Noname manuscript No.
(will be inserted by the editor)

Improving Arabic Neural Machine Translation via n-best list Re-ranking

Mohamed Seghir Hadj Ameur [1]
· **Ahmed Guessoum** [1]
· **Farid Meziane** [2]

Received: date / Accepted: date

Abstract Even though the rise of the Neural Machine Translation (NMT) paradigm has brought a great deal of improvement to the machine translation field, the current translation results are still not perfect. One of the main reasons for this imperfection is the decoding task complexity. Indeed, the problem of finding the one best translation from the space of all possible translations was and still is a challenging problem. One of the most successful ways to address it is via n-best list re-ranking which attempts to reorder the n-best decoder translations according to some defined features. In this paper, we propose a set of new re-ranking features that can be extracted directly from the parallel corpus without needing any external tools. The features set that we propose takes into account lexical, syntactic, and even semantic aspects of the n-best list translations. We also present a method for feature weights optimization that uses a Quantum-behaved Particle Swarm Optimization (QPSO) algorithm. Our system has been evaluated on multiple English-to-Arabic and Arabic-to-English machine translation test sets, and the obtained re-ranking results yield noticeable improvements over the baseline NMT systems.

Keywords Natural language Processing · Machine Translation · Neural Machine Translation · Quantum-behaved PSO

1 Introduction

Deep learning models have recently been proven to be very successful in various Natural Language Processing (NLP) applications such as Automatic Text Summarization (Chopra et al., 2016), Question Answering (Wang and Nyberg, 2015), Machine Translation (Cho et al., 2014a) and Document Classification (Kim, 2014). This success is due to the high capacity of these models to automatically learn important features from raw data without requiring any external knowledge (Collobert et al., 2011).

Mohamed Seghir Hadj Ameur
E-mail: mhadjameur@usthb.dz

Ahmed Guessoum
E-mail: aguessoum@usthb.dz

Farid Meziane
E-mail: f.meziane@salford.ac.uk

[1]Natural Language Processing and Machine Learning Research Group, Laboratory for Research in Artificial Intelligence, Computer Science Department, University of Science and Technology Houari Boumediene (USTHB), Algiers, Algeria

[2]School of Computing, Science and Engineering, University of Salford, M5 4WT, UK

For the task of Machine Translation (MT), an end-to-end approach called Neural Machine Translation (NMT) has recently been proposed by Bahdanau et al. (2014) and has been shown to achieve near human-level accuracy for several language pairs such as English-to-French and English-to-Spanish (Wu et al., 2016a).

Even though a great deal of improvement has been achieved in the field of MT, the current translation results are still not perfect. This is due to many factors, one of them being the decoding task complexity (or difficulty). Indeed, the decoder is not always capable of selecting the one best translation from the space of all possible translations (Freitag and Al-Onaizan, 2017). To this end, various solutions have been proposed. One of the most common methods are those which are based on a re-ranking process. Re-ranking can be seen as a two-pass procedure (Duh et al., 2010). First, the decoder is used to generate a list containing the top n best translations known as "the n-best list" [1]. Then a re-ranking methodology is used to select the best translation from the n-best list by re-ranking these translations according to a rich set of features. Re-ranking the decoder's n-best lists is an effective approach to improve the overall quality of the machine translation system for three reasons:

1. It allows the incorporation of various additional features to select the best translation from the n-best list.
2. The search space becomes significantly smaller given that it will be limited to only the translations found in the n-best list.
3. Re-ranking is a standalone model that can easily be incorporated into any other translation system.

In this work, we propose a set of sophisticated features that can be extracted solely from a parallel corpus without requiring any external linguistic tools. The features set that we have proposed covers the lexical, syntactic and semantic aspects of the translation candidates (the n-best list candidates) and can be grouped into five classes: (1) Translation-based Features: these features are related to a translation model and can be helpful for promoting translation adequacy; (2) Fluency Features: these features are used to promote syntactic fluency by incorporating language models; (3) Length-based Features: these features are used to promote the n-best list candidates according to the likelihood of their lengths; (4) N-best list Features: these features are extracted directly from the n-best list and are used to promote the most likely candidates in it; and (5) Embedding Features: these are based on bilingual word embeddings and are used to cover the semantic aspect of the translation candidates. For the problem of feature weights optimization, we present a methodology that is fairly similar to the one presented by Farzi and Faili (2015). It uses a Quantum-behaved Particle Swarm Optimization (QPSO) which guarantees the global convergence of the optimization process. Though our overall re-ranking framework is closely related to the one proposed by Farzi and Faili (2015), we diverge from them in various aspects:

- This work proposes several new re-ranking features mainly:
  - Position-based feature (section 5.1.1).
  - N-best alignment feature (section 5.1.2).
  - Word-to-word alignment feature (section 5.3.3).
  - Estimated length feature (section 5.2.1).
  - Global semantic similarity (section 5.5.1).
  - Alignment-based semantic similarity (section 5.5.2).
- All the features proposed in this work are language-independent, thus they can be used for any other languages.
- A new class of embedding-based features is proposed to take into account the semantic aspects of the translation candidates via bilingual word embeddings.
- Unlike Farzi and Faili (2015), our proposed objective function is based on the corpus-level, not the sentence-level, BLEU Score. Thus, feature weights are updated only if an improvement is achieved on the whole development corpus (section 4.2).

[1] Each translation in the n-best list is known as a translation candidate or a translation hypothesis.

– An in-depth investigation about features and classes impact is performed along with a discussion about their individual and combined effectiveness.
– This work focuses on two MT directions: English-to-Arabic and Arabic-to-English, in which very few n-best list re-ranking studies are performed.

The remainder of this paper is organized as follows. Section 2 presents the related work. The necessary background information needed to understand this work is provided in Section 3. Section 4 explains the overall system design. The details concerning all the incorporated reordering features are provided in Section 5. The evaluation methodology and all the experimentations are provided in Section 6. Section 7 gives a brief discussion about the obtained results. Finally, Section 8 gives a conclusion as well as a listing of some possible research directions.

2 Related Work

In this section, we first highlight some of the main NMT challenges. Then, we present the most important research studies that have been accomplished in regard to n-best list re-ranking.

2.1 Main NMT Challenges

Koehn and Knowles (2017) performed a wide range of experiments on several NMT systems to accurately identify their weaknesses. They reported the six most important ones as follows [2]:

1. NMT may produce lower translation quality when dealing with out-of-domain data, as they can completely sacrifice adequacy for the sake of fluency.
2. NMT system performance correlates strongly with the amount of data that are used for training. Indeed, the NMT models often produce low-quality translation under low-resource settings and higher performance under high resource ones.
3. Sub-word units (e.g. with byte-pair encoding) can simulate an open vocabulary with a fixed-size one, thus effectively handling the OOV words; yet they are still unable to accurately translate highly-inflected categories (e.g. verbs).
4. Even when using the attention mechanism, the NMT system translation quality still decreases when dealing with very long sentences (more than 60 words).
5. The attention mechanism in NMT may dramatically diverge.
6. A beam search decoder gives good translation results for smaller beam sizes; however, its performance deteriorates when dealing with a larger search space.

2.2 Re-ranking Research Studies

Many interesting ideas have been proposed to address the problem of n-best list re-scoring in MT. In the following, we will try to categorize the most important methods that have been proposed according to the way in which they address the re-ranking problem.

2.2.1 Re-ranking by Optimizing the Decoder Feature Weights

One research direction investigated the possibility of replacing the linear decoder scoring method with a more efficient one. Arun and Koehn (2007) investigated discriminative training of a phrase-based SMT using millions of features for the task of n-best list re-ranking. Their model parameters were optimized using two online learning algorithms, the structured perceptron and Margin-Infused Relax Algorithm (MIRA). Their experiments on the Czech-English

[2] For more details about the specifications of each weakness please refer to the original paper of Koehn and Knowles (2017).

translation task showed that the two methods produce very similar results while the perceptron has a more rapid convergence. Duh and Kirchhoff (2008) presented a boosting algorithm which they called BoostedMERT; it uses Minimum Error Rate Training (MERT) to boost the BLEU score on the n-best list re-ranking task. The results they reported on the IWSLT 2007 Arabic-to-English translation task showed an absolute improvement of 0.8 BLEU points over the baseline phrase-based statistical MT system. In their later work, Duh et al. (2010) addressed the n-best list re-ranking as a multi-task learning problem in which each n-best list is taken as a distinct task. First, they used a meta-algorithm that discovers the common feature representations across the n-best lists via multi-task learning. Then they used a conventional re-ranker to reorder the n-best list. They reported a 0.5 improvement in the overall BLEU score on the English-to-Japanese translation task. Sokolov et al. (2012) proposed an approach that uses a non-linear scoring function instead of the conventional phrase-based SMT linear scoring function via a Boosting algorithm. The experiments they carried out on the WMT10, WMT11 and WMT12 test sets resulted in a slight performance boost of about 0.4 BLEU points.

2.2.2 Re-ranking by Including Additional Features

The simplest way to improve the translation output is to include additional language models and use them to select the best translation from the n-best decoding list. Following this research direction, Kirchhoff and Yang (2005) investigated the effect of including additional language models on n-best list re-scoring. They used a 4-gram word-based language model with a modified KneserNey smoothing and interpolation, and a factored feature-based trigram language model. Their experiment results reported on the ACL05 Shared MT task for four language pairs (translation from Finnish, German, Spanish and French into English) showed that using additional language models did not result in a significant increase in the overall phrase-based SMT performance. Carter and Monz (2010) applied a large-scale discriminative language model to re-rank the n-best list translations generated by an SMT system. They reported an improvement of up to 0.4 BLEU points on the NIST Arabic-to-English translation benchmarks. Luong and Popescu-Belis (2016) proposed a method to better handle the translation of pronouns between English and French. They used a Pronoun-aware Language Model (PLM) which encodes the likelihood of generating a target pronoun given the gender and number of the nouns preceding it. They combined the phrase-based SMT (PSMT) decoder score and their PLM model score to re-rank the translation candidates and reported a 5% relative accuracy improvement in pronouns prediction over the PSMT baseline.

Various researchers used several linguistic features to boost the performance of a translation system as a standalone post-processing phase. Following this direction, Och et al. (2004) presented a method for n-best list re-ranking that uses a large number of features with different levels of syntactic representation. The features were combined using the log-linear model and their weights were optimized directly against the BLEU score using a Minimum Error Rate Training (MERT) on held-out data. They reported a significant improvement of 1.3% BLEU score on the task of Chinese-to-English translation. Hasan et al. (2007) investigated the usefulness of increasing the size of the n-best list produced by a statistical machine translation system. They showed that although it is possible to generate many distinct translation candidates, starting from a certain value of n, the increase in the overall system performance will become very minimal. They showed that going above n = 100 will not yield a significant improvement while noticeably increasing the decoding time. Specia et al. (2008) used Word Sense Disambiguation (WSD) features to re-rank the n-best list translations generated by a statistical machine translation system. Experiments with English-to-Portuguese translation showed a significant improvement that varied between 1.5 and 2.5 absolute BLEU points. Farzi and Faili (2015) used a set of non-syntactical features to re-rank the n-best translation candidates generated by a Phrase-based Statistical Machine Translation system. They investigated several feature weights optimization algorithms such as Particle Swarm Optimization (PSO), Quantum-behaved Particle Swarm Optimization (QPSO), Genetic Algorithms (GA), Perceptron and Averaged Perceptron. They reported an improvement of 1.09 and 1.73 points in BLEU score for English-to-Persian and German-to-English respectively. They concluded that QPSO

was better suited for weight optimization than the other investigated alternatives. Tong et al. (2016) investigated the use of new semantic and syntactic features in the re-ranking framework. The representations they used are basically sentence-embeddings that are learned using the recursive auto-encoder (RAE). They evaluated their proposal on the WMT2015 French-to-English translation data and reported a very noticeable improvement of about 1.23 BLEU points over the baseline SMT system.

2.2.3 Re-ranking via System Hybridization

Some researchers tried to re-rank the n-best list via the hybridization of different types of MT systems. In this spirit, Xiao et al. (2013) proposed an ensemble learning-based approach in which they first generate an ensemble of weak translation systems from a single SMT engine, and then they learn a strong translation system from that ensemble. One of their proposed system combination methods is a sentence-level one in which the best translation is selected from the union of all the weak translation systems n-best lists. They tested their approach on the NIST Chinese-to-English translation task and reported a significant improvement of all the three state-of-the-art statistical MT systems that they tested (phrase-based system, hierarchical phrase-based system, and syntax-based system). Neubig et al. (2015) described the results of applying a neural MT re-ranking to a baseline syntax-based MT system. Their tests on the WAT2015 translation task between English, Japanese and Chinese yielded a noticeable improvement in the overall BLEU Score result over the baseline system. Stahlberg et al. (2016) investigated the use of hierarchical phrase-based SMT lattices in an end-to-end NMT. They evaluated their proposed system on the English-to-German and English-to-French WMT 2014 news tests and found that the hybridization yielded a noticeable improvement over the individual models. Zhang et al. (2017) proposed a method that uses an existing phrase-based translation model to compute the phrase-based decoding cost for a given NMT output, then they used that decoding cost to re-rank the n-best list generated by their NMT system. They tested their proposal on several language pairs: English-to-Chinese, English-to-Japanese, English-to-German, and English-to-French. They reported some noticeable improvements over their NMT baseline.

2.2.4 Re-ranking by Improving the Decoder Search Strategy

A few studies attempted to address the main drawback of the beam search (BS) decoder which is the lack of diversity of its generated candidates. Indeed, the n-best list generated by means of a BS are generally very similar and differ slightly from each other. To address this problem, a branch of research has focused on diversifying the decoder n-best list candidates. Vijayakumar et al. (2016) proposed a Diverse Beam Search (DBS) decoder for neural machine translation (NMT) as an attempt to generate more diverse n-best list candidates than the ones that can be obtained from a classical beam search decoder. Their proposal optimizes a diversity-augmented objective function that divides the beam search space into smaller groups and promotes the diversity between them. Their test results on an English-German news test dataset showed a gain of up to 0.6 points in BLEU Score over the classical beam search decoder. Li and Jurafsky (2016) proposed a diversification heuristic that prevents the beam search decoder from producing highly similar candidates, thus, implicitly increasing the diversity of the produced n-best list. Their test results on WMT German-to-English and French-to-English translation tasks showed a consistent performance boost over their NMT baseline system. Some other studies attempted to recombine the hypothesis generated by a BS decoder to create new ones. Zhang et al. (2018) introduced a recombination method for NMT decoding based on the equivalence of partial hypotheses. They used an n-gram suffix-based heuristic approximation to determine partially equivalent hypothesis in the beam search space. They tested their proposal on two translation tasks: NIST Chinese-English and WMT English-German and reported a very small gain in BLEU score on both tasks. Tromble et al. (2008) presented a Minimum Bayes-Risk (MBR) decoding over a translation lattice that can encode a large number of translation candidates in a compact way. Their tests on Arabic-to-English, Chinese-to-English

and English-to-Chinese translation tasks showed moderate gains in translation performance over the classical N-best MBR decoding.

3 Background

This section gives a brief overview of the functioning mechanism of Neural Machine Translation from a theoretical standpoint and also introduces some concepts that will be important for a better understanding of this work.

3.1 Neural Machine Translation

We focus on the attention-based Neural Machine Translation (Bahdanau et al., 2014) which we will be using as our baseline system. Given a source sentence $X = (x_1, x_2, ..., x_d)$ and a target sentence $Y = (y_1, y_2, ..., y_{d'})$, where each x_t and y_t represent the source and target words at time-step t, d and d' represent the maximum source and target sentence lengths respectively [3]; the attention-based Neural Machine Translation (NMT) estimates the conditional probability of generating the target sentence Y given the source sentence X as $P(Y = (y_1, y_2, ..., y_{d'})|X = (x_1, x_2, ..., x_d))$.

The NMT architecture involves two components: an encoder and a decoder. The encoder is usually a bidirectional Recurrent Neural Network (bRNN) (Schuster and Paliwal, 1997) that reads the input sentence word by word from left-to-right (direct direction Eq. 1) and from right-to-left (reversed direction Eq. 2) :

$$\overrightarrow{h_t} = \Phi_{enc}(\overrightarrow{h_{t-1}}, x_t) \tag{1}$$

$$\overleftarrow{h_t} = \Phi_{enc}(\overleftarrow{h_{t-1}}, x_t) \tag{2}$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the hidden states at time-step t generated by the direct and reversed recurrent neural networks respectively. This is done by taking into consideration the previous hidden state h_{t-1} at time-step $t-1$ and the current input word x_t at time-step t. Φ_{enc} is the recurrent activation function responsible for combining the previous hidden state with the current input word. In practice the function Φ_{enc} is usually implemented as a Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or a Gated Recurrent Unit (GRU) (Cho et al., 2014b). Applying this process to the whole input sentence produces a direct encoder hidden representation $\overrightarrow{H} = (\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_d})$ and a reverse encoder hidden representation $\overleftarrow{H} = (\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_d})$ where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ represent the direct and reversed encoder hidden states at time-step t respectively. The direct and reversed hidden states will be concatenated at each time-step t to form what is called the "annotation vector" $H = (h_1, h_2, ..., h_d)$, where each single annotation h_t at time-step t (Eq. 3) is a tuple:

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{3}$$

Each annotation h_t conserves the information about the word at position t along with all the words surrounding it (its left and right contexts).

The decoder is generally a unidirectional recurrent neural network that uses an attention mechanism to select a target word at each time-step t based on its annotation vector. This is done by attributing a relevance weight α_{tj} to each annotation h_j via a feed-forward neural network that takes the annotation h_j, the previous output y_{t-1} and the previous decoder hidden state s_{t-1} to produce an output e_{tj} as shown in Eq. 4.

$$e_{tj} = f(s_{t-1}, h_j, y_{t-1}) \tag{4}$$

[3] Generally a padding process is used to pad all the sentences into the same length.

The function f is a dense neural network with one hidden layer. The output e_{tj} will then be normalized via Eq. 5.

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{d}(exp(e_{tk})} \tag{5}$$

The normalized scores are then used to compute the weighted sum of the annotation vectors (Eq. 6)

$$c_t = \sum_{j=1}^{d} \alpha_{tj} h_j \tag{6}$$

Then the decoder neural network can update its own hidden state using the encoder weighted sum of the annotation vectors c_t, the decoder previous hidden state s_{t-1} and the decoder previous output word y_{t-1}:

$$s_t = \Phi_{dec}(s_{t-1}, y_{t-1}, c_t) \tag{7}$$

where Φ_{dec} can be implemented as an LSTM or a GRU unit in a similar way to what we have seen for the encoder. The NMT model is trained to maximize the probability of generating the target sentence when given the source sentence in an end-to-end learning manner on the basis of a substantially large parallel corpus via the backpropagation algorithm (Goller and Kuchler, 1996).

3.2 Beam Search Decoder

Beam Search (BS) (Russell and Norvig, 2016; Koehn, 2009) is a heuristic search algorithm which can be seen as an improvement to the classical Greedy Search (GS) algorithm. Unlike the GS algorithm which keeps track of only the best next step as the solution sequence is constructed, the BS algorithm expands all possible next steps (tokens) and choses the B most likely ones resulting in B best partial solutions; where B is the unique parameter of the BS algorithm known as the beam width.

In the case of NMT, given a source sentence X, the BS algorithm can be used to generate the B best target translations of X. At each time step t, the BS decoder expands each of the B partial candidates with all the possible words $y_t \in V$, where V is the target vocabulary. Each newly constructed partial candidate $C_t = \{y_1, y_2, ..., y_t\}$ will be scored using Eq. 8.

$$P(C_t) = P(y_1, y_2, ..., y_t|X) = P(y_1, .., y_{t-1}|X) * P(y_t|X, y_1, .., y_{t-1}) \tag{8}$$

Where $P(y_t|X, y_1, .., y_{t-1})$ is the probability of generating the target word y_t at time step t by the NMT decoder. The B partial solutions with the most likely probabilities will be selected at each time step t. This same process will be repeated until the end of the sequence is reached.

3.3 Minimum Bayes Risk

The Minimum Bayes Risk (MBR) is a decoding method that finds the candidate with the least expected loss (González-Rubio et al., 2011; Shu and Nakayama, 2017; Kumar and Byrne, 2004)[4]. The Bayes risk of a given candidate y can be estimated using Eq. 9:

$$R(y) = \sum_{y' \in E} \triangle(y, y') P(y'|x) \tag{9}$$

where E refers to the evidence space [5], $P(y'|x)$ is the probability of generating the candidate y' as the translation of the source sentence x by the NMT decoder, and $\triangle(y, y')$ is the level of discrepancy between the two candidates y and y' which can be estimated using Eq. 10.

$$\triangle(y, y') = 1 - SBLEU(y, y') \tag{10}$$

[4] We follow the MBR re-ranking method given by González-Rubio et al. (2011).

[5] In this work the list of n-best list candidates is considered as the evidence space.

The function $SBLEU$ refers to the third smoothed sentence-level similarity metric proposed by (Chen and Cherry, 2014). The intuition behind this method is to select the candidate sharing the highest similarity with the candidates of the evidence space E (González-Rubio et al., 2011).

4 System Design

The global architecture of our system is presented in Figure 1. Its functioning mechanism involves two main consecutive steps: first, a re-ranking model is built; then it is used to re-rank the n-best list translation candidates generated by the NMT decoder.

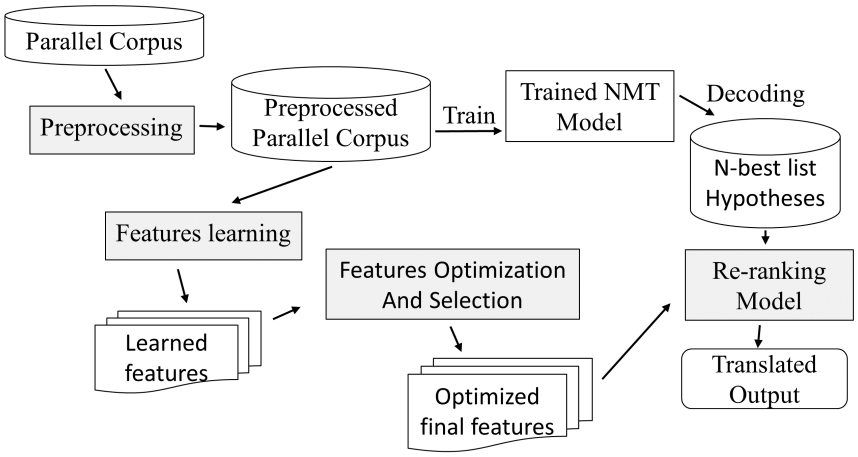

Fig. 1: Global Architecture of the proposed Re-ranking System

As shown in Figure 1, we first start by applying a preprocessing step to both the source and target sentences that are found in the parallel corpus. Then, a re-ranking model is built by using a set of predefined sentence-level linguistic features which are directly learned and optimized from the preprocessed parallel corpus. Finally, the re-ranking model is used to reorder the n-best candidates list generated by the NMT baseline decoder.

For the remaining of this section, we start by explaining in more detail the functioning mechanism of our re-ranking system. Then, we explain the feature weights optimization task along with the swarm-based algorithm that we incorporate to solve it.

4.1 The Re-ranking Process

Our final goal is to re-rank the n-best list candidates in a better way that allows us to pick the best translation among them. To do so a set of sentence-level features is defined; each feature will be used to assign a score in the interval [0, 1] to each individual candidate (translation) in the n-best list. We denote the n-best translations list of a given source sentence x by $H(x) = (h_1, h_2, ..., h_n)$ where n is the size of the n-best list and h_i is the i^{th} candidate translation of sentence x. We denote the features set by $F = \{f_1, f_2, ..., f_k\}$ where k is the number of features and each feature f_i is a function that takes as argument a translation candidate from the n-best list and assigns a probability to it. For example, $f_1(h_{12}) = 0.9$ means that the score of the 12^{th} candidate in the n-best list $H(x)$ according to the first feature f_1 is equal to 0.9.

By scoring each translation candidate using all the features defined in F, each candidate from the n-best list $H(x)$ will end up with multiple scores, one score per feature. An example is given in Table 1.

Table 1: An Example showing the process of scoring the n-best list candidates via a set of predefined features

$H(x)$	f_1	f_2	f_3	...	f_k
h_1	0.38	0.11	0.25	...	0.14
h_2	0.17	0.28	0.98	...	0.65
...	...	...	...	...	...
h_n	0.49	0.32	0.78	...	0.55

To re-rank the n-best list candidates we need to assign a single score to each candidate and re-rank them according to it. To this end, we need to define a way of combining all the individual feature scores assigned to a given candidate (Table 1) into a single score. One possible way to do that is to use a weighted linear combination method as follows:

$$S(h_i) = w_1 f_1(h_i) + w_2 f_2(h_i) + ... + w_k f_k(h_i) \tag{11}$$

$$with \qquad \sum_{j=1}^{k} w_j = 1 \tag{12}$$

where $w_j \in W$ is the weight (coefficient) of the j^{th} feature of F, h_i is the i^{th} candidate in the n-best list $H(x)$, and the total sum of all the feature weights w_j is always equal to one (Eq. 12). Then, the best translation of the source sentence x is selected from the n-best list $H(x)$ by simply picking the translation candidate that has the highest score as shown in Eq. 13.

$$best_{trans}(x|\ W = \{w_1, .., w_k\}) = argmax_{(h_i \in H(x))} S(h_i) \tag{13}$$

Using Eq. 13 we can determine the best translation candidate (the one with the highest score) from the n-best list of the source sentence x when the weight of each feature is provided.

4.2 The Task of Feature Weights Optimization

Our goal is to find the optimal feature weights vector $W = \{w_1, w_2.., w_k\}$ that gives the best re-ranking results. To this end, we use a development corpus which contains a set of source sentences $X = \{x_1, x_2, ..., x_t\}$, where each sentence $x_i \in X$ is associated with its n-best translation list $H(x_i) = (h_1, h_2, ..., h_n)$ generated by the NMT decoder. We define our objective function as a corpus-based BLEU Score (Papineni et al., 2002) that considers the first-best translation (using Eq.13) from each n-best list $H(x_i)$. Changing the feature weights will automatically lead to a change in the order of the n-best list candidates which will in turn lead to a change in the corpus-based BLEU Score of the development corpus as shown in Figure 2.

As shown in Figure 2, if we do not consider any feature by setting all the weights to zero (the top-right of the figure), then the original decoder ranking remains the same. Modifying the feature weights (the second and the third examples in the same figure) changes the ranks of the n-best list translation candidates; thus changing the overall corpus-based BLEU Score. This highlights the search space which consists of all the possible combinations of feature weight values. Each one of them can lead to a different BLEU Score on the development corpus.

4.3 Quantum-behaved Particle Swarm Optimization

Many methods have been proposed to tackle the feature weights optimization problem, such as the Perceptron algorithm (Carter and Monz, 2011), Particle Swarm Optimization and Quantum-behaved Particle Swarm Optimization (Farzi and Faili, 2015). The later work of

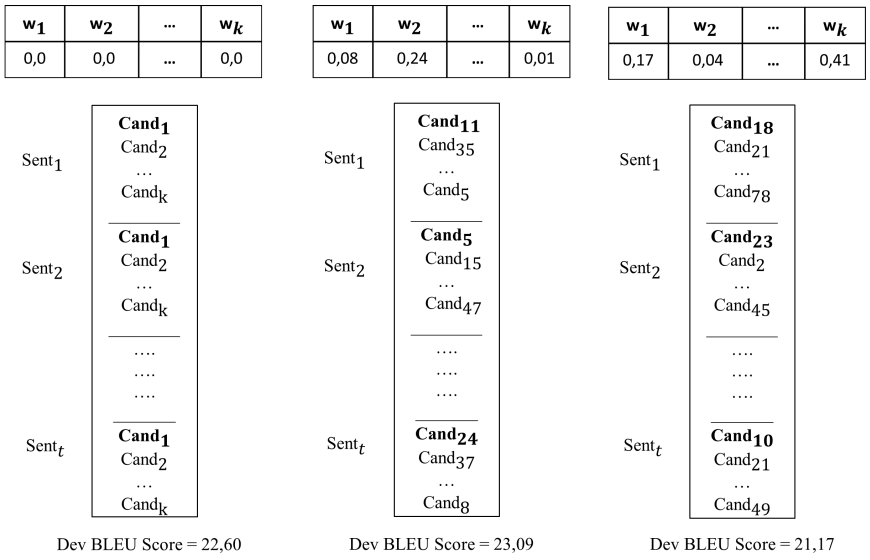

Fig. 2: The functioning mechanism of the weight-optimization process in the re-ranking system

Farzi and Faili (2015) reported that QPSO gives better re-ranking results in comparison to the remaining mentioned algorithms. Motivated by the work of Farzi and Faili (2015) we also make use of the QPSO algorithm.

QPSO (Sun et al., 2004) uses a swarm of m particles which attempt to find the optimal solution in a d-dimensional space. Each particle i is characterized by its position vector $X_i(t) = \{x_{i1}(t), x_{i2}(t), ..., x_{id}(t)\}$ and its best previous position *pbest* "Personal best". The information about the global best position $gbest(t)$ achieved by the m particles at each time t is also kept. Each particle moves according to the following equation:

$$X_i(t+1) = \begin{cases} P_i + \beta \cdot (mbest - X_i(t)) \cdot ln(\frac{1}{u}),\ if\ h > 0.5 \\ P_i - \beta \cdot (mbest - X_i(t)) \cdot\ ln(\frac{1}{u}),\ otherwise \end{cases} \tag{14}$$

$$P_i = \varphi\ pbest_i + (\varphi - 1)\ gbest_i \tag{15}$$

$$mbest = \sum_{i=1}^{m} \frac{pbest_i}{m} \tag{16}$$

where P_i is the "Inclination Point" which combines the personal best *pbest* and the global best *gbest* as shown in Eq. 15, *mbest* is the "Mean Best Position"; the center of gravity of all particles' best positions, φ, h and u are random numbers uniformly distributed on [0, 1], and β known as the "Contraction Expansion Coefficient" is the unique parameter of the QPSO algorithm which is used to control the convergence speed of the algorithm [6].

The high-level steps that explain the functioning mechanism of the QPSO-based weights optimization algorithm that we have incorporated in our re-ranking system are presented in Algorithm 1. The steps that we used are similar to the ones proposed by Farzi and Faili (2015). We take as input the development corpus and a set of features, then we find the optimal set of feature weights that gives the highest corpus-based BLEU Score on the development set (as explained earlier in Figure 2).

The algorithm starts by randomly initializing the position vectors that contain the feature weights for each particle. Then it updates the personal and global bests according to the

[6] The value of the contraction-expansion coefficient parameter is generally set to 0.75 as recommended by Sun et al. (2012).

Algorithm 1: Feature weights optimization algorithm

Input : A parallel development corpus
A set of features $F = \{F_1, F_2, ..., F_k\}$
Output: The optimal feature weights
Pseudo Algorithm:
begin
- For each particle i in the population, randomly initialize its position vector W_i (the feature weights vector) and set its personal best $pbest_i$ to W_i.
while *no termination condition is met* **do**
- Estimate the corpus-based BLEU score (objective function) based on the position vector W_i of each particle i in the population (as shown in Section 4.2).
- Update the personal best $pbest_i$ of each particle i, then, update the global best $gbest$ of the whole population.
- For each particle, update its position vector W_i (Eq. 14).
end
- Return the optimal feature weights of the global best $gbest$.
end

BLEU Score achieved by each particle in the swarm. Then the particles move toward the optimal solution using Eq. 14 and Eq. 15. This process is repeated until one of the following termination conditions is met:

1. Reaching the maximum number of iterations.
2. Reaching the maximum execution time limit.
3. Meeting the early stopping condition, which applies if no improvement is achieved for a certain number of iterations.

5 Proposed Features

As we mentioned earlier, we do not want to make the NMT system dependent on any language-specific tool. As such, we have tried to avoid using any external NLP tools (such as part-of-speech taggers, named entity recognizers, parsers, etc.) and we have used only the features that can automatically be extracted from the bilingual corpus. We have proposed five categories (classes) of features, each one of them containing a set of sophisticated features which serve different purposes.

For the remainder of this section, the following notations will be respected. The source English sentence is denoted by x with d being its length. The n-best translation list of x is denoted by $H(x) = (h_1, h_2, ..., h_n)$ where h_i is the i^{th} Arabic translation candidate.

To illustrate the functioning mechanism of each feature we will consider the following English source sentence[7]:

$x =$ "something stiffened inside me"

Let us assume that the NMT decoder generated an n-best list that contains three candidates $H(x) = (h_1, h_2, h_3)$ as follows:

- $h_1 =$ "هناك شيء ب داخل ي"
- $h_2 =$ "هناك شيء تشدد ب داخل ي"
- $h_3 =$ "هناك شيء ما في ال داخل"

We also consider that the NMT decoder alignments for those three candidates h_1, h_2 and h_3 are the ones provided in Figure 3. We note that the Arabic language candidates are written from left-to-right only to match the English language writing direction as indicated by the numbering of their words (Figure 3).

We denote the decoder alignment concerning the source sentence x and its candidate translation h by $Alig(x, h) = \{A_s(h[1]), A_s(h[2]), ..., A_s(h[t])\}$ where t is the length of the candidate

[7] We note that this example along with the notation will be used throughout this section.

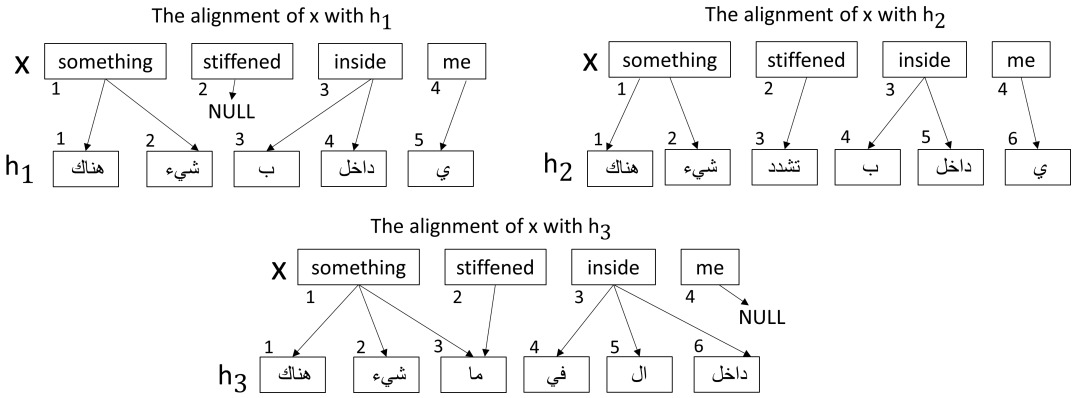

Fig. 3: The NMT decoder source-to-candidate alignments for all the n-best list candidates

h and $A_s(h[j])$ is a list of all word positions of x that are aligned to the word $h[j]$ found at position j of h. For example, $A_s(h_3[3]) = A_s(\text{ما}) = [1, 2]$ because the first and second words of the source sentence x are aligned to the third word of the candidate h_3. We note that we have obtained these word-to-word alignments from the NMT decoder source-to-candidate attention weights. For each source word from the source sentence we select the target word that has the highest attention weight as its alignment. An example demonstrating how word-to-word alignments are extracted from the NMT decoder attention weights is provided in Fig. 4.

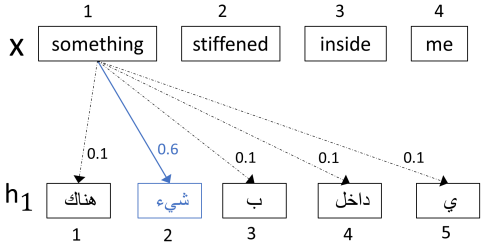

Fig. 4: An example showing the process of extracting word-to-word alignments from the NMT decoder attention weights

The first source word "something" is aligned with the target word "شيء" given that it has the highest attention weight 0.6. This same principle is applied to the remaining words of x to obtain their word-to-word alignments.

5.1 N-best list Features

This class of features relies on the knowledge that can be extracted from the n-best list to promote the most promising candidates in it. It includes two features: Position-based Score and N-best Alignment Score.

5.1.1 Position-based Score

This feature is a generalization of the one proposed by Farzi and Faili (2015). Farzi and Faili (2015) presented a feature that promotes the candidates having the most probable word distribution in the n-best list. First, they estimate $P(h, i)$ the probability of appearance of the i^{th}

word of candidate h in the i^{th} position of h:

$$P(h,i) = \frac{\sum_{h' \in H(x)} E(h[i], h'[i])}{n} \tag{17}$$

where $E(h[i], h'[i])$ is the equality function that returns 1 if $h[i] = h'[i]$ and 0 otherwise. Then the total probability of candidate h is estimated as the product of the probability at each position i in h using Eq. 18:

$$P(h) = \prod_{i=1}^{t} P(h,i) \tag{18}$$

With respect to this feature we propose a generalization of that defined in (Farzi and Faili, 2015), in that we consider n-grams (up to four grams) instead of unigrams. Thus, we propose the following new equation instead of Eq. 17:

$$P(h,i) = \sum_{k=1}^{4} \frac{1}{k} \frac{\sum_{h' \in H(x)} E(h[i:i+k], h'[i:i+k])}{n} \tag{19}$$

Equation 19 considers the case of n-grams with $k \in \{1,2,3,4\}$ (for $k = 1$, this equation is exactly the same as Eq. 17). To estimate the final probability of candidate h (the score of candidate h), we make use of the same equation (Eq. 18) proposed by Farzi and Faili (2015).

Example 1 Following the example given at the beginning of Section 5, to estimate the probability of the first candidate h_1 according to Farzi and Faili (2015) (Eq. 17), we first estimate $P(h_1, i)$, the probability of appearance of the i^{th} word of h_1 in the i^{th} position of h_1, as follows:
$P(h_1, 1) = \text{P}(\text{هناك}) = 3/3 = 1$
$P(h_1, 2) = \text{P}(\text{شيء}) = 3/3 = 1$
$P(h_1, 3) = \text{P}(\text{ب}) = 1/3$
$P(h_1, 4) = \text{P}(\text{داخل}) = 1/3$
$P(h_1, 5) = \text{P}(\text{ي}) = 1/3$

Then, the final probability of h_1 is estimated as the product of these quantities:
$P(h_1) = P(h_1, 1) * P(h_1, 2) * ... * P(h_1, 5) = 1/27$

The only difference in our proposed feature is that we use n-grams instead of unigrams, thus we estimate the probability of finding the i^{th} n-gram of h_1 in the i^{th} position of h_1 as follows:
$P(h_1, 1) = \frac{1}{4}\ \text{P}(\text{هناك}) + \frac{1}{4}\ \text{P}(\text{هناك شيء}) + \frac{1}{4}\ \text{P}(\text{هناك شيء ب}) + \frac{1}{4}\ \text{P}(\text{هناك شيء ب داخل})$
$P(h_1, 1) = \frac{1}{4} * \frac{3}{3} + \frac{1}{4} * \frac{3}{3} + \frac{1}{4} * \frac{1}{3} + \frac{1}{4} * \frac{1}{3} = 2/3$
$P(h_1, 2) = ...$
...
$P(h_1, 5) = ...$

Then, the final probability of h_1 is estimated as we have done previously:
$S(h_1) = P(h_1) = P(h_1, 1) * P(h_1, 2) * ... * P(h_1, 5)$

As illustrated in this example, the generalization that we have proposed in this feature allows us to make decisions based on the likelihood of appearance of entire segments (n-grams) on a given position instead of solely relying on individual words. We believe that this addition increases the usefulness of this feature as it takes into consideration the context of each word (the words that follow it) at each given position of the translation candidate.

5.1.2 N-best Alignment Score

The second feature that we propose in this class promotes the candidates that have the most probable alignments in the n-best list. First, for each word $h[j]$ found at position j in h we estimate the correctness of its source-words alignments list $A_s(h[j])$ based on its likelihood of appearance in the n-best list, as given in Eq. 20:

$$P(h,j) = \frac{\sum_{h' \in H(x)} E(A_s(h[j]), A_s(h'[j]))}{n} \tag{20}$$

where $E(A_s(h[j]), A_s(h'[j]))$ is the equality function that returns 1 if the list of source word positions aligned to $h[j]$ is equal to that of $h'[j]$, and 0 otherwise. Then the final probability (the score) of the candidate h is estimated as the product of the probabilities that we calculate for each position j in h:

$$S(h) = P(h) = \prod_{j=1}^{t} P(h,j) \tag{21}$$

Example 2 With the same example introduced at the beginning of this section, to estimate the probability of the first translation candidate h_1, we first estimate $P(h_1, j)$ the likelihood of having the j^{th} word of h_1 aligned to the list of positions $A_s(h[j])$ in the n-best list. This is done as follows:
$P(h_1, 1) = \frac{3}{3} = 1$, we obtained this probability because the alignment link found between the first word of h_1 and the first word of x is also present in the two other candidates h_2 and h_3.
$P(h_1, 2) = \frac{3}{3} = 1$, we obtained this probability for the same reason as for $P(h_1, 1)$.
$P(h_1, 3) = \frac{1}{3}$, we obtained this probability because the alignment link found between the third word of h_1 and the third word of x is not present in any other candidate.
And in a similar way we estimate the remaining probabilities:
$P(h_1, 4) = \frac{3}{3} = 1$
$P(h_1, 5) = \frac{1}{3}$
Then, the final probability of h_1 is estimated as the product of the probabilities:
$S(h_1) = P(h_1) = P(h_1, 1) * P(h_1, 2) * ... * P(h_1, 5) = 1/9$

5.2 Length-based Features

This class of features is used to promote a candidate translation based on the likelihood of its length. Thus, a candidate in the n-best list having a highly probable length will be given a high score and the one having a less probable length will be penalized (by giving it a lower score).

5.2.1 Estimated Length Score

The first feature in this class of length-based features is used to estimate the probability of generating a target translation h of length t for a given source sentence x of length d. This is done by estimating the likelihood of finding a source sentence of length d aligned to a target sentence of length t in the parallel corpus as shown in Equation 22.

$$S(h) = P(h) = p(len(h) = t \mid len(x) = d) = \frac{c(d,t)}{c(d)} \tag{22}$$

where $c(d)$ is the number of times a sentence of length d appears in the source side of the parallel corpus, and $c(d,t)$ is the number of times a source sentence of length d is found aligned to a target sentence of length t in the parallel corpus.

Example 3 Using the example introduced at the beginning of Section 5:
$d = len(x) = 5$
$t_1 = len(h_1) = 5, \quad t_2 = len(h_2) = 6, \quad t_3 = len(h_3) = 6$

Let us assume that the statistics that we get from the parallel corpus tell us that: $c(d = 5, t = 5) = 2540$, $c(d = 5, t = 6) = 1200$ and $c(d = 5) = 5105$.
Then, the probability of each candidate will be estimated as follows:
$P(h_1) = \frac{c(d=5,t=5)}{c(d=5)} = \frac{2540}{5105} = 0.49$
$P(h_2) = P(h_3) = \frac{c(d=5,t=6)}{c(d=5)} = \frac{1200}{5105} = 0.23$
The first candidate h_1 has the more probable length according to the considered statistics.

5.2.2 Length-based Penalty

The second feature of this class is used to penalize the candidates based on their distance from the most probable candidate's length in the n-best list. Equation 23 defines the most probable candidate's length in the n-best list as the most frequent one as follows:

$$f_{len}(H(x)) = \underset{len(h), h \in H(x)}{\arg\max} \; (freq(len(h))) \tag{23}$$

Then, the score of each candidate $h \in H(x)$ is estimated in a way that penalizes the candidates based on their absolute distance from $f_{len}(H(x))$ (Eq. 24).

$$S(h) = \frac{1}{|len(h) - f_{len}(H(x))| + 1} \tag{24}$$

Example 4 Using the example introduced at the beginning of Section 5:
$len(h_1) = 5, \quad len(h_2) = 6, \quad len(h_3) = 6$
$f_{len(H(x))} = 6$, because the most frequent candidate length in our case is 6.
The score of each candidate will then be estimated as follows:
$S(h_1) = \frac{1}{|5-6|+1} = \frac{1}{2} = 0.5$
$S(h_2) = S(h_3) = \frac{1}{|6-6|+1} = \frac{1}{1} = 1$

The first candidate h_1 received a slight penalty because of its distance from the most frequent candidate's length. On the other hand, the two other candidates h_2 and h_3 were not penalized because their lengths are equal to the most frequent one.

5.3 Translation-based Features

This class incorporates some features related to the translation model. These features are used to promote the candidates that have the highest translation adequacy.

5.3.1 Reverse Translation Score

This feature is used to prioritize the candidates based on the quality of their reverse translation. We denote the direct NMT model that translates from source to target as $\overrightarrow{NMT}$ and the reverse target-to-source model as $\overleftarrow{NMT}$. Each candidate h will be scored using the reverse translation model (target-to-source model):

$$S(h) = \overleftarrow{NMT}(h) \tag{25}$$

where $\overleftarrow{NMT}(h)$ is the score given to the candidate h by the reversed target-to-source NMT translation model.

5.3.2 Original Rank Score

The second feature of this class is added to take into consideration the original n-best list order produced by the NMT decoder. To this end, we propose a function that gives a high score to any candidate that is highly ranked in the n-best list and a low score otherwise:

$$S(h) = \frac{1}{\log_2(rank(h)+1)} \tag{26}$$

This function produces a score of 1 if the candidate is ranked first in the n-best list ($rank(h) = 1$). The assigned score will decrease as the rank of the candidate in the n-best list increases. The score converges towards zero when $rank(h)$ converges towards infinity.

Example 5 Using the example as above, we suppose that the n-best list candidates were generated by the NMT decoder in the following order $H(x) = (h_1, h_2, h_3)$. Thus we have:
$rank(h_1) = 1, \quad rank(h_2) = 2, \quad rank(h_3) = 3$
The score of each candidate will then be estimated as follows:

$S(h_1) = \frac{1}{log_2(1+1)} = \frac{1}{1} = 1$
$S(h_2) = \frac{1}{log_2(2+1)} = \frac{1}{1.58} = 0.63$
$S(h_3) = \frac{1}{log_2(3+1)} = \frac{1}{2} = 0.5$

The first candidate was not penalized given that it is ranked first by the decoder. The other two candidates have been penalized based on their ranks.

5.3.3 Word-to-word Alignment Score

The third feature of this class is proposed to promote the n-best list candidates that have the most probable alignments. We measure the quality of a candidate alignment by using a word-to-word alignment model. The model is trained using a parallel corpus via the IBM alignment algorithms 1 to 5 (Brown et al., 1993).

To estimate the probability of aligning a candidate word to a source word, we just rely on the IBM word-to-word alignment statistics obtained from the parallel corpus. If we suppose that the j^{th} word of the candidate h was aligned to the i^{th} word of the source sentence x, then their alignment probability is estimated as follows:

$$P(h[j] \mid x[i]) = \frac{c(h[j], x[i])}{c(h[j])} \tag{27}$$

where $c(h[j], x[i])$ is the number of times a target word $h[j]$ was aligned to a source word $x[i]$, and $c(h[j])$ is the count of the target word $h[j]$ in the parallel corpus, respectively.

To address the general case in which a candidate word $h[j]$ can be aligned to multiple source words $A_s(h[j])$ (such as the 3^{rd} word of h_3 in Figure 3) we just take their average word-to-word alignment probability as shown in Eq. 28.

$$P(h, j) = \frac{\sum_{i \in A_s(h[j])} P(h[j] \mid x[i])}{|A_s(h[j])|} \tag{28}$$

where $|A_s(h[j])|$ is the number of source words that $h[j]$ is aligned to. Then, the probability of the whole candidate h (the score of h) is estimated as the product of all the alignment probabilities estimated at each position j of h as shown in Eq. 29:

$$S(h) = P(h) = \prod_{j=1}^{t} P(h, j) \tag{29}$$

We note that the case of null probability ($P(h, j) = 0$) is handled by assigning a very small value to it, in our case $\epsilon = 10^{-5}$.

Example 6 Still using the example introduced at the beginning of Section 5 and the alignment given in Figure 3, to estimate the probability of the first candidate h_1 according to this feature, we first estimate $P(h_1, j)$ the likelihood of the alignment link of the j^{th} word of h_1 as follows from the parallel corpus
$P(h_1, 1)$ = P(هناك | something) = c(هناك, something) / c(هناك)
$P(h_1, 2)$ = P(شيء | something) = c(شيء, something) / c(شيء)
The remaining probabilities $P(h_1, 3)$, $P(h_1, 4)$ and $P(h_1, 5)$ can be estimated in a similar manner.
Then, the final probability of h_1 is estimated as the product:
$S(h_1) = P(h_1) = P(h_1, 1) * P(h_1, 2) * ... * P(h_1, 5)$

5.3.4 Right-to-left Translation Score

Unlike standard NMT models which generate the translation from left-to-right, a right-to-left (R2L) NMT model generates the translation from right-to-left (in the reverse order). Some recent studies have shown that a R2L NMT model can be beneficial for the task of n-best list re-ranking (Liu et al., 2016, 2018; Hassan et al., 2018). The R2L model is trained in a similar way to the L2R model, the only difference is that the target-side sentences of the parallel corpus need to be reversed. After training the model, each translation candidate in the n-best list is first reversed, then the R2L NMT is used to assign a score to it as shown in equation 30.

$$S(h) = P(h) = NMT_{R2L}(reverse(h)) \tag{30}$$

where h is the translation candidate and $reverse(h)$ is h in its reversed order.

5.4 Fluency Features

This class of features has been included to promote the candidates based on their level of fluency. To this end, two models have been used: a recurrent neural network and an n-gram language model.

5.4.1 n-gram Language Model Score

An n-gram language model assigns a probability to a candidate sentence based on the likelihood of occurrence of each word in it given a few previous words (history). We use this feature to assign a score (probability) to each candidate in our n-best list based on a statistical n-gram Markov language model (Kirchhoff and Yang, 2005).

5.4.2 RNN Language Model Score

An RNN language model (Mikolov et al., 2010) can keep track of long-term dependencies and is often shown to be very effective in practice. We incorporate this feature to assign a score to each candidate in the n-best list.

5.5 Embedding Features

In this class, we use bilingual word embedding features to take into account the semantic and syntactic aspects of the translated candidates. To build a bilingual word embeddings vector we followed the approach proposed by (Smith et al., 2017). First, monolingual embeddings are built from the source and the target sides of the parallel corpus. Then the source and target embeddings are aligned using a small bilingual word-to-word translation dictionary. The alignment is performed via a linear transformation between the two embedding distributions (Smith

et al., 2017). Under this bilingual word embedding model, the similarity between a source and a target word-embedding vectors reflects the degree of their semantic correspondence. This interesting property constitutes the core of all the features that we will be presenting under this class.

We define $ES(x[i])$ as the source embedding vector of the word $x[i]$ and $ET(h[j])$ as the target embedding vector of the word $h[j]$.

5.5.1 Global Semantic Similarity

This first feature is used to consider the global semantic similarity between the source sentence and its candidate translation in a bag-of-words fashion. The idea is simple: we first need to represent the whole source sentence as a fixed-size vector by taking the average of all its word embeddings. Then, in a similar manner, we estimate the average vector representation of each translation candidate in the n-best list. Having these fixed-size vector representations for the source sentence and each one of its translation candidates, we can easily estimate the semantic similarity (cosine similarity) between the source sentence and one of its translation candidates.

To estimate the similarity between the source sentence x and one of its candidate translations h, we first estimate the average embeddings of all their words as follows:

$$ES_{avg}(x) = \frac{\sum_{i=1}^{d} ES(x[i])}{d} \tag{31}$$

$$ET_{avg}(h) = \frac{\sum_{j=1}^{t} ET(h[j])}{t} \tag{32}$$

where d is the length of the source sentence x and t is the length of its candidate translation h. Then, the global similarity between x and h is estimated using Equation 33.

$$S(h) = sim(x, h) = \frac{1 + Cosine(ES_{avg}(x), ET_{avg}(h))}{2} \tag{33}$$

where $Cosine(ES_{avg}(x), ET_{avg}(h))$ is the Cosine similarity between the average embedding vectors of x and h. We add one to the nominator and divide by two only to map the Cosine result from the interval $[-1, 1]$ to $[0, 1]$.

Example 7 Using the example introduced at the beginning of Section 5, to estimate the score of the first candidate h_1 according to this feature, we first calculate $ES_{avg}(x)$ and $ET_{avg}(h_1)$ the average word embedding of the source sentence x and the average word embedding of the first candidate h_1, respectively, as follows:
$ES_{avg}(x)$ = (ES(something) + ES(stiffened) + ES(inside) + ES(me))/4
$ET_{avg}(h_1)$ = (ET(هناك) + ET(شيء) + ET(ب) + ET(داخل) + ET(ي))/5
Then, the final score of h_1 is estimated as the cosine similarity between x and h_1.
$S(h_1) = sim(x, h_1) = \frac{1+Cosine(ES_{avg}(x), ET_{avg}(h_1))}{2}$

5.5.2 Alignment-based Semantic Similarity

The second feature of this class is similar to the feature presented in Section 5.3.3; the only difference is that this feature relies on bilingual word embeddings instead of a word-to-word alignment.

As previously stated (in Section 5.3.3) the NMT decoder generates a word-to-word alignment for the input source sentence x and each one of its translation candidates. This feature uses this alignment and the bilingual word embedding information to assign a score to each candidate. Figure 5 takes the previous example provided at the beginning of this section and illustrates its first part which involves the alignment of the source sentence x and its translation candidate h_1.

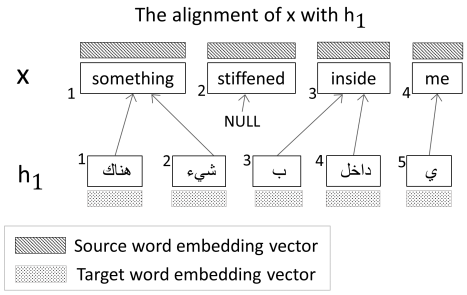

Fig. 5: An example illustrating the functioning mechanism of the alignment-based semantic similarity feature

As shown in Figure 5, to estimate the score of the candidate h_1, we go through each word of h_1 and measure its semantic similarity to the source words aligned to it. For instance, if we take the first word of h_1, we need to estimate its similarity to the English word "something" (because it is aligned to it), then for the second word of h_1, we also estimate its similarity with "something" given that it is also aligned to it. Since a single target word (candidate word) of h can be aligned to multiple words of x, we propose Eq. 34 to estimate the average embeddings vector of all the source words that the target word $h[j]$ is aligned to.

$$ES_{avg}(j) = \frac{\sum_{i \in A_s(h[j])} ES(x[i])}{|A_s(h[j])|} \tag{34}$$

where $A_s(h[j])$ is the list of source word positions of x that the target word $h[j]$ is aligned to, and $|A_s(h[j])|$ is the size of that list. Then Eq. 35 is used to estimate the probability of aligning $h[j]$ to its list of source word positions $A_s(h[j])$.

$$P(h, j) = max(\frac{1 + Cosine(ES_{avg}(j), ET(h[j]))}{2}, \varepsilon) \tag{35}$$

where ε is a very small value, e.g. 10^{-5}, that is used to avoid having null probabilities. We add one to the nominator and divide by two as we did for the previous feature to simply map the Cosine result from the interval $[-1, 1]$ to $[0, 1]$. Then the score of the candidate sentence h is estimated using Equation 36.

$$S(h) = P(h) = \prod_{j=1}^{t} P(h, j) \tag{36}$$

Example 8 Using the same example from the beginning of Section 5 and the alignment given in Figure 3, to estimate the score of the candidate h_1 according to this feature, we first calculate the alignment probability $P(h_1, j)$ at each position j of the candidate sentence h_1 as follows:
$P(h_1, 1)$ = max(1 + Cosine(ET(هناك), ES(something))/2 , ε)
$P(h_1, 2)$ = max(1 + Cosine(ET(شيء), ES(something))/2 , ε)
...
$P(h_1, 5)$ = max(1 + Cosine(ET(ي), ES(me))/2 , ε)
Then, the final score of h_1 is estimated as the product of all the individual probabilities:
$S(h_1) = P(h_1) = P(h_1, 1) * P(h_1, 2) * ... * P(h_1, 5)$

6 Experimentation and Evaluation

In this section, we start by presenting the software and hardware that we have used along with the data that we have incorporated. Then, we present the different tests that we have done.

6.1 Software and Hardware Setup

For all our tests, a Desktop Computer with the following characteristics has been used: an Intel Core I5 6500 Skylake Quad Core processor with a 3.20 GHz frequency, a 16 GB of DDR4 system RAM and a Gigabyte GeForce GTX 1070 GPU with 8GB of GDDR5 memory.

The following software packages were used:

- GIZA toolkit: we have used GIZA++ V2 (Och and Ney, 2003) [8] which performs word alignment by means of IBM Models 1 to 5 (Koehn, 2009). We have used it for the word-to-word alignment feature of the translation-based feature class (see Sect. 5.3.3).
- KenLM toolkit: We have used the KenLM toolkit (Heafield, 2011) [9] to train a 6-gram language model feature (see Sect. 5.4.1) from the monolingual target-side (Arabic) of the parallel training corpus. A modified Kneser-Ney smoothing was incorporated to handle the case of unseen words.
- OpenNMT toolkit: We have used the OpenNMT toolkit (Klein et al., 2017) [10] to train the RNN language model feature (see Sect. 5.4.2) and also to train our NMT baseline.
- SentencePiece toolkit: We have used the SentencePiece toolkit [11] to perform a byte-pair-encoding (BPE) language-independent segmentation on both the English and Arabic training data.
- FastText Multilingual toolkit (Smith et al., 2017) [12]: This is a toolkit to learn multilingual word embeddings via a linear transformation from their monolingual word embeddings. We have used it to learn our bilingual word embeddings from an English-Arabic parallel corpus to use it in our Embedding Features (see Sect. 5.5).
- NLG evaluation toolkit (Sharma et al., 2017): We have used the nlg-eval toolkit [13] to estimate the BLEU (Papineni et al., 2002) and METEOR[14] (Denkowski and Lavie, 2014) scores.

6.2 Data and Preprocessing

To train our baseline NMT model we have used the English-Arabic United Nations parallel corpora, which can be obtained for free from the "lingfil" website[15]. We have devised our tests into two groups:

1. In-domain tests: For these tests, we have used the UNv1.0 English-to-Arabic and Arabic-to-English development and test sets (Ziemski et al., 2016)[16].
2. Out-of-domain tests: For these tests, we have used the IWSLT 2015 and 2016 English-to-Arabic and Arabic-to-English test sets [17].

We kept only sentence pairs with both the source and the target sentence lengths having less than 50 words. "Bad" sentence pairs, i.e. for which the length difference between their source and target exceeds a certain threshold have been removed. We also removed all sentence pairs that contain more than 20% out-of-vocabulary words.

For the Arabic language, a language-specific preprocessing was applied; it includes:

- Removal of diacritical marks: all the Arabic diacritics such as "Fathah", "Dammah" and "Kasrah" are removed. This is a text normalization step that is performed to decrease the

[8] https://github.com/moses-smt/giza-pp/tree/master/GIZA%2B%2B-v2
[9] https://github.com/kpu/kenlm
[10] http://opennmt.net/
[11] https://github.com/google/sentencepiece
[12] https://github.com/Babylonpartners/fastText_multilingual
[13] https://github.com/Maluuba/nlg-eval
[14] For the remainder of this section "MET" will be used as an abbreviation for "METEOR".
[15] http://opus.lingfil.uu.se
[16] https://cms.unov.org/UNCorpus/
[17] http://workshop2016.iwslt.org/59.php

vocabulary size (i.e. decrease the number of Arabic unique words), which eases the training process [18]. For example, the two words رَأيتَ ("you saw") and رَأيتُ ("I saw"), will be joined under the same unvocalized word رأيت.

– Character normalization: all the Arabic characters are normalized to their most basic forms. This is so because characters such as أ ("Alif") can be written as ا ("Alif" without "HAMZA") in various texts. So this step normalizes the different writings. For example, the "HAMZA" in the phrase أنا أرى ("I see") is normalized to its most basic form, yielding انا ارى.

For the English side of the parallel corpus, only word-tokenization is performed using the Moses tokenizer available with the Python NLTK toolkit [19]. The tag <*nbr*> was used to group all the numbers that are present in the corpus, such as 12.1, 124, etc.

After applying all the aforementioned preprocessing to the training data, a Byte Pair Encoding (BPE) technique (Sennrich et al., 2015) is used to segment the Arabic and English words into smaller units. We set the BPE vocabulary size to 40k subword symbols for both the English and Arabic languages. This BPE segmentation allows us to simulate an open vocabulary with a limited set of subword symbols, which elegantly addresses the unknown word problem. The statistics about the corpus that we obtained before and after the BPE segmentation are provided in Table 2.

Table 2: Statistics about the used parallel training corpus

	Sentences	Unique words	Total words
English	1273841	91843	24596431
Arabic	1273841	211221	22125765
English-BPE	1273841	38581	25668988
Arabic-BPE	1273841	39882	22973496

6.3 Baseline Model

Our baseline model follows Google's Neural Machine Translation (GNMT) architecture (Wu et al., 2016b) which is an extension of the standard attention-based NMT (Bahdanau et al., 2014) [20]. We have used the Adam function (Kingma and Ba, 2014) for the Stochastic Gradient Descent (SGD) learning rate adjustment. We have used 2 GRU encoders and 2 GRU decoders with 500 units each, and a dropout layer with a dropout-rate of 0.3. We have set the size of the word embedding vectors to 300. Our model has been trained for 13 epochs (which is the default parameter suggested by the OpenNMT toolkit).

6.4 Classes/Features Impact

For the remainder of this section, B will be used to denote the NMT baseline system, and C_i and F_j will be used to denote the i^{th} class and the j^{th} feature, respectively, as follows:

1. N-best list Features (C_1): It includes the Position-based Score (f_1) and N-best Alignment Score (f_2).

[18] The effectiveness of this preprocessing step has been investigated in the work of Habash and Sadat (2006).

[19] `http://www.nltk.org/`

[20] For a detailed overview about the specificities of GNMT we refer the readers to the original paper of Wu et al. (2016b).

2. Length-based Features (C_2): It includes the Estimated Length Score (f_3) and Length-based Penalty (f_4).
3. Translation-based Features (C_3): It includes the Reverse Translation Score (f_5), Original Rank Score (f_6), Word-to-word Alignment Score (f_7), and Right-to-left Translation Score (f_8).
4. Fluency Features (C_4): It includes the n-gram Language Model Score (f_9) and RNN Language Model Score (f_{10}).
5. Embedding Features (C_5): It includes the Global Semantic Similarity (f_{11}) and Alignment-based Semantic Similarity (f_{12}).

The test results for incorporating each individual class of features to re-rank the n-best list generated by the NMT baseline decoder are presented in Table 3. The feature weights of each individual class are optimized using the QPSO swarm optimization algorithm over the development (DEV) dataset. The Minimum Bayes-Risk (MBR) results are also reported for both the EN-AR and the AR-EN translation directions.

Table 3: Translation results for the individual feature classes

AR-EN	Dev-UN		Test-UN		IWSLT15		IWSLT16	
	BLEU	MET	BLEU	MET	BLEU	MET	BLEU	MET
B	42.03	46.48	42.50	47.09	28.13	40.59	28.79	39.27
B+C1	42.51	46.81	**42.82**	46.95	29.02	**42.40**	**30.21**	42.09
B+C2	39.61	45.83	41.29	45.57	27.11	38.83	26.35	39.30
B+C3	42.89	47.07	42.59	47.54	28.97	41.55	30.11	**42.39**
B+C4	40.21	46.13	41.83	47.20	26.42	40.49	27.07	39.30
B+C5	**43.03**	**47.21**	42.41	47.91	28.40	41.50	29.83	42.13
MBR	42.07	46.78	42.47	**48.05**	**29.21**	42.29	29.64	42.28
EN-AR	Dev-UN		Test-UN		IWSLT15		IWSLT16	
	BLEU	MET	BLEU	MET	BLEU	MET	BLEU	MET
B	33.13	43.65	34.11	44.60	19.01	40.15	19.21	39.31
B+C1	33.67	**44.98**	**34.68**	45.81	19.52	**41.81**	19.95	42.29
B+C2	33.03	43.79	33.15	44.52	17.73	39.11	18.63	40.15
B+C3	**33.72**	44.87	34.59	45.87	19.51	41.59	**20.49**	41.97
B+C4	32.07	42.19	31.33	44.21	17.97	39.88	17.31	40.25
B+C5	33.61	44.82	34.47	45.32	**19.59**	41.34	20.35	41.43
MBR	32.23	44.51	34.50	**46.13**	19.22	41.47	19.82	**42.31**

As shown in Table 3, adding a new class of features to the re-ranking system can result in either an increase or a decrease in the overall NMT baseline performance. Indeed, Classes C_2 (length-based features) and C_4 (fluency features) gave a negative impact by decreasing the baseline NMT system results in terms of both BLEU and METEOR scores. This is to be expected since relying solely on the length of the n-best list candidates (C_2) or on their degree of fluency (C_4) does not provide us with the necessary information to re-rank them properly. On the other hand, using classes C_1 (N-best list Features), C_3 (Translation-based Features) and C_5 (Embedding Features) has led to an increase in the overall baseline performance. This indicates that each one of these three classes holds enough information that enables it to make good re-ranking decisions, thus being able to select the best translation among the n-best list candidates. Even though the difference between them has been very minor, C_1 and C_3 gave the highest increase, followed by C_5. The MBR re-ranking method also gave very close results to those three feature classes (but still slightly worse overall in terms of BLEU and METEOR).

Our second experiment aims at testing the effect of removing each feature class. First, we present the results of using all the feature classes (the abbreviation "all"), then we remove one class at a time and see how this affects the overall re-ranking performance. We note that each time a class of features is removed, the QPSO algorithm is incorporated to optimize the weights of the remaining features. The results of this experiment are provided in Table 4.

Table 4: Translation results of removing each individual feature class

AR-EN	Dev-UN		Test-UN		IWSLT15		IWSLT16	
	BLEU	MET	BLEU	MET	BLEU	MET	BLEU	MET
All	**43.11**	48.68	**43.31**	47.90	29.76	**42.62**	31.08	43.17
All - C1	42.71	48.60	42.89	47.11	29.88	42.22	30.61	42.07
All - C2	43.02	**48.73**	43.22	**47.95**	**29.81**	42.55	30.98	42.23
All - C3	42.91	48.61	42.81	47.77	29.11	41.95	31.10	42.15
All - C4	43.22	48.50	43.12	47.51	29.66	42.59	31.13	42.10
All - C5	42.81	47.42	43.27	48.01	29.59	42.40	**31.11**	**43.19**
EN-AR	Dev-UN		Test-UN		IWSLT15		IWSLT16	
	BLEU	MET	BLEU	MET	BLEU	MET	BLEU	MET
All	34.72	**45.82**	**35.71**	46.49	**20.41**	**42.48**	20.48	42.47
All - C1	34.52	45.80	35.39	46.29	20.07	42.11	20.22	41.97
All - C2	**34.83**	45.79	35.61	46.53	20.53	42.39	**20.51**	**42.62**
All - C3	33.98	45.29	34.92	45.89	19.83	42.22	20.32	42.49
All - C4	34.68	45.77	35.53	46.42	20.39	42.35	20.49	42.41
All - C5	34.81	45.73	35.63	**46.69**	20.11	42.41	20.19	42.35

As shown in Table 4, removing the C_1 and C_3 classes have led to a noticeable decrease in the re-ranking performance. Removing the C_4 and C_5 classes also decreased the overall re-ranking performance. However, the decrease was not as substantial as for C_1 and C_3. Removing the C_2 feature class did almost no damage at all to the re-ranking performance and, in some cases, slightly improved it.

The third experiment that we made investigated the effect of combining various feature classes, by adding one feature class at a time. As we did for the first experiment, we include the MBR results (that have been presented before in Table 3) for comparison purposes. We note that each time a new class of features is added to the re-ranking system, a feature weights optimization is performed by means of the QPSO algorithm. The results of this experiment are provided in Table 5.

Table 5: Translation results for the accumulated feature classes

AR-EN	Dev-UN		Test-UN		IWSLT15		IWSLT16	
	BLEU	MET	BLEU	MET	BLEU	MET	BLEU	MET
B	42.03	46.48	42.50	47.09	28.13	40.59	28.79	39.27
B+C1	42.51	46.81	42.82	46.95	29.02	42.40	30.21	42.09
B+C12	42.48	47.50	43.07	47.30	29.21	42.33	30.49	43.01
B+C123	42.91	47.69	43.11	47.25	29.30	42.64	30.81	**43.21**
B+C1234	42.81	47.42	43.27	48.01	29.59	42.40	**31.11**	43.19
B+C12345	**43.11**	**48.68**	**43.31**	47.90	**29.76**	**42.62**	31.08	43.17
MBR	42.07	46.78	42.47	**48.05**	29.21	42.29	29.64	42.28
EN-AR	Dev-UN		Test-UN		IWSLT15		IWSLT16	
	BLEU	MET	BLEU	MET	BLEU	MET	BLEU	MET
B	33.13	43.65	34.11	44.60	19.01	40.15	19.21	39.31
B+C1	33.67	44.98	34.68	45.81	19.52	41.81	19.95	42.29
B+C12	33.73	44.92	34.51	45.88	19.61	41.79	19.93	42.25
B+C123	34.13	45.33	34.79	45.91	19.79	42.17	20.11	42.37
B+C1234	**34.81**	45.73	35.63	**46.69**	20.11	42.41	20.19	42.35
B+C12345	34.72	**45.82**	**35.71**	46.49	**20.41**	**42.48**	**20.48**	**42.47**
MBR	32.23	44.51	34.50	46.13	19.22	41.47	19.82	42.31

As shown in Table 5, the impact of accumulating the feature classes is generally positive. We can see that even though the fluency feature class (C_4) had no positive impact when used individually, it still yields a very slight increase when used along with other classes. This suggests that the information it holds can be relevant to the overall re-ranking system when

combined with other classes of features. As we stated earlier, the MBR re-ranking results were similar to those obtained from the C_1 feature class (yet slightly below it).

The weights (importance values) that have been assigned to each feature in our final re-ranking system using the QPSO optimization algorithm are shown in Figure 6. The QPSO

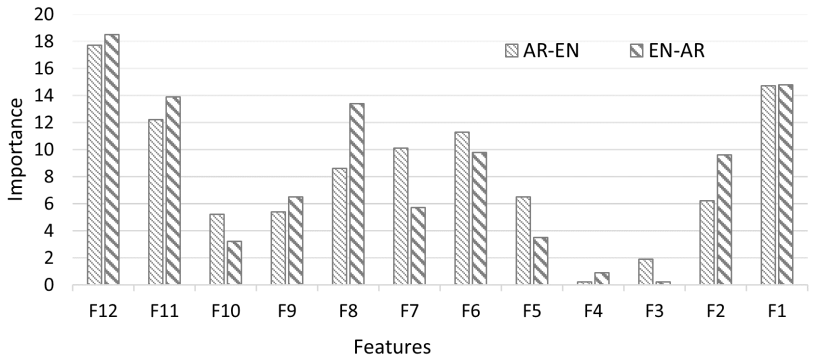

Fig. 6: Features importance in the Re-ranking System

algorithm gave a very high importance to the 1^{st} (Position-based Probability) and the 12^{th} (Alignment-based Semantic Similarity). This means that their influence on the re-ranking process is considerable. The estimated length Score (F_3) and length-based penalty (F_4) have been given very low importance which indicates that their impact on the re-ranking task was very minimal. All the remaining features have been given considerable importance which indicates that they played an important role in determining the best candidate in the re-ranking system.

Figure 7 presents the results of the QPSO feature weights optimization algorithm per class. These results have been obtained by simply summing up the feature weights belonging to the same class (from Figure 6).

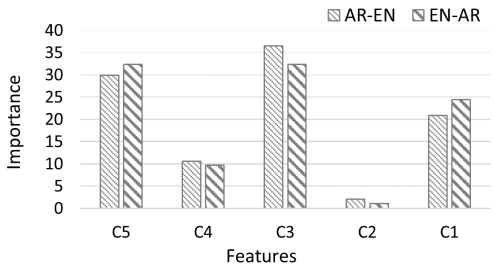

Fig. 7: Classes importance in the Re-ranking System

As shown in Figure 7, a high importance has been assigned to the C_3 (Translation-based Features), C_5 (Embedding Features), and C_1 (N-best list Features) classes. Less importance has been given to the 4^{th} (Fluency Features) feature class. Very tiny importance has been assigned to the 2^{sd} (Length-based Features) class. These results suggest that the length-based features are not very helpful for the task of n-best list re-ranking. The fluency features (language models) had a small positive impact on the re-ranking performance.

6.5 N-best List Impact on the Decoding Time

We have conducted this test to investigate the rate of increase of the decoding time as a function of the n-best list size. The results of this experiment are reported on the UN development set. The effect of increasing the n-best list size on the decoding time is shown in Figure 8.

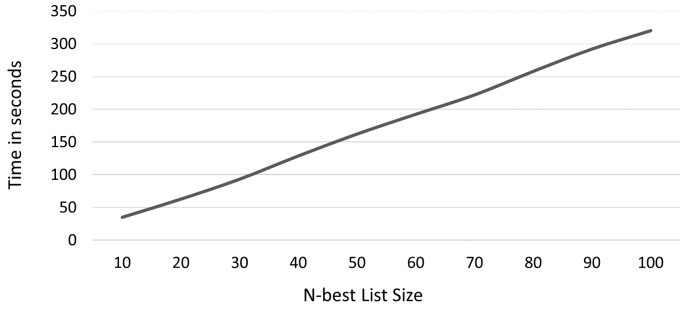

Fig. 8: The effect of the n-best list size on the decoding time

The latter shows that the increase of the decoding time is almost linear in respect to the increase in the n-best list size. It increases from $50s$ for $n = 10$ to $350s$ for $n = 100$.

6.6 N-best List Candidates Selection

When working with an n-best list of size n, the re-ranking system will have a tendency of picking candidates that are present in certain positions of the n-best list more than others. Figure 9 shows the results of the experiment that we have performed on our development set. In this experiment, the selection rate of each position (rank) from the n-best list by our re-ranking system is reported for an n-best list of size $n = 100$.

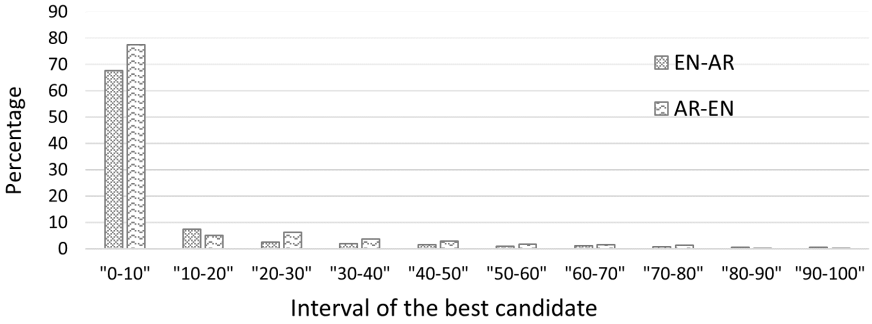

Fig. 9: Statistics about the ranks selected by the re-ranking system

As shown in Figure 9, the best candidate translation is chosen from the first 10 positions of the n-best list almost 70% of the time (for both directions). This is to be expected since the decoder original ordering is fairly well founded which means that the best candidate is more likely to be there. As we go further towards $n = 100$, the rate of selection of a best candidate keeps decreasing until it becomes almost null when reaching $n = 100$. This result confirms that using a larger n-best list (more than $n = 100$) will not lead to any significant improvement as also suggested by the work of Hasan et al. (2007).

7 Discussion

In this work, we have tackled the n-best list re-ranking problem by incorporating several features whose weights were optimized using a swarm-based optimization algorithm. The test results that have been presented in the previous section show that the features belonging to the translation (C_3), the embedding (C_5), and the n-best list (C_1) feature classes were the most effective for the task of n-best list re-ranking. Indeed, these feature classes alone

have managed to achieve more than 80% importance value (Figure 7) among all the other feature classes incorporated in this study. The two most useful individual features were the alignment-based semantic similarity (F_{12}) and the position-based feature (F_1). The usefulness of the embedding features suggests that the semantic knowledge that can be automatically learned from a bilingual word embedding is indeed helpful for the re-ranking process. With the exception of the length-based features, all the remaining features had mid-to-high importance values (Figure 7). Another thing worth mentioning is that the language model features also did not have a very noticeable impact on the re-ranking performance. We believe that this is due to the nature of the NMT encoder-decoder model which has a built-in capacity for learning language model information, hence for producing more fluent translations. For these reasons, we believe that additional language model features are not very helpful for the re-ranking process especially when using an NMT baseline system.

Overall, our proposal has some clear advantages, though it still suffers from some limitations as discussed hereafter:

- Strengths:
  - The features proposed in this paper are automatically learned from the parallel corpus without requiring any external tools. Thus our proposal is fairly general and can be used to tackle the translation task between any pair of languages.
  - This work has addressed the semantic level by introducing two features that use bilingual word embedding. The effectiveness of these features has been demonstrated via practical tests.
  - Our method can automatically estimate the effectiveness of each feature and assign an importance value to it via a swarm-based weights optimization algorithm. Thus, our proposal can deal with marginally useful features by decreasing their importance.
  - Even though our proposal does not use external tools, the improvement that we have achieved was still very noticeable, which encourages us for future improvements in this same direction.
- Limitations:
  - Even though general purpose features are interesting in the task of re-ranking, it is always helpful to use language-specific features to address linguistic phenomena that are related to a specific language such as Arabic.
  - Our system is able to ignore the irrelevant features by giving them very minor importance weights. However, having a more sophisticated filtering method that can remove the features when they are deemed irrelevant would be a valuable improvement.

8 Conclusion

In this work, we have presented a re-ranking system that uses a set of new sophisticated features to effectively reorder n-best list translations. All our proposed features can directly be extracted from the parallel corpus without needing any language-specific NLP tools. We also present a method for feature weights optimization that uses a Quantum-behaved Particle Swarm Optimization algorithm which guarantees the global convergence of the optimization process. The effectiveness of our re-ranking system has been tested on the UN and the IWSLT evaluation benchmarks, and the obtained results have shown noticeable increase in the overall translation performance.

The contributions of this work can be summarized as follows:

- We proposed a re-ranking system that does not have any language-specific tool dependencies.
- We proposed a new feature class that can capture the semantic level of the translation candidates via bilingual word embeddings.
- We used a swarm-based QPSO optimization algorithm that has been trained to maximize the BLEU score on a holdout data.

– We tested our proposal on an English-to-Arabic and an Arabic-to-English Neural Machine Translation (NMT) systems.

This work can be developed further in various directions. One such direction is to introduce a feature filtering mechanism that detects and removes the non-useful features. Also merging the n-best lists of multiple translation systems prior to the re-ranking task may yield better results. Additional features can also be added to appropriately handle the problem of pronoun resolution.

References

Arun A, Koehn P (2007) Online learning methods for discriminative training of phrase based statistical machine translation. Proc of MT Summit XI 2(5):29

Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. CoRR abs/1409.0473

Brown PF, Pietra VJD, Pietra SAD, Mercer RL (1993) The mathematics of statistical machine translation: Parameter estimation. Computational linguistics 19(2):263–311

Carter S, Monz C (2010) Discriminative syntactic reranking for statistical machine translation. In: Ninth Conference of the Association for Machine Translation in the Americas (AMTA 2010), Denver, CO, USA

Carter S, Monz C (2011) Syntactic discriminative language model rerankers for statistical machine translation. Machine translation 25(4):317–339

Chen B, Cherry C (2014) A systematic comparison of smoothing techniques for sentence-level bleu. In: Proceedings of the Ninth Workshop on Statistical Machine Translation, pp 362–367

Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014a) Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, pp 1724–1734, DOI 10.3115/v1/D14-1179, URL `http://www.aclweb.org/anthology/D14-1179`

Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014b) On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:14091259

Chopra S, Auli M, Rush AM (2016) Abstractive sentence summarization with attentive recurrent neural networks. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp 93–98

Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. Journal of Machine Learning Research 12(Aug):2493–2537

Denkowski M, Lavie A (2014) Meteor universal: Language specific translation evaluation for any target language. In: Proceedings of the ninth workshop on statistical machine translation, pp 376–380

Duh K, Kirchhoff K (2008) Beyond log-linear models: boosted minimum error rate training for n-best re-ranking. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, Association for Computational Linguistics, pp 37–40

Duh K, Sudoh K, Tsukada H, Isozaki H, Nagata M (2010) N-best reranking by multitask learning. In: Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR, Association for Computational Linguistics, pp 375–383

Farzi S, Faili H (2015) A swarm-inspired re-ranker system for statistical machine translation. Computer Speech & Language 29(1):45–62

Freitag M, Al-Onaizan Y (2017) Beam search strategies for neural machine translation. In: Proceedings of the First Workshop on Neural Machine Translation, pp 56–60

Goller C, Kuchler A (1996) Learning task-dependent distributed representations by backpropagation through structure. In: Neural Networks, 1996., IEEE International Conference on, IEEE, vol 1, pp 347–352

González-Rubio J, Juan A, Casacuberta F (2011) Minimum bayes-risk system combination. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:

Human Language Technologies-Volume 1, Association for Computational Linguistics, pp 1268–1277

Habash N, Sadat F (2006) Arabic preprocessing schemes for statistical machine translation. In: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, Association for Computational Linguistics, pp 49–52

Hasan S, Zens R, Ney H (2007) Are very large n-best lists useful for smt? In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, Association for Computational Linguistics, pp 57–60

Hassan H, Aue A, Chen C, Chowdhary V, Clark J, Federmann C, Huang X, Junczys-Dowmunt M, Lewis W, Li M, et al. (2018) Achieving human parity on automatic chinese to english news translation. arXiv preprint arXiv:180305567

Heafield K (2011) Kenlm: Faster and smaller language model queries. In: Proceedings of the Sixth Workshop on Statistical Machine Translation, Association for Computational Linguistics, pp 187–197

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780

Kim Y (2014) Convolutional neural networks for sentence classification. arXiv preprint arXiv:14085882

Kingma D, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980

Kirchhoff K, Yang M (2005) Improved language modeling for statistical machine translation. In: Proceedings of the ACL Workshop on Building and Using Parallel Texts, Association for Computational Linguistics, pp 125–128

Klein G, Kim Y, Deng Y, Senellart J, Rush AM (2017) Opennmt: Open-source toolkit for neural machine translation. arXiv preprint arXiv:170102810

Koehn P (2009) Statistical machine translation. Cambridge University Press

Koehn P, Knowles R (2017) Six challenges for neural machine translation. In: Proceedings of the First Workshop on Neural Machine Translation, Association for Computational Linguistics, pp 28–39, DOI 10.18653/v1/W17-3204, URL http://aclweb.org/anthology/W17-3204

Kumar S, Byrne W (2004) Minimum bayes-risk decoding for statistical machine translation. Tech. rep., JOHNS HOPKINS UNIV BALTIMORE MD CENTER FOR LANGUAGE AND SPEECH PROCESSING (CLSP)

Li J, Jurafsky D (2016) Mutual information and diverse decoding improve neural machine translation. arXiv preprint arXiv:160100372

Liu L, Utiyama M, Finch A, Sumita E (2016) Agreement on target-bidirectional neural machine translation. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp 411–416

Liu Y, Zhou L, Wang Y, Zhao Y, Zhang J, Zong C (2018) A comparable study on model averaging, ensembling and reranking in nmt. In: CCF International Conference on Natural Language Processing and Chinese Computing, Springer, pp 299–308

Luong NQ, Popescu-Belis A (2016) A contextual language model to improve machine translation of pronouns by re-ranking translation hypotheses. In: Proceedings of the 19th Annual Conference of the European Association for Machine Translation, pp 292–304

Mikolov T, Karafiát M, Burget L, Černockỳ J, Khudanpur S (2010) Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association

Neubig G, Morishita M, Nakamura S (2015) Neural reranking improves subjective quality of machine translation: Naist at wat2015. arXiv preprint arXiv:151005203

Och FJ, Ney H (2003) A systematic comparison of various statistical alignment models. Computational linguistics 29(1):19–51

Och FJ, Gildea D, Khudanpur S, Sarkar A, Yamada K, Fraser A, Kumar S, Shen L, Smith D, Eng K, et al. (2004) A smorgasbord of features for statistical machine translation. In: Proceedings of the Human Language Technology Conference of the North American Chapter

of the Association for Computational Linguistics: HLT-NAACL 2004
Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics, pp 311–318
Russell SJ, Norvig P (2016) Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,
Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45(11):2673–2681
Sennrich R, Haddow B, Birch A (2015) Neural machine translation of rare words with subword units. arXiv preprint arXiv:150807909
Sharma S, El Asri L, Schulz H, Zumer J (2017) Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. CoRR abs/1706.09799, URL http://arxiv.org/abs/1706.09799
Shu R, Nakayama H (2017) Later-stage minimum bayes-risk decoding for neural machine translation. arXiv preprint arXiv:170403169
Smith SL, Turban DH, Hamblin S, Hammerla NY (2017) Offline bilingual word vectors, orthogonal transformations and the inverted softmax. arXiv preprint arXiv:170203859
Sokolov A, Wisniewski G, Yvon F (2012) Non-linear n-best list reranking with few features. In: Association for Machine Translation in the Americas
Specia L, Sankaran B, Nunes MdGV (2008) N-best reranking for the efficient integration of word sense disambiguation and statistical machine translation. In: International Conference on Intelligent Text Processing and Computational Linguistics, Springer, pp 399–410
Stahlberg F, Hasler E, Waite A, Byrne B (2016) Syntactically guided neural machine translation. arXiv preprint arXiv:160504569
Sun J, Xu W, Feng B (2004) A global search strategy of quantum-behaved particle swarm optimization. In: Cybernetics and Intelligent Systems, 2004 IEEE Conference on, IEEE, vol 1, pp 111–116
Sun J, Fang W, Wu X, Palade V, Xu W (2012) Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection. Evolutionary computation 20(3):349–393
Tong Y, Wong DF, Chao LS (2016) Exploiting rich feature representation for smt n-best reranking. In: Wavelet Analysis and Pattern Recognition (ICWAPR), 2016 International Conference on, IEEE, pp 101–106
Tromble RW, Kumar S, Och F, Macherey W (2008) Lattice minimum bayes-risk decoding for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp 620–629
Vijayakumar AK, Cogswell M, Selvaraju RR, Sun Q, Lee S, Crandall D, Batra D (2016) Diverse beam search: Decoding diverse solutions from neural sequence models. arXiv preprint arXiv:161002424
Wang D, Nyberg E (2015) A long short-term memory model for answer sentence selection in question answering. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), vol 2, pp 707–712
Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K, Klingner J, Shah A, Johnson M, Liu X, Kaiser L, Gouws S, Kato Y, Kudo T, Kazawa H, Stevens K, Kurian G, Patil N, Wang W, Young C, Smith J, Riesa J, Rudnick A, Vinyals O, Corrado G, Hughes M, Dean J (2016a) Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR abs/1609.08144, URL http://arxiv.org/abs/1609.08144, 1609.08144
Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K, et al. (2016b) Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:160908144
Xiao T, Zhu J, Liu T (2013) Bagging and boosting statistical machine translation systems. Artificial Intelligence 195:496–527

Zhang J, Utiyama M, Sumita E, Neubig G, Nakamura S (2017) Improving neural machine translation through phrase-based forced decoding. arXiv preprint arXiv:171100309

Zhang Z, Wang R, Utiyama M, Sumita E, Zhao H (2018) Exploring recombination for efficient decoding of neural machine translation. arXiv preprint arXiv:180808482

Ziemski M, Junczys-Dowmunt M, Pouliquen B (2016) The united nations parallel corpus v1. 0. In: LREC