The Usage of Differential Evolution in a Statistical Machine Translation

Jani Dugonik, Borko Bošković, Mirjam Sepesy Maučec, Janez Brest
Faculty of Electrical Engineering and Computer Science
University of Maribor
2000 Maribor, Slovenia
Email: jani.dugonik@um.si

***Abstract*—Translations in statistical machine translation (SMT) are generated on the basis of statistical models, the parameters of which are derived from the analysis of aligned bilingual text corpora. Different models' parameters provide various translations, which can be evaluated by the BiLingual Evaluation Understudy (BLEU) metric. The problem of finding a suitable translation can be regarded as an optimization problem and some optimization can be done using the decoder itself - the optimization of models parameters. The main goal of this paper was to build SMT systems for the language pair English-Slovenian, and improve their translation quality using a global optimization algorithm - Differential Evolution (DE) algorithm. Experiments were performed using English and Slovenian JRC-ACQUIS Multilingual Parallel Corpora. The results show improvement in the translation quality.**

I. Introduction

Natural language is alive and constantly changing, the rules are complex and do not take creativity into account. Natural language processing (NLP) [2] is a field situated between Computer Science and Linguistics and is concerned with the interaction between computers and natural (human) language, spoken and written. NLP deals with problems ranging from ambiguity resolution on both the lexical and syntax levels, part-of-speech tagging (POS-tagging), speech and text segmentation, to syntactic and semantic parsing, and machine translation.

Machine translation (MT) [2] is a process where a computer analyses the text in the source language and produces translated text in the target language without human intervention. MT systems include single or multilingual lexicons, software for morphological analysis and synthesis, software for syntax analysis and synthesis, software for resolving ambiguities, software for identifying multi word semantic units, and other complex mechanisms that serve the automation of the translation process. Translation is a challenging and creative act. In some cases, machine translation can ease the work of a translator or even replace it as a rough translation, which would be reviewed and corrected by a translator at a later date, or as a draft which serves as an aid to the translation. There are several methods of machine translation, like the rule-based, statistical, example-based and hybrid methods.

The statistical method, also known as statistical machine translation (SMT) [2], is nowadays by far the more widely studied and used machine translation method [1]. Statistical methods were originally used for translating single words (word-based translation) but now they have progressed to the level of translating sequences of words called blocks or phrases (phrase-based translation). Currently the most successful SMT systems are based on phrase-based translation.

Translations in SMT are generated on the basis of statistical models - translation and language models, the parameters of which are derived from the analysis of aligned bilingual text corpora. Different models' parameters can provide various translations and finding the best translation is called decoding. The problem of finding the best translation can be regarded as an optimization problem. Some optimization can be done using a decoder - the optimization of the models parameters. This optimization can be done using evolutionary algorithms, which are simple and effective algorithms for global optimization. Therefore, we chose the Differential Evolution (DE) algorithm, where each individual from the population is described as a vector of the model's parameters (weights) used to translate a text in the source language to a text in the target language. Each individual is evaluated using BLEU metric, which is one of the more popular and inexpensive automated metrics for achieving a high correlation with human judgements of quality [3].

In this paper we present the usage of a DE algorithm within SMT systems. The remainder of the paper is organized as follows. Section II shows some past work of when using the DE algorithm within a SMT system and describes the basic DE algorithm. Section III describes SMT in detail. Our experiment: the used corpus, SMT systems, how we improved SMT systems and results, are given in Section IV. For our experiment we used the language pair English-Slovenian in both direction, so our examples in the paper are be based on these two languages. We conclude this paper with Section V, where we will give our opinion about the obtained results.

II. Background

A. Previous Work

In the paper [14] the authors addressed the problem of SMT from highly inflective language to less inflective language. Existing translation systems often treat different word forms of the same lemma as if they were independent of each other, although some interdependence does exist because the characteristics of inflective languages are not generally taken

into account by the SMT system. Thus if we reduce inflected word forms to common lemmas, some information is lost. So it would be reasonable to only eliminate those variations in inflected word forms which are not relevant for the translation. Inflectional features of words are defined by morpho-syntactic descriptions (MSD) tags and authors wanting to reduce them. The authors' idea was to find the information-bearing MSDs within a source language using a data-driven approach. The task was performed by the DE algorithm, which is a simple and effective algorithm for global optimization.

In SMT an alignment defines a mapping between the words within a source and within a target sentence. Alignments are used to train statistical models and during the decoding process to link the words within a source sentence to words within the partial hypotheses generated. In both cases, the quality of alignments is crucial for the success of the translation process. The authors of the paper [19] focused on the decoder. They state that a good decoding algorithm is critical for the success of any SMT system. Optimal decoders that guarantee the finding of optimal solutions are not used during practical SMT implementation because a decoding problem is NP-complete [10]. Authors introduce an evolutionary decoding algorithm in order to improve the efficiency of the translation task within a SMT framework. They performed the translation from Spanish to English and made a comparison with the public greedy decoder.

In the paper [22] the authors proposed several evolutionary algorithms for computing alignments between two sentences within a parallel corpus. In regard to their experiments it was notable that because of the limitations of well-known statistical alignment models, new improvements in alignment qualities could not be achieved by using only improved search algorithms.

B. Differential Evolution

The DE [5], [15], [24], [25], [21] algorithm is a simple and effective algorithm for global optimization. Due to its simplicity and effectiveness it is used for solving various practical problems. DE can be described as a simple mathematical model of a complex evolutionary process, therefore it is classified as an evolutionary algorithm (EA). This mathematical model is based on using the differences between vectors or individuals. The differences between vectors or individuals are defined with the help of fast and simple arithmetic operations.

The DE algorithm is a real-coded population-based algorithm, where each generation g has a current population $\mathrm{P}_{\mathbf{x},g}$ and a trial population $\mathrm{P}_{\mathbf{u},g}$. The current population contains those individuals who survived the previous generation and the trial population contains those individuals who were created by the DE algorithm within the current generation and will compete against the individuals from the current population. Both populations contain N_p individuals. These individuals are represented in the forms of *D*-dimensional vectors. The elements of the vectors are real numbers the values of which are from specified intervals. These intervals are provided by the user with the help of the lower and upper bounds.

Algorithm 1 Differential Evolution Algorithm

```
initialization(P_{x,0}, N_p, D, b_L, b_U)
for g = 1 to g_max do
    for i = 1 to N_p do
        v_{i,g} = mutation(P_{x,g}, i, F, N_p, D)
        u_{i,g} = crossover(x_{i,g}, v_{i,g}, C_r, D))
    end for
    selection(P_{x,g},P_{u,g}, N_p, D)
end for
```

A DE algorithm contains parameters that can modify the properties of an algorithm and with these parameters we can adjust the algorithm in regard to different problems. The following parameters represent control parameters that are specified by the user, and affect the behavior of the algorithm:

- mutation parameter (F),
- crossover parameter (C_r),
- population size (N_p),
- algorithm strategy (s) and
- maximum number of generations (g_{max}).

The following parameters determine the problem being solved:

- dimension problem (D),
- lower bounds ($\mathbf{b}_L$) and
- upper bounds ($\mathbf{b}_U$).

Based on the type of problem, we can choose between the more varied strategies of the DE algorithm. The strategy s determines the method of mutation and crossover. The pseudo code of the DE algorithm is shown in Algorithm 1.

A DE's most versatile implementation maintains a pair of vector populations, both of which contain N_p $D-$dimensional vectors of real-valued parameters. The current population $P_{\mathbf{x}}$ is composed of vectors $\mathbf{x}_{i,g}$ which have already been found as acceptable either as initial points or by comparison with other vectors:

$$\begin{aligned} P_{\mathbf{x},g} &= \{\mathbf{x}_{1,g}, \mathbf{x}_{2,g}, ..., \mathbf{x}_{N_p,g}\} \\ \mathbf{x}_{i,g} &= \{x_{1,i,g}, x_{2,i,g}, ..., x_{D,i,g}\} \\ i &= 1, ..., N_p, \;\; g = 1, ..., g_{max} \end{aligned} \tag{1}$$

The index $g = 1, 2, ..., g_{max}$ indicates the generation to which a vector belongs. In addition, each vector is assigned a population index i which runs from 1 to N_p. Parameters within vectors are indexed with j, which runs from 1 to D.

Once initialized, DE mutates randomly chosen vectors to produce an intermediary population $P_{\mathbf{v},g}$ of N_p mutant vectors $\mathbf{v}_{i,g}$:

$$\begin{aligned} P_{\mathbf{v},g} &= \{\mathbf{v}_{1,g}, \mathbf{v}_{2,g}, ..., \mathbf{v}_{N_p,g}\} \\ \mathbf{v}_{i,g} &= \{v_{1,i,g}, v_{2,i,g}, ..., v_{D,i,g}\} \\ i &= 1, ..., N_p, \;\; g = 1, ..., g_{max} \end{aligned} \tag{2}$$

Each vector in the current population is then recombined with a mutant to produce a trial population $\mathrm{P}_{\mathbf{u}}$ of N_p trial

vectors $\mathbf{u}_{i,g}$:

$$\begin{aligned} P_{\mathbf{u},g} &= \{\mathbf{u}_{1,g}, \mathbf{u}_{2,g}, ..., \mathbf{u}_{N_p,g}\} \\ \mathbf{u}_{i,g} &= \{u_{1,i,g}, u_{2,i,g}, ..., u_{D,i,g}\} \\ i &= 1, ..., N_p, \;\; g = 1, ..., g_{max} \end{aligned} \tag{3}$$

During recombination, trial vectors overwrite the mutant population, so a single array can hold both populations.

1) Initialization: Before the population can be initialized, both upper and lower bounds for each parameter must be specified. These values can be collected into two *D*-dimensional initialization vectors $\mathbf{b}_L$ and $\mathbf{b}_U$, for which subscripts L and U indicate the lower and upper bounds. Once initialization bounds have been specified, a random number generator assigns each vector component a value within the prescribed range. For example, the initial value of the j-th parameter of the i-th vector is:

$$x_{j,i,0} = rand_j(0,1) \cdot (b_{j,U} - b_{j,L}) + b_{j,L}. \tag{4}$$

The random number generator rand$_j$(0,1) returns a uniformly distributed random number within the range [0,1). The subscript *j* indicates that a new random value is generated for each vector component.

2) Mutation: Once initialized, DE mutates and recombines the population to produce a population of N_p trial vectors. Differential mutation adds a scaled, randomly sampled vector difference to a third vector. Eq. (5) shows how to combine three different, randomly chosen vectors to create a mutant vector $\mathbf{v}_i, g$:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r_1,g} + F \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}). \tag{5}$$

The scale factor $F \in$ [0,2] is a positive real number that controls the rate at which the population evolves. Usually it is less than 1.

The base vector index r_1 can be determined in a variety of ways, but for now it is assumed to be a randomly chosen vector index that is different from the target vector index *i*. Except from being distinct from each other and from both the base and target vector indices, the difference vector indices r_2 and r_3 are also randomly selected once per mutant.

3) Crossover: After mutation DE algorithm employs uniform crossover operation. Crossover builds trial vectors with the parameter values that have been copied from two different vectors. DE crosses each vector with a mutant vector v_i to get trial vector $\mathbf{u}_{i,g}$:

$$\begin{aligned} \mathbf{u}_{i,g} &= \{u_{1,i,g}, u_{2,i,g}, ..., u_{D,i,g}\}, \\ u_{j,i,g} &= \begin{cases} v_{j,i,g} & if \;\; (rand(0,1) \leq C_r \;\; or \;\; j = rand_j \\ x_{j,i,g} & otherwise, \end{cases} \\ j &= 1, ..., D, \;\; i = 1, ..., N_p, \;\; g = 1, ..., g_{max}. \end{aligned} \tag{6}$$

The crossover probability $C_r \in$ [0,1] is a user-defined value that controls the fraction of parameter values that are copied from the mutant. To determine which source contributes a given parameter, uniform crossover compares C_r to the output of a uniform random number generator $rand$(0,1). If the random number is less than or equal to C_r, the trial parameter is inherited from the mutant $\mathbf{v}_{i,g}$, otherwise, it is copied from the vector $\mathbf{x}_{i,g}$. In addition, the trial parameter with randomly chosen index $rand_j$ is taken from the mutant to ensure that the trial vector does not duplicate $\mathbf{x}_{i,g}$. Because of this additional demand, C_r only approximates the true probability that a trial parameter will be inherited from the mutant.

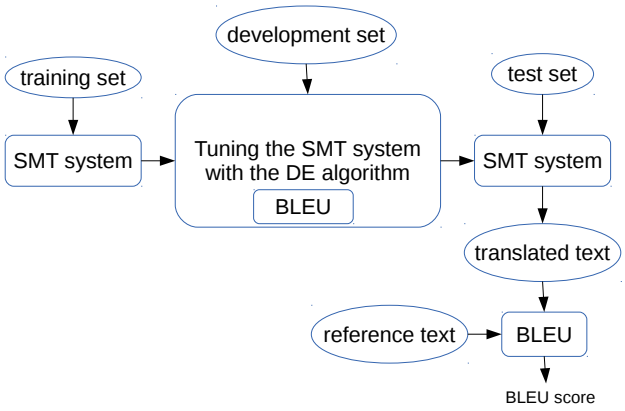

Figure 1. Using the DE algorithm in the SMT system

4) Selection: If the trial vector $\mathbf{u}_{i,g}$ has an equal or lower objective function value than that of its target vector $\mathbf{x}_{i,g}$, it replaces the target vector in the next generation, otherwise, the target retains its place in the population for at least one more generation, as shown in Eq. (7). By comparing each trial vector with the target vector from which it inherits parameters:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & if \;\; f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & otherwise. \end{cases} \tag{7}$$

Once the new population is installed, the process of mutation, recombination and selection is repeated until the optimum is located or a pre-specified termination criterion is satisfied, for example number of generations reaches a maximum number of generations g_{max}.

III. Statistical Machine Translation

The aim in phrase-based translation is to reduce the restrictions of a word-based translation by translating whole sequences of words, where the lengths may differ. The sequences of words are called blocks or phrases. These phrases are not linguistic phrases but phrasemes found using statistical methods from corpora. SMT systems are based on parametric statistical models which are learned on the aligned bilingual corpora. The SMT system looks for general patterns which appear in the everyday language. The text is translated according to the probability distribution - the translation with the highest probability is selected. In SMT, probability models

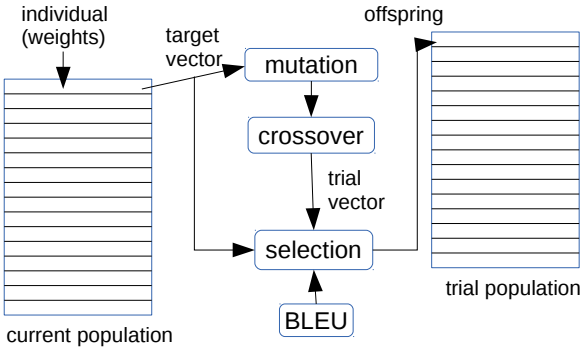

Figure 2. The DE algorithm

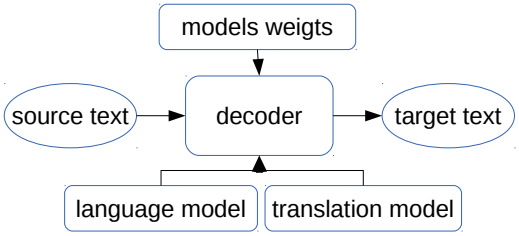

Figure 3. The SMT system

are used to find the best possible translation TL^* out of all possible translations TL for a given sentence SL, where SL is the source language sentence and TL is the target language sentence. Finding the best translation is called decoding, as shown in Figure 3. Probability models are estimated from monolingual and bilingual training data and may include translation models, language models, reordering models, etc. In practice discriminative linear model with logarithmic probabilities is used. If the models of a search algorithm are $h_i, ..., h_r$, which are dependent on TL and SL, then we obtain the best translation with:

$$TL^*(\lambda) = \arg\max_{TL} P(TL, SL) = \arg\max_{TL} \sum_{i=1}^{r} w_i h_i(TL, SL) \tag{8}$$

where λ is a set of models' weights, r is the number of models and $w_i, ..., w_r$ are models' weights. Model h_i may be a number or a function, for example word count, which will penalize too long or too short sentences. The problem appears on how to find a set of weights that will provide the best translation quality.

The main objective of SMT is to generate translations from statistical models based on the bilingual corpus. The bilingual corpus could be compared to a bilingual dictionary. The idea of using bilingual corpora is not new, it dates back to the early days of machine translation but it was not used in practice until 1984 [8]. The use of bilingual parallel corpora seems to offer a promising tool for the future, thanks to the progress which has been made in terms of storage and computing capacities, as well as acquisitions of large amounts of text. Moreover, bilingual corpora are richer in information about the language than monolingual corpora, since it provides situational equivalency information on the possibilities of the language system when in contact with different linguistic systems.

A. Corpora

The SMT system that is based on probabilistic models which are generally trained using parallel corpora. A parallel corpus is a sentence aligned pair of documents in which each pair of aligned sentences are translations of each other.

In SMT, as shown in Figure 1, a corpus is divided into two sets: seen and unseen. The seen set is the set used for training, and is further divided into two sets: training and development. The unseen set is the set used for evaluation and is called the test set. During the training, a model learns the statistics over the training set provided to it and estimates its parameters accordingly. The translations that are produced are very dependent on the parallel corpus used for training. The larger the size of the training corpus the better the parameter estimation is, and thus the better translation quality. The translation quality is evaluated on a test set. During translation, sentences can be split, merged, deleted, inserted or reordered by the translator. However, one of the major challenges faced by SMT is the scarce availability of parallel corpora. There are some language pairs for which a huge set of parallel corpora are readily available but for some language pairs there is a scarcity of parallel corpora.

Manual creation of a large parallel corpora can be very costly in terms of effort and time, so we need some methods for automatically and efficiently creating parallel corpora. Parallel corpora are objects of interest at present because of the opportunity offered to align the original with the translation, and thus gain insights into the nature of the translation.

B. N-gram Language Model

The n-gram language model (LM) is a type of probabilistic language model for predicting the next item within a sequence in the form of a (n-1)-order Markov model. It predicts w_i based on w_{i-n+1}, ..., w_{i-1}. In probability terms, this is $P(w_i|w_{i-n+1}, ..., w_{i-1})$.

When used for language modeling, independence assumptions are made so that each word depends only on the last n-1 words, called history. This assumption is important because it massively simplifies the problem of learning the language model from the data. In a simple n-gram language model the probability of a word, conditioned on some number of previous words, can be described as a categorical distribution, often called a multinomial distribution where the probabilities of words sum to 1. In practice, the probability distributions

are smoothed by assigning non-zero probabilities to unseen words or n-grams. A n-gram is a contiguous sequence of n items from a given sequence of text or speech. Items can be phonemes, syllables, letters, words or phrases. N-grams are typically collected from a text or speech corpus. A n-gram of size 1 is referred to as *unigram*, a n-gram of size 2 to as *bigram* and a n-gram of size 3 to as *trigram*. Larger sizes are sometimes referred to by the value of n, for example *fourgram*, *fivegram*, and so on.

Assume we have a string of English words. The language model answers the question "How likely is a string of English words good English?". The language model assigns a probability to a sequence of words by means of a probability distribution. They are used in many NLP applications like speech recognition, machine translation, part-of-speech tagging, parsing and information retrieval. In machine translation the more likely sentences are probably better translations. Language models also support predicting the completion of a sentence.

- Word choice: P_{lm}(I am going home) > P_{lm}(I am going house)
- Reordering: P_{lm}(the house is small) > P_{lm}(small the house is)

N-gram language models estimate the probability of each word given prior context: P(John | Hello my name is). Number of parameters required grows exponentially with the number of words of prior context. A n-gram model uses only n-1 words of prior context:

- unigram: $P(John)$
- bigram: $P(John|is)$
- trigram $P(John|name\ is)$

The Markov assumption is the presumption that the future behavior of a dynamic system only depends on its recent history. In particular, in a k-order Markov model, the next state only depends on the k most recent states, therefore a n-gram model is a (n-1)-order Markov model.

N-gram model formulas:

- Word sequence: $W = w_1^l = w_1...w_l$, where l is the number of words in sentence
- Chain rule of probability: $P(W) = \prod_{i=1}^{l} P(w_i|w_1^{i-1}) = = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1^2) \cdot ... \cdot P(w_l|w_1^{l-1})$
- N-gram approximation: $P(W) = \prod_{i=1}^{l} P(w_i|w_{i-n+1}^{i-1})$

Note that n-gram approximation uses a specific n-gram model for approximating the whole k^{th} probability of a sentence. N-gram conditional probability $P(w_i|w_{i-n+1}^{i-1})$ can be estimated from raw text based on the relative frequency of word sequences:

$$P(w_k|w_{i-n+1}^{i-1}) = \frac{count(w_{i-n+1}^{i-1} w_i)}{count(w_{i-n+1}^{i-1})}$$

where $count$ is a function that returns a number of n-grams in the brackets from the raw text.

To have a consistent probabilistic model, append a unique start (<s>) and end (</s>) symbol to every sentence and treat these as additional words.

Example: Calculating probabilities for the two sentences.
P(<s> i want english food </s>) =
P(i | <s>) · P(want | i) · P(english | want) · P(food | english) · P(</s> | food) = 0.1 · 0.25 · 0.0009 · 0.3 · 0.8 = $5.4e^{-6}$

P(<s> i want chinese food </s>) =
P(i | <s>) · P(want | i) · P(chinese | want) · P(food | chinese) · P(</s> | food) = 0.1 · 0.25 · 0.004 · 0.6 · 0.8 = $4.8e^{-5}$

The values in the example were calculated from our own test corpus. From the above example we can see that the sentence "<s> i want chinese food </s>" has a higher probability of being chosen.

The model can be evaluated based on its ability to predict a high probability for a disjointed test corpus - testing on the training corpus would give an optimistically biased estimate. Ideally, the training and test corpus should be representative of the actual application data. Estimating the probability of sequences can become difficult within corpora, in which phrases or sentences can be arbitrarily long and hence some sequences are not observed during the training of the language model - data sparseness problem of over-fitting. It is for that reason, these models are often approximated using smoothed n-gram models.

Smoothed n-gram models, like Laplace (Add-one) smoothing [13], are also one of the methods for handling words within the test corpus which did not occur in the training data, i.e. out of vocabulary (OOV) words. Laplace smoothing is a technique used to smooth categorical data. Imagine additional training data in which each possible n-gram occurs exactly once and adjust estimates accordingly:

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{count(w_{i-n+1}^{i-1} w_i)+1}{count(w_{i-n+1}^{i-1})+V}$$

where V is the total number of possible (n-1)-grams, i.e. the vocabulary size for a n-gram model.

C. Translation Model

The translation model (TM) is a "backwards" model, i.e. if we translate from English to Slovenian, we would actually translate from Slovenian to English, and it is trained on bilingual parallel corpora.

IBM translation models 1-5 [12], [11], [16] form the basis for many SMT models used today. But phrase-based translation models give much improved translations over the IBM models and provide state-of-the-art translations for many pairs of languages. They allow lexical entries with more than one word on either the source or target language side. For example, we might have a lexical entry of a Slovenian string "pes" and a English string "the dog", specifying that the string "pes" in Slovenian can be translated as "the dog" in English. The option of having multi word expressions in either the source or target language is a significant departure from IBM models, which are essentially word-to-word translation models, i.e. they assume each word in a source language is generated from a single word in English. Multi word expressions are extremely useful in the translation and

this is the main reason for improvements that phrase-based translation models give.

Example 1: Translating from Slovenian (sl) to English (en) with the use of the translation model $P(sl|en)$.
P(Lačen sem|What hunger have) $= 0.000014$
P(Lačen sem|Hungry I am) $= 0.000001$
P(Lačen sem|I am hungry) $= 0.0000015$
P(**Lačen sem**|**Have I hunger**) $= \mathbf{0.00002}$

In Example 1 the correct translation would be "I am hungry", but we can see that the translation model assigned higher probability to the translation "Have I hunger". With the use of a language model, we can complement deficiencies of the translation model, as shown in Example 2.

Example 2: Translating from Slovenian (sl) to English (en) with the use of the translation and language model $P(en) * P(sl|en)$.
P(Lačen sem|What hunger have) $= 0.000001 \cdot 0.000014 = 1.4e^{-11}$
P(Lačen sem|Hungry I am) $= 0.0000014 \cdot 0.000001 = 1.4e^{-12}$
P(**Lačen sem**|**I am hungry**) $= 0.0001 \cdot 0.0000015 = \mathbf{1.5e^{-10}}$
P(Lačen sem|Have I hunger) $= 0.00000098 \cdot 0.00002 = 1.96e^{-11}$

From Example 2 we can see that the translation model assigned higher probability to the translation "I am hungry", which is the translation a human would choose. Values in both examples were calculated from our own test corpus and were used to show the difference between the translation model without a language model, and the translation model with a language model.

D. BiLingual Evaluation Understudy metric

Various metrics exist for MT quality evaluation, like BiLingual Evaluation Understudy (BLEU) [20], Metric for Evaluation of Translation with Explicit ORdering (METEOR) [6], etc. Translation quality is considered to be the correspondence between a machine translation and professional human translation. One of the first metrics to achieve a high correlation between a machine translation and professional human translation is BLEU metric. It is one of the more popular automated and inexpensive metrics. The closer a machine translation is to a professional human translation, the better it is. That is the central idea behind the BLEU metric. BLEU is quick, inexpensive, language-independent and correlates highly with human evaluation. There are many good translations of a given source text. These translations may vary in word choice or in word order even when they use the same words. Humans can clearly distinguish between a good and bad translation.

The primary task in BLEU metric is to compare n-grams of the candidate with the n-grams of the reference translation and count the number of matches. These matches are position-independent. The more matches, the better the candidate translation. The cornerstone of BLEU metric is the modified n-gram precision measure. Modified n-gram precision is computed similarly for any n:

- All candidate n-gram counts and their corresponding maximum reference counts are collected, and
- the candidate counts are clipped by their corresponding reference maximum value, summed, and divided by the total number of candidate n-grams.

Example:

Candidate: the the the the the the the.
Reference 1: The cat is on the mat.
Reference 2: There is a cat on the mat.

From the above Example we can calculate the standard unigram precision for the candidate sentence. Number of all words in the candidate sentence is 7. All of seven words "the" from the candidate sentence appear in the reference translations, so the maximum total count is 7, thus the standard unigram precision is $\frac{7}{7}$. For each unique word in the candidate sentence, we check how many times it appears in reference translations. Since we have only one unique word "the", it appears twice in reference 1 and once in reference 2, so the maximum total count is 2, thus the modified unigram precision is $\frac{2}{7}$.

In the above example, the candidate achieves a modified bigram precision of 0. This sort of modified n-gram precision scoring captures two aspects of translation: adequacy and fluency. The unigram scores are found to account for the adequacy of the translation - how much information is retained. The longer n-gram scores account for the fluency of the translation, or to what extent it reads like "good" English.

In practice, however, using individual words as the unit of comparison is not optimal. Instead, BLEU computes the same modified precision metric using n-grams. Another problem with BLEU scores is that they tend to favor short translations, which can produce very high precision scores, even when using modified precision.

BLEU's output is always a number between 0 and 1. This value indicates how similar the candidate and reference texts are, with values closer to 1 representing more similar texts. However, few human translations will attain a score of 1. The candidate text must be identical to a reference translation. For this reason, it is not necessary to attain a score of 1. Because there are more opportunities to match, adding additional reference translations will increase the BLEU score.

E. Usage of a Differential Evolution algorithm in Statistical Machine Translation

As shown in Figure 1, after we have built the SMT system, we can improve its translation quality by tuning it. Weights used by the Moses toolkit [11] to weight different models against each other are not optimized - there are default values within Moses configuration file. We use the DE algorithm to find better weights on the development set to get the tuned Moses configuration file that contains new weights. We translate using a tuned Moses configuration file and test set, and evaluate using BLEU metric to get the BLEU score.

The DE algorithm, as shown in Figure 2, uses a population of individuals, where each individual represent a vector of weights. From Figure 3 we have at least two or more weights, depending on the used models. For each vector from the current population g, called target vector, the DE algorithm generates a new vector, called trial vector, using mutation and crossover operations. At the selection we evaluate target and trial vectors using BLEU metric. The vector with the higher BLEU score, called offspring, "survives" to the next generation $g + 1$.

IV. Experiments

A. JRC-Acquis Corpus

Experiments were carried out for the English-Slovenian language pair in both directions using the freely available JRC-Acquis corpus [23]. It contains 23,545 English and 20,642 Slovenian texts. These texts have been written from the 1950s until now, and are a collection of legislative texts from European Union (EU) law. The corpus was preprocessed in three steps:

- tokenization of the text,
- lowercasing the entire text,
- removing too short and too long sentences.

The corpus was split and preprocessed into training (600,000 sentences), development (500 sentences) and test (2,000 sentences) sets. For the training set we removed sentences that were longer than 80 words. For the development and test set we removed sentences that were shorter than 8 and longer than 60 words.

B. Building SMT systems

In our experiments we built two SMT systems:

- English-Slovenian
- Slovenian-English

In our experiment we built a 5-gram language model with the IRST Language Modeling (IRSTLM) toolkit [7], removed singletons, smoothed with improved Kneser-Ney and added sentence boundary symbols. The translation model was built with word-alignment (using GIZA++ [18]), phrase extraction, and scoring. The language model was used to ensure fluent output, so it was built with the target language. The translation model created lexicalized reordering tables. The SMT system generated a Moses configuration file which contained models' parameters (weights). We then used this configuration file together with models to decode, i.e. translate. We binarized the phrase and reordering tables, i.e. compiled them into a format that could be loaded quickly because loading tables into the memory was very slow. We needed to find better weights because the weights had some default values, i.e. 0.2, 0.3, ...

C. Tuning SMT systems

In practice, currently the most popular way to find the parameters of SMT system is the Minimum Error Rate Training (MERT) [17], [9] method, and in this article we used the DE algorithm. DE optimizes this problem by maintaining a population of individuals and creates new individuals by combining existing ones according to its simple formula, and then keeping whichever individual has the highest fitness.

The following parameters defining the problem were:

- dimension problem D = 14,
- lower bounds ($\mathbf{b}_L$) = -1, and
- upper bounds ($\mathbf{b}_U$) = 1.

The following control parameters specified by us were:

- mutation parameter (F) = 0.5,
- crossover parameter (C_r) = 0.9,
- population size (N_p) = 20,
- maximum number of generations (g_{max}) = 50, and
- algorithm strategy (s) was DE/rand/1/bin.

The population was composed of 20 individuals where each individual represented a different vector of weights used to translate. The vector of weights was composed as following:

- 1 parameter for word penalty, annotated as wp,
- 1 parameter for phrase penalty, annotated as pp,
- 4 parameters for the translation model, annotated as tm,
- 6 parameters for the lexical reordering model, annotated as lr,
- 1 parameter for the distortion model, annotated as d, and
- 1 parameter for the language model, annotated as lm.

During the initial generation we generated 19 individuals with random weights within the lower and upper bounds, and 1 individual with default weights from the initial Moses configuration file. After initialization, the evolution process started. During the first generation, for each target vector we created a trial vector by choosing three individuals at random from the population which were different from the target vector, and performed crossover and mutation operations. During the selection we evaluated target and trial vectors using BLEU metric, and if BLEU score of a trial vector was better (higher) than the BLEU score of a target vector, the trial vector "survived" to the next generation, otherwise the target vector "survived" to the next generation. We repeated this process for g_{max} generations. After the final generation, we chose the best individual from the population and used its weights to test the SMT system.

D. Results

In Table I we have the BLEU score for the initial (default) weights. Slovenian-English and English-Slovenian SMT systems started tuning with the same initial weights. The Slovenian-English SMT system was evaluated using the BLEU score of 57.46% and the English-Slovenian SMT system with 56.35% on the development set. Table I contains the BLEU score for final (best) weights for Slovenian-English and English-Slovenian SMT systems. There was an improvement from 57.46% to 63.51% for the Slovenian-English SMT system, and for the English-Slovenian SMT system there was an improvement from 56.35% to 60.03% on the development set.

We tested both SMT systems on the test set using the initial Moses configuration file and the tuned Moses configuration

Table I
INITIAL AND FINAL PARAMETERS FOR THE SLOVENIAN-ENGLISH AND ENGLISH-SLOVENIAN MODELS

Weights	BLEU (%) ↑	
	Slovenian-English	English-Slovenian
Initial	57.46	56.35
Final (best)	63.51	60.03

file. We also tested MERT on the same test set for comparison. The results are shown in Table II. For the Slovenian-English SMT system using the DE algorithm, the BLEU score had improved from 40.61% to 46.05%, where MERT had a BLUE score of 45.74%. For the English-Slovenian SMT system using the DE algorithm, the BLEU score had improved from 38.78% to 39.62%, where MERT had a BLEU score of 39.10%.

Table II
RESULTS ON THE TEST SET

Algorithm	BLEU (%) ↑	
	Slovenian-English	English-Slovenian
Baseline	40.61	38.78
MERT	45.71	39.10
DE algorithm	46.05	39.62 †

MultEval [4] was used to help when evaluating the impact of internal experimental variations on translation quality. The improvement that is not statistically significant is marked with † in Table II. MultEval is a recognized tool within the field of SMT, which takes MT hypotheses from several runs of an optimizer and provides three popular metric scores, as well as standard deviations (via bootstrap resampling) and p-values (via approximate randomization). This allows us to mitigate some of the risk when using unstable optimizers.

V. CONCLUSION

We successfully built two phrase-based SMT systems for English-Slovenian and Slovenian-English language pairs. The DE algorithm was successfully used within the SMT systems. It improved the translation quality of a translated text according to BLEU metric within both SMT systems. In MT, getting a 100% BLEU score is virtually impossible, so if we can improve translation quality by only 1%, it means a lot - 1% closer to the perfect translation. Our SMT systems were both comparable or better than the SMT systems tuned using the MERT method. For the Slovenian-English SMT system the DE algorithm was significantly different statistically from the MERT, while for the English-Slovenian SMT system the DE algorithm did not differ significantly. From the obtained results we noticed that improving translation quality from English to Slovenian is a more difficult task than from Slovenian to English. As can be seen from the results, the basic DE algorithm performed well on improving the translation quality because DE is very efficient at finding the best parameters. In the future work we will try other DE versions, like jDE, GaDE, SaDE, etc.

REFERENCES

[1] T. F. Albat. US Patent 0185235, Systems and Methods for Automatically Estimating a Translation Time, 2007.

[2] L. Bungum and B. Gambck. Evolutionary Algorithms in NLP. *Norwegian Artificial Intelligence Symposium*, pages 7–18, 2010.

[3] C. Callison-Burch, M. Osborne, and P. Koehn. Re-evaluating the Role of BLEU in Machine Translation Research. *EACL*, pages 249–256, 2006.

[4] J. Clark, C. Dyer, A. Lavie, and N. Smith. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the Association for Computational Lingustics*, 2011.

[5] S. Das and P. N. Suganthan. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation*, pages 27–54, 2011.

[6] M. Denkowski and A. Lavie. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

[7] M. Federico, N. Bertoldi, and M. Cettolo. IRSTLM: an open source toolkit for handling large scale language models. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association*, pages 1618–1621, 2008.

[8] M. Guidre. Toward Corpus-Based Machine Translation for Standard Arabic. *Translation Journal*, 2002.

[9] M. Hopkins and J. May. Tuning as ranking. *EMNLP*, pages 1352–1362, 2011.

[10] P. Koehn. *Statistical Machine Translation*. Cambridgre University Press, 2011.

[11] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. J. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. *ACL Demo and Poster Session*, 2007.

[12] P. Koehn, F. Och, and D. Marcu. Statistical Phrase-Based Translation. *HLT-NAACL*, 2003.

[13] C.D. Manning, P. Raghavan, and M. Schtze. Introduction to Information Retrieval. *Cambridge University Press*, page 260, 2008.

[14] M. S. Maučec and J. Brest. Reduction of Morpho-syntactic Features in Statistical Machine Translation of Highly Inflective Language. *INFORMATICA*, 21:95–116, 2010.

[15] F. Neri and V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, pages 61–106, 2010.

[16] F. Och and H. Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, pages 19–51, 2003.

[17] F. J. Och. Minimum Error Rate Training for Statistical Machine Translation. *ACL*, pages 160–167, 2003.

[18] F. J. Och and H. Ney. Improved Statistical Alignment Models. In *ACL*, pages 440–447, 2000.

[19] E. Otto and M. C. Riff. EDA: An Evolutionary Decoding Algorithm for Statistical Machine Translation. *Applied Artificial Intelligence*, pages 605–621, 2007.

[20] K. Papineni, S. Roukos, T. Ward, and W-J. Zhu. BLEU: a method for automatic evaluation of machine translation. *ACL*, pages 311–318, 2002.

[21] K. Price, R. Storn, and J. Lampinen. *Differential Evolution, A Practical Approach to Global Optimization*. Springer, 2005.

[22] L. Rodrguez, I. Garca-Varea, and J. A. Gmez. On the application of different evolutionary algorithms to the alignment problem in statistical machine translation. *Neurocomputing*, pages 755–765, 2008.

[23] R. Steinberger, B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, D. Tufis, and DAniel Varga. The JRC-Acquis: A Multilingual Aligned Parallel Corpus with 20+ Languages. *LREC*, 2006.

[24] R. Storn and K. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

[25] R. Storn and K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimisation Over Continuous Spaces. *Journal of Global Optimization*, pages 341–359, 1997.