Rémi Vincent

Multilingual Poetry Generation

Roses are red, Les violettes sont bleues

Master's thesis in Information Systems
Supervisor: Björn Gambäck
August 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Master's Thesis

Roses are red, Les violettes sont bleues

Multilingual Poetry Generation

Rémi Vincent

August 2019

Master's Thesis in Information Systems
Supervised by Björn Gambäck
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

At the crossroad of Natural Language Processing and Computational Creativity, poetry generation is a complex task, partly due to the different forms it can take and the subjectivity of its evaluation. Because of their inherent nature, poetic texts cannot be translated from one language to another using machine translation models without losing too much information and properties of the poem. This thesis explores different solutions to generate poetry in multiple languages.

Four different deep learning models are developed to produce multilingual poetry, focusing on characteristics such as grammaticality, meaningfulness and poeticness of the resulting texts. The last system — called *MultiLingual Poetry Generator* — combines a multilingual phoneme-based with a language-specific LSTM. It is able to output poetry in both French and English with a 15.2% increase in terms of poeticness, assessed by human evaluation.

The results show that it is possible for a model to successfully learn and generate poetry in multiple languages, opening the way to multilingual poetry generation.

Preface

This Master's Thesis has been written for the Norwegian University of Science and Technology (NTNU). It is the result of 6 months of work, as a conclusion of the degree of Computer Science.

The thesis was supervised by Björn Gambäck, who is Professor of Language Technology at NTNU.

I would like to personally thank my supervisor Björn for the time he dedicated and the guidance he provided in this subject. I would also like to thank Jianlong Fu, who gave me hints on his marvelous paper and led me towards the study of multilingual poetry generation. Finally, thank you to my friends at MILA who supported me in this project.

Poetry is what gets lost in translation.

— Robert Frost

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the initial development of computer science, attempts have been made at closing the gap between humans and machines. It has been quite obvious so far that computers are able to surpass humans in tasks involving a highly-reliable memory or expanded data capabilities. Yet other tasks that appear simple to us keep challenging even the most advanced computers.

Among these tasks, many involve the use of natural languages to solve a given problem. Instinctively, one can comprehend the difficulties that are raised by the learning of natural languages, compared to other fields such as computer vision. Natural languages usually contain million words that can be put together in billions of different ways. It is not uncommon to see long-term relationships between these words and identical words having very different meaning depending on the given context. Whereas an image contains millions of pixels with a large continuous range of possible values, a sentence contains a few words that look like very sparse information in the view of the computer.

Of course, recent developments, especially in the field of deep learning, have led to some breakthroughs. Computer models have been successfully applied to real-life language processing tasks such as translation (Edunov et al. [2018], Wu et al. [2019]) or grammatical error correction (Zhao et al. [2019], Ge et al. [2018]), with results closing the gap in accuracy with human attempts. However there is still a crucial difference between the way humans and computers learn a new natural language, which could explain the persisting issues that are faced. At a young age, humans

learn their native language from a few examples given in different contexts. In the meantime, computers are usually flooded with billions of examples and different strategies are applied to help them figure out linguistic information.

Some of the most complex tasks that computers can face as regards to natural language are the ones involving creativity. Creativity is often deemed to be a human-specific characteristic that would be impossible for machines to reproduce. Yet many efforts have been made in the past to develop computational creativity.

In this work, the focus is on computer-based generation of poetry. Can a computer write poems in a way that would be considered creative by human evaluators? Can this generative process be generalized to different languages or is it necessarily language-specific?

A few attempts at generating poetry in multiple languages have been made in the past (Oliveira et al. [2017]), but they have rarely leveraged the development of deep learning methods and have always involved a lot of manual work every time a new language was to be added.

Poetry is more than just writing meaningful sentences. If the focus is on meaning only, then working with multiple languages presents few advantages. However some characteristics of poetry — including aesthetics, phonetic attributes, rhythm and overall creativity — translate in multiple languages. With that in mind, generalizing the task to multiple languages can present many advantages, including helping the model to achieve true poeticness and to avoid overfitting to a specific dataset.

In addition, while our knowledge of human learning is often applied to the development of algorithms, attempting to make computers solve creative tasks in multiple languages could help us understand how creativity operate in different languages.

1.1 Main objectives

The general objectives of this project are:

1. To understand the role played by the chosen language in the process of learning poetry generation. Are there differences? Are some languages easier to use for

training than others? In what aspects?

2. To reuse this knowledge to improve traditional networks in the task of generating human-like poetry.

In particular, there are different approaches that are explored to achieve multilingual poetry generation:

1. Using the same network for each language but train them on different datasets. This could be useful to highlight the possibilities and limitations of poetry generation in different languages.

2. Using both the same network and the same training data, whatever the language. This makes the whole task more complex for the network since it has to learn the differences between the languages. This approach stresses the complexity of the task and the differences in learning for each language.

3. Using a combination of the two previous points and build a network in two parts. The first one is common to each language and trained with a single combined dataset. The second is specific to each language, trained on different datasets. This way the network is fed with way more data and such a system leverages the fact that there is not so much difference in the structure of the network needed to learn a language rather than another.

1.2 Contributions

The following contributions are made to the field through this work:

1. The development of a *MultiLingual Poetry Generator*, able to produce poetry in both French and English

2. A dataset of English and French poems, totaling around 50k verses, built from scratch

3. A general study of regular and multilingual poetry generation, with state-of-the-art models that have been recently developed

4. An exploration of the possible approaches to multilingual poetry generation, with the corresponding architectures and results (1.1)

1.3 Plan of the report

The general structure of the report is as follows:

1. Understanding what has been done in the field so far in terms of natural language processing and more specifically in terms of poetry generation (ch. 2)
2. Exploring the latest results and how different methods have been combined to always push the field further (ch. 3)
3. Introducing the conducted experiments to explore multilingual poetry generation (ch. 4)
4. Detailing and discussing the results as well as what could be done in further work (ch. 5, 6)

Chapter 2

Background Theory

Many concepts and theories are involved in the development of a multilingual poetry generation system. Decades of research in the fields of Natural Language Processing, Deep Learning and Computer Creativity have led to many major results that are essential to the understanding of this project, mainly because they are often implicitly considered in experiments and discussions.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a field of computer science that focuses on the interaction between computers and humans, using a natural language, namely a language that has evolved in a natural way, as opposite to an artificial language.

As such, the study of NLP attempts to find ways to analyse and process large amounts of natural language data.

Among the most common goals of NLP, a wide range of tasks can be mentioned (tab. 2.1).

All the tasks are usually quite different from each other and require different methods. Although most of them can relate to poetry generation, two of them have a strong impact and are described below.

Task	Goal
Automatic speech recognition	To recognize and translate natural speech
Part-of-speech tagging	To associate a word with its corresponding grammatical category
Machine translation	To translate a sentence from one language to another
Question answering	To answer close, closed-ended or open-ended questions
Sentiment analysis	To classify a text in terms of polarity (positive, negative, neutral)
Summarizing	To render the meaning of a given text with a smaller text
Text classification	To assign a text to a category among a given list
Text generation	To predict the next character or word in a given document

Table 2.1: Different NLP tasks with the corresponding objective

2.1.1 Language modeling

Language modeling aims at computing the statistical probability of a sentence. Specifically, given a sequences of entities $(w_1, w_2, w_3, ..., w_k)$, the goal is to compute the probability P so that $P(s) = P(w_1, w_2, w_3, ..., w_k)$.

This is particularly important for text generation, since it becomes possible to deduce the probability of an upcoming entity $P(w_{k+1}|w_1, w_2, w_3, ..., w_k)$.

There are two types of language models:

1. **Character-based language models**. The sentence is looked upon at a character-level, which means that we try to generate a new character given the previous ones. These models usually need a smaller vocabulary, so they are more memory-efficient and suit better rich morphologies such as the Finnish or Russian languages.

2. **Word-based language models**. The sentence is looked upon at a word-level, which means that we try to generate a new word given the previous ones. These models are usually less expensive to train and display a higher accuracy, however tokenization is needed to account for the large vocabulary (Graves [2013]).

Whatever the model chosen, the Markov assumption is usually made to simplify the problem and make the computation possible. Under this assumption, we state that $P(w_{k+1}|w_1, w_2, w_3, ..., w_k) = P(w_{k+1}|w_{k-n}, w_{k-n+1}, ..., w_k)$ with $0 < n < k$. In other terms, the next word/character only depends on a subset of the previous ones. The

corresponding model is called an *n-gram* model and most text-generating algorithms are based on this property.

2.1.2 Part-of-speech tagging

Part-of-speech (POS) tagging is the task of asserting the grammatical category of a word in a given sentence. This task is particularly important in the view of generating syntactically correct sentences as they help constrain the generated text.

An example of such task is depicted in tab. 2.2.

Sentence	POS tags	Meaning
Generating	VBG	Verb (present participle)
poetry	NN	Noun (singular)
is	VBZ	Verb (3rd person singular present)
amazing	JJ	Adjective
!	.	Punctuation

Table 2.2: An example of POS-tagging for an English sentence

Different methods can be used for POS-tagging. Lexical-based methods look at the frequency of a given word in the corpus. Rule-based methods implement many different rules on the patterns that must be present in a word to have a given tag. Probabilistic methods use Conditional Random Fields or Hidden Markov Models to assign the tag. Finally, deep learning methods use artificial neural networks such as Recurrent Neural Networks to classify words according to tags.

Although difficult and highly dependent on context, POS-tagging has reached high accuracy. Usually its performance in English is evaluated on the Penn Treebank dataset, with an accuracy over 97%.

2.2 Text Generation

In a text generation task, the model attempts to predict the next character — or word — in a given document. By stacking together the predicted characters — or words, a full text is being generated.

There are two different approaches to text generation:

- The system can produce the text by selecting the appropriate word or sentence from a pool of candidates.
- The system can produce the text from scratch, without comparing possible candidates.

2.2.1 Retrieval-Based Text Generation

Building text generation systems is a challenging task that can easily lead to a lot of struggle in the learning process. Developing a system that provides meaningful and appropriate sentences altogether is hard, particularly if one operates on an open domain, i.e. there is no specific topic. To ease the work of the system, one can predefine a set of allowed sentences, possibly through the use of templates, and train the agent to identify and fetch the most relevant of them. This was one of the historical approaches to text generation, and some real-worlds systems still follow this pattern, which provides a few advantages:

- The sentences are always syntactically correct and grammar-error free (although not necessary meaningful with regards to the current context).
- The system is easier to train, and one can concentrate on the choice of sentences rather than their elaboration.
- The domain is tightly controlled, and there are fewer risks of ethical issues in the answer.

2.2.1.1 Information Retrieval

Building Retrieval-Based Text Generators is an Information Retrieval problem, since the intent is to find material (the next utterance) of an unstructured nature that satisfies an information need from within large collections (the set of possible sentences). The concerns are similar to those of Information Retrieval systems and the same components need to be used:

1. A collection of documents with a given index

2. A user query that must return one or several documents from the collection

3. A ranking method that is learned by the system and allows the retrieval of the most relevant documents

The main difference lies in the fact that the agent is its own user as it provides queries and results by itself, without any external intervention. This leads to a slightly different functioning in the case of a text generator, with the following components:

1. A knowledge base containing the pairs of message-response that have been stored during training and during the previous part of the discussion. In some systems, it is possible to use templates rather than fixed strings of text, for better performances.

2. A candidate generation process, that retrieves (in one or multiple steps) the most relevant answers.

3. A response generator, that outputs the best match.

To evaluate such a system, the common metric that is being used is called **recall**. Recall is a measure of the probability for the system to retrieve relevant answers, defined as follows:

$$recall = \frac{|\{relevant\_answers\} \cap \{retrieved\_answers\}|}{|\{retrieved\_answers\}|}$$

Other metrics could also be used, such as precision, which measures the fraction of answers retrieved that are relevant. However recall is usually a better way to go, since we only return one sentence in the end, and we are mainly interested in how relevant this answer is. In that case, recall is called *recall at 1* and is abbreviated *r@1*. We can also define *r@N* with $N > 1$ if we wish to evaluate our systems on the possible answers it could have delivered but were not actually classified at the top position.

2.2.1.2 TF-IDF

The standard way of capturing the relevance between an answer and a query is through distributional methods, by evaluating the occurrence of words in both strings. To do so, a straightforward approach is to work with TF-IDF, which stands for *Term Frequency - Inverse Document Frequency*. TF-IDF is a numerical statistical method that assesses the weight of a word in a given document by taking into account the whole sets of documents. A word is important in a given document if it is frequently used in it, but barely used in other documents of the corpus.

Mathematically, we can define the Term Frequency of a word w_0 to be:

$$TF(w_0) = \frac{c(w_0, D)}{\sum_{w_i \in D} c(w_i)}$$

where c is a function that maps a word to its number of occurrences.

In the same way, we can define the Inverse Document Frequency of a word w_0 to be:

$$IDF(w_0) = log(\frac{N}{N(w_0)})$$

where N is the total number of documents and N(w) the number of documents containing the word w.

Finally we compute TF-IDF(w) as the product of the two previous results:

$$TF - IDF(w) = TF(w) * IDF(w)$$

2.2.1.3 Assessing Similarity

The main challenge of Information Retrieval Systems, in particular when applied to text generation, is to find a way to estimate the similarity between the query and the possible outputs. One way to achieve it is by embedding the sentences and then using cosine similarity, where the similarity S between two word vectors $\vec{q}$ and $\vec{a}$ is computed as follows:

$$S = cos((\vec{q}, \vec{a})) = \frac{\vec{q} * \vec{a}}{||\vec{q}|| * ||\vec{a}||}$$

This is a simple but efficient approach than can be found in a wide range of systems and problems.

On the other hand, it requires to have a set of possible answers, and for this reason does not generalize to any system, especially when the quantity of information being stored is important. Case-Based Reasoning approaches can be used to ensure a better generalization. In this case the goal is to solve a problem given similar past problems. In particular, one can use k-Nearest Neighbors search methods to select the set of possible answers while avoiding partially the curse of dimensionality that arises from the traditionally high number of dimensions and the resulting sparsity of the search space. It includes different methods to outperform linear search, such as space partitioning or locality sensitive hashing.

2.2.1.4 Word Embedding

To compute more directly the relevance of an answer given an utterance, one needs a mathematical operation that could return a result in an ordered vector space, in which it would make sense to compare the relevance of possible answers. Such an operation can hardly be found in a textual space, and as a consequence we need a way to embed our strings of text into another vector space. In addition to being ordered, we ask this space to be continuous (for an easier manipulation of the vectors), usually with a lower dimension (to remedy the curse of dimensionality). A common approach is to represent the words of the vocabulary as one-hot vectors having the size of the vocabulary itself, and map them to vectors of real numbers: $W : word \mapsto \mathbb{R}^n$. If correctly chosen and trained, such a word embedding can capture the meaning of the words and allow the ones that are close in meaning to be also close in Euclidean distance (fig. 2.1).

The embedding W can be done with a matrix of shape (voc_size, $embedding\_dim$), which serves as a look-up table to get the mapping of each word of the vocabulary. The matrix is initialized with random values and then these values are updated through training to achieve the relevant representation, the one that captures the best the meaning of the words for the current task.

It is also possible to use a pretrained representation. Even though it is not specialized to any task, it has already captured the meaning of words and can then be fine-tuned

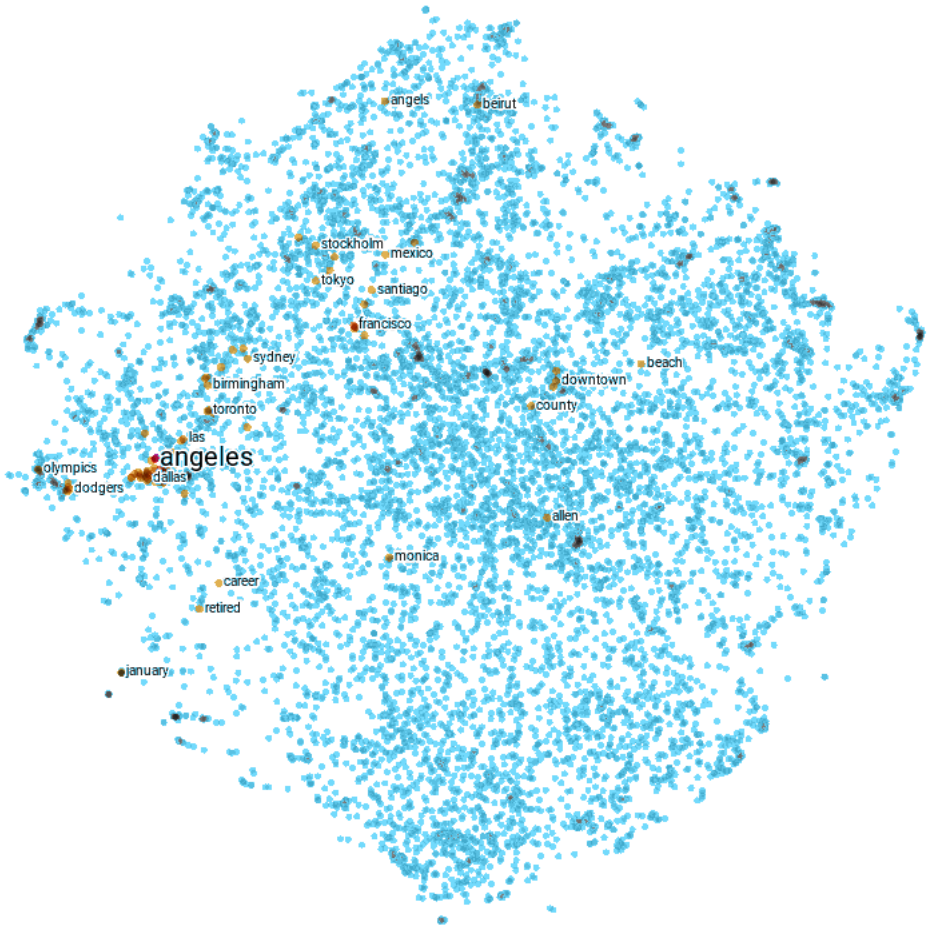


Figure 2.1: t-SNE representation of words, embedded into a 2D space of real numbers, using tensorboard

to get closer to the desired representation. It allows a faster learning since the network does not need to build a word representation from scratch.

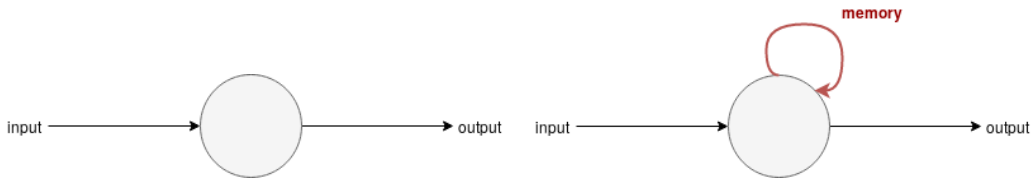
Among the pretrained representations used in state-of-the-art projects, GloVe initialization is performed in multiple projects. Built upon the *word2vec* method (Mikolov et al. [2013]), the model uses matrix factorization methods to exploit global statistical information and capture co-occurrence in the data. Trained on *Gigaword* and *Wikipedia* corpora, it achieved state-of-the-art results on various word analogy and similarity tasks (Pennington et al. [2014]). The pretrained vectors can be used with different embedding dimensions and have been applied so far to a wide range of NLP tasks.

2.2.1.5 Deep Learning Methods

Among the recent methods that have been used efficiently in the NLP field, most of them are related to Deep Learning. Either as a way to extract relevant features in the word representation (Prakash et al. [2016]) or to compute the similarity (Lowe et al. [2015]), Deep Learning provides a key data-driven approach for building the retrieval process.

A popular method for text generation is the use of Recurrent Neural Networks (Xie [2017]).

Recurrent Neural Networks (RNNs) are a subclass of artificial neural networks where connections between nodes form a directed cycle. As a consequence, RNNs display an internal state that is not present with regular feedforwarding networks (fig. 2.2).



(a) Feedforward neural node (b) Recurrent neural node

Figure 2.2: Main difference between recurrent and feedforward neural networks

Using this internal state allows the network to memorize previous inputs and use them to compute the next one (fig. 2.3). By doing so, they get closer to n-gram models and are a good fit for sequence generation.

This is particularly useful in text generation, since inputs and outputs have arbitrary lengths that are hardly managed by feedforward neural networks. By using their internal state, RNNs are able to process the temporal information present in the textual data and output a new word or a confidence score in a given match.

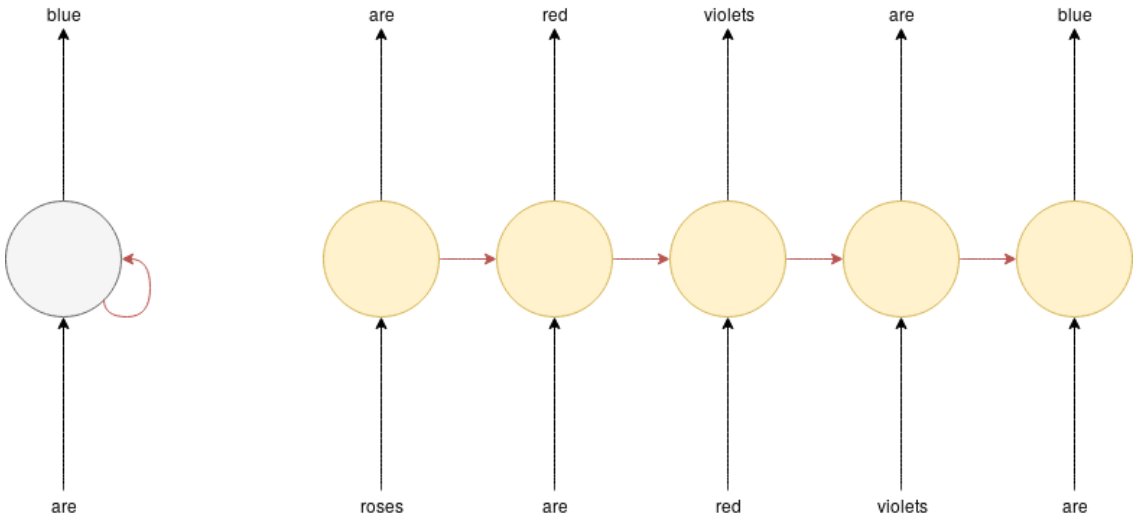
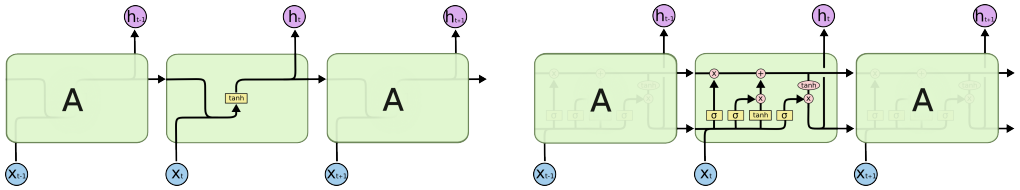


Figure 2.3: Unfolded RNN for a word-based text generation task

The main problem of RNNs as such is the lack of management for long-term dependencies, which makes it hard for the network to remember information beyond the last few words or sentences, depending on the scale at which we are operating. To remedy the short-term memory of RNNs, Long Short-Term Memory (LSTM) has been developed as a special kind of RNNs focusing on the management of stored information. As opposed to traditional RNNs, LSTM provides two gated units that learn to open and close access to error flow within each memory cell (Hochreiter and Schmidhuber [1997]).

This is illustrated in fig. 2.4, which highlights the differences in the repeating modules of RNN vs. LSTM, with the latter providing a much more complex architectures with the presence of input, output and forget gates in the form of sigmoid operations.

LSTM has been applied successfully to Information Retrieval. In Palangi et al. [2014], LSTM is applied to capture contextual dependencies in sequences of words in order to retrieve relevant documents in a web search. In Kadlec et al. [2015], a Bi-Directional LSTM processes context information in both directions to achieve state-of-the-art results in the Ubuntu Dialog Corpus challenge.



(a) Repeating module in the RNN (b) Repeating module in the LSTM

Figure 2.4: Main difference between RNN and LSTM. Illustrations taken from Olah [2015] with the permission of the author.

2.2.2 Generative Text Generation

As opposed to Retrieval-Based Text Generators, generative systems do not fetch the next utterance from a set of existing sentences, but rather build a new utterance from scratch. As a consequence, there is no inherent limitation in what the output of the system could be and it is theoretically able to handle any type of discussion, which makes them especially important whenever creativity is needed. In practice, building such a system is hard and some constraints are provided to the system. Although some systems work at a character level, by providing the next letter given the past ones, most of the existing generative systems are built at the word level, where a new word is being fetched given the previous ones. On the other hand, retrieval-based systems usually operate at a sentence level.

The main issues with such systems is the difficulty to achieve meaningful conversations, in addition to the issues that applied to the retrieval systems, such as consistency in the output and relevance. Generative Models are currently a major focus in language modeling and many researches are performed to train them (Wolf et al. [2019], Al-Rfou et al. [2019]).

2.2.2.1 RNNs and LSTMs

Generative Text Generators are mostly being trained using Deep Learning methods, since manual engineering of features is a challenging task for such a wide range of output. Many approaches tend to use Recurrent Neural Networks and Long Short-Term Memory to generate an utterance from a given context (Mikolov et al. [2014], Serban et al. [2016]). These methods can include working character by character or word by word and training the system on huge quantities of unlabeled

data, in an unsupervised way. Although the network learns quickly how to generate possible words, it hardly extends to meaningful and relevant sentences.

In Mikolov et al. [2014], an upgraded RNN is successfully built to capture context features in sequential data before being trained and evaluated on the Penn Treebank Corpus. The system outperforms LSTM results on the same set and manages to predict efficiently the next word of the current utterance. In Serban et al. [2016], a Generative Hierarchical Neural Network Mode is built upon encoder and decoder RNNs. The system achieved state-of-the-art results on the *MovieTriples* dataset, in the task of modeling the next word in the utterance. Other approaches have been investigated in the task of building a Generative Model, such as n-gram methods, but the task of building a functional generative agent is still far from being solved.

2.2.2.2 GANs

Recently, a new deep learning architecture has been successful at generating data for various purposes. Called Generative Adversarial Networks (GANs), this architecture was introduced by Ian Goodfellow in 2014 (Goodfellow et al. [2014]). In a GAN implementation, two adversarial systems face each other. One is a Generator, whose goal is to produce candidates that match the training data distribution, while the other is a Discriminator, whose goal is to distinguish the candidates generated by the Generator from the true data distribution. By doing so, both systems eventually get better at their task, and the final form of the Generator can be used to generate new data that would match the initial data distribution.

Although widely used for image generation, GANs have also been applied to NLP problems. In (Chen et al. [2017]), adversarial training is successfully used for cross-domain image captioning. In Yu et al. [2017], GANs are applied directly to the generation of sequences of discrete tokens, via the use of policy gradient. GAN is still a recent breakthrough, but both these architectures tend to show the promises of such a model for the generation of text.

2.3 Computational Creativity

Computational creativity is the use of computers to generate results that would be regarded as creative if produced by humans alone (Besold et al. [2015]). A result can be anything ranging form a theory to a piece of music, although the term is often limited to results of artistic nature.

Creativity has long been deemed an exclusivity of humans because of the nature and difficulty of the task. Indeed, for a result to be considered as creative, it would have to present the following characteristics (Boden [2004]):

- Be novel
- Be surprising
- Be valuable

Boden [2007] distinguishes two ranges of creativity. H-creativity is the production of ideas that are highly-valuable, meaning that nobody has ever had them before. On the other hand, P-creativity is the production of ideas that are new for the one producing them, but not for everyone else. Although H-creativity is obviously more difficult to reach, P-creativity is often the focus as it is much easier to obtain and the idea-generation process remains basically the same.

It can be interesting to sort creativity outputs by the psychological process being performed (Boden [2007]). As such, creativity can be:

- Combinational, in which familiar ideas are combined to produce unfamiliar ones. This is usually the most commonly referred form of creativity, and it is widely present in poetry, as poets like to juxtapose different or even contrary concepts in order to surprise the reader.
- Exploratory, in which existing rules or conventions are used to generate novel ideas, by testing their potential and limits. The new idea is often a variation of an existing one, and as such this is the type of creativity that can usually be found in arts and sciences.

- Transformational, in which the previous variation goes further and becomes a real transformation of an existing idea. The resulting idea becomes way different from the initial one and thus difficult to accept. This type of creativity is more rare and traditionally valued higher than the others. However, its use is less common in written forms of art, which already use a constraining language.

2.4 Automated Poetry Generation

At the crossroad of Computer Creativity and Natural Language Processing, automated poetry generation attempts to capture the complexity of a language in order to produce poetry that is both creative and meaningful.

2.4.1 Defining poetry

> "Poetry is the record of the best and happiest moments of the happiest and best minds."
> — Percy Bysshe Shelley

Although everyone has read some poetry, coming up with a good definition of it can be an arduous task. What is poetry? The genre has had so many different forms in the past, ranging from prose to verses, that it is difficult to identify common characteristics.

Since the first known forms of poetry were written over five thousand years ago, many definitions have been attempted. American poet Rita Dove described it as a *language at its most distilled and most powerful.* Edgar Allan Poe defined it as *the rhythmical creation of beauty in words.* Although not precise, these definitions show that the focus of poetry lies in the use which is made of a language to produce a rhythmic and artistic writing.

As such, poetry could be described as another way of using a language that conveys meaning while stressing the aesthetics and rhythm of the language through phonaesthetics and meters.

In practice, poetry is often recognized by its dependence to the *line*, a unit of language that divides the poem while not necessary coinciding with the grammatical markers of the text. Sometimes, the line is a verse, and the identification becomes facilitated by the use of rhymes, a repetition of similar sounds in different words.

Among the different types of rhymes, we can mention:

- End rhymes, where the lines' final words are rhyming.

  Roses are red, violets are blue,
  I like rhymes, and so do you.

- Internal rhymes, where the repetition occurs within the same line.

  The jar was round upon the ground.

- Slant rhymes, where an imperfect repetition is used and only the vowel sounds are the same.

  Between my finger and my thumb
  The squat pen rests; snug as a gun.

Producing computer-generated verses represents a challenge, but it also makes the output more easily recognizable as poetry and for this reason it is common to generate verses rather than prose. However, simply having a machine capable of producing verses is obviously not enough to achieve human-like creativity.

2.4.2 Evaluating poetry

What makes a text poetic? Given the difficulty of coming up with a good objective definition of poetry, we can understand the complexity of such a task. Yet it is only by defining poetic criteria that we can decide whether a given output is relevant or not.

To lower a bit the complexity of this NLP task, researchers have come up with three criteria when it comes to evaluating poetry (Manurung [2003]):

- Grammaticality
- Meaningfulness
- Poeticness

With grammaticality, we wonder whether the text obeys the linguistic rules set by a given grammar/lexicon. What we want is to generate syntactically correct sentences and avoid the generation of other random word that comes with undertrained systems.

With meaningfulness, we look at the conceptual message that is conveyed and whether it is meaningful under some interpretation. If a word or image is used as a topic, as we shall see later, we want the output to relate to it.

Finally, poeticness attempts to assess whether the text exhibits the poetic features that have previously been described. Is there expressiveness in the output? Is there aesthetics? This is definitely the most subjective criteria and as a result it is often hard to compare two systems based on this metric.

2.4.3 Generating poetry

Automated poetry generation is the act of using a computer program to assist in or be fully responsible for the production of poetic text.

In the past fifty years, many systems have been developed in this regard, each one attempting to push further the limits of computational creativity.

Gervás [2019] distinguishes different types of poetry generators based on the overall architecture being used by such systems:

1. Template Based Poetry Generation
2. Generate and Test Approaches
3. Evolutionary Approaches
4. Case-Based Reasoning Approaches

In Template Based Poetry Generation, the generation comes from the building of a set of words, placeholders and possible transformations that are applied to generate a poetic text. In this approach, the number of versions of the poem that can be produced by the given input is limited. An example of such a process could be to combine the sentence structure of one poem with the vocabulary of another one, by identifying and replacing specific grammatical categories in the sentence. One could also add constraints on rhymes, meters, word frequencies and similarities. The resulting poems tend to be syntactically correct but one can wonder whether they are truly creative, due to their inherent lack of novelty (Oliveira [2017]). *RACTER* (Chamberlain [1983]) is an example of such a system. Some models also use this approach partially, for instance *Full-FACE Poetry Generation* from Colton et al. [2012].

In Generate and Test Approaches, a set of requirements is defined as an initial step. The system then iterates over user inputs — they can be a selection of vocabulary, prior poems, keywords — to produce random sequences and test them against both the inputs and requirements. After a few iterations, the output tends to perform well based on rhymes, metrics and other phonetic properties. However because the freedom of the system is greatly constrained, the amount of semantic creativity of such outputs is often limited. *WASP* (Gervás [2000]) is an example of such a system.

In Evolutionary Approaches, an initial population of poem candidates is generated. Inspired by biological evolution mechanisms, the candidates are subjected to reproduction, mutation and recombination that produce a new population, of which only the best elements are kept. The process is repeated until it reaches a local extremum. The advantage of such a system is that it is able to search a large space of possible outputs. However, it requires a way to tell which one of two candidates performs better, an evaluation that can be difficult to process automatically. *MCGONAGALL* (Manurung [2003]) is an example of such a system.

In Case-Based Reasoning Approaches, similarity measurement is performed between a given user input and a set of existing lines or verses. The closest matches are identified by the system (RETRIEVE) and they are adapted to fit the required content, for instance using the POS structure of the input (REUSE). A manual validation can be asked to the user (REVISE), before adding the generated poem to the set of existing lines (RETAIN). *ASPERA* (Gervás [2001]) is an example of such a system.

Structuring poetry generation by the technical process involved works with most of the original systems, yet in the past few years some other techniques have emerged that hardly classify into one of the above categories.

Lamb et al. [2017] use a different taxonomy based on the purpose for which a given process is used rather than its technicality.

1. Mere generation gathers all the methods that produce text based on a random or deterministic algorithm. In this category, we find methods such as:
   (a) Templates
   (b) Markov chaining
   (c) Context-free grammars
   (d) Found poetry
2. Human enhancement gathers all the methods that build upon the interaction with a human to perform better. An example of such a method is Gnoetry (Elshtain [2006]), in which the generation process is based on a dialogue between the human and the computer.
3. Computer enhancement gathers all the methods that make use of optimization processes to drive the generated poetry towards coherence and artistic style. In this category, we find methods such as:
   (a) Data mining and knowledge representation
   (b) Optimization, whether it be:
      i. Hill-climbing search
      ii. Generate-and-test method
      iii. Genetic algorithms
      iv. Case-based reasoning approaches
      v. Recurrent Neural Networks (RNN, LSTM) and Autoencoders (VAE). Those can be:
         A. Character-based
         B. Word-based

Chapter 3

Related Work

Since the first attempts at generating poetry over fifty years ago (Queneau [1961]), a lot of results and breakthroughs have been made in the field. This chapter covers the most important progress that has been made as regards generating creative and human-like poetry.

3.1 Historic breakthroughs

3.1.1 Pablo Gervás and the Case-Based Reasoning Approach

Gervás [2001] introduces ASPERA and becomes one of the first researchers to compose formal poetry in a semiautomatic manner.

For the most, ASPERA, which stands for Automatic Spanish Poetry Expert and Rewriting Application, uses Case-Based Reasoning techniques to find the most relevant verse that corresponds to a set of message fragments, selected vocabulary and strophic forms determined upon user input.

The different steps are as follows:

1. The user inputs the desired parameters into the system (mood, message, other settings).

2. The system searches the knowledge base for the most appropriate strophic form.

3. A first draft is made by combining the message with the retrieved strophic form.

4. A vocabulary as well as a corpus of relevant examples are retrieved for each fragment of the poem. One of the examples is isolated and its POS structure is extracted then filled in with words from the vocabulary.

5. The corresponding output is validated by the user.

6. Validated poems serve to enrich the knowledge base.

The use of ASPERA leads to poetic and syntactically correct outputs. However, semantics is not strictly enforced and poems are not necessary fully meaningful (Gervás [2001]):

Andando con arbusto fui pesado vuestras hermosas nubes por mirarme quien antes en la liebre fue templado.

which literally translates to English as:

Walking with bush I was heavy your beautiful clouds for looking at me who before in the hare was tempered.

Using a Case-Based Reasoning approach can be time-consuming since the system has to go through multiple steps during the generation of the poem. Moreover, such a system is heavily dependent on its knowledge base and fine-tuning.

3.1.2 Hisar Manurung and the Evolutionary Approach

Manurung [2003] sees poetry generation as a state space search problem, which is explored by a genetic algorithm in order to construct poems. The developed system, McGONAGALL, uses a linguistic formalism to represent its genomics information, and is loosely constrained by semantic information.

More specifically, the system performs the following steps:

- Generating a set of candidates, based on some initial data as well as the target output form.
- Evaluating the candidates using a score function that is based on surface form, phonetic pattern and semantics.
- Evolving the population of candidates into a new one, using traditional evolutionary processes such as selection, mutation and reproduction.

According to the author, the system is able to generate meaningful, poetic (although limited to meter similarity) and somehow grammatically correct text (Manurung [2003]):

A lion, it dwells in a waste.
A lion, it dwells in a waste.
A waste will be rare.
Its head will be rare.
Its waist, that is small, will be rare.

However, the poems also tend to stay close to the input data and the author deliberately chose to put aside "the subjects of creativity and artistic theory" (Manurung [2003]).

3.1.3 Eric Elshtain and the Interactive Generate-and-Test Approach

Elshtain [2006] use a different approach for poetry generation. Rather than looking for full automation besides the initial user input, the Gnoetry program allows a human user to participate in the poem generation by evaluating and correcting the generated n-grams. As stated by the authors, the result becomes *a true collaboration of equals.*

At each step, Gnoetry displays an interface that can be used by the human to select relevant sets of words for regeneration. For every generated sequence, the first word

makes a bigram with the word before it and the last word makes another bigram with the word after it, linking sequences together.

Because it is a fully interactive system, Gnoetry cannot be really qualified as an automated poetry generator. The produced poems can often be mistaken for human-written poems, but it is mostly due to the fact that a heavy human intervention was involved:

It is not what I had judged.
It is the gift of desire absorbed in
itself. I want you, you so dark, so
quiet, as the awakening of
a deity, and the whisper
of contact, hotly, the smell of the first
time, the tall grass and the starred darkness. A
door opened, closed. And we crept on,
and looked about. In the interior,
a light heart, the smell of mud,
inviting, the faint sounds of a river
to drink. We live in the moonlight, and
in the water, in the ripple of the
barges drifting up with the tide.

3.1.4 Hugo Gonçalo Oliveira and the Template-Based Approach

Oliveira [2012] details the functioning of PoeTryMe, a highly customizable platform for automatic generation of poetry.

The user can choose one of the poem templates, along with seed words, that are combined through a generation strategy to produce new poems.

The generation is based on a modular architecture and built upon a knowledge base. More specifically:

1. The Generation Strategy gathers the user input and a poem template. It seeds

the Sentence Generator with input terms and waits for the returned verse.

2. The Sentence Generator uses the seed terms to select a semantic subgraph from the Triples Manager. It calls the Grammar Processor to retrieve the grammar rule that are used to produce a syntactically correct sentence.

3. The resulting sentences (verses) are gathered and returned by the Generation Strategy.

Thanks to the Grammar Processor, PoeTryMe performs well based on grammatical correctness. The use of the Generation Strategy allows heuristics to apply to ensure features like meters, rhymes and coherence between lines are present, which can lead to poeticness (Oliveira [2012]):

o seu macaco era duas maquinas
horaciano antes dos poetas
para as consolas dos computadores
num mundo de poesias e carmes

Using machine translation, this translates to:

his monkey was two machines
Horatian before the poets
for the computer consoles
in a world of poetry and carmine

However, as with every template-based approach, and even though PoeTryMe is a highly customizable system, the use of templates tends to make the creativity of the predictions limited. Finally, the lexical knowledge is built at a word-based level and so does not necessary capture the meaning of words, as it would do if any kind of tokenization was performed.

3.2 Recent Research Based on Deep Learning

To avoid interactions with the user and try to overcome the lack of creativity of previous systems, recent research has focused on the rise of Deep Leaning methods

to generate poetry. These methods are less rule-oriented and qualify as statistical approaches.

3.2.1 Using Sequential Neural Networks

3.2.1.1 Straightforward use of RNNs and LSTMs

RNNs and LSTMs have been widely use for the generation of text (Mikolov et al. [2014], Serban et al. [2016]). Using their hidden state, they can effectively return an output that depends on the given input but also on previous inputs, making them attractive for sequential processing.

Simple models have been used from the beginning of the 2010s to generate poetry. In Karpathy [2015], a 3-layer LSTM is trained on Shakespeare poetry. Although the model is kept simple, it already produces readable and coherent poems:

Thou dost digest you! If accident doth see,
What says old George and Stafford and know me,
Even yonder where he should prove struck in praise in it;
Could be received black scurvy sight?

The work of Shakespeare is used as training data, which represents a 4.5 MB file. The training is performed on a character-level, but still the network manages to successfully learn words, sentence structure as well as poem structure (short sentences separated by line breaks).

This model has been subsequently reused and optimized to give implementations such as PoetRNN (Ballas [2015]). If these models manage to produce good overall results, improving them is not as simple as stacking more layers together, and other approaches are still to be found to consistently reproduce poetic features within a text.

3.2.1.2 Implementing and evaluating more complex models for the generation of Chinese poetry

In Zhang and Lapata [2014], an RNN is successfully trained on 300k poems to produce Chinese poetry. At its core, a Convolutional Sentence Model converts the poem line into a vector and attempts to capture how each line reinforces and constrains others. Attempts at evaluating the RNNPG model include Perplexity Evaluation, BLEU-based Evaluation and Human Evaluation. The model is compared to other models, including a random model, the current state-of-the art model for Chinese poetry evaluation (He et al. [2012]) and human generation:

- In terms of *fluency*, the model performs 59.1% better than a random model, 42.7% better than the state-of-the-art system and 7.5% worse than human generation.
- In terms of *meaning*, the model performs 58.4% better than a random model, 37.3% better than the state-of-the-art system and 12.8% worse than human generation.
- In terms of *poeticness*, the model performs 58.2% better than a random model, 34.6% better than the state-of-the-art system and 17.5% worse than human generation.

RNNPG achieves state-of-the-art performance in these criteria and closes the gap between automated and human-like poetry generations, although poeticness remains the biggest challenge.

3.2.1.3 Generating rhythmic verses with phoneme-based neural networks

In Hopkins and Kiela [2017], a neural language model is trained not on characters but on phonetic encoding to output English sonnets. The model uses the Carnegie Mellon University (CMU) Pronouncing Dictionary to convert the input poetry corpus into a more trainable dataset, with phonemes being embedded as 256-dimensional vectors. A LSTM is applied to the phonetic representation and basic backpropagation is

performed to train the model. An orthographic decoding step is needed to transform the output from phonemes into their most likely orthographic forms.

By constraining the model further and training it on various sonnets of similar structure, including Shakespeare's, the model is mistaken for a human poet by 54% of human evaluators (Hopkins and Kiela [2017]). It is able to insert end rhymes and many outputs manage to give a sense of meaning to the reader:

The crow crooked on more beautiful and free,
He journeyed off into the quarter sea.
his radiant ribs girdled empty and very -
least beautiful as dignified to see.

Tikhonov and Yamshchikov [2018a] investigate the problem of author-stylized text generation. In addition to a simple bidirectional character-based LSTM, they transcribe each word to corresponding phonemes from the International Phonetic Alphabet. The resulting phonemes are fed through a bidirectional phoneme-based LSTM that is combined with the first LSTM to output a concatenated word representation. The results show that, for a subset of authors considered to write in a recognizable style, the generated poems manage to reproduce the style of the authors good enough to be mistaken for them by human evaluators.

3.2.2 Combining Computer Vision and Natural Language Processing

In the past two years, a lot of exploration has been made in the use of visual inputs as a source of inspiration for poetry generation. More complex, these models have exploited specific techniques such as Convolutional Neural Networks (CNNs), Gated Recurrent Units (GRU) or Reinforcement Learning (RL).

By using images as inputs rather than words, sentences or a full corpus of text, they have further reduced the interaction with the user while enriching the system's seed.

3.2.2.1 Image Inspired Poetry Generation

Loller-Andersen and Gambäck [2018] make one of the first attempts at generating poetry by drawing inspiration from visual inputs. The model is composed of two parts:

1. A CNN based on Inception (Szegedy et al. [2015]) classifies an input image and outputs the five most probable candidates. Using ConceptNet(Speer and Havasi [2012]), the possible classes are used to generate related concepts and some corresponding rhyme pairs are produced using CMU Dictionary (Carnegie Mellon University [2014]).

2. A LSTM with two hidden layers is trained on 40.7k song lyrics extracted from MLDB. Some constraints on poetic generation — number of stanzas, number of syllables, rhyming scheme and focused words previously extracted — are added and the network is used to sequentially generate the final poem.

Although the model is not able to generate poetry consistently perceived as aesthetically good by the panel of human reviewers, some poems have a good overall quality, are closely related to the input image and positively evaluated by the panel.

Cheng et al. [2018] share a similar idea. They notice that if poets usually benefit from the objects and sentimental imprints perceived in images to compose their poetry, the same could be true for computers.

To make it happen, they extract keywords from input images by combining two CNNs sharing the same overall architecture. One of the networks is used for object detection in the input, the other for sentiment detection. Keywords are eventually expanded to account for low-confidence and rare words. The resulting data goes through a standard RNN that is responsible for outputting the poem.

To train the XaioIce network, a dataset of 50k Chinese poem lines is used. Human evaluation is performed using a specific interface that allows the evaluator to compare poems generated by different baseline methods to the input image.

Comparing the model to a state-of-the-art image captioning model — Image2caption (Fang et al. [2015]) — shows improvements in some areas (*relevance, imagination,*

emotion) while underachieving in others (*fluency, relatedness*). However, for a first attempt, the use of visual inputs seems promising, especially in terms of poeticness:

Unknown grass in the second month of spring
Petals and green woods honest in noon
Want to ask the bee where to go
Pair of butterfly shadows want to walk through path

3.2.2.2 Beyond Narrative Description

In late 2018, Bei Liu et al. made the first attempt at generating end-to-end image-inspired poetry using a holistic framework (Liu et al. [2018]).

To do so, they developed and used a training dataset that consists of two combined sets of data:

1. **MultiM-Poem**, a set of 8,292 images and, for each, a related human-written poem.
2. **UniM-Poem**, a poem corpus made of 93,265 poems.

While *MultiM-Poem* is obtained by scraping Flickr and is specific to computer vision tasks, *UniM-Poem* is a pure textual dataset that can be used in most of the tasks related to poetry generation. It is obtained by crawling several online repository of English poems, including Poetry Foundation, Poetry Soup, best-poem.net and poets.org. The dataset is further filtered by removing short poems (fewer than three lines), long poems (more than ten lines), duplicated poems, poems with infrequent characters and poems that include non-English languages.

The developed architecture is then based on two main parts:

1. On one hand, a **deep coupled visual-poetic embedding model** is used to build a multi-modal space from the training dataset. The model is made of three CNNs that learn different poetic features — object, sentiment, scene — from MultiM-Poem, and enhanced by a skip-thought model — a sentence encoder described in Kiros et al. [2015] — that is trained exclusively to retrieve

relevant poems from the UniM-Poem. The goal of this part of the framework is to learn an embedding representation from the resulting augmented MultiM-Poem (Ex) dataset.

2. On the other hand, a **RNN-based generator** is trained on the previously learned multi-modal representation by using two discriminators that provide rewards by adversarial training. In this configuration, the RNN acts as an *agent* whose parameters define a *policy* for generating a poem. The policy predicts the next *action*, i.e. picking the next word of the poem, and once there is no other word to pick, a cross-modal discriminator and a poetic discriminator provide a *reward* that updates the generator through *policy gradient*.

The system performs well compared to the four proposed baseline models:

- Show and tell (1CNN), a CNN-RNN trained with one CNN feature by VGG-16
- Show and tell (3CNNs), a CNN-RNN trained with three CNN features by VGG-16
- SeqGAN (Yu et al. [2017]), a CNN-RNN trained using adversarial training
- Regions-Hierarchical (Krause et al. [2017]), a hierarchical method for long image descriptions

To conduct the evaluation, both automatic (*BLEU, novelty, relevance, overall*) and human-based methods are used. In every case, the multi-adversarial network outperforms the baseline models (Liu et al. [2018]).

3.2.3 Mutual Reinforcement Learning

Yi et al. [2018] assert that traditional generative methods, which are based on Maximum Likelihood Estimation, put too much emphasis on word-level and tends to reduce innovation (which is important for an art like poetry), to fail at considering the whole line/poem and to ignore some criteria along the way (fluency, coherence, meaningfulness, overall quality).

To overcome these limitations, Xiaoyuan Yi et al. implement a mutual Reinforcement Learning model in which two learners are learning from the rewarder but also from

each other. The learners are basic generative seq2seq models, based on GRU units, and the rewarder combines neural models such as a neural language model and an adversarial training based classifier, as well as statistical metrics like Mutual Information and TF-IDF.

The MRL model is trained on 490k Chinese lines, mainly taken from quatrains — stanzas of four lines. Evaluation is performed using baseline statistical models, including the state-of-the-art model for Chinese quatrain generation (Mem). Different automatic evaluation metrics are defined, such as rewarder score, diversity/innovation score, TF-IDF and topic distribution. In particular, traditional statistical evaluation metrics such as BLEU or METEOR are excluded since they fail to capture a sentence-level sense of the poem, which is yet the main scope adopted by human experts when conducting an evaluation. In addition, human evaluation is used since the chosen automatic metrics fail at evaluating the poeticness of a poem.

The use of two learners in a Reinforcement Learning setting leads to significant improvement both on automatic and human evaluation scores (tab. 3.1):

A mosquito is flying around and feeling too hungry at night. It flies out of the window because of the smoke. It is just like me, sharing the same worry: if driven by hunger, we both choose to fly even if we are already exhausted.

Models	Fluency	Coherence	Meaning	Overall Quality
Base	3.28	2.77	2.63	2.58
Mem	3.23	2.88	2.68	2.68
MRL	4.05	3.81	3.68	3.60
Human poems	4.14	4.11	4.16	3.97

Table 3.1: Human evaluation performed on MRL and the compared models, according to different criteria. The results come from Yi et al. [2018].

3.3 Towards Multilingual Poetry Generation

Although text has been quite successfully generated in different languages, little research has been made on multiple language generation.

3.3.1 Simultaneous generation of text in multiple languages

While machine translation has gone a long way, with established models leading to successful real-life implementations, the idea of directly generating multiple outputs in different languages has been less studied. This is mainly due to the fact that machine translation is more trustful in most applications (Edunov et al. [2018], Wu et al. [2019]), yet in some others such a method remains impractical.

Hinaut et al. [2015] conduct one of the first analyses of multiple language acquisition. Using a neuro-inspired model, they attempt to reproduce human learning of different languages. The resulting model is an *Echo State Network*, with a very sparse hidden layer acting as a Reservoir for the network's knowledge. Every sentence is processed as follows:

1. The sentence is decomposed into semantic words, defining the main thematic of the sentence, and function words.

2. Semantic words are replaced by corresponding SW symbols.

3. All resulting words or symbols are stored in the network's memory through its hidden layer.

4. Words are processed sequentially and the network outputs the corresponding action described by the input.

A few mechanisms are added in order to make the learning more realistic, such as giving a higher focus to semantic words and applying a specific processing to infrequent words.

Performance is evaluated in terms of generalization error as regards the thematic role of the sentence. The results are promising, with only a 9.71% drop in performance from training on two languages rather than one.

A more straightforward approach is adopted in Östling and Tiedemann [2016]. Instead of attempting to imitate child learning, the authors focus on building a continuous vector representation of the languages to allow a standard LSTM to learn them.

The network is a stacked character-based LSTM, where characters are embedded into a 128-dimensional space. Training is performed on various Bible translations, gathering nearly a thousand languages and totaling 3 billion characters. Once again, the authors manage to limit the drop in performance resulting from each added language. In addition, the model can be used to interpolate multiple language models and perform text generation on the resulting language.

3.3.2 Multilingual approaches to poetry generation

As opposed to text generation, no successful approach has been developed from the perspective of generating poetry simultaneously in multiple languages. However, some systems have been developed of extended in order to achieve multilingual results.

Oliveira et al. [2017] supplement their existing system PoeTryMe to support three languages: Portuguese, Spanish and English. Many manual steps are needed in order to isolate and instantiate language-specific components. While offering sonnets generation in multiple language, the system is not able to easily extend to new languages due to the necessary refactoring of the set of language-dependent components.

Tikhonov and Yamshchikov [2018b] emphasize the importance of capturing phonetic information in order to correctly produce poems both in English and Russian. To achieve this, they build a concatenated embedding that contains information on the phonetics of every word preprocessed by a bi-directional LSTM network, alongside with its vectorized semantic representation. The final concatenated embedding stacks the following information:

1. Word embedding information
2. Phoneme-based information
3. Char-based information
4. Author embedding information
5. Document embedding information

Training is performed on 300 Mb of poetry, gathering verses from more than 20k authors. The resulting system demonstrates interesting results on BLEU and cross-entropy, especially when it comes to producing author-stylized poetry.

Chapter 4

Experiments

To explore the generation of multilingual poetry, several conducted experiments are presented hereunder. For the most part, they range by ascending complexity, from the use of both a common dataset and network to the distinction of a language-specific network from phoneme-based components.

The experiments highlight how the choice of architecture impacts the ability of the network to generate poems in multiple languages, and the resulting quality in grammaticality, meaningfulness and poeticness.

Four experiments are conducted:

1. A traditional approach of generating poetry using a character-based LSTM trained on distinct language-specific dataset

2. A similar method where the character-based LSTM is replaced by a phoneme-based model in order to increase the poeticness of the generated poems

3. A straightforward approach of training the same model on a multilingual dataset to observe whether it is is able to distinguish the different languages when generating poetry

4. A last method that combines a multilingual network with a language-specific network, in an attempt to learn poetic information on a multilingual dataset and linguistic information on a single-language dataset

The results of the experiments are detailed and discussed in the next two chapters.

4.1 Language selection

Previous work on poetry generation has led to successful production of text in many languages, including English, Spanish, Portuguese, Chinese and Russian.

Working with non-Latin alphabets, such as in Chinese or Russian, raises many problems that are out of the scope of this project. Furthermore, these languages would be hard to evaluate without advanced knowledge of them.

In the following experiments, the generation of poetry will be limited to English and French, which are among the most spoken languages with a Latin alphabet.

4.2 Training a common network on different datasets

Two different architectures are developed to explore the generation of poetry in multiple languages.

The first one is straightforward and generated poetry character by character using a shallow LSTM. The lack of success in terms of poeticness — later described in ch. 5.2 — leads to the development of a second network working on phonemes rather than pure characters.

4.2.1 Character-based LSTM

To tackle the problem of training a common network to learn both English and French with different datasets, a straightforward approach is first implemented before being expanded in the next experiment.

Since rule-based methods have been extensively worked on in the past decades, a simple method based on deep learning is used in the form of a basic recurrent neural network that learns to generate poetry in two languages.

This method seems naïve at first. The network works at a character level while humans produce language at a higher word or sentence level. In addition, it is unlikely that poetry can come down to a statistical succession of characters. Yet this approach might highlight promises and challenges when it comes to building a multilingual poetry generator.

4.2.1.1 Architecture of the character-based LSTM

The model is a simple LSTM network with three hidden layers of dimension 256 (fig. 4.1). It is based on Karpathy [2015] but one layer was found to give better results, with a 20.2% decrease in training loss on average.

Experimentation has been conducted with deeper networks since the training time was becoming too significant.

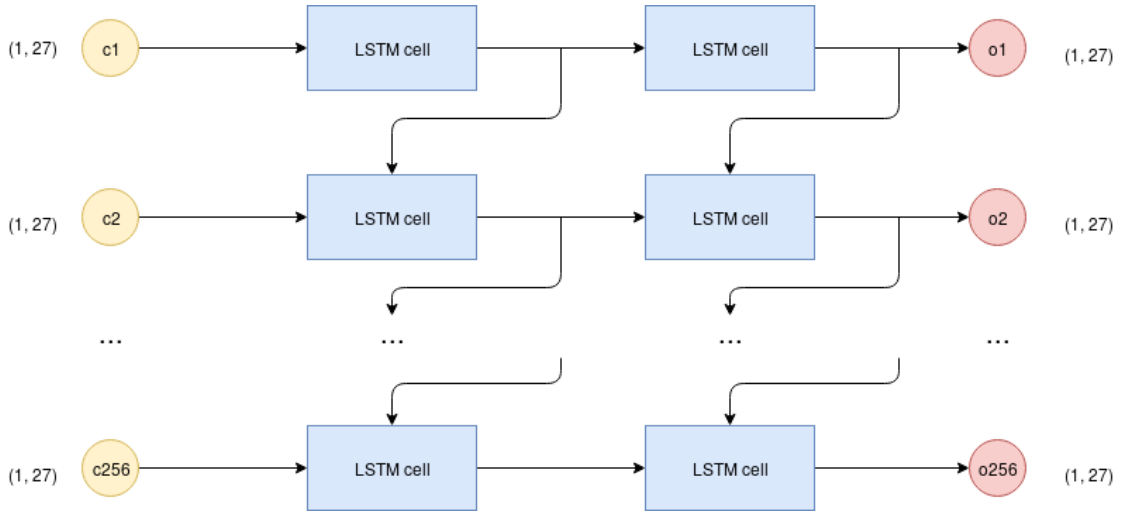


Figure 4.1: A representation of the two-layer character-based LSTM

The input data is one-hot encoded before being passed to the network. It requires the use of twenty-seven characters:

- The twenty-six lowercase letters
- The apostrophe character ’
- A space character to split words

Other characters like punctuation marks are discarded at training. As a result, any input is a 28-dimensional vector.

4.2.1.2 Dataset fed through the character-based LSTM

Two datasets are needed to carry out the experiment. In both cases, a list of poems from various authors is gathered, without any specific order. No preprocessing is added to this simple approach.

After obtaining the dataset, it is split into a training set (95%) and a validation set (5%). The small size of the validation set should be enough to give a rough estimation of how the network performs on unseen data.

4.2.1.2.1 English Corpus

The English corpus is built from the UniM-Poem dataset described in ch. 3.2.2.2. This dataset is interesting because of it significant size and because it gathers a lot of different poem structures. Plus, it has already demonstrated success in Liu et al. [2018].

Pandas is used to transform the JSON file into a simple text file that aggregates the poems. Because the dataset is quite heavy (over 600k lines), only a random subset of about 10% of the data is kept, which speeds up training.

There is no further preprocessing. The final version of the dataset contains 14.5k unique words (tab. 4.1).

Poets	File Size	Poems	Lines	Unique Words	Words
Various	1.2 MB	10.2k	52.2k	14.5k	229.9k

Table 4.1: Description of the English training dataset

4.2.1.2.2 French Corpus

There is no complete dataset available for French poetry, a new one has to be built from scratch.

To that purpose, a small web scraper is developed using *Scrapy*, a web crawling framework for Python. It is used to crawl Glehello [2007], one of the biggest online collection of French poems. The crawler browses the list of French authors, opens all relevant links to access a given author's page, then evaluates and crawls every subsequent link.

The final dataset is a list of 11,719 poems, composing over 200k lines. Once again, such a quantity is not needed for now, so only a 10% random fraction of the data is kept, ending up with a dataset of approximately 33k unique words.

4.2.1.3 Training of the character-based LSTM

At training, decayed learning rate is used to better control the gradient descent. A standard 50% dropout is added in both layers, meaning that half the neural nodes are randomly dropped during training. Although it adds noise in the training process, it is beneficial to the network by improving learning and reducing overfitting scenarios in which the network only learns to repeat the same character multiple times. Experimentation was not conducted with other dropout rates but it could be interesting to evaluate the impact it has on the training speed and accuracy.

Training is performed on a single Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz, using mini-batch gradient descent with cross-entropy loss. Computation is stopped after 50 epochs, which corresponds to a local minimum being reached. At this point, the learning rate is very small due to the decay learning rate, so that the network is only slowly updating its nodes to a new example.

4.2.2 Phoneme-based LSTM

The second model is more advanced and built in response to the lack of poeticness in the generated poems of the previous model (ch. 5.2).

In particular, a focus is made on the following aspects:

- Preprocessing the corpus, since the training dataset is likely to play a key role in the learning ability of the network.

- Building a training dataset based on phonemes rather than characters, which should lead to more poetic sounds and rhymes, and generalize better to different languages.

- Adding constraints to the poetry generation, to improve syntactic correctness.

4.2.2.1 Architecture of the phoneme-based LSTM

The architecture is based on the previous works from Hopkins and Kiela [2017] and, more recently, from Benhart et al. [2018], which won the *2018 PoetiX Literary Turing Test Award for computer-generated poetry.*

Contrary to what can be seen in computer vision (Szegedy et al. [2015]), NLP networks are rarely built with many layers, mainly because they become very hard to train when more than a few layers are stacked (Pascanu et al. [2013]). As a result, the chosen model is a 3-layer LSTM, with 1024 hidden cells. The representation of the model in fig. 4.1 is still relevant, but with one more layer, and larger input and output vectors, since tokenization will be used (more about this later).

There is still a key difference compared to the first approach. Instead of working forward, generating the words in a readable order, the generation is made backwards. By doing so, the model is able to fix the end rhyme and ensure more poeticness in the output. This way of decoding the state of the network was introduced in Ghazvininejad et al. [2016] and has been subsequently implemented in different projects, including Benhart et al. [2018].

4.2.2.2 Dataset fed through the phoneme-based LSTM

A few approaches have underlined the fact that focusing on one author yields better results in terms of poetry generation (Tikhonov and Yamshchikov [2018b], Benhart et al. [2018]). The consistency in style seems to help the network learn patterns in the data, which makes the whole output more coherent.

4.2.2.2.1 Building the dataset

English Corpus. Some English poetry datasets are available online. Kaggle hosts a few corpora from the *poetryfoundation*, and some papers have released poems they use for training (Liu et al. [2018]). Yet, most of these datasets get too small when restricted to a single author.

To build the training corpus, the focus is made on the work of John Bradburne. John Bradburne is one of the most prolific English poets, with more than 5000 written poems. He lived in the 20th century and as a consequence used a vocabulary very close to modern English. In addition, his texts tend to focus on the same religious topic, which should make his style easier to learn.

Once again, the dataset is not available, so it has to be made from scratch. After scraping and cleaning the dataset, in particular to remove some of the unnecessary epistolary work, a 6.3 MB file of poems is available.

In addition, a second and much smaller dataset is prepared, with the work of Walt Whitman, whose written sonnets are supposed to be more easily learned by a network (Benhart et al. [2018]).

The two datasets are summarized in tab. 4.2.

Date	Poet	File Size	Poems	Lines	Unique Words	Words
1921–1979	Bradburne	5.8 MB	5.5k	171k	49.1k	1096.9k
1819–1892	Whitman	722 kB	1.4k	16.1k	15.1k	122.6k

Table 4.2: Description of the two English training datasets

French Corpus. The French dataset described in ch. 4.2.1.2 is grouped by author and only three of them are kept: Victor Hugo, Paul Verlaine and Alphonse de Lamartine. Why these poets? They have a stressed style — which should help the learning — and have produced a large quantity of poetry throughout their respective lives (tab. 4.3).

Unfortunately no contemporary author was found to provide a more recent dataset.

Date	Poet	File Size	Poems	Lines	Unique Words	Words
1802–1885	Hugo	793 kB	354	18.9k	15.4k	139.1k
1844–1896	Verlaine	470 kB	762	11.5k	12.3k	81.7k
1790–1869	Lamartine	214 kB	80	5.1k	5.9k	38.0k

Table 4.3: Description of the three French training datasets

4.2.2.2.2 Processing the dataset

In the previous experiment, the network was learning at a character level and so the dimension of the problem was quite small. As a result, every character could easily be converted to its one-hot encoding representation. Every input was a (1, 28) vector, which is totally manageable for a network.

This time, the goal is to leverage the phonemes in the dataset, which means that is necessary to work at a word level. Given the number of unique words in the corpora (over 10k words, whatever the language chosen), a straightforward one-hot encoding of the inputs would lead to very sparse and big vectors. This would be a problem as it would make the whole learning very difficult. To overcome it, the input vectors are embedded into a continuous space of dimension 300 (ch. 2.2.1.4). In addition, they are initialized with pretrained vectors to help the network learn a useful representation of the words more quickly.

English Dataset. In English, one of the largest datasets of pretrained vectors is GloVe (ch. 2.2.1.4). The small version already contains 6B tokens trained with a 400K vocabulary size and it has a 300-dimension representation that fits perfectly the task at hand (Pennington et al. [2014]).

Before embedding the input lines into vectors, they are converted into a set of tokens. The default NLTK tokenizer is used, as it is has been specifically trained for English tokenization, using the English Penn Treebank (Loper and Bird [2002]).

Some basic preprocessing of the data is also implemented, including removing digits and special characters. More importantly, the large Bradburne dataset is reduced from 49k unique words to about 10k. This is done by keeping the most common words and replacing the others by an *unk* token that is then discarded during the

generation process. The final version of the corpus is less likely to overfit to specific examples and has a vocabulary size way easier to fit into the computer memory.

French Dataset. Unfortunately, GloVe is not available in other languages. The same model could be trained again on another dataset, yet this has already been done by Carnegie Mellon University (CMU) on the TED corpus (Ferreira et al. [2016]) and by Fauconnier and Kamel [2015] on Wikipedia. Only CMU resources were used as Wikipedia seemed less suitable for the generation of poetry, but this hypothesis would have to be confirmed by comparing the results with Fauconnier's *Word2Wac* pretraining.

Using CMU dictionary, the embedding takes the form of a (40417, 300)-dimensional matrix, much smaller than the (400000, 300) GloVe representation, but still suitable given the limited vocabulary size of the French corpus.

For the embedding step, the NLTK tokenizer has to be replaced since it has not been built in regard to French tokenization. This is important to prevent phrases like "j'allai voir" from being tokenized into the two words "j'allai" and "voir". Instead, the goal is to have another split between "j'" and "allai", since the former is the subject and can be linked with many other words than just the latter verb. Rather than developing a tokenizer from scratch, the French version of the Moses Tokenizer is used (Koehn et al. [2007]).

4.2.2.3 Training of the phoneme-based LSTM

Once again, mini-batch (64) gradient descent is relied upon to smooth the learning. The learning rate is progressively decayed as the network progresses, using a cosine decay, and warm restarts are introduced to avoid unstable local minima, as proposed in Loshchilov and Hutter [2017]. The various related parameters are tweaked depending on the dataset, especially in the case of the Bradburne dataset, which is much larger and requires more steps before starting the decay.

The loss is computed using a basic weighted cross-entropy, and many keys ideas from Benhart et al. [2018] are kept in the implementation, including the smaller 30% dropouts to further reduce the risk of overfitting and the manual rules applied to the model, namely:

- Fixed part-of-speech constraints are added to the model to limit the possible categories of words that can be generated at a given step. Such rules prevent the network from producing sequences of words in a grammatically wrong order, although it does not remove all risks of syntactic incorrectness.
- A system of weights is used at generation to discourage repetition of words, by down-weighting the likelihood of producing a recently-used word compared to unseen words. It increases novelty and prevents the network from abusing of repetitions of words once they are learned.

Training is heavier than previously and is performed on a Nvidia Tesla V100 GPU with 16 GB of memory, hosted on *Amazon Web Services*.

4.2.2.4 Generation with the phoneme-based LSTM

At generation, a seed word serves as input to the network in order to inspire the model. The seed is used to orientate the poetry generation (fig. 4.2):

1. The seed is embedded into a vector, just like any dataset word, making use of the pretrained dataset.
2. Distances between the seed and the other words from the corpus are computed, and the four nearest are picked.
3. Corresponding rhyming words are selected, building five pairs of words.
4. Two other pairs of rhyming words are randomly picked from the most common words included in the corpus.
5. The 14 resulting words are allocated at the end of each poem line and the remaining words are generated by the network in reverse order.

To select the rhyming words and ensure more poeticness in the output, a dictionary of rhyming words is built for each language. This is done by using pronunciation dictionaries:

1. The Carnegie Mellon University (CMU) Pronouncing Dictionary from Carnegie Mellon University [2014] in English

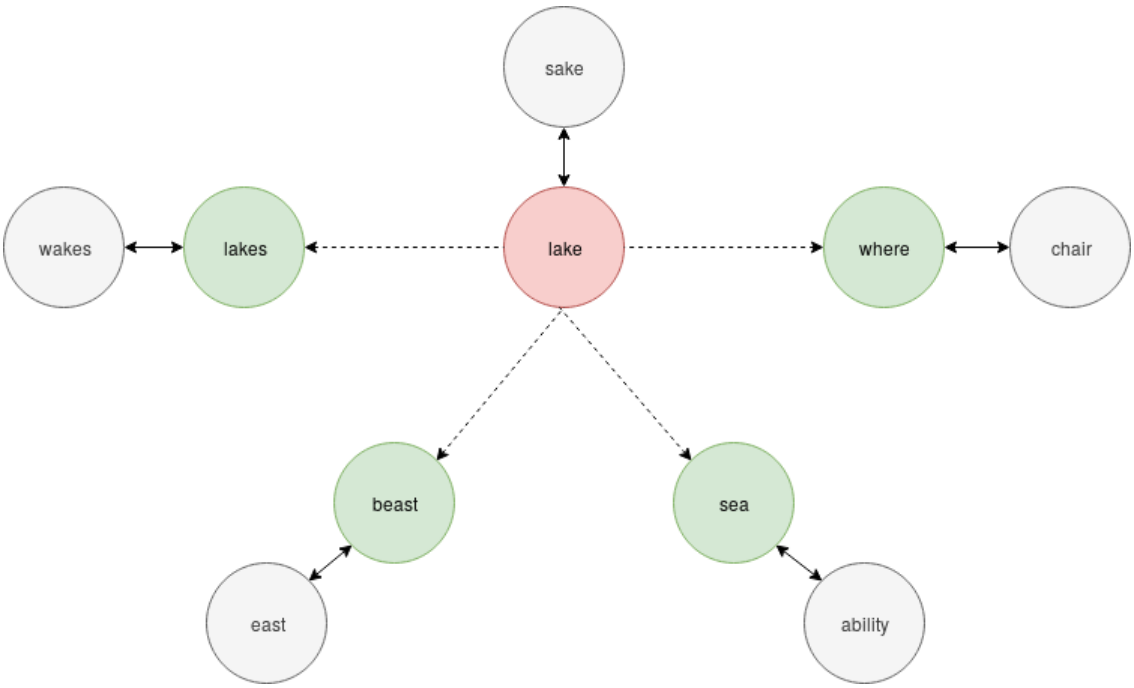


Figure 4.2: Enforcing rhymes at text generation

2. The Lexique Database from New and Pallier [1999] in French

These dictionaries are applied to the training corpora and used to extract the most common pairs of rhyming words. The output is a set of 21k pairs in English and 6k pairs in French.

4.3 Training a common network on a single dataset

As an intermediary study, inspiration is drawn from Östling and Tiedemann [2016] to train a single network with continuous vector representation on a bilingual dataset.

The dataset is built using a combination of English and French poems. The poems are drawn from the same datasets that are described in 4.2.1.2 and contain the work of various poets in English and of Victor Hugo in French. At each epoch, the poems are randomly grouped into batches so that the rate of English data as opposed to French data is always varying.

The architecture is a standard character-based LSTM, as described in 4.2.1.1. No batch regularization is used since the learning task is likely to be already challenging

enough for the network.

4.4 Combining multilingual and language-specific networks

So far the focus has been on word embeddings to efficiently learn linguistic information from the dataset. Yet poetry generation is much more impacted by characteristics such as aesthetics and rhythm, which are less likely to be correctly learned by a general text embedding approach.

The previous experiment highlights the use that can be made of a phoneme-based network and the results detailed in ch. 5.3 are promising enough so that it makes sense to build on them.

4.4.1 Architecture of MLPG

Since phonetic properties transfer across languages, the architecture is split into two networks:

1. The first network is a **shared phoneme-based LSTM**, responsible for learning a useful phonetic representation of the poems in both languages. The network's focus is on *poeticness*.

2. The second network is a **language-specific LSTM**, responsible for learning linguistic information in order to output a text of human-like quality. The network's focus is on *grammaticality* and *meaningfulness*.

The resulting model is named the **MLPG**, standing for Multilingual Poetry Generator. The detailed architecture is represented in fig. 4.3.

The architecture of the phoneme-based LSTM is entirely based on the architecture described in ch. 4.2.2.1.

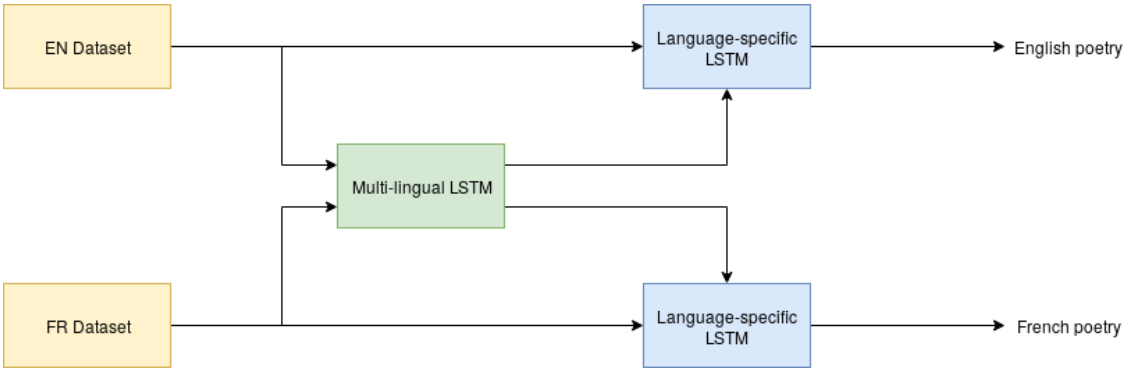


Figure 4.3: MLPG architecture

4.4.2 Datasets fed through MLPG

In this approach, two datasets have to be gathered for each network composing the model.

4.4.2.1 Language-specific datasets

Poems in both English and French are gathered as inputs for the language-specific LSTMs. The poems are drawn from the previous datasets:

- English poems are taken from the UniM-Poem dataset, supplemented with the works of Walt Whiteman and John Bradburne. The final set is a corpus of around 40k verses.
- French poems are taken from the works of Hugo, Verlaine and Lamartine, which have been gathered previously. The final set is a corpus of around 40k verses.

The datasets are described in greater details in 4.2.1.2 and 4.2.2.2.

4.4.2.2 Common dataset

The previous datasets are randomly merged into a common dataset of 80k verses that is used to train the multilingual LSTM (tab. 4.4).

Poets	File Size	Poems	Lines	Unique Words	Words
Various, incl. Whitman & Bradburne	716 kB	1388	21.6k	15.1k	122.5k
Hugo, Verlaine & Lamartine	784 kB	1388	19.9k	14.6k	139.1k

Table 4.4: Description of the combined datasets

To prevent the network from overfitting too much on meaning, only the 10k most frequently encountered words are kept — which account for about two thirds of the data — while the other ones are replaced with an *unk* token.

4.4.3 Training of MLPG

In order to further reduce the risk of overfitting, the common network is highly regularized with batch normalization.

Most of the implementation choices made in 4.2.2.3 are still relevant to the system, although tweaking of the hyperparameters is needed to obtain a successful learning, including lowering the learning rate from $2 * 10^{-4}$ to $1.8 * 10^{-4}$.

Because of the size of the dataset and the complexity of the architecture, composed of two networks rather than one, training takes longer than before. Almost one day is needed before the network stops learning. Once again, a Nvidia Tesla V100 GPU with 16GB RAM is necessary to avoid weeks of training.

4.4.4 Generation with MLPG

The seed system defined in ch. 4.2.2.4 is used at generation. Since some words are not necessary in the limited vocabulary, they are skipped when searching for the next word in the generated poem.

Chapter 5

Results

The results of the four previous experiments are detailed below.

When human evaluation is performed, a panel of 23 bilingual reviewers was asked to score the generated poems in terms of poeticness, grammaticality and meaningfulness (ch. 2.4.2). Each reviewer had to evaluate between 5 and 10 poems drawn from random seeds. The score given ranges between 0 and 10, where 0 means "the poem does not exhibit the considered characteristic at all" while 10 means that "the poem performs as well as a human poet would".

Although such an evaluation remains subjective to the panel of reviewers, it is better than using traditional statistical evaluation metrics such as text perplexity or BLEU. Such metrics are not really suitable to poetry, as opposed to regular text generation tasks (Yi et al. [2018]).

5.1 Common LSTM on a single dataset

In the intermediary experiment, a regular LSTM is trained to produce poetry both in English and French. After around 30 epochs, the network seems to stop learning, also the corresponding outputs sound gibberish:

amour pure
Aes x goni

Esedsd me
Ae fe poule r ces ve loi tou sanktti ta wesn wone sei
Soir
E doar ded pecan son ge jten chlecre de pni ca morlent le
Eu te dour repoume puu
e gans lan
Leus lonnre toue fandar jou-eve
Si se gopie jor ra f yemes orori cé foun li le cuveCd de
Aoer ce soes de in re fet ce
Ditrs sanget afset pouvsre
Cess sipired les fupgee sar
Das chons dun lhe Ler nar fên ta bce de senfse
Vt qou de tecge pur
Lu les an pe tous an

Whatever the parameters and the generation seed chosen, the network fails to distinguish both languages with such a simple architecture, highlighting the difficulty of the task. As can be seen in the last lines, the network sometimes manages to keep a single language for each word, but often mix both languages while generating a single line (tab. 5.1).

Lu	les	an	pe	tous	an
FR	FR	EN	∅	FR	EN

Table 5.1: Identification of multiple languages within a line

The study of the training loss (fig. 5.1) shows that a plateau is quickly reached by the network (around the second epoch), after which the network starts slowly overfitting to the training dataset.

Among the few positive things, it is interesting to note that the structure of the input sonnets is slightly learned since most of the outputs have 16 verses.

Given the obvious low quality of the output, the system was not evaluated any further.

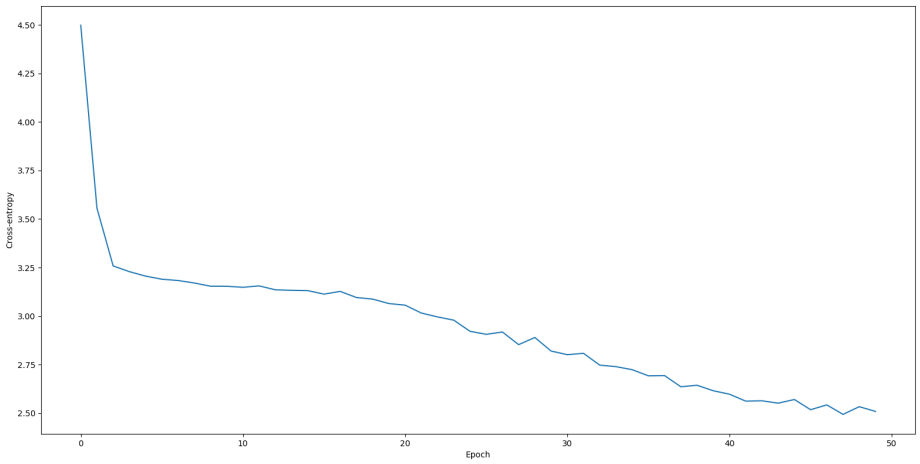


Figure 5.1: Training of the common network

5.2 Character-based LSTM on multiple datasets

Despite the underwhelming results previously met, the same character-based architecture is kept for the second experiment. Yet this time training is performed distinctly on both datasets, leading to two sets of network weights for the English and French languages.

One of the underestimated advantages of training two models on such a similar task is that a high difference in quality most likely means one of the networks is under-optimized. As a result, optimization was performed until both models approximately reached the same performance. Among other things, it was crucial for the English network, where training accuracy got an increase from 0.17 to 0.42 by tweaking hyperparameters — essentially increasing the hidden size from 128 to 512 and decreasing the batch size from 100 to 64 — and further processing the dataset (ch. 5.2).

Epoch	Loss	Training Accuracy	Validation Accuracy
50	2.05	0.17	0.14

Table 5.2: The first training results for the English character-based LSTM. Here the network is under-optimized and the training dataset hard to learn from.

In particular, the most important issue was with the UniM-Poem dataset, where it

appears that some poems are of very poor quality:

heeya
sick 'im puppy
loo
loo
lulu
loot
loot
loot

Since the model is using only a subset of it, this has a high impact on the quality of the results and prevents the system from learning a proper representations of words.

To overcome this problem, the process of drawing poems was slightly updated. Instead of taking a random subset of the dataset, Pandas was used to select a subset of poems that range from 100 to 120 characters, leading to a dataset of medium-sized poems that are less likely to slow down the learning process.

These changes lead to a smoother learning (fig. 5.2) and better results (tab. 5.3).

In both cases, the networks are able to learn words and even some sentence structures, although they are only trained on generating a single character at a time.

Some rhymes are even present while there was no specific strategy to encourage them. The output poems are generally quite long, which is likely due to the use of a dataset with long poems.

Finally, the networks sometimes come up with their own made-up words.

5.2.1 English poetry

After 50 epochs, the cross-entropy loss function has reached a local minimum and training is stopped (ch. 5.2).

In addition to the loss, training and validation accuracy are computed by comparing every output character to the ground truth. Although this is not necessarily a valid

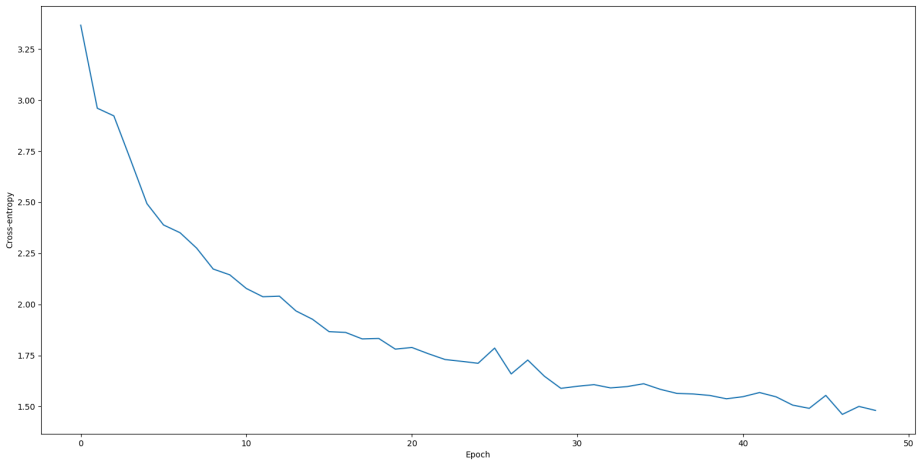


Figure 5.2: Training of the English character-based LSTM

evaluation metric for poetry, it gives a first sense of whether the network is learning or not. Gathered results are displayed in tab. 5.3.

Network	**Epoch**	**Loss**	**Training Accuracy**	**Validation Accuracy**
2-layer LSTM	50	1.95	0.45	0.42
3-layer LSTM	60	1.40	0.48	0.49

Table 5.3: Impact of the number of layers on the training, for the English dataset

To overcome the limitations of training and validation accuracy, human evaluation is performed on a random sample of produced poems. The results are displayed in tab. 5.4.

EN	Grammaticality	Meaningfulness	Poeticness
AVERAGE	8	5.82	6.78
MEDIAN	8	6	7

Table 5.4: Human evaluation of the English character-based LSTM

Below is an example of generated poetry. Note that the first letter of each line has been capitalized for display purpose.

And likewise is the dount of down
And the cended stall is in the daan

I hear of the saying of the bors
The streath I would was way more
And a sick of the sun and the delar
Come to soul with the veny lake and all

Obviously there are a few non-existing words, even though they are sometimes not so far from existing ones: *dount, cended, daan, streath, delar, veny.*

5.2.2 French poetry

Training is similar for the French dataset, although the final loss is lower and the learning curve appears slightly smoother (fig. 5.3).

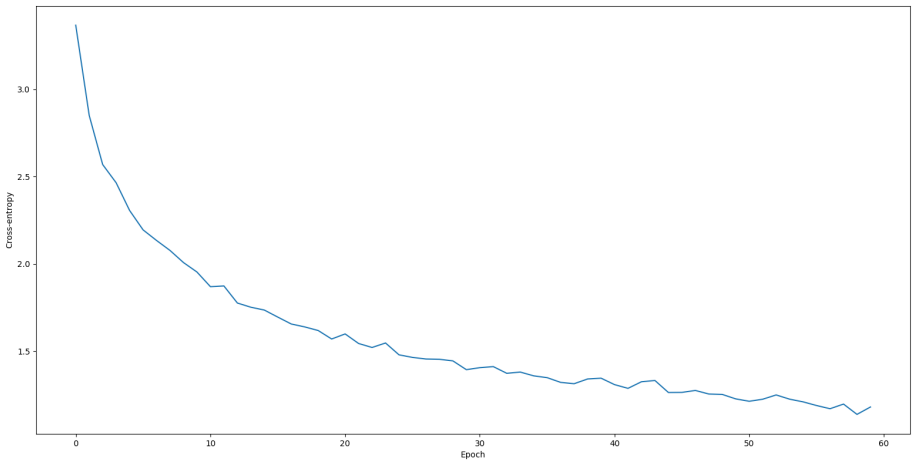


Figure 5.3: Training of the French character-based LSTM

The accuracy of the training and validation sets is a bit lower than expected as regards the English version of the model (tab. 5.5). This could be due to some under-optimization of the model, a higher complexity of the French language or simply that one dataset is harder to learn than the other. More experiments should be conducted to narrow the set of possible reasons down, but it has not been done as part of this work.

Once again, human evaluation is conducted (tab. 5.6). The model performs worse in all three categories, with a very low score in the meaningfulness of the output.

Network	**Epoch**	**Loss**	**Training Accuracy**	**Validation Accuracy**
2-layer LSTM	50	1.63	0.34	0.33
3-layer LSTM	60	1.43	0.45	0.45

Table 5.5: Impact of the number of layers on the training, for the French dataset

FR	Grammaticality	Meaningfulness	Poeticness
AVERAGE	6.65	3.65	4.52
MEDIAN	7	3	4

Table 5.6: Human evaluation of the French character-based LSTM

Yet it is expected that a network generating poetry character by character would be lacking sense of what is meaningful and what is not, given that this characteristic operates at a higher level.

Below is an example of a generated poem:

Le tourment le sourire et plus de tresse
Et de l'ame aux vers de son ecorde et le bien
C'est pas et de leur deux cortes de ma mole
Ou de mon bras de la reve et le loin
Ne chante a le tendre aux cheveux de ce lange
On me sourier rendre et te vanteur
Ne me changer les vieux souvents des frais sens de ses forts
X de l'enfun pour comme voir au monde
Heureux que le porde a son cher de tout le let de mon pera

5.3 Phoneme-based LSTM on multiple datasets

The phoneme-based architecture is much more complex, with a larger LSTM, a further preprocessing of the data and more constraints on the generative process.

As a result, the task's main difficulty becomes to find a balance between enforcing enough constraints for the network to generate meaningful content and allowing

enough freedom to avoid the generation of *boring* predictable poetry — with repetitions of words or a high similarity to the input poetry.

5.3.1 English Poetry

Training is performed on both Bradburne and Whitman poetry, with little difference in terms of results, apart from the fact that training takes longer on the much larger Bradburne dataset.

After 20-30 epochs, the loss function achieves a local minimum (fig. 5.4).

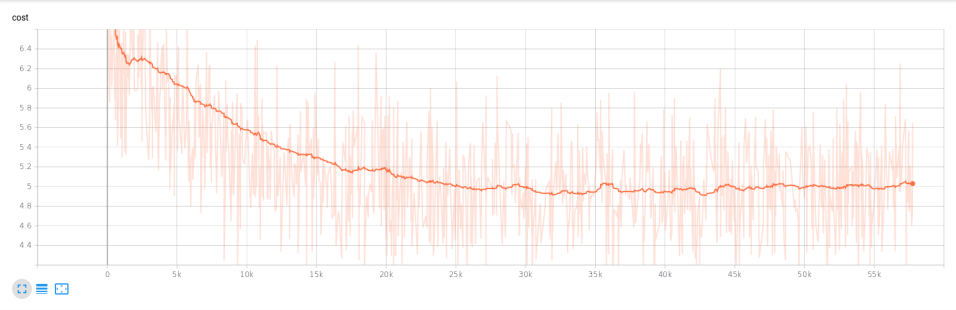


Figure 5.4: Training of the phoneme-based LSTM on the Whitman dataset

A small portion of the dataset is used as validation, scoring the weighted cross-entropy loss of the sequence (tab. 5.7).

Epoch	**Training loss**	**Validation loss**
30	4.37	5.62

Table 5.7: Training results of the phoneme-based LSTM on the Bradburne dataset

The quality of the model is assessed using human evaluation (tab. 5.8). All scores are higher than the previous character-based model, even if the poeticness does not increase as much as expected.

EN	Grammaticality	Meaningfulness	Poeticness
AVERAGE	9.13	7.17	7.3
MEDIAN	9.13	7	7

Table 5.8: Human evaluation of the phoneme-based LSTM on the Bradburne dataset

Below is an example of generated poem when learning from the Bradburne dataset:

THE LAKE.

Is, our lady of, the holy lake
Is, our lady of the woods and lakes
Of heavens king and queen of glory make
A man is better not for the mistakes.
To, wait and see if he will take it back
Of them that walk along the farther shore
In christ, the son of god is, like a shack,
Be still and know that it is time for your.
With our lady of the morning dam
Is on the height of every heart and sight,
The word and, god, the voice is like a jam
It is to praise the lord of, day and night.
And blood of jesus christ the lord of light
Together with the lord of day and night.

It is interesting to note that the poem certainly conveys a strong religious atmosphere that is often displayed in the work of Bradburne. In addition, the network is able to learn more complex phrases and reuse them in the right way (such as in "Christ, the son of god").

5.3.2 French Poetry

Training the French model presents a few quality-related issues. Indeed, a straightforward application of the previous model leads to low quality poems:

LE LAC.

Or, vers le pied du grand souffle de lac
A pas de terre sous le printemps clair
A pas de terre, on a tout de sac
De songer sous le grand milieu de mer.
De lui vous a loin de notre large
De argent, sous le grand milieu de, bosse
Du plus de terre, on a tout de marge
Milieu de terre, on a tout de fosse.

A sous le pied du grand souffle de eau
A vers le pied du grand souffle de place
A clair de terre, sous le printemps, oh
Or, vers le pied du grand souffle de face.
Or, et de terre, on a fait de place
Or, vers le pied du grand souffle de face.

There are two reasons that could explain such a performance:

1. When predicting the last word of the verse, the model searches for a pair of rhyming words to use while the rhyme dictionary has not been converted to French.
2. POS tags are used to enforce grammatical constraints on the generated verses, yet current POS tags are still being generated using the NLTK tagger, which has been trained on an English dataset and is based on the English POS classification.

The first issue is easy to overcome by building a custom dictionary of rhyming words specific to the French language, as detailed in ch. 4.2.2.4.

To solve the second issue, it is necessary to adapt Benhart's POS constraints defined in Benhart et al. [2018] to the French language. In this regard, a conversion table is built from scratch, allowing each EN POS tag to be translated into a matching FR POS tag. The resulting table is used to translate existing constraints (tab. 5.9).

This allows the model to generate more grammatically-constrained poetry. The final learning characteristics are displayed in tab. 5.10.

Human evaluation leads to slightly worse results than the English model (tab. 5.11), even though all three characteristics are improved compared to the character-based model.

Although the constraints add more syntactic correctness to the text, it also enforces in some cases a heavy use of comparisons, suggesting that the network is slightly overfitting to this specific structure:

LA DAME.

EN Description	**EN Tag**	**FR Match**	**FR Matching Descriptions**
Coordinating conjunction	CC	CC	Coordination conjunction
Cardinal number	CD	DET	Determiner
Determiner	DT	DET	Determiner
Existential there	EX	/	/
Foreign word	FW	/	/
Preposition or subordinating conjunction	IN	P	Preposition
Adjective	JJ	ADJ	Adjective
Adjective, comparative	JJR	ADJ	Adjective
Adjective, superlative	JJS	ADJ	Adjective
List item marker	LS	PRO	Full pronoun
Modal	MD	V	Indicative or conditional verb
Noun, singular or mass	NN	N, NC	Common noun
Noun, plural	NNS	N, NC	Common noun
Proper noun, singular	NNP	NPP	Proper noun
Proper noun, plural	NNPS	NPP	Proper noun
Predeterminer	PDT	ADJ	Adjective
Possessive ending	POS	/	/
Personal pronoun	PRP	CLS	Subject clitic pronoun
Possessive pronoun	PRP$	DET	Determiner
Adverb	RB	ADV	Adverb
Adverb, comparative	RBR	ADV	Adverb
Adverb, superlative	RBS	ADV	Adverb
Particle	RP	/	/
To	TO	P	Preposition
Interjection	UH	/	/
Verb, base form	VB	V, VINF	Indicative or conditional verb, infinitive verb
Verb, past tense	VBD	VIMP	Imperative verb
Verb, gerund or present participle	VBG	VPR	Present participle
Verb, past participle	VBN	VPP	Past participle
Verb, non-3rd person sing. pres.	VBP	V	Indicative or conditional verb
Verb, 3rd person singular present	VBZ	V	Indicative or conditional verb
Wh-determiner	WDT	ADJWH	Interrogative adjective
Wh-pronoun	WP	ADJWH	Interrogative adjective
Possessive wh-pronoun	WP$	ADJWH	Interrogative adjective
Wh-adverb	WRB	ADVWH	Interrogative adverb

Table 5.9: Conversion table between English and French POS tags

Epoch	**Training loss**	**Validation loss**
20	4.87	5.91

Table 5.10: Training of the phoneme-based LSTM on the French dataset

FR	Grammaticality	Meaningfulness	Poeticness
AVERAGE	8.21	6.43	6.08
MEDIAN	8	6	6

Table 5.11: Human evaluation of the French phoneme-based LSTM

Seuil, comme une ombre déesse
Rien comme elle vient sonder
Une lueur du ciel tourné
Toi du ciel toujours alla.
Jeune âme, comme une sultan
Âme, comme une lune torrent
Du seuil, comme une génie
Est aigu, comme une douleur.
Ce seuil, comme une mari
Pensée, comme une terre frappés
Jeune âme, comme une forêt
Toi du ciel bien resté.
Était comme une autre sujet
Une pensée comme une instinct.

which literally translates to English as:

THE LADY.
Threshold, like a shadow goddess
Nothing like she's probing
A gleam of the sky turned
You from heaven always went.
Young soul, like a sultan
Soul, like a torrential moon
From the threshold, like a genius
Is sharp, like pain.

This threshold, like a husband
Thought, like a struck land
Young soul, like a forest
You from heaven stayed well.
Was like another subject
A thought like an instinct.

5.4 Multilingual Poetry Generator

The Multilingual Poetry Generator (MLPG) architecture is more complex and thus harder to train than the previous model, mainly due to the fact that there are twice the number of networks. In addition, two different flows of data need to be set, a bilingual one for the multilingual network and a language-specific one for the other network.

The preliminary experiment had highlighted the difficulty of training a network on a bilingual dataset, and though this time the shared network is only expected to learn phonetic information, a lot of tweaking in the parameters is needed.

5.4.1 English evaluation

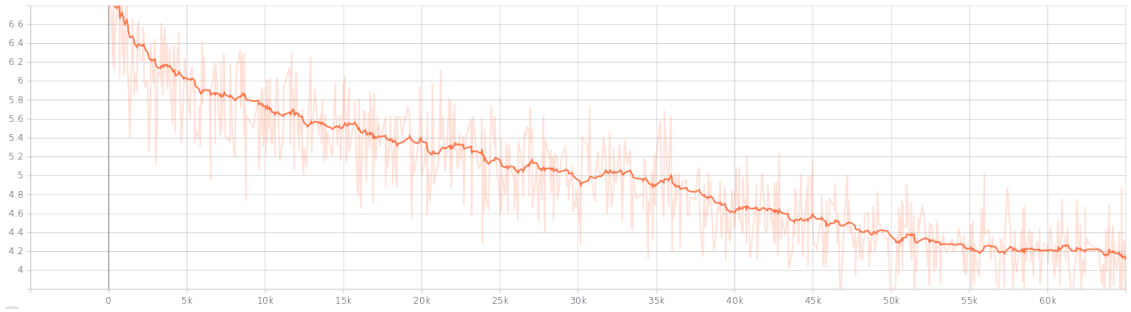
Best results are achieved with *SGDR* (Loshchilov and Hutter [2017]) rather than simple decay learning. After 30 epochs, the learning curve starts to straighten up a bit (fig. 5.5), meaning that the network is no longer learning from the dataset — either because a local minimum has been reached or the decayed learning rate has gotten too low.

Some training characteristics are described in tab. 5.12.

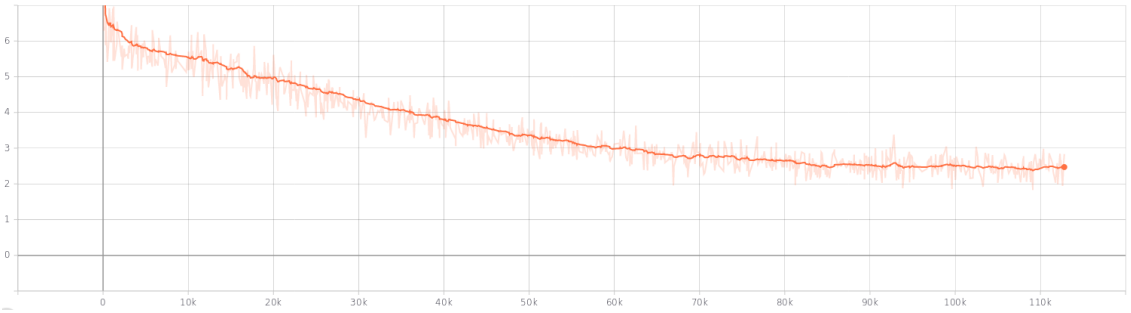
Epoch	**Training loss**	**Testing loss**
50	10.17	10.61

Table 5.12: Training results of the MLPG on the English dataset

Before evaluating the network, it is interesting to project the multilingual LSTM weights using Principal Component Analysis. The result clearly shows that the



(a) Evolution of the training loss for the multilingual network



(b) Evolution of the training loss for the language-specific network

Figure 5.5: Training of the MLPG

trained network has considered the input data as split into two categories, one for each language (fig. 5.6).

Human evaluation is performed using the same settings as previously. Both grammaticality and meaningfulness decrease compared to a simple phoneme-based network, yet this is offset by an increase in poeticness (tab. 5.13).

EN	Grammaticality	Meaningfulness	Poeticness
AVERAGE	8.39	6.69	8.69
MEDIAN	9	7	8

Table 5.13: Human evaluation of the MLPG on the English dataset

Below is an example of a produced poem:

THE LAKE.

Subordinated intermission lake
And o of the to a and savage beast
And at the two the speculation sake

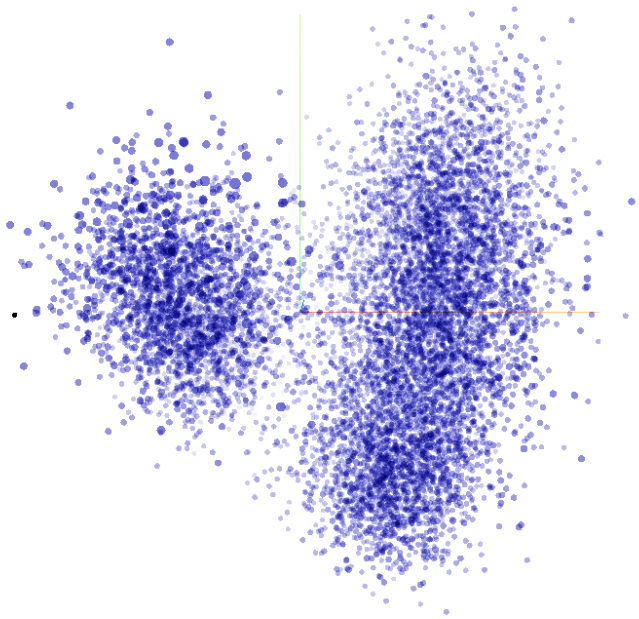


Figure 5.6: PCA performed on the shared network weights

Transfers to and the respiration east.
Unoccupied with the ability
Its, unannounced, to of, and o the lakes
And of to or a strange the current, sea,
And to a and of the transported, wakes.
Of, and, perpetuating with the chair
And undisguised to with the in and sight
And narragansett population where
To intermission each and of the night.
Pronounce with the and presentation light,
Perpetuating, each, and of the night.

5.4.2 French evaluation

French performs quite similarly, with training characteristics described in tab. 5.14.

Human evaluation leads to the same findings, namely both grammaticality and meaningfulness get a decrease while poeticness increases (tab. 5.15).

Epoch	Training loss	Testing loss
50	10.46	11.04

Table 5.14: Training results of the MLPG on the French dataset

FR	Grammaticality	Meaningfulness	Poeticness
AVERAGE	7.52	5.13	6.73
MEDIAN	7	5	7

Table 5.15: Human evaluation of the MLPG on the French dataset

Below is an example of produced poem:

LE LAC.

Est terrestre trottoirs retrouverez cassées
Françaises souffrances chinois sa vallée
Comptaient ressources, retrouverez être, mâchoires
Souffrances retrouverez, âme parfois alla.
Ressources prendraient est blessures prendrait,
Avait ressources est du, trajet,
Insecte, grève, insecte du croissant
Ressource retrouverez tranquilles souffrances dieux.
Sa sanglantes est jusque piquant
Souffrances ouvraient trottoirs retrouverez restez
Terrestre souffrances retrouverez squelettes tracée
Laissaient souffrances ouvraient françaises cadran.
Laissaient souffrances ouvraient françaises hôtel
Laissaient souffrances ouvraient françaises décret.

which literally translates to English as:

THE LAKE.

East terrestrial sidewalks will find broken sidewalks
French women suffer Chinese suffering in the valley
Counted resources, will find being, jaws
Suffering will find again, soul sometimes went.

Resources would take and injuries would take,
Had resources is due, journey,
Insect, strike, crescent insect
Resource will find quiet suffering gods.
Its bloody is even spicy
Suffering opened sidewalks will find you stay
Earthly suffering will find skeletons traced
Let sufferings open French dial.
Let sufferings open French hotel
Let sufferings open French decree.

Chapter 6

Discussion

The different methods used to achieve multilingual poetry generation all show promises and limits.

The first experiment, building a bilingual network trained on a common dataset, is the easiest to implement but demonstrates the difficulty for a simple network to split the data into different language categories. While the network starts learning some words both in English and French, they are usually mixed within the sentence. The results that were obtained in Östling and Tiedemann [2016] are not accessible by such a simple network without the use of appropriately trained language vectors.

In the second experiment, the network is still simple but this time it is trained once per language. The model does not need to distinguish between languages, only to learn poetry generation with the given language at hand. Grammaticality is quite easily achieved by the network but both meaning and poeticness score quite low, especially in French. The difference of quality in both languages whatever the datasets chosen tend to show that French is a harder language to learn, which can be explained by the numerous grammatical rules, the multiple genders per word and other complex rules in the structure of the sentence. Yet the network is able to generate simple words although it is only trained at a character-level.

In the third experiment, the network is more complex and many rules are manually implemented to constrain the learning, including constraints on POS tags and on rhymes. As a consequence, grammaticality scores really high and state-of-the-art results found by Benhart et al. [2018] are retrieved. It is noteworthy that poeticness

scores higher than previously, most likely due to the rhythms that are enforced by the model. Once again, French achieves lower results, especially in meaning, which could suggest limitations in the training dataset. The process of translating the network from one language to the other makes it possible to extend this approach to other languages, although the rules need to be re-implemented in each language, which can be a tedious process as demonstrated by the POS tag conversion in 5.3. Yet the resulting difficulty does not overtake the manual work described by Oliveira et al. [2017] when extending his system.

Finally, the fourth experiment combines previous results by stacking together a phoneme-based LSTM, trained on multiple languages, with a language-specific LSTM that outputs the final poem. The former network manages to learn some phonetic structure in the data but still distinguishes both languages, while the latter achieves to produce meaningful and grammatically correct text. The results are lower than the previous model, but the increase in poeticness is promising as well as the fact that new languages become easy to add.

Both objectives stated in the introduction are fulfilled to some extent and are detailed below:

1. Understanding the role played by a chosen language in the process of learning poetry generation

Different languages display various degrees of difficulty and perform differently, as highlighted by the experiments conducted in English and French. The gap in results between two languages seems to grow wider as the network gets more complex.

Although it would be interesting to extend the studies to other languages, there is no expected major difference by using other languages with a Latin alphabet. Only languages with a non-Latin one would have to be treated specifically. In particular some languages do not have POS-tagging (Chinese), so the rules added in the third and fourth models would have to be removed or updated.

2. Reusing this knowledge to improve traditional networks in the task of generating human-like poetry

The final model managed to produce high quality poetry in both English and French, as demonstrated by the human evaluation that has been performed. A complex

architecture was needed for it, requiring the use of multiple networks and datasets.

The first experiment makes it obvious that a simple network is not able to correctly learn the difference between two languages, which is consistent with previous research, in particular with the work of Dhar, who failed to find any transfer of linguistic information from one language to the other when simultaneously learning multiple languages:

> "There is currently no evidence that syntactic transfer occurs in our setup. A possible explanation is that the bilingual model has to fit the knowledge from two language systems into the same number of hidden layer parameters and this may cancel out the benefits of being exposed to a more diverse set of sentences." (Dhar and Bisazza [2018])

Chapter 7

Conclusion

In this project, different methods for poetry generation were investigated in the context of multilingual outputs. The different experiments have led to a better understanding of the mechanisms involved in bilingual poetry generation and to the development of a MultiLingual Poetry Generator (MLPG), which is able to produce poetry in different languages by learning from a combined and a language-specific dataset, all gathered from scratch.

While the final network does not achieve state-of-the art results, it displays poeticness at a level that was evaluated as greater than previous state-of-the-art networks, in both English and French. Yet performance assessment in the field of poetry generation still relies on human evaluation, which can vary greatly between different evaluators.

7.1 Contributions

The following contributions were made to the field as part of this work:

1. A *MultiLingual Poetry Generator* was developed. The model combines a multilingual phoneme-based and a language-specific networks to produce poetry in both French and English.

2. A dataset of English and French poems was gathered, totaling 41.6k verses,

261.7k words and 29.2k unique words.

3. Different models developed for poetry generation were studied, from traditional rule-based approaches to multilingual models, including a wide range of deep learning methods based on textual or visual inputs.

4. Challenges raised by multilingual poetry generation were listed, including the difficulty for a network to distinguish languages, the strenuousness of converting manual rules from a language to another and the general complexity of making a network learn aesthetics information rather than pure linguistic features in the data.

7.2 Further work

As part of possible further work, evaluation remains limited and subjective in the conducted experiments. While automated evaluation metrics rarely fit the field of poetry generation, it would be interesting to assess our models based on ROUGE (Lin [2004]) or the more recent BERTSCORE (Zhang et al. [2019]). More generally, developing reliable automated evaluation metrics specific to poetry generation seems necessary to compare models in a more relevant way.

Another key point is the addition of new languages. While the use of English is very standard in poetry generation research, for obvious reasons, implementing the production of French poems was more challenging in many ways, including rhymes retrieval, POS constraints and validation accuracy. It would be relevant to train the MLPG on more languages to compare results and ensure that adding any other language does not involve unforeseen extra steps.

In addition, while most French poets composing the datasets were from the 18th and 19th centuries, no modern poet was considered, which does not allow to draw a comparison with modern English poems like those of Bradburne.

Finally, one can point out limitations of transfer learning in NLP, that would have to be overcome. The final model still uses GloVe or an equivalent as a pretrained representation, which are trained on very general datasets such as Wikipedia to learn linguistic information. While such a method greatly facilitates learning, the

datasets used for pretraining are quite different from poetry and thus the context of the words in the pretraining dataset can be quite far from the one in the training dataset. One possible way of solving this issue would be to use a deep language model for transfer learning, possibly the recent BERT model from Devlin et al. [2018].

Bibliography

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166, 2019.

Samuel Ballas. Generating poetry with PoetRNN. Website, 2015. URL http://sballas8.github.io/2015/08/11/Poet-RNN.html. Last accessed on 2019-04-13.

John Benhart, Tianlin Duan, Peter Hase, Liuyi Zhu, and Cynthia Rudin. Shall i compare thee to a machine-written sonnet? An approach to algorithmic sonnet generation. *arXiv preprint arXiv:1811.05067*, 2018.

Tarek R Besold, Marco Schorlemmer, and Alan Smaill. *Computational Creativity Research: Towards Creative Machines*. Atlantis Press/Springer, 2015.

Margaret A Boden. *The Creative Mind: Myths and Mechanisms*. Routledge, 2004.

Margaret A Boden. How creativity works. In *Creativity: Innovation and Industry conference*. Creativity East Midlands, 2007.

Carnegie Mellon University. The Carnegie Mellon University Pronouncing Dictionary. Website, 2014. URL http://www.speech.cs.cmu.edu/cgi-bin/cmudict#about. Last accessed on 2019-05-02.

Bill Chamberlain. Racter. Website, 1983. URL https://www.atariarchives.org/deli/write_about_itself.php. Last accessed on 2019-05-11.

Tseng-Hung Chen, Yuan-Hong Liao, Ching-Yao Chuang, Wan-Ting Hsu, Jianlong Fu, and Min Sun. Show, adapt and tell: Adversarial training of cross-domain image captioner. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 521–530, 2017.

Wen-Feng Cheng, Chao-Chung Wu, Ruihua Song, Jianlong Fu, Xing Xie, and Jian-Yun Nie. Image inspired poetry generation in XiaoIce. *arXiv preprint arXiv:1808.03090*, 2018.

Simon Colton, Jacob Goodwin, and Tony Veale. Full-FACE poetry generation. In *Proceedings of the 3rd International Conference on Computational Creativity (ICCC)*, pages 95–102, 2012.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Prajit Dhar and Arianna Bisazza. Does syntactic knowledge in multilingual language models transfer across languages? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 374–377, 2018.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

Eric Elshtain. Gnoetry. Website, 2006. URL http://www.beardofbees.com/gnoetry.html. Last accessed on 2019-05-20.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482, 2015.

Jean-Philippe Fauconnier and Mouna Kamel. Discovering hypernymy relations using text layout. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 249–258, 2015.

Daniel C Ferreira, André FT Martins, and Mariana SC Almeida. Jointly learning to embed and predict with multiple languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 2019–2028, 2016.

Tao Ge, Furu Wei, and Ming Zhou. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*, 2018.

Pablo Gervás. An expert system for the composition of formal Spanish poetry. In *Applications and Innovations in Intelligent Systems VIII*, pages 19–32. Springer, 2001.

Pablo Gervás. Exploring quantitative evaluations of the creativity of automatic poets. In *Computational Creativity*, pages 275–304. Springer, 2019.

Pablo Gervás. WASP: Evaluation of different strategies for the automatic generation of Spanish verse. In *Proceedings of the AISB-OO Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pages 93–100, 2000.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, 2016.

Didier Glehello. Poésie française. Website, 2007. URL https://www.poesie-francaise.fr. Last accessed on 2019-04-13.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Jing He, Ming Zhou, and Long Jiang. Generating chinese classical poems with statistical machine translation models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Xavier Hinaut, Johannes Twiefel, Maxime Petit, Peter Dominey, and Stefan Wermter. A recurrent neural network for multiple language acquisition: Starting with English and French. In *NIPS 2015 Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory*, volume 9, pages 1735–1780. MIT Press, 1997.

Jack Hopkins and Douwe Kiela. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 168–178, 2017.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. Improved deep learning baselines for Ubuntu Corpus Dialogs. In *Neural Information Processing Systems Workshop on Machine Learning for Spoken Language Understanding*, 2015.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*, 21, 2015.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics*, pages 177–180, 2007.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 317–325, 2017.

Carolyn Lamb, Daniel G. Brown, and Charles L.A. Clarke. A taxonomy of generative poetry technique. *Journal of Mathematics and the Arts*, 11:159–179, 2017.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Bei Liu, Jianlong Fu, Makoto P. Kato, and Masatoshi Yoshikawa. Beyond narrative description: Generating poetry from images by multi-adversarial training. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 783–791. ACM, 2018.

Malte Loller-Andersen and Björn Gambäck. Deep learning-based poetry generation given visual input. In *ICCC*, pages 240–247, 2018.

Edward Loper and Steven Bird. NLTK: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2017.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The Ubuntu Dialogue Corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*, 2015.

Hisar Manurung. An evolutionary algorithm approach to poetry generation. PhD. Thesis, University of Edinburgh, 2003.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

Boris New and Christophe Pallier. The lexique database. Website, 1999. URL http://www.lexique.org/. Last accessed on 2019-05-02.

Christopher Olah. Colah's blog. Website, 2015. URL http://colah.github.io. Last accessed on 2019-05-11.

Hugo Gonçalo Oliveira. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21, 2012.

Hugo Gonçalo Oliveira. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–20, 2017.

Hugo Gonçalo Oliveira, Raquel Hervas, Alberto Diaz, and Pablo Gervás. *Multilingual Extension and Evaluation of a Poetry Generator*, volume 23, pages 929–967. Cambridge University Press, 2017.

Robert Östling and Jörg Tiedemann. Continuous multilinguality with language vectors. *arXiv preprint arXiv:1612.07486*, 2016.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and R Ward. Semantic modelling with long-short-term memory for information retrieval. *arXiv preprint arXiv:1412.6629*, 2014.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Abhay Prakash, Chris Brockett, and Puneet Agrawal. Emulating human conversations using convolutional neural network-based IR. *arXiv preprint arXiv:1606.07056*, 2016.

Raymond Queneau. 100.000. 000.000. 000 de poemes. *Gallimard Series. Schoenhof's Foreign Books, Incorporated*, 1961.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Aleksey Tikhonov and Ivan Yamshchikov. Sounds wilde. phonetically extended embeddings for author-stylized poetry generation. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 117–124, 2018a.

Alexey Tikhonov and Ivan P Yamshchikov. Guess who? Multilingual approach for the automated generation of author-stylized poetry. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 787–794. IEEE, 2018b.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*, 2019.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

Ziang Xie. Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534*, 2017.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. Automatic poetry generation with mutual reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3143–3153, 2018.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, 2014.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. *arXiv preprint arXiv:1903.00138*, 2019.

NTNU
Norwegian University of
Science and Technology