Humans Outperform Machines at the Bilingual Shannon Game

Marjan Ghazvininejad *† and Kevin Knight †

Information Sciences Institute, University of Southern California, 4676 Admiralty Way #1001, Marina Del Rey, CA 90292, USA; knight@isi.edu
* Correspondence: ghazvini@isi.edu; Tel.: +1-310-822-1511
† Both authors contributed equally to this work.

Academic Editors: Kevin H. Knuth and Raúl Alcaraz Martínez
Received: 3 October 2016; Accepted: 27 December 2016; Published: 30 December 2016

Abstract: We provide an upper bound for the amount of information a human translator adds to an original text, i.e., how many bits of information we need to store a translation, given the original. We do this by creating a Bilingual Shannon Game that elicits character guesses from human subjects, then developing models to estimate the entropy of those guess sequences.

Keywords: compression; multilingual; translation

1. Introduction

Zoph et al. [1] ask the question “How much information does a human translator add to the original? ” That is, once a source text has been compressed, how many additional bits are required to encode its human translation? If translation were a deterministic process, the answer would be close to zero. However, in reality, we observe an amount of free variation in target texts. We might guess, therefore, that human translators add something like 10% or 20% extra information, as they work.

To get an upper bound on this figure, Zoph et al. [1] devise and implement an algorithm to actually compress target English in the presence of source Spanish. The size of their compressed English is 68% of the size of compressed Spanish. This bound seems rather generous. In setting up a common task, Zoph et al. [1] encourage researchers to develop improved bilingual compression technology [2].

In this paper, we investigate how good such algorithms might get. We do not do this by building better compression algorithms, but by seeing how well human beings can predict the behavior of human translators. Because human beings can predict fairly well, we may imagine that bilingual compression algorithms may one day do as well.

Shannon [3] explores this exactly question for the simpler case of estimating the entropy of free text (not translation). If a human subject were able to write a probability distribution for each subsequent character in a text (given prior context), these distributions could be converted directly into entropy. However, it is hard to get these from human subjects. Shannon instead asks a subject to simply guess the next character until she gets it right, and he records how many guesses are needed to correctly identify it. The character sequence thus becomes a guess sequence, e.g.:

T h e _ b r o k e n _ v
2 1 1 1 1 3 2 5 1 1 1 1 5

The subject’s identical twin would be able to reconstruct the original text from the guess sequence, so in that sense, it contains the same amount of information.

Let $c_1, c_2, \ldots, c_n$ represent the character sequence, let $g_1, g_2, \ldots, g_n$ represent the guess sequence, and let $j$ range over guess numbers from 1 to 95, the number of printable English characters plus newline. Shannon [3] provides two results.
(Upper Bound). The entropy of $c_1, c_2, \ldots, c_n$ is no greater than the unigram entropy of the guess sequence:

$$-\frac{1}{n} \log(\prod_{i=1}^{n} P(g_i)) = -\frac{1}{n} \sum_{i=1}^{n} \log(P(g_i)) = -\sum_{j=1}^{95} P(j) \log(P(j))$$

This is because this unigram entropy is an upper bound on the entropy of $g_1, g_2, \ldots, g_n$, which equals the entropy of $c_1, c_2, \ldots, c_n$. In human experiments, Shannon obtains an upper bound of 1.3 bits per character (bpc) for English, significantly better than the character n-gram models of his time (e.g., 3.3 bpc for trigram).

(Lower Bound). The entropy of $c_1, c_2, \ldots, c_n$ is no less than:

$$\sum_{j=1}^{95} j \cdot [P(j) - P(j+1)] \cdot \log(j)$$

with the proof given in his paper. Shannon reported a lower bound of 0.6 bpc.

1.1. Contributions of This Paper

Table 1 gives the context for our work, drawing prior numbers from Zoph et al. [1]. By introducing results from a Bilingual Shannon Game, we show that there is significant room for improving bilingual compression algorithms, meaning there is significant unexploited redundancy in translated texts. Our contributions are:

1. A web-based bilingual Shannon Game tool.
2. A collection of guess sequences from human subjects, in both monolingual and bilingual conditions.
4. An upper bound on the amount of information in human translations. For English given Spanish, we obtain an upper bound of 0.48 bpc, which is tighter than Shannon’s method, and significantly better than the current best bilingual compression algorithm (0.89 bpc).

<table>
<thead>
<tr>
<th></th>
<th>Monolingual</th>
<th>Bilingual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine</td>
<td>1.39</td>
<td>0.89</td>
</tr>
<tr>
<td>Human</td>
<td>1.25</td>
<td>0.42 (this paper)</td>
</tr>
</tbody>
</table>

Related Work

Compression has attracted research attention for a long time, e.g., [4–11]. The Hutter Prize [12], a competition to compress a 100 m-word extract of English Wikipedia, was designed to further encourage research in text compression. Bilingual and multilingual text compression is a less-studied field [1,13–18]. These papers provide different algorithms for compressing text in multilingual format, but they do not demonstrate how humans perform on this task.

Shannon [3] devised an experimental method to estimate the entropy of written English. After that, many other papers used Shannon’s method to calculate the entropy of English on different passages and context lengths [19–22]. Other papers use Shannon’s technique to measure the entropy of other languages [23–27]. Shannon’s method was modified by Cover and King [28] who asked their subjects to gamble on the next character.

Nevill and Bell [29] describe a parallel-text Shannon Game, but they work in an English-English paraphrasing scenario, with different versions of the Bible. Zoph et al. [1] briefly mention a Shannon Game experiment in which human subjects guessed subsequent characters in a human translation.
They report a Shannon upper bound for English-given-Spanish guess sequences as 0.51 bpc, but they do not give details, and they do not appear to separate testing sequences from training.

We note that different text genres studied in the literature yield different results, as some genres are more predictable than others. Different alphabet sizes (e.g., 26 letters versus 95 characters) have a similar effect. Our interest is not to discover an entropy figure that holds across all genres, but rather to study the entropy gap between humans and machines, and between monolingual and bilingual settings. For this purpose, we use the state of the art Monolingual text compressor Prediction by partial matching, Variant C (PPMC) [6], and the state of the art Bilingual text compressor presented in [1].

2. Materials and Methods

2.1. Shannon Game Data Collection

Figure 1 shows our bilingual Shannon Game interface. It displays the current (and previous) source sentence, an automatic Google translation (for assistance only), and the target sentence as guessed so far by the subject. The tool also suggests (for further assistance) word completions in the right panel. Our monolingual Shannon Game is the same, but with source sentences suppressed.

![Predict the next character:](image)

**Figure 1.** Bilingual Shannon Game interface. The human subject reads the Spanish source and guesses the translation, character by character. Additional aids include a static machine translation and a dynamic word completion list.

To gather data, we asked 3 English-speaking subjects plus a team of 4 bilingual people to play the bilingual Shannon game. For each subject/team, we assigned a distinct 3–5 sentence text from the Spanish/English Europarl corpus v7 [30] and asked them to guess the English characters of the text one by one. We gathered a guess sequence with 684 guesses from our team and a guess sequence with 1694 guesses from our individuals (2378 guesses in total). We also asked 3 individuals and a team of
3 people to play the monolingual Shannon game. We gathered a guess sequence with 514 guesses from our team and a guess sequence with 1769 guesses from our individuals (2283 guesses in total).

Figure 2 shows examples of running the monolingual and bilingual Shannon Game on the same sentence.

Monolingual Shannon Game (no source sentence)

I t _ i s _ d e f e n d e d _ t h r o u g h _ r e a s o n i n g .
D w h i m e i f a ,
m t a s _ c o n
co n
n c i f
m o d
p t

Bilingual Shannon Game (source sentence = "Se defiende con argumentos.")

I t _ i s _ d e f e n d e d _ t h r o u g h _ r e a s o n i n g .

Figure 2. Example guess data collected from the Shannon Game, in both monolingual (top) and bilingual (bottom) conditions. The human subject’s guesses are shown from bottom up. For example, in the bilingual condition, after seeing ‘...reason’, the subject guessed ‘.’ (wrong), but then correctly guessed ‘i’ (right).

2.2. An Estimation Problem

Our overall task is now to estimate the (per-guess) entropy of the guess sequences we collect from human subjects, to bound the entropy of the translator’s text. To accomplish this, we build an actual predictor for guess sequences. Shannon [3] and Zoph et al. [1] both use a unigram distribution over the guess numbers (in our case 1 to 95). However, we are free to use more context to obtain a tighter bound.

For example, we may collect 2-gram or 3-gram distributions over our observed guess sequence and use those to estimate entropy. In this case, it becomes important to divide our guess sequences into training and test portions—otherwise, a 10-gram model would be able to memorize large chunks of the guess sequences and deliver an unreasonably low entropy. Shannon [3] applies some ad hoc smoothing to his guess counts before computing unigram entropy, but he does not split his data into test and train to assess the merits of that smoothing.

We set aside 1000 human guesses for testing and use the rest for training—1378 in the bilingual case, and 1283 in the monolingual case. We are now faced with how to do effective modeling with limited training data. However, before we turn to that problem, let us first work in a less limited playground, that of machine guess sequences rather human ones. This gives us more data to work with, and furthermore, because we know the machine’s actual compression rate, we can measure how tight our upper bound is.

2.3. Machine Plays the Monolingual Shannon Game

In this section, we force the state-of-the art text compressor PPMC [6] to play the monolingual Shannon Game. PPMC builds a context-dependent probability distribution over the 95 possible character types. We describe the PPMC estimator in detail in Appendix A. For the Shannon Game,
we sort PPMC’s distribution by probability, and continue to guess from the top down until we correctly identify the current character.

We let PPMC warm up on 50 m characters, then collect its guesses on the next 100 m characters (for training data), plus an additional 1000 characters (our test data). For the text corresponding to this test data, PPMC’s actual compression rate is 1.37 bpc.

The simplest model of the training guess sequence is a unigram model. Table 2 shows the unigram distribution over 100 m characters of training, for both machine and human guess data (These numbers combine data collected from individuals and from teams. In the bilingual case, teams outperformed individuals, guessing correctly on the first try 94.3% of the time, versus 90.5% for individuals. In the monolingual case, individuals and teams performed equally well).

Table 2. Unigram probabilities of machine and human guesses, in both monolingual and bilingual conditions. Amounts of training data (in characters) are shown in parentheses.

<table>
<thead>
<tr>
<th>Guess #</th>
<th>Monolingual</th>
<th></th>
<th></th>
<th>Bilingual</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Machine (100 m)</td>
<td>Human (1283)</td>
<td>Machine (100 m)</td>
<td>Human (1378)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.732</td>
<td>0.744</td>
<td>0.842</td>
<td>0.916</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.105</td>
<td>0.086</td>
<td>0.074</td>
<td>0.035</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.047</td>
<td>0.047</td>
<td>0.024</td>
<td>0.013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.027</td>
<td>0.030</td>
<td>0.014</td>
<td>0.011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.017</td>
<td>0.020</td>
<td>0.009</td>
<td>0.005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.012</td>
<td>0.012</td>
<td>0.007</td>
<td>0.004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.009</td>
<td>0.008</td>
<td>0.005</td>
<td>0.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.007</td>
<td>0.007</td>
<td>0.004</td>
<td>0.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.006</td>
<td>0.005</td>
<td>0.003</td>
<td>0.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.005</td>
<td>0.004</td>
<td>0.003</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>7.05 x 10^{-8}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>7.69 x 10^{-8}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>1.09 x 10^{-7}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We consider two types of context—the g guess numbers preceding the current guess, and the c characters preceding the current guess. For example, if c = 3 and g = 2, we estimate the probability of the next guess number from previous 3 characters and previous 2 guess numbers. In:

\[
\text{T h e - c h a p}
\]

we calculate \( P(?) \mid \text{character context } = c, h, a; \text{guess context } = 2, 4 \).

The context gives us more accurate estimates. For example, if \( c = 1 \) and the previous character is ‘q’, then we find the machine able to correctly guess the next character on its first try with probability 0.981, versus 0.732 if we ignore that context. Likewise, having \( g \) previous guesses allows us to model “streaks” on the part of the Shannon Game player.

As \( g \) and \( c \) grow, it becomes necessary to smooth, as test guess sequences begin to contain novel contexts. PPMC itself makes character predictions using \( c = 8 \) and \( g = 0 \), and it smooths with Witten-Bell, backing off to shorter n-gram contexts \( c = 1...7 \). We also use Witten-Bell, but with a more complex backoff scheme to accommodate the two context streams \( g \) and \( c \). If \( g \geq c \) we back off to the model with \( g - 1 \) previous guesses and \( c \) previous characters, and if \( g < c \) we back off to the model with \( g \) previous guesses and \( c - 1 \) previous characters.

Table 3 shows test-set entropies obtained from differing amounts of training data, and differing amounts of context. We draw several conclusions from this data:
• Character context \((c)\) is generally more valuable than guess context \((g)\).
• With large amounts of training data, modest context \((g = 1, c = 2)\) allows us to develop a fairly
tight upper bound (1.44 bpc) on PPMC’s actual compression rate (1.37 bpc).
• With small amounts of training data, Witten-Bell does not make effective use of context. In fact,
adding more context can result in worse test-set entropy!

The last column of Table 3 shows entropies for necessarily-limited human guess data, computed
with the same methods used for machine guess data. We see that human guessing is only a bit more
predictable than PPMC’s. Indeed, PPMC’s guesses are fairly good—its massive 8-gram database is
a powerful counter to human knowledge of grammar and meaning.

**Table 3.** Entropies of monolingual test guess-sequences (1000 guesses), given varying amounts
of context \((c = \text{number of previous characters}, \ g = \text{number of previous guess numbers})\) and different
training set size (shown in parentheses). Witten-Bell smoothing is used for backoff to shorter contexts.
The best number in each column appears in bold.

<table>
<thead>
<tr>
<th>(c)</th>
<th>(g)</th>
<th>Machine Guessing</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(100 m)</td>
<td>(10 m)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.72</td>
<td>1.72</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1.70</td>
<td>1.70</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1.69</td>
<td>1.69</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1.54</td>
<td>1.54</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.52</td>
<td>1.52</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1.50</td>
<td>1.52</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1.45</td>
<td>1.51</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.44</td>
<td>1.53</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.48</td>
<td>1.56</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1.37</td>
<td></td>
</tr>
</tbody>
</table>

2.4. Modeling Human Guess Sequences

How can we make better use of limited training data? Clearly, we do not observe enough instances
of a particular context to robustly estimate the probabilities of the 95 possible guess numbers that may
follow. Rather than estimating the multinomial directly, we instead opt for a parametric distribution.
Our first choice is the geometric distribution, with one free parameter \(p\), the chance of a successful guess
at any point. For each context in the training data, we fit \(p\) to best explain the observations of which
guesses follow. This one parameter can be estimated more robustly than the 94 free parameters of
a multinomial.

Figure 3 shows that the geometric distribution is a decent fit for our observed guess data, but it
does not model the head of the distribution well—the probability of a correct guess on the first try is
consistently greater than \(p\).

Therefore, we introduce a smoothing method (“Frequency and Geometric Smoothing”) that only
applies geometric modeling to guess numbers greater than \(i\), where data is sparse. For each context,
we choose \(i\) such that we have seen all guess numbers 1..\(i\) at least \(k\) times each, where

\[
k = \min\left(\frac{\text{Number of samples seen in context}}{20}, 4\right)
\]

Table 4 (left half) demonstrates the effect of different smoothing methods on estimated entropies
for human guess data. The monolingual Witten-Bell smoothing column in this figure is the same of
last column of Table 3.

The right half of Table 4 shows the bilingual case. For the machine case, we use the algorithm
of Zoph et al. [1]. Note that the machine and human subjects both make use of source-sentence
context when predicting. However, we do not use source context when modeling guess sequences,
only target context.
Figure 3. Guess number distributions from human monolingual Shannon Game experiments (training portion). Plot (a) shows all 1238 guesses, while plots (b–d) show guesses made in specific character contexts ‘ ’ (space), ‘a’ and ‘p’. The y-axis (probability of guess number) is given in log scale, so a geometric distribution is represented by a straight line. We observe that the single-parameter geometric distribution is a good fit for either the head or the tail of the curve, but not both.

Table 4. Entropies of human guess-sequences (1000 test-set guesses), given varying amounts of context (c = number of previous characters, g = number of previous guess numbers) and different smoothing methods. Prediction models are trained on a separate sequence of 1283 guesses in the monolingual case, and 1378 guesses in the bilingual case. The best entropy of Monolingual/Bilingual human guessing appears in bold.

<table>
<thead>
<tr>
<th>c</th>
<th>g</th>
<th>Witten-Bell Smoothing</th>
<th>Geometric Smoothing</th>
<th>Frequency and Geometric Smoothing</th>
<th>Witten-Bell Smoothing</th>
<th>Geometric Smoothing</th>
<th>Frequency and Geometric Smoothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.68</td>
<td>2.02</td>
<td>1.62</td>
<td>0.54</td>
<td>0.73</td>
<td>0.67</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1.75</td>
<td>2.06</td>
<td>1.62</td>
<td>0.56</td>
<td>0.72</td>
<td>0.66</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1.92</td>
<td>2.06</td>
<td>1.65</td>
<td>0.61</td>
<td>0.72</td>
<td>0.67</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1.74</td>
<td>1.57</td>
<td>1.50</td>
<td>0.65</td>
<td>0.57</td>
<td>0.48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2.18</td>
<td>1.55</td>
<td>1.48</td>
<td>0.84</td>
<td>0.56</td>
<td>0.48</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2.32</td>
<td>1.52</td>
<td>1.49</td>
<td>0.93</td>
<td>0.56</td>
<td>0.49</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2.20</td>
<td>1.65</td>
<td>1.60</td>
<td>0.94</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2.58</td>
<td>1.57</td>
<td>1.57</td>
<td>1.10</td>
<td>0.63</td>
<td>0.62</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.70</td>
<td>1.59</td>
<td>1.58</td>
<td>1.18</td>
<td>0.63</td>
<td>0.63</td>
</tr>
</tbody>
</table>

3. Results

For calculating final entropy bounds, we first divide our guess sequence into 1000 for training data, 100 for development, and remainder for test (1183 for the monolingual case and 1278 for the bilingual case). We use the development set to find the best context model and smoothing model. In all
experiments, using $c = 1$ previous characters, $g = 1$ previous guesses, and Frequency and Geometric Smoothing works best.

Table 5 summarizes our results. As shown in the figure, we also computed Shannon lower bounds (see Section 1) on all our guess sequences.

For the bilingual case of English-given-Spanish, we give a 0.48 bpc upper bound and a 0.21 bpc lower bound. In the case of machine predictors, we find that our upper bound is loose by about 13%, making it reasonable to guess that true translation entropy might be near 0.42 bpc.

Table 5. Summary of our entropy bounds.

<table>
<thead>
<tr>
<th>Guesser</th>
<th>Shannon Upper Bound</th>
<th>Our Improved Upper Bound</th>
<th>Compression Rate</th>
<th>Shannon Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolingual</td>
<td>Machine</td>
<td>1.76</td>
<td>1.63</td>
<td>1.39</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>1.65</td>
<td>1.47</td>
<td>~1.25</td>
</tr>
<tr>
<td>Bilingual</td>
<td>Machine</td>
<td>1.28</td>
<td>1.01</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>0.54</td>
<td>0.48</td>
<td>~0.42</td>
</tr>
</tbody>
</table>

4. Information Loss

So far, we estimate how much information a human translator adds to the source text when they translate. We use $H(E|S)$ to represent the conditional entropy of an English text $E$ given Spanish text $S$, i.e., how many bits are required to reconstruct $E$ from $S$. A related question is how much information from the original text is lost in the process of translation. In other words, how much of the precise wording of $S$ is no longer obvious when we only have the translation $E$? We measure the number of bits needed to reconstruct the $S$ from $E$, denoted $H(S|E)$. We could estimate $H(S|E)$ by running another (reversed) bilingual Shannon game in which subjects predict Spanish from English. However, fortunately we can skip this time-consuming process and calculate $H(S|E)$ based on the definition of joint entropy [31]:

$$H(E|S) + H(S) = H(S|E) + H(E)$$

where $H(E)$ and $H(S)$ are the monolingual entropies of $E$ and $S$.

We can estimate $H(S)$ using the monolingual Spanish Shannon game like what we did for estimating $H(E)$. However, as we show in this paper, PPMC compression is close to what we get from the monolingual human Shannon game (1.39 vs. 1.25). So we can estimate $H(S) \simeq 1.26$, using PPMC on Spanish Europarl data, as reported by [1]. Using this estimate, we obtain the amount of information lost in translation as $1.26 + 0.42 - 1.39 = 0.29$.

We see that in the case Spanish and English, the translation process both adds and subtracts information. Other translation scenarios are asymmetric. For example, when translating the word “uncle” into Persian, we must add information (maternal or paternal uncle), but we do not lose information, as “uncle” can be reconstructed perfectly from the Persian word.

5. Conclusions

We have presented new bounds for the amount of information contained in a translation, relative to the original text. We conclude:

- Bilingual compression algorithms have plenty of room to improve. There is substantial distance between the 0.95 bpc obtained by [1] and our upper bound of 0.48 and lower bound of 0.21.
- Zoph et al. [1] estimate that a translator adds 68% more information on top an original text. This is because their English-given-Spanish bilingual compressor produces a text that is 68% as big as that produced by a monolingual Spanish compressor. Using monolingual and bilingual Shannon Game results, we obtain a revised estimate of $0.42/1.25 = 34\%$ (Here, the denominator
is monolingual English entropy, rather than Spanish, but we assume these are close under human-level compression).

Meanwhile, it should be possible to reduce our 0.48 upper bound by better modeling of guess sequence data, and by use of source-language context. We also conjecture that the bilingual Shannon Game can be used for machine translation evaluation, on the theory that good human translators exhibit more predictable behavior than bad machine translators.

Acknowledgments: This work was supported by ARO grant W911NF-10-1-0533.

Author Contributions: Marjan Ghazvininejad and Kevin Knight contributed equally to the conception, experiments, and write up. Both authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. PPMC Compression

This description is based on our previous work [1].

Prediction by partial matching (PPM) is the most well-known adaptive, predictive compression technique [5]. It predicts by producing a complete probability distribution for the next character \( P(X|\text{context}) \), based on the previous \( n - 1 \) characters. It adaptively constructs empirical character \( n \)-gram tables (usually \( n = 1...5 \)) as it compresses. In a given context, a \( n \)-gram table may predict only a subset of characters, so PPM reserves some probability mass for an escape (ESC), after which it executes a hard backoff to the \((n-1)\)-gram table. PPM models are different in assigning probabilities to ESC.

In PPMA, \( P(ESC) = 1/(1+D) \), where \( D \) is the number of times the context has been seen. PPMB uses \( q/D \), where \( q \) is the number of distinct character types seen in the context. PPMC uses \( q/(q+D) \), also known as Witten-Bell smoothing. PPMD uses \( q/2D \).

For compression, after the model calculates the probability of the next character given the context, it sends it to the arithmetic coder [4,6]. Figure A1 sketches the technique. We produce context-dependent probability intervals, and each time we observe a character, we move to its interval. Our working interval becomes smaller and smaller, but the better our predictions, the wider it stays. A document’s compression is the shortest bit string that fits inside the final interval. In practice, we do the bit-coding as we navigate probability intervals. In this paper, when we force the machine to play the Shannon game, we are only interested in the probability distribution of the next character, so we skip the arithmetic coding of the probabilities.

Figure A1. Arithmetic coding.
References

10. Witten, I.; Moffat, A.; Bell, T. Managing Gigabytes: Compressing and Indexing Documents And Images; Morgan Kaufmann: San Francisco, CA, USA, 1999.

© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).