Modifying Existing Analogy-based N-gram Language Model

Meng Tian & Yves Lepage
IPS, Waseda University, Kitakyushu, Japan
tianmeng0112@asagi.waseda.jp & yves.lepage@aoni.waseda.jp

Abstract

By investigating the occurrence of different proportional analogies in corpora, this paper describes an approach to increase the performance of existing analogy-based N-gram language models evaluated by perplexity. Our approach consists in using analogy to reconstruct N-grams from the test data so as to give higher probabilities to these N-grams. By giving different weights to different patterns, we also except that some N-grams which can be reconstructed by different patterns will get more accurate probabilities. The use of suffix arrays for data searching leads to a lesser computation time on text scoring tasks.

1 Introduction

A language model assigns a score to a sentence. This score evaluates the naturalness of the sentence, i.e. whether it is natural or not or how often it is used in real texts. For example, the sentence “I like this book” is a sentence which may occor many times in a text, but the sequence of words “like this book I” may never occur. For a language model, “I like this book” is more natural than “like this book I”, and the former will get a higher score than the latter.

Language modeling is used in many natural language processing applications such as speech recognition, part-of-speech tagging, parsing, information retrieval and machine translation. The most important part in language modeling is smoothing. Smoothing is used to assign non-zero probabilities to a unseen N-grams [1]. In this paper, we inquire the use of proportional analogy to smooth language models. The basic idea is as follows. If we use the proportional analogy pattern ab : ac :: db : dc, with the test data: ’he has’, training data: ’she is’, ’she has’, ’he is’, then we can say that ’he has’ can be reconstructed using the above proportional analogy pattern. ’he has’ should be considered safer by the language model. The goal of this paper is to modify previously proposed analogy-based N-gram language models [8] and try to improve their performance. Our contribution is manifold: firstly, previously proposed N-gram analogy language models relied on 2 parameters. In previous experiments their parameters were set to some values, that need to be improved. We modify the formula used to assign probabilities to different N-grams. We also select the best parameters by the brute-force technique automatically and get a better performance. Experiments are conducted for all the languages in version 3 of the Europarl corpus (11 languages). Secondly, we use new analogy patterns and assign different weights to different analogy patterns. As is standard, we use perplexity to estimate the performence of our language model. The results show that our proposed technique yields improvements over previously proposed techniques.

The rest of this paper is as follows. Section 2 introduce proportional analogy. Section 3 introduces patterns used in building language models. Section 4 presents the improvements. A conclusion is given in Section 5.

2 Proportional analogy in language models

The use of analogy in this paper takes advantage of the notion of proportions. Consider Equation 1. By using the notion of proportion, this equation can be solved: x is “□♦”.

$$\triangledown\lozenge : \square\lozenge :: \triangledown\blacklozenge : x \tag{1}$$

2.1 Definition of Analogy

In the field of natural language processing, several works have explored the use of proportional analogy such as [2], [7] and [4]. The definition of proportional analogy we use comes from [5]. It imposes three constrains for four sequences of symnols to form a proportional analogy. These constraints are given in Equation 2.1.

$$A : B :: C : D \Longleftrightarrow \begin{cases} |A|_a - |B|_a = |C|_a - |D|_a,\ \forall a \\ dist(A,B) = dist(C,D) \\ dist(A,C) = dist(B,D) \end{cases}$$

The above constraints can be applied to many fields by changing the data types for “A, B, C, D” and “a”. In this paper, “A, B, C, D” are N-grams in a corpus and “a” stands for words in N-grams. dist(A, B) refers to the edit distance which uses only insertions and deletions [3]. This will be detailed in the following section.

2.2 Edit Distance

Edit distance is often used to refer specifically to Levenshtein distance [6]. The value of the edit distance between two sequences of symbols (= sequences of words in this paper) is the minimal number of edit operations needed to transform the first sequence into the second sequence.

Mathematically, the Levenshtein distance between two strings a, b is given by $\mathrm{dist}_{\mathrm{lev}}(|a|,|b|)$, the formula shows in the following equation.

$$\mathrm{dist}_{\mathrm{lev}}(i,j) = \begin{cases} \max(i,j) & \text{if} \min(i,j)=0, \\ \min \begin{cases} \mathrm{dist}_{\mathrm{lev}}(i-1,j)+1 & (\mathit{insertion}) \\ \mathrm{dist}_{\mathrm{lev}}(i,j-1)+1 & (\mathit{deletion}) \\ \mathrm{dist}_{\mathrm{lev}}(i-1,j-1)+[a_i \neq b_j] & (\mathit{substitution}) \end{cases} & \text{otherwise} \end{cases} \tag{2}$$

The edit distance that we use in this paper is a bit different from Levenshtein distance, as substitution is not used in our setting. Only insertions and deletions are used. It is the simplest edit distance that can be thought of. It can be considered a special case of Levenshtein distance, by setting the cost of substitutions to two. The following example in Figure 1 shows how to calculate edit distance in this paper. The unit is word as is used in this paper. In Figure 1, the number of word "to" in each N-gram is calculated and checked to see if it matches the definition of analogy, edit distance is also calculated as definition of analogy requires [8].

to walk : I walk :: *to* laugh : I laugh

to: 1 - 0 = 1 - 0

d(*to* walk, *I* walk) = d(*to* laugh, *I* laugh) = 2

d(to *work*, to *laugh*) = d(I *walk*, I *laugh*) = 2

Figure 1: An example of proportion analogy edit distance

3 Building Language Models

3.1 Previous work: efficiency of patterns

To check the efficiency of the various proportion analogy patterns that can be formed between bigrams or trigrams, we check the percentages of unseen n-grams which can be reconstructed be a given pattern. Table 1 compares the number of occurrences of patterns of trigrams and shows the cumulated number of occurrences of patterns for trigrams. The index we use to evaluate different patterns is the number of trigrams in the test data that can be reconstructed using a pattern. The fact that one N-gram in the test data may can be reconstructed by more than one pattern explains the differences in Table 1. Our training data consists of 350,000 lines from the English part of the Europarl Corpus. The test data consists of 36,237 lines.

Table 1 shows only the first three most efficient analogical patterns. By contrast, in previous work [8] only the pattern $abc : abd :: efc : efd$ and the pattern $abc : bcf :: ade : def$ were used to reconstruct unseen n-grams. To give a real example: the most productive pattern for trigrams is:

Table 1: Percentages of unseen n-grams which can be reconstructed by a given pattern (%)

Pattern	Separate	Accumulated
abc:abd::efc:efd	76%	76%
abc:bcf::ade:def	55%	96%
bcf:abc::def:ade	19%	99%
...	...	100%

$$abc : abd :: efc : efd$$

In real text, this corresponds to the following example:

opportunity to serve : opportunity to bring :: that could serve : that could bring

The second most productive pattern for trigrams is:

$$abc : bcf :: ade : def$$

It can be illustrated by the following real data:

opportunity to serve : to serve people :: opportunity for it's : for it's people

3.2 Rationale for analogy-based language models

The rationale for analogy-based language models is that not all unseen N-grams are equal. We belive that, if an N-gram does not appear in the training data, but can be reconstructed by proportional analogy pattern, it should be considered safer than an N-gram which cannot be reconstructed. The more patterns can generate this N-gram, the safer it should be. An N-gram language should reflect this fact and these separate unseen N-grams according to whether they can be reconstructed or not.

3.3 Previous work: Parameters of analogy-based language model

Known N-grams are those N-grams which appear in training data at least once. We give them a probability defined in Equation 3:

$$\frac{P(h_i.w_i)}{P(h_i)} = \frac{C(h_i.w_i)}{C(h_i)+\delta V} \tag{3}$$

For unknown N-grams that donot appear in the training data, but which can be reconstructed by analogy, we assign a probability as given by Equation 4:

$$\frac{P(h_i.w_i)}{P(h_i)} = \frac{\alpha_1}{C(h_i)+\delta V} \tag{4}$$

For unknown N-grams that cannot be reconstructed by analogy, we give them the probability defined in Equation 5:

$$\frac{P(h_i.w_i)}{P(h_i)} = \frac{\alpha_2}{C(h_i)+\delta V} \tag{5}$$

The probabilities above are inspired by Lidstone smoothing. In these formulas, the following notations are used:

- w_i: a word.
- h_i: n-1 words ahead of w_i.
- $C(h_i.w_i)$: counts of the $h_i.w_i$ in training data.
- α_1: a number less than 1 to assign probability to n-gram that can be reconstructed.
- α_2: a number less than 1 to assign probability to n-gram that cannot be reconstructed (α_1 should be bigger than α_2).
- δ: a parameter to make all the probabilities add up to 1.
- V: the length of vocabulary.

We must note that this formula is an approximate one, so all the probabilities add up to a number which is close to 1. In test data, suppose λ is the proportion of unseen N-grams to all n-grams, and μ is the proportion of the N-grams that can be reconstructed by analogy to unseen N-grams.
By summing up all the possibilities for N-grams in the test set, taking into consideration Equation 3, 4, 5, in this model, δ is calculated as Equation 6.

$$\delta = \mu\lambda\alpha_1 + (1-\mu)\lambda\alpha_2 \qquad (6)$$

4 Experiments

Our training data consists in in 378,237 lines from the English part of Europarl corpus. Our test data consists in 100 randomly selected lines from the remaining 4000 lines. We repeat our experiments 10 times and the final average results are shown in Table 2. The parameters in

Table 2: Perplexity for new language model and additive smoothing

	additive	analogy
average	232.45	**180.35**
std dev	32.83	**26.44**

this experiment for additive smoothing seems not to be the best (α_1=0.0003, α_2=0.00003 as default).

4.1 Introducing brute-force attack to tune parameters for analogy

In order to select the two previous parameters α_1 and α_2, we choose brute-force search to get the best values for our language models.

- For α_1, start from 10^{-6} to 10^{-3}, the accuracy is 10^{-6}, it means 1000 candidates.
- For α_2, start from 10^{-7} to 10^{-4}, the accuracy is 10^{-7}, it means 1000 candidates.
- In total there are about $1000 \times 1000/2$ candidate pairs for α_1 and α_2. The division by 2 comes from the fact that α_1 should be smaller than α_2 according to our formula.

Using one machine with 4 GB RAM for 20.6 days, we tune the best values for the parameters α_1 and α_2 ($\alpha_1 = 8.1 \times 10^{-4}$, $\alpha_2 = 9 \times 10^{-5}$). Table 3 shows the result by new parameters.

Table 3: Perplexity for new language model and additive smoothing

	additive	analogy
average	232.45	**161.35**
std dev	32.83	**23.44**

4.2 Select parameters for other languages

In the same way, we obtained the necessary parameters for all other languages of Europarl. The values for each language are given in Table 4.

Table 4: Parameters for 3-gram

parameters	da	de	el	en	es	fi	fr	it	nl	pt	sv
$\alpha_1 \times 10^{-4}$	5	6	6	8	8	7	9	6	6	7	6
$\alpha_2 \times 10^{-5}$	8.0	9.4	8.9	9.0	10.0	8.8	6.8	7.1	10.0	8.0	9.5

4.3 Using more patterns

Table 1 shows that a third pattern can cover 19% of the unknow N-grams. By taking into consideration this third pattern, the perplexity of our language models can still be reduced as shown in Table 5.

Table 5: Perplexity of additive smoothing method compared with analogy based smoothing method using the third pattern of Table 1 in addiction to other patterns

	additive	analogy(weights)
average	231.05	**149.78**
std dev	26.63	**22.69**

4.4 Giving weights to more reliable N-grams

Another way to improve the result may be to give weights to n-grams which can reconstruct more proportional analogy patterns. By doing this we may get a better result by assigning probabilities more precisely, the previous formula is as below:

$$p(w_i|h_i) = \frac{\alpha_1}{C(h_i) + \delta V} \tag{7}$$

We use three patterns in the research reported here. If an N-gram can be reconstrcted by several patterns, we consider it is safer. Consequentlyn we change the weights of this N-gram by assuming that the more patterns can generate a trigram from training corpus, the safer this trigram is.

If a trigram can be reconstructed by the three patterns we use, i.e $abc : abd :: efc : efd$, $abc : bcf :: ade : def$, $bcf : abc :: def : ade$ the probability we assign to it is given in Equation 8.

$$p(w_i|h_i) = \frac{a * \alpha_1}{C(h_i) + \delta V} \tag{8}$$

If a trigram can be reconstructed by any two of the above patterns, the probability we assign to it is given in Equation 9.

$$p(w_i|h_i) = \frac{b * \alpha_1}{C(h_i) + \delta V} \tag{9}$$

Finally, if a trigram can be reconstructed by only one of the patterns, the probability we assign to it is given in Equation 10.

$$p(w_i|h_i) = \frac{c * \alpha_1}{C(h_i) + \delta V} \tag{10}$$

Using tuning, we are able to determine the three parameters (a, b, c). In our experiments with English, we get a=1.5, b=1.2 and c=1.0. With these parameters, the perplexity we obtain is shown in Table 6. We see that the perplexity still decreased in comparison with Table 5.

Table 6: Perplexity of additive smoothing method compared with analogy based smoothing method giving different weight to trigrams which can be reconstructed by more patterns

	additive	analogy(weights)
average	230.00	**146.18**
std dev	26.33	**15.82**

5 Conclusions

In this paper, (1) we modified an existing smoothing technique based on analogy to smooth N-gram language models. We used a new formula to assign probabilities to N-grams, and used several other methods to improve the performance. (2) We used brute-force search to tune the parameters . The result shows that by using the new parameters our method outperforms the existing smoothing technique based on analogy by about 15%.

References

[1] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

[2] Arnaud Delhay and Laurent Miclet. Analogical equations in sequences: Definition and resolution. In *Grammatical Inference: Algorithms and Applications*, pages 127–138. Springer, Athens, Greece, 2004.

[3] Yves Lepage. Solving analogies on words: an algorithm. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL '98)*, volume 1, pages 728–734, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics.

[4] Yves Lepage. Analogy and formal languages. *Electr. Notes Theor. Comput. Sci.*, 53:180–191, 2001.

[5] Yves Lepage. Lower and higher estimates of the number of true analogies between sentences contained in a large multilingual corpus. In *Proceedings of the 20th international conference on Computational Linguistics (COLING 2004)*, pages 736–742, Geneva, Switzerland, 2004. COLING.

[6] Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI '98)*, 20 (5):522–532, 1998.

[7] Nicolas Stroppa and François Yvon. Analogical learning and formal proportions: Definitions and methodological issues. *Technical Report ENST-2005-D004*, page no pagination, 2005.

[8] Ding Yi. Building n-gram language models using analogy, July 2013.