Test), and (ii) a small gold dataset (TestG), which is a subset of Test. The genres of the datasets were specified, and thus there was no need to train a system for automatic genre identification.

2.1 Modern Standard Arabic Preprocessing

For training purposes, some datasets for Modern Standard Arabic (MSA) were provided in addition to the Egyptian Arabic data (SMS, CHT, CTS) described above. These datasets consisted mostly of newswire text (i.e., a different genre), and thus we processed them using a standard MSA tool: MADAMIRA in MSA mode (Pasha et al., 2014). The MSA data was used to build a second phrase table to be then combined with some of the genre-specific phrase tables. We planned to match the same morphological tokenization schemes used for Egyptian Arabic, but at the end, we only used ATB segmentation for MSA. For more information on Arabic morphology challenges and tokenization schemes, see ?).

2.2 Genre-Specific Preprocessing

The CTS data contained speech tags, markers for spelling corrections, etc., which we removed before training. The Arabic side of the SMS and CHT data had elongations and spelling mistakes. Therefore, we normalized the elongations and we standardized the spelling variations.

Parts of the SMS and CHT data contained romanized Arabic text (aka Arabizi), which is Arabic written using the Roman alphabet. In order to homogenize the input data, we converted the Arabizi into the standard Arabic script (utf8).

Given the conversational nature of SMS and chat, a further complication with converting romanized text into the Arabic script was that the text was usually affected by code-switching into English; therefore, it was important for us to identify the English words and not to try to convert them to Arabic. We solved these issues using the 3ARRIB tool (Al-Badrashiny et al., 2014).

On the English side of the data, there were multiple translation options, where typically the first option was the intended meaning and the second one the literal meaning of the Arabic text. We chose to use the intended meaning only.

Finally, in all processing, for both Arabic and English, we made sure the emoticons were not affected by the tokenization or the translation.

2.3 Egyptian Arabic Segmentation

A major issue when training an SMT system for the present edition of the task is the small size of the provided SMS, CHT and CTS datasets, which means that data sparseness is a severe problem. One common way to reduce it, at least on the Arabic side, where it is more severe, is to segment the Arabic words into multiple tokens, e.g., by separating the main word from the attached conjunctions, pronouns, articles, etc. Since these are separate words in English, such a segmentation not only reduces sparseness, but also yields improved word mapping to English, thus ultimately helping word alignments and translation model estimation for SMT. The value of Arabic tokenization for SMT, especially under low resource conditions, has been demonstrated by a number of researchers in the past, e.g., (Badr et al., 2008; Habash et al., 2013; Al-Mannai et al., 2014).

We experimented with common segmentation schemes such as D3, S2 and ATB (Badr et al.,)
Table 2: Experiments in adapting Egyptian to look like MSA.

<table>
<thead>
<tr>
<th></th>
<th>SMS Test</th>
<th>TestG</th>
<th>CHT Test</th>
<th>TestG</th>
<th>CTS Test</th>
<th>TestG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Egyptian, no segmentation</td>
<td>21.02</td>
<td>21.64</td>
<td>20.27</td>
<td>22.34</td>
<td>20.60</td>
<td>23.36</td>
</tr>
<tr>
<td>Egyptian adapted to MSA, no segmentation</td>
<td>21.54</td>
<td>21.82</td>
<td>20.70</td>
<td>22.77</td>
<td>21.30</td>
<td>23.81</td>
</tr>
<tr>
<td>Egyptian adapted to MSA, then ATB-segmented</td>
<td>21.32</td>
<td>21.06</td>
<td>21.55</td>
<td>23.70</td>
<td>21.73</td>
<td>24.30</td>
</tr>
</tbody>
</table>

2008, ?). For tokenization, we used MADAMIRA (Pasha et al., 2014), a fast and efficient implementation of MADA for MSA (Habash and Rambow, 2005, ?), and MADA-ARZ, a version of MADA for Egyptian Arabic (Habash et al., 2013). Table 1 compares the results of using different segmentation schemes including no segmentation. We see gains of up to 3 BLEU points absolute on Test when using segmentation compared to no segmentation. However, the differences between the various schemes (ATB, S2 and D3) are small.

2.4 Egyptian to MSA Conversion

Another way to reduce data sparseness is by using additional out-of-domain data, e.g., newswire; this means a double domain shift: (i) from an informal text genre to newswire, and also (ii) from dialectal Arabic to MSA. While it is hard to do anything about the domain shift, the dialectal shift is somewhat easier to address. Changes between dialects are often systematic and many of the differences are at the level of individual words.

Previous work has shown that converting Egyptian to MSA makes it easier to use MSA resources for translating dialectal Arabic (?; Salloum and Habash, 2011; Zbib et al., 2012; Salloum and Habash, 2013; Sajjad et al., 2013a; Durrani et al., 2014a). So, we experimented with converting Egyptian to MSA using an in-house tool (Sajjad et al., 2013a), which performs character-level transformations for each Egyptian word in isolation to generate an MSA version thereof. We then trained an SMT system on this converted data.

The results are shown in Table 2. We can see that converting Egyptian to MSA yields improvements that are systematic across the three genres and also across the two test datasets. However, this improvement is not very large and ranges from 0.18 to 0.70 BLEU points absolute.

Further segmenting the MSA-like Egyptian using MADA ATB yielded very mixed results: in some cases, it added 0.93 BLEU points absolute, but in other there was a drop of 0.76. This could be because of the highly dialectal nature of the NIST data, which differs from MSA in lexical choice, which our tool cannot handle. Our Egyptian to MSA tool works at the character level and converts to MSA only those Egyptian words that are different at the character level.

In general, full conversion of dialectal Arabic to MSA would require not only word-level transformations but also phrase-level ones (?), while taking context into account (?), and also modeling morphological phenomena (?). There are also potential gains from a smarter character alignment models (?), or even from using a specialized decoder (?). Ultimately, the real benefit is when combining the adapted version of the smaller dialect with a large dataset in the bigger dialect (?; ?), which we will do below.

3 Translation System Characteristics

We started our experiments from a strong baseline system, which was originally designed for MSA to English translation (Sajjad et al., 2013b). We then extended it with some additional models and features (Durrani et al., 2014b). Most notably, we used minimum Bayes risk decoding (MBR) (Kumar and Byrne, 2004), monotone-at-punctuation reordering, dropping of out-of-vocabulary words, operation sequence model for reordering (OSM) (Durrani et al., 2011; Durrani et al., 2013), a smoothed BLEU+1 version of PRO for parameter tuning (Nakov et al., 2012), etc.

Given this baseline system, we experimented with several further extensions, which we will describe below. Some of them were eventually included in our final submission.

3.1 Genre-Based Hierarchically-Interpolated Language Model

First, we experimented with building a hierarchically-interpolated language model for each genre (i.e., CTS, SMS, and CHT). We tuned each model to minimize the perplexity on a held-out set for that target genre.

We examined the text resources that were available for training an English language model, and
we split them into six groups: (1) Egyptian-source (the target sides of the CTS, CHT and CTS training bi-texts), (2) MSA GALE News (GALE P3 \{R1,R2\}, P4 \{R1,2,3\}), (3) Chinese GALE (GALE P2 \{BC,BC,BL,NG\}), (4) MSA NEWS (newse-terir, news-par, news-trans, ISI), (5) MSA GALE non-news (GALE P1 \{BLOG\} P2 \{BC1, BC2, WEB\}), and (6) Gigaword v5, split into four subgroups by year (Guzmán et al., 2012) (1994-1997, 1998-2001, 2002-2005, 2006-2010).

For each such group, \(i\) we built an individual 5-gram language model with Kneser-Ney smoothing for each member of the group, and \(ii\) we interpolated these language models into a single language model, minimizing the perplexity for the target genre. Then, \(iii\) we performed a second (hierarchical) interpolation, this time combining the resulting six group language models, again minimizing the perplexity for the target genre. We used the SRILM toolkit (Stolcke and others, 2002) to build the language models.

Table 2? shows the results of using interpolated language model for each domain in comparison with using a language model built on the concatenation of the SMS, CHT and CTS corpora. We can see that sparse features only helped for CHT, while they were harmful for CTS, and yielded mixed results for SMS. Thus, in our final system, we only used them for CHT.

### 3.3 Class-Based Language Models

Next, we experimented with using automatic word clusters, which we computed on the source and on the target sides of the training bi-text using mkcls. We also experimented with OSM models (\(\text{OSM}\)) over cluster IDs (Durrani et al., 2014c; Bisazza and Monz, 2014). Normally, the lexically-driven OSM model falls back to context sizes of 2-3 operations due to data sparseness, but learning operation sequences over cluster IDs enabled us to learn richer translation and reordering patterns that can generalize better in sparse conditions.

Table 4 shows the experimental results when adding a target language model and an OSM model over cluster IDs. We can see that these class-based models yielded consistent improvements in all cases. We also tried using word2vec (Mikolov et al., 2013) for clustering, but the results did not improve any further and they were occasionally worse than those with mkcls. We tried both with 50 and 500 classes, but using additional classes did not help.

### 3.4 Unsupervised Transliteration Models

A consequence of data sparseness is that at test time, the SMT system would see many unknown or out-of-vocabulary (OOV) words. One way to cope with them is to just pass them through untranslated. This works somewhat fine with newswire text and for languages with (roughly) the same alphabet, e.g., English and Spanish, as many OOVs are likely to be named entities (persons, locations, organizations), and are thus likely to be
preserved in translation.

However, for languages with different scripts, such as Arabic and English, passing through is not a good idea, especially when translating into English as words in Arabic script do not naturally appear in English. In that case, it is much safer just to drop the OOVs, which is best done at decoding time; this was indeed our baseline strategy.

A better way is to transliterate OOV words either during decoding or in a post-processing step (7). We also experimented with this approach. For this purpose, we built an unsupervised transliteration model (Durrani et al., 2014) based on EM as proposed in (Sajjad et al., 2011; Sajjad et al., 2012). Unfortunately, it did not help much, probably because in these informal genres the OOVs are rarely named entities; they are real words, which need actual translation, not transliteration.

3.5 Neural Network Joint Language Model

Recently, neural networks have come back from oblivion with the promise to revolutionize NLP. Major performance gains have already been demonstrated for speech recognition, and there have been successful applications to semantics. Most importantly for us, last year, very sizable performance gains were also reported for SMT using a neural joint language model or NNJM (Devlin et al., 2014).

We tried the Moses implementation of JNLM using the settings described in (Birch et al., 2014). While we managed to achieve consistent improvements for all three genres and for both test sets, as Table 5 shows, these gains are modest, 0.04–0.57 BLEU points absolute, which is far from what was reported in (Devlin et al., 2014). It is unclear what the reasons are, but it could have to do with the small size of our training bitexts and the informal genres we are dealing with.

3.6 Domain Adaptation

We experimented with different techniques for domain adaptation, trying to combine bi-texts from different genres, e.g., our Egyptian SMS, CHT, and CTS, but also MSA newswire.

First, we experimented with concatenating our SMS, CHT and CTS bitexts for training, but then using genre-specific tuning sets; this did not work as well as some other alternatives. Next, we experimented with building separate phrase tables, one in-domain and one out-of-domain, and then (a) using phrase table backoff, or (b) merging phrase tables and reordering tables as in (7; 7; Sajjad et al., 2013b).

The results of our domain adaptation experiments when testing on SMS as the target genre are shown in Table 6. Note that the results on Test and on TestG differ a lot, and thus we focus on Test as it is much larger. We can see that the best way to combine SMS, CHT and CTS is simply to concatenate them, which yields +2.5 BLEU points of improvement absolute over training on SMS data only. Further small gains can be achieved by merging the resulting SMS+CHT+CTS phrase table with a phrase table trained on MSA, where the two tables are merged using extra indicator features as described in (7).

3.7 Tuning

Our phrase-based SMT system combines different features in a log-linear model. We tune the weights for the individual features of that model by optimizing BLEU (7) on a tuning dataset from the same genre as that in the test. We use PRO (7), but with smoothed BLEU+1 as proposed in (Nakov et al., 2012). We allowed the optimizer to run for up

---

**Table 5:** Experiments with class-based language models.

<table>
<thead>
<tr>
<th></th>
<th>SMS Test</th>
<th>TestG</th>
<th>CHT Test</th>
<th>TestG</th>
<th>CTS Test</th>
<th>TestG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>24.22</td>
<td>24.33</td>
<td>23.02</td>
<td>25.60</td>
<td>21.93</td>
<td>24.88</td>
</tr>
<tr>
<td>+ class-based models</td>
<td><strong>24.63</strong></td>
<td><strong>25.16</strong></td>
<td><strong>23.18</strong></td>
<td><strong>26.30</strong></td>
<td><strong>22.20</strong></td>
<td><strong>25.04</strong></td>
</tr>
</tbody>
</table>

**Table 6:** Experiments with a neural network joint language model.

<table>
<thead>
<tr>
<th></th>
<th>SMS Test</th>
<th>TestG</th>
<th>CHT Test</th>
<th>TestG</th>
<th>CTS Test</th>
<th>TestG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>24.58</td>
<td>24.33</td>
<td>24.02</td>
<td>27.11</td>
<td>22.64</td>
<td>24.95</td>
</tr>
<tr>
<td>+ NNJM Model</td>
<td><strong>25.01</strong></td>
<td></td>
<td>25.72</td>
<td>24.24</td>
<td>27.41</td>
<td>22.68</td>
</tr>
</tbody>
</table>

---
Table 7: Experiments with different training data combinations (bitext concatenations, phrase table merging, and backoff), testing on SMS as the target genre.

<table>
<thead>
<tr>
<th>Combination</th>
<th>Test</th>
<th>TestG</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS</td>
<td>21.30</td>
<td>21.99</td>
</tr>
<tr>
<td>CAT(SMS, CHT, CTS)</td>
<td><strong>23.78</strong></td>
<td>23.20</td>
</tr>
<tr>
<td>SMS, Backoff(CHT,CTS)</td>
<td>22.55</td>
<td>23.00</td>
</tr>
<tr>
<td>CAT(SMS, CHT), Backoff(CTS)</td>
<td>22.54</td>
<td>23.20</td>
</tr>
<tr>
<td>MergePT(CAT(SMS, CHT), CTS)</td>
<td>23.69</td>
<td><strong>24.40</strong></td>
</tr>
<tr>
<td>CAT(SMS, CHT, CTS), Backoff(MSA)</td>
<td>23.70</td>
<td>23.64</td>
</tr>
<tr>
<td>MergePT(CAT(SMS, CHT, CTS), MSA)</td>
<td><strong>23.83</strong></td>
<td>23.60</td>
</tr>
</tbody>
</table>

Table 8: Results using a length-filtered vs. an unfiltered dataset for tuning.

<table>
<thead>
<tr>
<th>Genre</th>
<th>Test</th>
<th>TestG</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS - unfiltered tune</td>
<td>23.57</td>
<td>23.56</td>
</tr>
<tr>
<td>SMS - filtered tune</td>
<td>23.35</td>
<td>24.36</td>
</tr>
<tr>
<td>CTS - unfiltered tune</td>
<td>22.07</td>
<td>25.10</td>
</tr>
<tr>
<td>CTS - filtered tune</td>
<td>22.70</td>
<td>25.22</td>
</tr>
</tbody>
</table>

Table 7: Experiments with different training data combinations (bitext concatenations, phrase table merging, and backoff), testing on SMS as the target genre.

4 Final Submission and Output Combination

We recombined hypotheses produced (a) by our best individual systems and (b) by other systems that are both relatively strong and can contribute diversity, e.g., using a different word segmentation scheme. For this purpose, we used the Multi-Engine MT system, or MEMT, (Heafield et al., 2009), which has been proven effective in such a setup and has contributed to achieving state-of-the-art results in a related competition in the past (Sajjad et al., 2013b).

The results are shown in Table ?? . We can see that using output combination yields notable improvement for SMS and CHT. However, for CTS, BLEU dropped by 0.45 points on Test. Thus, we submitted as a primary system the output combination for SMS and CHT, but our best individual system for CTS (which uses D3 segmentation).

5 Conclusion

We presented the Egyptian Arabic-to-English SMT system of the QCN team for the NIST OpenMT’2015 evaluation campaign. The system was ranked second in the competition on all three genres: SMS, chat, and speech.

Given the informal dialectal nature of these genres, we benefited from careful pre-processing, cleaning, and normalization, which yielded an improvement of up to 3 BLEU points over a strong baseline.

We further added a number of extra advanced features, which yielded 2.5 more BLEU points of absolute improvement on top of that due to pre-processing. In particular, sparse features contributed 0.7 BLEU points for CHT, class-based models added 0.7 and 0.6 BLEU points for CHT and SMS, respectively, and NNJM yielded gains of up to 0.4 BLEU points absolute.

References


<table>
<thead>
<tr>
<th></th>
<th>SMS Test</th>
<th>SMS TestG</th>
<th>CHT Test</th>
<th>CHT TestG</th>
<th>CTS Test</th>
<th>CTS TestG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best system with D3 segmentation</td>
<td>25.28</td>
<td>26.05</td>
<td>23.87</td>
<td>27.07</td>
<td>23.34</td>
<td>26.05</td>
</tr>
<tr>
<td>Best system with S2 segmentation</td>
<td>24.93</td>
<td>25.61</td>
<td>24.09</td>
<td>27.01</td>
<td>22.11</td>
<td>24.50</td>
</tr>
<tr>
<td>Best system with ATB segmentation</td>
<td>25.13</td>
<td>25.80</td>
<td>24.24</td>
<td>27.41</td>
<td>22.83</td>
<td>25.56</td>
</tr>
<tr>
<td>System with ATB segmentation + MSA phrase table as a backoff</td>
<td>25.20</td>
<td>25.04</td>
<td>23.48</td>
<td>26.16</td>
<td>23.01</td>
<td>25.67</td>
</tr>
<tr>
<td>Output combination</td>
<td>26.13</td>
<td>26.79</td>
<td>24.86</td>
<td>27.95</td>
<td>22.89</td>
<td>25.88</td>
</tr>
</tbody>
</table>

Table 9: Results for system recombination.


