# Towards Automated Translation of Poetry

Michael A.

Exeter Mathematics School
2017

**Abstract**

As a prerequisite to automated machine translation of poetry, I present possible methods of improving current word- and phrase-based statistical methods to make them more suitable for poetic and literary works.

# Introduction

*"Poetry is that which is lost in translation"*
*Robert Frost*

Translation is hard. It is practically impossible to translate long sections of text from one language to another without at least some cultural awareness of both languages. If you were to leave a monolingual English speaker in a room with an English-Chinese dictionary and, say, a copy of the *Wǎngchuān jí* (a collection of Tang-dynasty poetry by Wang Wei and Pei Di, from around 740), and ask them to translate it, the translation you will get is unlikely to be any good. Even experienced translators and poets struggle with poetry translation; they will often misinterpret meaning, change the style or tone of the poem, or in some other way alter it to their liking.[1] Translation is a fickle mistress, and not for the faint-hearted.

Machine translation is, of course, even harder. In fact, machine translation is considered AI-complete.[i] Translating non-fiction texts (newspapers, etc.) is hard enough: sparse data, ambiguity, and insufficient computing resources mean that even state-of-the-art systems cannot quite reach perfect translation quality. When it comes to translating poetry, the stumbling blocks are even higher. Classical machine translation systems focus (quite rightly) on preserving the content of text above all else[ii] — in essence, translating only the top level of meaning. The small parts of text — vocabulary choices, word order, metaphor — are invariably 'lost in translation' when machine translation is used. But it is these small details that make poetry as beautiful as it is; content is tightly interwoven with form, and translating only the top

[i] Solving a problem which is AI-complete is, in practice, the same as managing to create an artificial intelligence which can do anything a human can.

[ii] Interestingly, some professional translators do the same — Eugin Nida, for example, says: "Only rarely can one reproduce both content and form in a translation, and hence in general the form is usually sacrificed for the sake of the content."[2]

level of meaning may end up leaving the poem completely nonsensical.

This research does not intend to fully solve the machine translation problem: in fact, I doubt it will ever be solved. Poetry is at its core a human expression, and machines cannot hope to emulate that. Discussing the philosophical issues behind machine translation is beyond the scope of this report: I will leave those issues to the philosophers. Instead, I hope for this report to suggest ideas which may act as an aid to human translators, not replace them.

This report will be split into three parts. As a preliminary, I will explain the history of both computational linguistics and machine translation, to give the reader an overview of the two tightly woven fields and a historical context for this research; there is also a discussion on *why* one would want to translate poetry computationally. Secondly, I will describe the current techniques in machine translation: I will cover the statistical modeling of language, and current approaches to machine translation, with a focus on word- and phrase-based statistical methods. Finally, using Goethe's "Über allen Gipfeln" I will evaluate the current approaches to manual and computational translation, and discuss my research into improving these methods.

# Background

*"Prose: words in their best order;*
*poetry: the best words in the best order"*
*Samuel Taylor Coleridge*

Computational linguistics (or Natural Language Processing, NLP) is defined by the Association for Computational Linguistics as: 'the scientific study of language from a computational perspective'.[3] The field has its origins in the United States, immediately following the Second World War.

Warren Weaver, an American scientist, wrote a now-famous memorandum[4] in 1947 to Norbert Wiener at MIT:

> I have wondered if it were unthinkable to design a computer which would translate. Even if it would translate only scientific material (where the semantic difficulties are very notably less), and even if it did produce an inelegant (but intelligible) result, it would seem to me worth while.

Weaver made four core arguments about possible approaches to machine translation (MT):[5]

1. The problem of one word having multiple meanings can be solved by looking at the $n$ words (or just nouns) around it to disambiguate from context.

2. A theorem proved in 1943 by McCulloch and Pitts[6] suggests that "a robot... is capable of deducing any legitimate conclusion from a finite set of premises". Under the assumption that language is logical, language understanding may be possible.

3. Translation problems can be looked at as a cryptographic process: "a book written in Chinese is simply a book written in English which was coded into the 'Chinese code'," and cryptographic methods can therefore be used to translate text.

4. Instead of translating language-to-language, we should descend "down to the common base of human communication — the real but as yet undiscovered universal language" to express a concept, then express it in the desired language.

Weaver's theorems, while not necessarily state-of-the-art today, were enough to fuel an increase in interest for language processing and machine translation. Most work in the early days was military funded: it concerned Russian

→ English (and to a minor extent English → Russian) translation, intended for use by the United States government against the USSR during the Cold War.[7] Its development continued to the mid-1960s, although it was strongly confined by the computing power available; systems were primarily rule-based, using limited dictionaries. Researchers of the famous Georgetown experiment[iii] estimated that machine translation would be a solved problem in three to five years.[8] However, the development was not fast enough: in 1966 the Automatic Language Processing Advisory Committee (ALPAC) published a skeptical report on the future of machine translation:[9]

> At present [MT] serves no useful purpose without postediting [...] we will only have adequate mechanical translation when the machine can "understand" what it is translating [...] perhaps our attitude might be different if there were some pressing need for machine translation, but we find none.

The report did not bode well for NLP research: none of its nine recommendations supported any sort of research in MT or NLP.[10] Research in MT stalled, especially in the US. Work began to be done in applying linguistics, rather than raw computing power, to problems. A great deal of work was done in natural language understanding (for example with SHRDLU[11]). Eventually, by the 1980s, developments in corpus linguistics and machine learning began to reawaken the slumbering MT community. The focus of research shifted from closed to open domains, and from hand-written rules to corpus-based statistical approaches.[12] In 1990, IBM published a seminal work in the field, describing progressively more complex statistical models for word-based machine translation.[13] As research boomed, so too did commercial usage of MT systems: the most famous examples of which being

[iii] A demonstration of machine translation in 1954, which translated 60 Russian sentences to English with quite astounding accuracy and drew a large amount of press attention: however, the methodology was flawed and the system was not nearly as impressive as it seemed.

SYSTRAN (used by the US Department of Defense) and METEO (used for English-French weather bulletin translation in Canada).

The fields of machine translation and computational linguistics continue to grow today. Most major technology companies (Apple, Google, Facebook, etc.) have dedicated research departments for computational linguistics. Some uses are ubiquitous: Siri, Google Translate, Cortana. The rapid pace of development is underscored by the speed and scale of improvements in related fields such as machine learning. Some of today's most advanced methods of machine translation come close to achieving human-level accuracy in translation tasks.[14]

But, machine translation is still mostly used for translating non-fiction text like scientific papers or newspaper articles. So why do we want to translate poetry? A couple of reasons come to mind, which specifically motivated my interest in this problem:

- Translated poems have historically built up the rich tapestry of English literature: where would we be without Dante, Virgil, Homer? The ability to computationally translate poems may allow yet more foreign poems to enter the consciousness of the English language.[18]

- Furthering the previous point, there are many poems written in languages which do not have an active, or for that matter any, translation community. Machine translation techniques could be used to translate these poems where humans cannot.[iv]

- Some argue that a truly faithful translation requires the translator to have intimate familiarity with both the source and target languages:[17] if a computer could do the job of being familiar with one of the lan-

[iv] For discussion of machine translation for resource-poor languages, see Nakov & Ng (2014),[15] Avramidis & Koehn (2008).[16]

guages, it would lighten the cognitive load on the translator and perhaps allow for a better-quality translation.

- Translated poems are often worth reading in their own right: "when translator and original are in tune. . . a third poem is created";[18] indeed, it would be an interesting creative exercise to analyze the poems which result from machine translation.

- The hope that a more computational, quantitative treatment of poetry and poetic analysis will encourage the more scientifically-minded to explore poetry and the beauty contained therein.

# Theory

*"How tired I am of stories, how tired I am of neat little phrases that come down so beautifully with all their feet on the ground!"*
*Virginia Woolf*

**A cognitive model of language** Before we can begin to discuss computational approaches to language modeling, it is best that we take a moment to discuss cognitive models of language: essentially, a way of trying to approximate and describe human cognitive (thinking) processes. For all that humans are very good at learning and speaking languages — both as children and adults — we really have no idea *why* we're so good at it. As early as Roger Bacon's 1245 work 'Overview of Grammar',[19] linguists (or philosophers and grammarians, as they were called in those days) have suggested that all languages are built upon a common cognitive grammar: that is, human languages are an innate experience. Noam Chomsky formalized the idea of a 'universal grammar' in 1965:[20] the idea that there is an innate language faculty in humans that 'knows' the rules that allow for language to be developed and used, for example distinguishing nouns from verbs. But exactly

what this universal grammar is, is not a question which has been answered yet.

At multiple points throughout the history of NLP, researchers have tried to computationally emulate this internal grammar structure, especially within the context of machine translation. It stood to reason that if we could build a computational grammar good enough, machine translation would become easy — in a similar way to how Weaver suggested we use the "universal language". The simplest of these attempts were rule-based machine translation systems (RBMT), which came in three groups:[21]

1. Direct systems, or dictionary-based machine translation, which consisted purely of dictionary lookups.

2. Transfer systems, which broke translation up into three steps:

   (a) Analysis of the source language input to find its grammatical structure by, for example, morphological analysis (classifying each word as eg. noun, verb, as well as gender, tense, etc), or word-sense disambiguation;

   (b) Transfer of the resulting structure into a structure useful for the target language; and

   (c) Generation of text in the target language from the final structure.

3. Interlingua systems, which converted source input to an intermediate language-independent representation, then converted that to the target language.

For all that these approaches became very popular in the MT community, they did have some major problems. Rule-based systems work reasonably well for closed domains (where you know roughly what format your input and output must take, like in the case of METEO), but when it comes to open systems (where the input and output could be anything), the amount of

work needed to maintain complex rules, including accounting for ambiguity and idiomatic expressions, quickly becomes insurmountable.

**A statistical model of language** All these rule-based, computational models of cognitive grammar are bound to be incomplete: simply put, we do not yet have enough knowledge of how the brain interprets and creates language — what internal structures it uses, and so forth — to be able to express these methods and structures computationally. So we compromise. Natural language processing and generation (of the kind needed in machine translation) doesn't require fully emulating human use of language: it only needs to approximate it, mimic it in such a way that it seems close to indistinguishable from the real thing. A good way of doing this is a probabilistic approach.

The best way to explain the probabilistic model is in context: so let's take a look at the statistical model of machine translation. Say we want to translate a foreign sentence $f$ into an English sentence $e$. Then, by a probabilistic model,[22] we want our translation engine to maximize the probability that a given English sentence is the correct translation of the foreign sentence: that is, maximize $\Pr(e|f)$. Intuitively, we can interpret this probability as the probability a translator will produce $e$ in the target language when presented with $f$ in the foreign language.[13]

Modeling this probability distribution is difficult: we can simplify the problem by applying Bayes' theorem:[23] by Bayesian decomposition, we can rewrite $\Pr(e|f)$ as:

$$\Pr(e|f) = \frac{\Pr(e)\Pr(f|e)}{\Pr(f)}$$

Note that the denominator $\Pr(f)$ does not depend on $e$, and so is the same for all possible values of $e$: therefore we can just simplify our equation to be:

$$\Pr(e|f) = \Pr(e)\Pr(f|e)$$

This gives us two new variables to maximize: $\Pr(e)$, the *language model*, and $\Pr(f|e)$, the *alignment model*.

**Corpuses** But before we can start, we need to acquire a corpus. A corpus, roughly speaking, is just a collection of text in our target language. It can come from any source, and indeed corpora used in linguistics vary widely: examples include the British National Corpus (consisting of 100 million words, covering British English in the late 20th century), Brown Corpus (1 million words, and tagged for parts-of-speech), and Google Books N-Gram corpus (multilingual, and searchable online).

Within machine translation, we need two types of corpus: a monolingual (language model) corpus, and a parallel bilingual corpus. When looking for a corpus to use, the most important thing to consider is the domain and topic of the text. For machine translation, the domain and topic of the corpus should, as closely as possible, match the domain and topic of the text to be translated. So, for example, if we were to be translating scientific documents, then the best choice for a corpus to learn from would be a corpus of other scientific documents. In an ideal world, to translate a given poem, we'd have: a monolingual corpus, entirely consisting of poems in the target language in the same style as our poem to translate; and a bilingual corpus of poems in the source language, of the same style as our input poem, translated by professionals into the target language. However, that does throw up some problems, which will be discussed later.

**Sentence alignment** Once we have acquired a parallel corpus to work with, we still need to perform sentence alignment. When a human translator translates some text, they do not necessarily translate it word-for-word or sentence by sentence: this is especially true for languages where the writing system does not distinguish sentence boundaries. By aligning sentences within a corpus, we can more easily find word and phrase pairs when building our alignment model.

Formally,[22] given a set of sentences in a foreign language $f = f_1, f_2 \dots f_{n_f}$ and their translations into English $e = e_1, e_2 \dots e_{n_e}$, we want to find a sentence alignment $S = \{s_1, s_2 \dots s_n\}$ such that all sentences in both the source and target language are present within the alignment, and each sentence only occurs in one sentence pair. The simplest way to do this is with the Gale-Church sentence alignment algorithm.[25] The intuition behind their algorithm is that longer sentences in one language tend to be translated to longer sentences in the other language, and vice versa. A probabilistic score is assigned to sentence pairs based on both the ratio of character lengths in the sentences, and the variance of this ratio. Then a dynamic programming framework is used to find the most likely sentence alignment.

Two steps are required for this algorithm: first, paragraphs of parallel text are aligned, then sentences within each paragraph are aligned. Paragraph alignment is reasonably easy, as paragraphs are almost always already aligned in translated text, and tend to be very clearly marked out. This means that they can be aligned automatically, with the output being corrected by hand if needed.

First we define a distance function $d$ which allows for sentence insertion, deletion, and substitution, which takes four arguments, $x_1, y_1, x_2, y_2$. We let:[25]

1. $d(x_1, y_1, 0, 0)$ be the cost of substituting $x_1$ with $y_1$,
2. $d(x_1, 0, 0, 0)$ be the cost of deleting $x_1$,
3. $d(0, y_1, 0, 0)$ be the cost of inserting $y_1$,
4. $d(x_1, y_1, x_2, 0)$ be the cost of combining $x_1$ and $x_2$ to make $y_2$,
5. $d(x_1, y_1, 0, y_2)$ be the cost of expanding $x_1$ to $y_1$ and $y_2$
6. $d(x_1, y_1, x_2, y_2)$ be the cost of merging $x_1$ and $x_2$ and matching with $y_1$ and $y_2$

Then we can use a dynamic programming algorithm to align the sentences, defined as follows:[25]

> Let $s_i$, $i = 1 \ldots I$, be the sentences of one language, and $t_j$, $j = 1 \ldots J$ be the translations of those sentences in the other language. Let $d$ be the distance function defined previously, and let $D(i, j)$ be the minimum distance between sentences $s_1 \ldots s_i$ and their translations $t_1 \ldots t_j$, under the maximum likelihood alignment. $D(i, j)$ is computed by minimizing over six cases (substitution, deletion, insertion, contraction, expansion, and merger) which, in effect, impose a set of slope constraints. That is, $D(i, j)$ is defined by the following recurrence with the initial condition $D(i, j) = 0$:

$$D(i,j) = \min \begin{cases} D(i, j-1) & + & d(0, t_j, 0, 0) \\ D(i-1, j) & + & d(s_i, 0, 0, 0) \\ D(i-1, j-1) & + & d(s_i, t_j, 0, 0) \\ D(i-1, j-2) & + & d(s_i, t_j, 0, t_{j-1}) \\ D(i-2, j-1) & + & d(s_i, t_j, s_{i-1}, 0) \\ D(i-2, j-2) & + & d(s_i, t_j, s_{i-1}, t_{j-1}) \end{cases}$$

**Language model** With a language model, we can assign a probability to a sequence of words. Intuitively, the sentence 'the house is small'[22] should be more likely than 'small the is house'.[v] We can use a statistical language model to accurately describe just how much more likely it is. To build our language model, we take a single-language corpus and use $n$-grams by way of the Markov assumption,[24] in a similar way to Weaver's original suggestions.

[v] However, to quote Chomsky: "It must be recognized that the notion 'probability of a sentence' is an entirely useless one". The probability of a sentence has no intrinsic linguistic meaning, because humans do not produce language on a statistical basis.

Let's say that we want to estimate the probability of a sequence of words, $W = w_1, w_2, \ldots w_n$. The naïve approach to this would be to take a sufficiently large corpus, count how many times $W$ appears in it, and divide by the total number of $n$-word sequences to get a probability for $W$: however, even for an extremely large corpus, it is very unlikely that $W$ will have appeared at all in its exact form, making probability distributions extremely sparse. What we can do instead is use the chain rule to decompose the probability of $W$:[22]

$$\Pr(w_1, w_2, \ldots w_n) = \Pr(w_n | w_1 \ldots w_{n-1})$$

This is still cumbersome to calculate, so we can restrict our calculations to only a history of words, that is

$$\Pr(w_n | w_1 \ldots w_{n-1}) \approx \Pr(w_n | w_{n-m} \ldots w_{n-1})$$

We use the Markov assumption here by saying that only a limited number of previous words affect the probability of the $n$th word.[vi] Most commonly, language models will restrict their search space to the previous two words (called *trigrams*, because they consider sets of three words) or just the previous word (correspondingly called *bigrams*). Expanding the search space for larger values of $n$ is usually a matter of diminishing returns.

If we therefore wanted to estimate the probability of a given trigram consisting of the sequence $T = w_1, w_2, w_3$, we would count the amount of times we see the sequence $w_1, w_2, w_3$ in the corpus, and divide it by the amount of times we see just the sequence $w_1, w_2, x$ for all words $x$: that is, we divide the count of that specific trigram by the count of all trigrams which start with its history. Then by maximum likelihood estimation,[22] the probability of $T$ is therefore:

$$\Pr(T) = \frac{\text{count}(w_1, w_2, w_3)}{\sum_x \text{count}(w_1, w_2, x)}$$

[vi] While this assumption is technically wrong, it is a convenient simplification, because shorter word histories have less sparse probability distributions.

The formula for a general $x$ is defined similarly. We can calculate this for all occurring $n$-grams in a corpus to give us an $n$-gram probability distribution for that corpus.

It is worth noting at this point that these resulting probabilities are not (and should not be!) used as they are: $n$-gram models are very sensitive to the training corpus,[21] and as such we have two issues: sparse data (probabilities for most $n$-grams will generally be quite low, by Zipf's law,[28] and will be zero for $n$-grams not seen in the training corpus), and how to deal with out-of-vocabulary (OOV) tokens (any $n$-gram containing a word not seen in the training corpus will have a probability of 0). To deal with OOV tokens, we can either simply insert an 'unknown' token (usually 'UNK') into the training data, or we can replace the first occurrence of each vocabulary word with 'UNK', and then calculate its probability distribution in the same manner as any other word. The best method of dealing with the sparse data issue is by using smoothing methods: these allow us to get better estimates for $n$-grams which occurred rarely in the training data, or not at all. There are numerous methods, which I will not go into here, including Kneser-Ney Smoothing and Good-Turing estimation. Backoff and interpolation methods may also be used.[21]

**Alignment model** How we approach our alignment model depends on if we are using *word-based* or *phrase-based* machine translation.[22]

**Word-based** While word-based models are no longer state-of-the-art, the methods which are used for them remain relatively applicable to modern MT problems, as well as being interesting in their own right. The following section will be based on the IBM models of machine translation first presented in 1990,[13] which are still used to an extent in the GIZA and GIZA++ toolkits for alignment. Word-based models, are, predictably, based on translating words in isolation, or *lexical translation*. Firstly, we take a corpus of texts

in a foreign language, and their translations into English, and then calculate statistics (a lexical translation probability distribution) from these.

Formally speaking, we want to calculate the probability that a given foreign word $f$ translates into a given English word $e$, given the data in our corpus. The most straightforward way of doing this is, again, maximum likelihood estimation: we take the count of the amount of times we see $f$ translated to $e$, and divide it by the amount of times we see $f$ overall in the corpus. Smoothing can again be used here to account for data sparsity and OOV tokens.

Given this probability distribution, we can now get to the task of building our alignment model. A foreign sentence $F$ can be translated to an English sentence $E$ by use of an alignment: a mapping of foreign words to English words. We can map this using an alignment function, which may allow for adding, removing, and duplicating words: in this case, each English word must be linked to exactly one input word, which may be a special NULL token. On the other hand, one input word may be linked to multiple output words, or none at all.[22]

The IBM Model 1 is a generative model (meaning it breaks the problem up into smaller steps, and then generates a final answer using these intermediate steps), and is based only on the aforementioned lexical translation probability distribution. It is described in quite some detail in Brown et al. (1993),[26] so I will only give a superficial gloss. In short, the IBM Model 1 defines the probability of translating a foreign sentence $f = f_1, f_2 \dots f_n$ of length $l_f$ into an English sentence $e = e_1, e_2 \dots e_n$ of length $l_e$, with an alignment of each English word $e_j$ to a foreign word $f_i$ according to the alignment function $a(j) = i$. The formula to calculate this probability is:

$$\Pr(e, a|f) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

where the leading fraction is used to normalize the distribution so all probabilities are between 0 and 1, and $t(e_j|f_{a(j)})$ is the probability of a given English word being translated into its aligned foreign word, from our earlier lexical translation probabilities.

Higher IBM Models increase in complexity and improve upon the first model's flaws; in short, they are:[22]

- Model 1: lexical translation, as discussed above.
- Model 2: adds an explicit alignment model, which allows for a more statistically likely alignment.
- Model 3: adds a fertility model, which estimates the amount of output words that a foreign word 'produces', allowing for input words to be dropped (eg. flavoring particles), and a distortion probability, which allows for more reordering in translation.[vii]
- Model 4: adds a relative alignment model, which fixes issues with Model 3's distortion model by looking at relative distortions instead.
- Model 5: fixes issues with previous models where multiple output words could be placed in the same positions, which wasted probability mass.

**Phrase-based** Phrase-based models are widely considered as state-of-the-art today[22] and outperform word-based models by a large margin.[27] As the name suggests, they are based on phrases (short sequences of words), and processing is done on these phrases. The motivation behind this is clear to speakers of foreign languages: in many cases, a word in one language cannot be easily translated to another language because it has phrase-specific meanings. Phrase-based translation segments an input sentence into phrases, translates those phrases, and reorders them as needed. This has multiple

[vii] In practice, the Model 3 is sufficient for modern uses: using any models after this is superfluous.

advantages compared to the word-based model: it allows for one-to-many and many-to-one mappings, resolves translation ambiguities, and is conceptually much simpler.[22]

We start with exactly the same base equation as for word based models: that is, we look to maximize

$$\Pr(e|f) = \Pr(f|e)\Pr(e)$$

However, for the phrase-based model, we further decompose $\Pr(f|e)$ into phrase translations:[22]

$$\Pr(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{j} \phi(\bar{f}_i|\bar{e}_i)\ d(\text{start}_i - \text{end}_{i-1} - 1)$$

We break up each foreign sentence $f$ into some $I$ phrases $\bar{f}_i$, and the same for each English sentence; each segmentation of phrases is equally likely. $\phi$ is a mapping of the probability of each English phrase being translated into a foreign sentence. The distance model $d$ is used to reorder the phrases.

Splitting a sentence into phrases can be done using a word alignment, like those seen in the earlier word-based translation section. We collect all aligned phrase pairs that are consistent with the word alignment. A phrase pair ($\bar{f}$, $\bar{e}$) is *consistent* with an alignment $A$ if all words in $\bar{f}$ with alignment points in $A$ have corresponding alignment points with words in $\bar{e}$, and vice versa: as a result, the words in a consistent phrase pair are only aligned with each other, and not to words outside that phrase pair.[viii] We can then estimate the phrase translation probability distribution in much the same way as we estimate $n$-gram probabilities, as

$$\phi(\bar{f}|\bar{e}) = \frac{\text{count}(f, \bar{e})}{\sum_{\bar{f}} \text{count}(\bar{f}, \bar{e})}$$

[viii] This can leave us with phrases which are non-intuitive, for example 'house the': interestingly, pruning out these non-intuitive, syntactically incorrect pairs actually can decrease the performance of a model.[27]

The reordering of our English phrases, relative to the input foreign phrases, is done by the relative distortion probability function $d(s_i - e_{i-1})$, where $s_i$ is the start position, in terms of words, of the foreign phrase which was translated to the $i$th English phrase, and $e_{i-1}$ is the end position of the foreign phrase translated to the $(i-1)$th English phrase. For simplicity, $d$ is modeled by an exponentially decaying cost function, $d(x) \approx \alpha^{|x|}$, for some value of $\alpha$, $0 \leq \alpha \leq 1$, so larger distortions are much less probable, although some work has been done on modeling distortion using joint probability distributions.[29]

# Discussion

*"What is translation? On a platter*
*A poet's pale and glaring head"*
*Vladimir Nabokov*

Following the heavy math of the previous section, I return here to the crux of the problem: poetry, and translating it. To dig into the meat of the problem, let's consider an exemplar poem, and how we might approach translating it, both by hand and computationally.

Vladimir Nabokov, in the notes of his controversial translation of Aleksandr Pushkin's work 'Eugene Onegin',[30,31] argues that all translations of poetry will inevitably fall under three categories:

**Paraphrastic** A free version of the original: words and phrases are toyed with — added, removed, or changed — in order to conform to some form that the translator wishes.

**Lexical** Translating the basic meaning of words and their order.

**Literal** Translating as closely as possible the original, with the exact contextual meaning being preserved.

Nabokov believed that only the final item, literal translation, was a 'true' translation, and all others only served to tarnish the poem:

> The hack who has never read the original, and does not know its language, praises an imitation as readable because easy platitudes have replaced in it the intricacies of which he is unaware.

Nabokov's translation of 'Eugene Onegin', predictably, fell into the 'literal' camp of translation. The original poem consisted of stanzas of iambic tetrameter, with the rhyme scheme 'AbAbCCddEffEgg'. Note that uppercase letters represent feminine rhymes (a rhyme which matches two or more syllables, in which the final syllables are unstressed), and lowercase letters represent masculine rhymes (a rhyme with matches only one syllable, which is stressed). Nabokov argued that "to reproduce the rhymes and yet translate the entire poem literally is mathematically impossible", and as such decided to eschew translating the rhyme scheme: his final version contained no rhymes, but translated meticulously every word of vocabulary and every structural choice which Pushkin made while writing the poem, as well as containing extensive commentary. It is considered the most 'faithful' translation of the original, and it serves as a useful gloss for those wanting to read the original poem, but not possessing a suitable level of Russian.

With these three options in mind, let's explore the manual and computational approaches one may take to translating what is considered to be one of the most beautiful works in the German language:[32] Goethe's 'Über allen Gipfeln' [ix] Here is the text of the original:

[ix] Also called 'Wanderer's Nightsong II', as it is the second part of Goethe's 'Wandrers Nachtlied' series.

Über allen Gipfeln
Ist Ruh,
In allen Wipfeln
Spürest du
Kaum einen Hauch;
Die Vögelein schweigen im Walde.
Warte nur, balde
Ruhest du auch.

Johann Wolfgang von Goethe (1749–1832)

Firstly, let us consider a lexical translation — that is, a translation which considers only the basic preservation of words and their order. The following gloss was done using an English-German dictionary, with alternative words marked using slashes:

Over all peaks/summits/tops
Is rest/peace/silence,
In all treetops
You (informal) sense/feel
Hardly a breath/breeze;
The little birds remain silent/keep still in the wood/forest.
Just wait, soon
You rest/repose too/also.

TU Chemnitz Dictionary[33]

Doing this by hand only requires a bilingual dictionary and knowledge of the target language: as such, a computational approach would, at most, require access to a good English-German dictionary. This is, then, the simplest

computational approach, requiring no large amounts of mathematics or programming. Note how the rhyme scheme has not been preserved, excepting the duplication of 'tops' in lines 1 and 3,[x] or the half-rhyme of peaks-peace in lines 1 and 2. Looking at preservation of rhythm is pointless, because the translation purposely sacrifices ease of reading for completeness of translation: it would not make sense to read this poem aloud.[xi]

Next, consider what Nabokov would have called the paraphrastic version: John Whaley's translation, which hangs on a plaque in a reproduction of the cabin where Goethe wrote the original poem:

> Over all of the hills
> Peace comes anew,
> The woodland stills
> All through;
> The birds make no sound on the bough.
> Wait a while,
> Soon now
> Peace comes to you.
>
> John Whaley

Let us consider which parts of the original poem this translation preserves, and which parts it changes. The rhyme scheme of the original is *ababcddc*; the translation, on the other hand, is *ababcdcb* — so the rhyme scheme has been preserved to an extent. But to allow for this preservation of rhyme scheme, the poem has been altered in a way that would disgust Nabokov. The first line is translated in the same way. The second line goes from 'is peace' to 'peace comes anew' — a change in tense which is only done to allow

[x] While this is technically a rhyme, a poet using a word as its own rhyme is committing a crime against poetry.

[xi] Unless one is E.E. Cummings, in which case it is fine.

a rhyme two lines later. The rest of the poem has been completely altered: it goes from 'in all treetops, you feel hardly a breath' to 'the woodland stills all though'; the birds move from the forest to the bough; and instead of you finding peace, the peace comes to you.

The original poem is considered one of the most beautiful works in the German language, primarily for two reasons. The first is the contrast Goethe draws between man and nature: man is restless, uncomfortable in the silence of the forest but expecting of death's eternal peace. On the other hand, nature is united in its silence. The scale of the poem is also interesting: it moves from a large scale (the summits), to a middle distance (the treetops), to the immediate surroundings (in the forest), and finally to the person. This sequence can be seen as encompassing the human universe, moving from the largest (visible from a forest) scale, to the smallest, most personal. The translation of the poem loses the large scale (summits turn to hills), although it preserves the progression of scale reduction. But man's restlessness is lost: instead of man having to wait for eternal peace, it becomes peace coming to man.

Instead of looking at this poem as a translation of Goethe's original, it might be better to look at it as a poetic work in its own right. Looking at it from that point of view, it is very much a good poem. There is a good rhythm, a good rhyme scheme, and it reads well: and, while there is not necessarily the same level of conciseness and beauty that Goethe's poem shows, it is still a read to be savored.

It is worth discussing here the work of Douglas Hofstadter in his book 'Le Ton Beau de Marot'.[34] The book covers possible translations of the Clément Marot's 'Ma Mignonne': by exploring various possible translations, the author explores a wide range of topics in linguistics, psychology, and mathematical theory. Hofstadter is perhaps the most vocal supporter (outside of translation circles) of the paraphrastic approach to translation. The

book contains a multitude of possible translations and transformations of the poem, most of which are barely recognizable to be related to the original. They are good reads, just as Whaley's translation is a good read. But they do not stay faithful to the original. While intended to be a proof of the superiority of the paraphrastic approach to translation, what Hofstadter's book really shows us is the unsuitability of paraphrastic translation to preserving the context and meaning of the poem. Instead, it can be seen as proof that paraphrastic translation creates a new, distinct poem.

How would one go about manually taking the literal approach to this poem? I believe the task is beyond the scope of this report. The main issue is that there is a deep understanding of both languages required: Nabokov himself was bilingual in both English and Russian from childhood, and thus had the kind of intimate understanding of both languages and cultures which I simply do not possess. Additionally, there are time constraints: Nabokov's own translation took many years of research,[30] and the time frame for this project was limited. Perhaps one might start with exploring Goethe's background, his other poetry, and so on, and explore the area, working and reworking a translation in much the same way as the poet might have, until a suitably literal translation is reached.

Considering all of the above, what is the best computational approach we could take to translate poetry? Manually, the best approach, in my opinion, is the literal approach: a preservation of the original poem's contextual meaning. A computational literal translation can be considered to be under the umbrella of 'AI-complete' problems. Paraphrastic translation, for all Nabokov disparaged it, is a good computational option: while it does not preserve the original, it does create a new poem which, as seen above, is still worth reading in its own right: it is also easier, computationally, to implement.

To get an idea of what we're working with, here is how Google Translate,

arguably the most popular translation service today, translates the poem.[xii]

> Above all peaks
> Is peace,
> In all tops
> Do you feel
> Hardly a breath;
> The birds are silent in the forest.
> wait, soon
> Are you retiring too?
>
> Google Translate

The first thing to notice about this translation is its somewhat perplexing tendency towards converting statements to questions. 'Spürest du' becomes 'do you feel' — a reasonable mistake to make, because the German language does not use an auxiliary ('do') to indicate a question: it simply reverses the verb and the noun to indicate its presence. However, the following and preceding lines make it clear that this is not an isolated phrase, which could be interpreted as a question, but rather part of a sentence clause. Another interesting thing is the vocabulary choices: 'Gipfeln' is translated to 'peaks', as opposed to the more common translation 'summits'; the word 'nur' is also cut completely, which eliminates the feeling of urgency and restlessness which its use in the original poem implies.

My research has consisted of hypothesizing changes to current (word- and phrase-based) machine translation methods which would allow for more fluent poetic output, compared to the type shown above. I have developed

[xii] It must be noted here that as of 2016, Google Translate uses neural machine translation for English ↔ German translation, the explanation and analysis of which is beyond the scope of this report. I will be focusing only on the types of translation I outlined in the theory section; nonetheless, analyzing its poetic output is an interesting exercise.

three hypotheses:

1. Training the language and alignment models on a poetic corpus improves poetic qualities of the output translation.

2. Altering the sentence alignment model for poetic works will improve line-by-line translation quality.

3. Poetic characteristics can be preserved by constraints on the hypothesis space, or recovered by post-processing of the output translation.

**Poetic corpus** I would argue that statistical translation of poetry faces the same problem as statistical modeling of language: we simply have no way of truly modeling the poet's thought processes while writing the original, and therefore no way of emulating it computationally. So, again, we have to compromise: instead of seeking to emulate the thought process, we instead seek to statistically model poetry in such a way that our translation sounds poetical.

The primary way of having a 'poetic' sounding translation is through corpus choices. Google Translate's language and alignment models for English $\leftrightarrow$ German translation are primarily trained on the Europarl corpus, which is a transcription of all European Parliament proceedings, and is available in all the official languages of the Parliament.[35] Multilingual corpus availability is a large problem in machine translation, and the Europarl corpus is large enough that it can be used as a basis for alignment models when translating the applicable European languages. However, alignment models, like language models, are sensitive to the input corpus (even with smoothing), and will prefer alignments with words which occur often in the corpus. Europarl proceedings are, predictably, very dry and not at all poetic. A non-poetic input implies a non-poetic output. The obvious solution here is to use a parallel corpus of German poems and their English translations. Unfortunately,

no such corpuses exist, and creation is extremely difficult, as very few parallel poetic translations are available in easily-processable digital form. In an ideal world, to translate a given poet's poem, we'd use a corpus of parallel translated poems from the same time period or style. Unfortunately, this is not the ideal world, so some compromise is needed.

Using a poetic language model is a much more reasonable and achievable solution. A suitable language model can be built, I believe, simply by collecting a large amount of poetry in the target language — in this case, English, and using this to train the language model. As part of this project, I scraped the top 500 English language works from Project Gutenberg tagged under 'poetry' and collated them into a corpus, which could possibly be used to train a language model with a toolkit that allows it. However, this method does raise some problems.

Kao (2011)[36] showed that there are certain characteristics which occur specifically in professionally-written poems. The primary features, for modern poetry, are firstly the reduced presence of rhyme and alliteration, secondly the use of concrete nouns, and thirdly the use of less common words. Preservation of these features is well-paired with phrase-based machine translation. The lack of rhyme and alliteration means we do not need to find ways to parse for them, which is a difficult task due to variations in pronunciation. The increased use of concrete nouns can be applied by adding an extra weighing to these tokens while training our language model. The use of less common words is more difficult, as language models, by design, favor more common words. A possible solution would be performing extra smoothing on the corpus which, for example, greatly increases the probability of less common words and decreases the probability of more common words. The problem with this, of course, is that less common words will begin to be used where a poet may not necessarily use them: for example, substituting 'to have' with 'to possess' or 'to retain'.

**Alignment model** Whereas sentences in prose are clearly defined, poetry does not have strong sentence distinctions. In some cases, a line break may indicate the end of a sentence: in others, a line break may indicate the end of a clause, or even just nothing. Compare the following extracts of poems:

> one's not half two. It's two are halves of one:
> which halves reintegrating,shall occur
> no death and any quantity;but than
> all numerable mosts the actual more
>
> ee cummings[37]

> Sunlight pouring across your skin, your shadow
> flat on the wall.
> The dawn was breaking the bones of your heart like twigs.
> You had not expected this,
> the bedroom gone white, the astronomical light
> pummeling you in a stream of fists.
>
> Richard Siken[38]

In the first poem, line breaks have no meaning: aside from adding a small pause when the poem is read out loud, they can completely be disregarded. There are no traditional sentence boundaries. This is an issue with phrase-based translation, as it relies on the existence of aligned sentences, and to have aligned sentences, one needs clearly defined sentence boundaries. An option for this would be to define sentence boundaries as new lines, rather than standard punctuation (. ! ?). However, in this case, it would lead to important phrases, such as 'than all' being split. In theory, this shouldn't be an issue — as already mentioned, phrase-based translation actually performs better when non-syntactically correct phrase alignments are allowed — how-

ever, with translation of meaning in poetry being so context-dependent (see the Google Translate example), it could be severely detrimental to translation quality.

The second poem is one in which using line-endings for separating sentence units would not work: in this case, parsing for traditional sentence endings will, in fact, work, because the line breaks have only been introduced due to the stylistic preferences of the author. The most striking part of this poem is actually the layout on the page, which I have tried my best to reproduce from the original. A translation would need to reproduce this indentation as well, but this could be done quite easily by parsing for whitespace at the start of a line in the original poem and using the same amount of whitespace in the translation.[xiii]

A possible solution to the sentence alignment issue would be to split poems into two types:

1. Poems whose sentences can be treated in the same way as prose, where sentences can be assumed to be marked by standard sentence boundaries.

2. Poems whose sentence boundaries are marked by line-endings, rather than punctuation.

The first feature is already used in common sentence alignment tools: the second feature could be implemented relatively easily by tagging new sentences on line breaks.

**Preserving poetic qualities** Now that we have dealt with the first two problems, we now face the third: how do we preserve poetic characteristics

[xiii]This would only work for systems where features like whitespace are not stripped before being input.

while translating? State-of-the-art systems, as mentioned already, do not put any weight on preserving aspects of poetry such as rhythm or rhyme. There are two possible ways: either hypothesis constraints, or post-processing.

Genzel, Uszkoreit, and Och (2010)[39] use hypothesis constraints to preserve poetic features such as meter, line length (syllable and word), or rhyme. By treating poetic features as constraints on the poem — for example, a haiku has a constraint on line length and syllable count — they implement a generic poetic form feature function for phrase-based statistical machine translation systems. This is a feasible solution to the problem of preserving certain poetic features. However, there is an issue in the output: while it preserves poetic features, it does not preserve fluency, which is of high concern when it comes to translating poetry. Here is an example from the paper of a baseline translation, and a 'couplet in amphibrachic tetrameter' (stressed syllables italicized):

> A policeman said that three people were arrested and that the material is currently being analyzed.
>
> An *of*ficer *sta*ted that *three* were ar*re*sted
> and *that* the e*quip*ment is *cur*rently *test*ed.

Another solution, instead of searching inside the hypothesis space, would be to use post-processing. In many uses of machine translation, the output already undergoes post-processing by experienced translators or native speakers to fine-tune the output. We could, in much the same way, recover poetic features which were 'lost in translation' by computational post-processing. For example, if we wanted to preserve the rhyme scheme of the original poem, it could be enough to look for the rhyming synonyms of words at the end of lines, and replace those words with their synonyms until the lines rhyme in the same way as the original. This is a very naïve approach, but I believe it

could work, given a good enough synonym dictionary. There could also be a distortion model, in much the same way as phrase-based translation systems, to allow for word reordering. This could essentially be seen as implementing a second machine translation system, but instead of translating from a foreign language to English, we translate from non-poetic to poetic English.

What I have assumed while discussing the two previous methods is that we already know what poetic features a poem has. While it is relatively easy for a trained human to identify and annotate such features, it is a non-trivial task for computers to do perfectly. Let us briefly consider the two main qualities we want to preserve: rhythm and rhyme.

Byrd and Chodorow (1985)[40] present a method for finding the stress patterns and rhyme-endings of unknown words from a pre-existing dictionary. For a given word, three things are looked at: the words surrounding it alphabetically, the words which are likely to rhyme with it, and words with similar word-end spelling. This method, however, relies on a good-quality, up-to-date dictionary for the target and source languages. For less common languages, such a dictionary is very unlikely to exist. Additionally, slant rhymes and changes in pronunciation mean that whether two words rhyme is not necessarily a yes-or-no question.

Reddy and Knight (2014)[41] continue the work of Greene et al. (2010)[42] to describe a method of unsupervised discovery of rhyme schemes which is very promising. Machine learning methods are used to infer the presence of rhyme schemes from corpuses of poetry. The major benefit of this is that it is language-independent: so long as one has some sort of corpus of rhyming poetry in a given language, it is feasible that some rhyme schemes are able to be automatically learned by the algorithm.

Luckily, the problem of identifying word stress is more easily solved. Due to advances in the field of speech synthesis,[43,44] as well as previous research into

stress-timing and syllable-timing,[45] finding word stress is relatively trivial.

# Conclusions and future work

*"Poetry is a sword of lightning, ever unsheathed,*
*which consumes the scabbard that would contain it."*
*Percy Bysshe Shelley*

This paper has discussed translation of poetry in a computational, statistically-based way. A discussion of the history of machine translation, the theory and practice of modern machine translation systems, and current and possible approachs to manual and computational translation, have all allowed me to explore the possibility of reaching high quality, fully automatic machine translation of poetry.

The hypotheses I have discussed, of how current machine translation methods may be improved to create better translations of poetry, all lead nicely into future work. Each of the methods I have proposed — training the language model with a custom-built corpus, altering the alignment model, and recovering poetic characteristics — can easily be incorporated into existing machine translation systems such as Moses.[46] The next thing to do with this research would be to actually implement these methods, and compare the results of the new system with existing systems. One could also look at how other state of the art systems (for example neural machine translation systems) could be integrated with these hypotheses.

In conclusion: translating poetry is hard. Both manually and computationally, there are many pitfalls which must be avoided to ensure a translation is faithful to the original and fluently written. I hope that with this report, the rarely-explored field of computational translation of poetry may find some new ideas. Perhaps in the future we may be able to update the quote which

the introduction started with:

> Poetry is that which is lost in translation,
> Unless we use a computational calculation.

# References


[1] E. Weinberger & O. Paz. *Nineteen Ways of Looking at Wang Wei*. 1987.

[2] E. Nida. *Principles of Correspondence* in *Translation Studies Reader*. 2007.

[3] *What is Computational Linguistics?* `http://www.aclweb.org/archive/misc/what.html`

[4] W. Weaver. *Translation.* `http://www.mt-archive.info/Weaver-1949.pdf`

[5] J. Hutchins. *Milestones in Machine Translation no.2* `http://www.hutchinsweb.me.uk/Milestones-2.pdf`. 1998.

[6] W. S. McCulloch & W. Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity* in *Bulletin of Mathematical Biophysics*. 1943.

[7] J. Hutchins. *Retrospect and prospect in computer-based translation*. 1999.

[8] L. E. Dostert. *The Georgetown-IBM experiment*. 1955.

[9] *Language and Machines: Computers in Translation and Linguistics* `http://www.mt-archive.info/ALPAC-1966.pdf`. 1966.

[10] J. Hutchins. *ALPAC - the (in)famous report* `http://www.hutchinsweb.me.uk/MTNI-14-1996.pdf`. 1996.

[11] T. Winograd. *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language* in *MIT AI Technical Report 235*. 1971.

[12] L. Liddy et al. *Natural Language Processing*. http://www-nlpir.nist.gov/MINDS/FINAL/NLP.web.pdf

[13] P. F. Brown et al. *A Statistical Approach to Machine Translation*. Computational Linguistics. 1990.

[14] Y. Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016.

[15] P. I. Nakov & H. T. Ng. *Improving Statistical Machine Translation for a Resource-Poor Language Using Related Resource-Rich Languages*. 2014.

[16] E. Avramidis & P. Koehn. *Enriching Morphologically Poor Languages for Statistical Machine Translation*. 2008.

[17] J. Dryden. *The Three Types of Translation* in *The Translation Studies Reader*. 2007. p. 53–54

[18] C. Rumens. *Translating poetry opens up new worlds of language*. https://www.theguardian.com/books/booksblog/2007/sep/28/translatingpoetryopensupne

[19] R. Bacon. *Summa Grammatica (Overview of Grammar)*. 1245.

[20] N. Chomsky. *Aspects of the Theory of Syntax*. 1965.

[21] D. Jurafsky & J. H. Martin. *Speech and Language Processing*. 2009.

[22] P. Koehn. *Statistical Machine Translation*. 2010.

[23] T. Bayes & R. Price. *An Essay towards solving a Problem in the Doctrine of Chance*. 1763. Also independently discovered by P. S. Laplace. *Mémoire sur la probabilité des causes par les événements*. 1774.

[24] A. A. Markov. *Theory of Algorithms*. 1954.

[25] W. A. Gale & K. W. Church. *A Program for Aligning Sentences in Bilingual Corpora*. 1993.

[26] Brown et al. *The Mathematics of Statistical Machine Translation*. 1993.

[27] P. Koehn et al. *Statistical Phrase-Based Translation*. 2002.

[28] G. K. Zipf. *The psycho-biology of language*. 1935.

[29] D. Marcu & W. Wong. *A phrase-based & joint probability model for statistical machine translation*. 2002.

[30] V. Nabokov. *Eugene Onegin: A Novel in Verse by Aleksandr Pushkin. Translated from the Russian*. 1964.

[31] *The Theory and Practice of Poetic Translation in Pushkin and Nabokov*

[32] Alan P. Cottrell. *Goethe's view of evil and the search for a new image of man in our time*. 1982.

[33] Technische Universität Chemnitz. *Beolingus*. `http://dict.tu-chemnitz.de/`

[34] D. Hofstadter. *Le Ton Beau de Marot: In Praise of the Beauty of Language*. 1997.

[35] P. Koehn. *Europarl: A Parallel Corpus for Statistical Machine Translation*. 2005.

[36] J. Kao. *A Computational Analysis of Poetic Craft in Contemporary Professional and Amateur Poetry*. 2011.

[37] ee cummings. *Selected Poems*. 2007.

[38] R. Siken. *Visible World* in *Crush*. 2005.

[39] D. Genzel & J. Uszkoreit & F. Och. *"Poetic" Statistical Machine Translation: Rhyme and Meter*. 2010.

[40] R. J. Byrd & M. S. Chodorow. *Using an on-line dictionary to find rhyming words and pronunciations for unknown words*. 1985.

[41] S. Reddy & K. Knight. *Unsupervised Discovery of Rhyme Schemes*. 2014.

[42] E. Greene et al. *Automatic analysis of rhythmic poetry with applications to generation and translation*. 2010.

[43] B. Williams. *Word stress assignment in a text-to-speech synthesis system for British English*. 1987.

[44] M. Y. Liberman & K. W. Church. *Text Analysis and Word Pronunciation in Text-to-speech Synthesis*. 1991.

[45] R. M. Dauer. *Stress-timing and syllable-timing reanalyzed*. 1983.

[46] P. Koehn et al. *Moses: Open Source Toolkit for Statistical Machine Translation*. 2007.