Revisiting Optimal Decoding for Machine Translation IBM Model 4

Sebastian Riedel[*†] **James Clarke**[‡]
[*]Department of Computer Science, University of Tokyo, Japan
[†]Database Center for Life Science, Research Organization of Information and System, Japan
[‡]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801
[*]sebastian.riedel@gmail.com [†]clarkeje@gmail.com

Abstract

This paper revisits optimal decoding for statistical machine translation using IBM Model 4. We show that exact/optimal inference using Integer Linear Programming is more practical than previously suggested when used in conjunction with the Cutting-Plane Algorithm. In our experiments we see that exact inference can provide a gain of up to one BLEU point for sentences of length up to 30 tokens.

1 Introduction

Statistical machine translation (MT) systems typically contain three essential components: (1) a model, specifying how the process of translation occurs; (2) learning regime, dictating the estimation of model's parameters; (3) decoding algorithm which provides the most likely translation of an input sentence given a model and its parameters.

The search space in statistical machine translation is vast which can make it computationally prohibitively to perform exact/optimal decoding (also known as search and MAP inference) especially since dynamic programming methods (such as the Viterbi algorithm) are typically not applicable. Thus greedy or heuristic beam-based methods have been prominent (Koehn et al., 2007) due to their efficiency. However, the efficiency of such methods have two drawbacks: (1) they are approximate and give no bounds as to how far their solution is away from the true optimum; (2) it can be difficult to incorporate additional generic global constraints into the search. The first point may be especially problematic from a research perspective as without bounds on the solutions it is difficult to determine whether the model or the search algorithm requires improvement for better translations.

Similar problems exist more widely throughout natural language processing where greedy based methods and heuristic beam search have been used in lieu of exact methods. However, recently there has been an increasing interest in using Integer Linear Programming (ILP) as a means to find MAP solutions. ILP overcomes the two drawbacks mentioned above as it is guaranteed to be exact, and has the ability to easily enforce global constraints through additional linear constraints. However, efficiency is usually sacrificed for these benefits.

Integer Linear Programming has previously been used to perform exact decoding for MT using IBM Model 4 and a bigram language model. Germann et al. (2004) view the translation process akin to the travelling salesman problem; however, from their reported results it is clear that using ILP naively for decoding does not scale up beyond short sentences (of eight tokens). This is due to the exponential number of constraints required to represent the decoding problem as an ILP program. However, work in dependency parsing (Riedel and Clarke, 2006) has demonstrated that it is possible to use ILP to perform efficient inference for very large programs when used in an incremental manner. This raises the question as to whether incremental (or Cutting-Plane) ILP can also be used to decode IBM Model 4 on real world sentences.

In this work we show that it is possible. Decoding IBM Model 4 (in combination with a bigram language model) using Cutting-Plane ILP scales to much longer sentences. This affords us the opportunity to finally analyse the performance of IBM Model 4 and the performance of its state-of-theart ReWrite decoder. We show that using exact inference provides an increase of up to one BLEU point on two language pairs (French-English and German-English) in comparison to decoding using the ReWrite decoder. Thus the ReWrite decoder performs respectably but can be improved slightly, albeit at the cost of efficiency.

Although the community has generally moved away from word-based models, we believe that displaying optimal decoding in IBM Model 4 lays the foundations of future work. It is the first step in providing a method for researchers to gain greater insight into their translation models by mapping the decoding problem of other models into an ILP representation. ILP decoding will also allow the incorporation of global linguistic constraints in a manner similar to work in other areas of natural language processing.

The remainder of this paper is organised as follows: Sections 2 and 3 briefly recap IBM Model 4 and its ILP formulation. Section 4 reviews the Cutting-Plane Algorithm. Section 5 outlines our experiments and we end the paper with conclusions and a discussion of open questions for the community.

2 IBM Model 4

In this paper we focus on the translation model defined by IBM Model 4 (Brown et al., 1993). Translation using IBM Model 4 is performed by treating the translation process a noisy-channel model where the probability of the English sentence given a French sentence is, $P(\mathbf{e}|\mathbf{f}) = P(\mathbf{f}|\mathbf{e}) \cdot P(\mathbf{e})$, where $P(\mathbf{e})$ is a language model of English. IBM Model 4 defines $P(\mathbf{f}|\mathbf{e})$ and models the translation process as a generative process of how a sequence of target words (in our case French or German) is generated from a sequence of source words (English).

The generative story is as follows. Imagine we have an English sentence, $\mathbf{e} = e_1, \ldots, e_l$ and along with a NULL word (e_o) and French sentence, $\mathbf{f} = f_1, \ldots, f_m$. First a fertility is drawn for each English word (including the NULL symbol). Then, for each e_i we then independently draws a number of French words equal to e_i's fertility. Finally we process the English source tokens in sequence to determine the positions of their generated French target words. We refer the reader to Brown et al. (1993) for full details.

3 Integer Linear Programming Formulation

Given a trained IBM Model 4 and a French sentence $\mathbf{f}$ we need to find the English sentence $\mathbf{e}$ and alignment $\mathbf{a}$ with maximal $p(\mathbf{a}, \mathbf{e}|\mathbf{f}) \backsimeq p(\mathbf{e}) \cdot p(\mathbf{a}, \mathbf{f}|\mathbf{e})$.[1]

Germann et al. (2004) present an ILP formulation of this problem. In this section we will give a very high-level description of the formulation.[2] For brevity we refer the reader to the original work for more details.

In the formulation of Germann et al. (2004) an English translation is represented as the journey of a travelling salesman that visits one English token (hotel) per French token (city). Here the English token serves as the translation of the French one. A set of binary variables denote whether or not certain English token pairs are directly connected in this journey. A set of constraints guarantee that for each French token exactly one English token is visited. The formulation also contains an exponential number of constraints which forbid the possible cycles the variables can represent. It is this set of constraints that renders MT decoding with ILP difficult.

4 Cutting Plane Algorithm

The ILP program above has an exponential number of (cycle) constraints. Hence, simply passing the ILP to an off-the-shelf ILP solver is not practical for all but the smallest sentences. For this reason Germann et al. (2004) only consider sentences of up to eight tokens. However, recent work (Riedel and Clarke, 2006) has shown that even exponentially large decoding problems may be solved efficiently using ILP solvers if a Cutting-Plane Algorithm (Dantzig et al., 1954) is used.[3]

A Cutting-Plane Algorithm starts with a subset of the complete set of constraints. In our case this subset contains all but the (exponentially many) cycle constraints. The corresponding ILP is solved by a

[1] Note that in theory we should be maximizing $p(\mathbf{e}|\mathbf{f})$. However, this requires summation over all possible alignments and hence the problem is usually simplified as described here.

[2] Note that our actual formulation differs slightly from the original work because we use a first order modelling language that imposed certain restrictions on the type of constraints allowed.

[3] It is worth mentioning that Cutting Plane Algorithms have been successfully applied for solving very large instances of the Travelling Salesman Problem, a problem essentially equivalent to the decoding in IBM Model 4.

standard ILP solver, and the solution is inspected for cycles. If it contains no cycles, we have found the true optimum: the solution with highest score that does not violate any constraints. If the solution does contain cycles, the corresponding constraints are added to the ILP which is in turn solved again. This process is continued until no more cycles can be found.

5 Evaluation

In this section we describe our experimental setup and results.

5.1 Experimental setup

Our experimental setup is designed to answer several questions: (1) Is exact inference in IBM Model 4 possible for sentences of moderate length? (2) How fast is exact inference using Cutting-Plane ILP? (3) How well does the ReWrite Decoder[4] perform in terms of finding the optimal solution? (4) Does optimal decoding produce better translations?

In order to answer these questions we obtain a trained IBM Model 4 for French-English and German-English on Europarl v3 using GIZA++. A bigram language model with Witten-Bell smoothing was estimated from the corpus using the CMU-Cambridge Language Modeling Toolkit.

For exact decoding we use the two models to generate ILP programs for sentences of length up to (and including) 30 tokens for French and 25 tokens for German.[5] We filter translation candidates following Germann et al. (2004) by using only the top ten translations for each word[6] and a list of zero fertility words.[7] This resulted in 1101 French and 1062 German sentences for testing purposes. The ILP programs were then solved using the method described in Section 3. This was repeated using the ReWrite Decoder using the same models.

5.2 Results

The Cutting-Plane ILP decoder (which we will refer to as ILP decoder) produced output for 986 French sentences and 954 German sentences. From this we can conclude that it is possible to solve 90% of our sentences exactly using ILP. For the remaining 115 and 108 sentences we did not produce a solution due to: (1) the solver not completing within 30 minutes, or (2) the solver running out of memory.[8]

Table 1 shows a comparison of the results, broken down by input sentence length, obtained on the 986 French and 954 German sentences using the ILP and ReWrite decoders. First we turn our attention to the solve times obtained using ILP (for the sentences for which the solution was found within 30 minutes). The table shows that the average solve time is under one minute per sentence. As we increase the sentence length we see the solve time increases, however, we never see an order of magnitude increase between brackets as witnessed by Germann et al. (2004) thus optimal decoding is more practical than previously suggested. The average number of Cutting-Plane iterations required was 4.0 and 5.6 iterations for French and German respectively with longer sentences requiring more on average.

We next examine the performance of the two decoders. Following Germann et al. (2004) we define the ReWrite decoder as finding the optimal solution if the English sentence is the same as that produced by the ILP decoder. Table 1 shows that the ReWrite decoder finds the optimal solution 40.1% of the time for French and 29.1% for German. We also see the ReWrite decoder is less likely to find the optimal solution of longer sentences. We now look at the model scores more closely. The average log model error per token shows that the ReWrite decoder's error is proportional to sentence length and on average the ReWrite decoder is 2.2% away from the optimal solution in log space and 60.6% in probability space[9] for French, and 4.7% and 60.9% for German.

Performing exact decoding increases the BLEU score by 0.97 points on the French-English data set and 0.61 points on the German-English data set with similar performance increases observed for all sentence lengths.

6 Discussion and Conclusions

In this paper we have demonstrated that optimal decoding of IBM Model 4 is more practical than previously suggested. Our results and analysis show that

[4]Available at `http://www.isi.edu/licensed-sw/rewrite-decoder/`

[5]These limits were imposed to ensure the Python script generating the ILP programs did not run out of memory.

[6]Based on $t(e|f)$.

[7]Extracted using the rules in the filter script `rewrite.mkZeroFert.perl`

[8]All experiments were run on 3.0GHz Intel Core 2 Duo with 4GB RAM using a single core.

[9]These high error rates are an artefact of the extremely small probabilities involved.

Len	#	Solve Stats			BLEU		
		%Eq	Err	Time	ReW	ILP	Diff
1–5	21	85.7	15.0	0.7	56.5	56.2	-0.32
6–10	121	64.5	7.8	1.4	26.1	28.0	1.90
11–15	118	47.9	5.9	2.7	22.9	23.7	0.85
16–20	238	37.4	6.3	13.9	20.4	20.8	0.41
21–25	266	30.5	6.6	70.1	20.9	22.5	1.62
26–30	152	25.7	5.3	162.6	20.9	22.3	1.38
1–30	986	40.1	6.5	48.1	21.7	22.6	0.97

(a) French-English

Len	#	Solve Stats			BLEU		
		%Eq	Err	Time	ReW	ILP	Diff
1–5	31	83.9	27.4	0.8	40.7	41.1	0.44
6–10	175	51.4	19.7	1.7	19.2	20.9	1.72
11–15	242	30.6	17.4	5.5	16.0	16.7	0.72
16–20	257	19.1	14.4	23.9	15.8	15.9	0.16
21–25	249	15.7	14.0	173.4	15.3	15.9	0.61
1–25	954	29.1	16.4	53.5	16.1	16.7	0.61

(b) German-English

Table 1: Results on the two corpora. Len: range of sentence lengths; #: number of sentences in this range; %Eq: percentage of times ILP decoder returned same English sentence; Err: average difference between decoder scores per token ($\times 10^{-2}$) in log space; Time: the average solve time per sentence of ILP decoder in seconds; BLEU ReW, BLEU ILP, BLEU Diff: the BLEU scores of the output and difference between BLEU scores.

exact decoding has a practical purpose. It has allowed us to investigate and validate the performance of the ReWrite decoder through comparison of the outputs and model scores from the two decoders. Exact inference also provides an improvement in translation quality as measured by BLEU score.

During the course of this research we have encountered numerous challenges that were not apparent at the start. These challenges raise some interesting research questions and practical issues one must consider when embarking on exact inference using ILP. The first issue is that the generation of the ILP programs can take a long time. This leads us to wonder if there may be a way to provide tighter integration of program generation and solving. Such an integration would avoid the need to query the models in advance for *all* possible model components the solver may require.

Related to this issue is how to tackle the incorporation of higher order language models. Currently we use our bigram language model in a brute-force manner: in order to generate the ILP we evaluate the probability of all possible bigrams of English candidate tokens in advance. It seems clear that with higher order models this process will become prohibitively expensive. Moreover, even if the ILP could be generated efficiently, they will obviously be larger and harder to solve than our current ILPs. One possible solution may be the use of so-called delayed column generation strategies which incrementally add parts of the objective function (and hence the language model), but only when required by the ILP solver.[10]

[10]Note that delayed column generation is dual to performing cutting planes.

The use of ILP in other NLP tasks has provided a principled and declarative manner to incorporate global linguistic constraints on the system output. This work lays the foundations for incorporating similar global constraints for translation. We are currently investigating linguistic constraints for IBM Model 4 and other word-based models in general. A further extension is to reformulate higher-level MT models (phrase- and syntax-based) within the ILP framework. These representations could be more desirable from a linguistic constraint perspective as the formulation of constraints may be more intuitive.

Acknowledgements

We would like to thank Ulrich Germann and Daniel Marcu for their help with the ISI ReWrite Decoder.

References

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics* 19(2):263–311.

Dantzig, George B., Ray Fulkerson, and Selmer M. Johnson. 1954. Solution of a large-scale traveling salesman problem. *Operations Research* 2:393–410.

Germann, Ulrich, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2004. Fast and optimal decoding for machine translation. *Artificial Intelligence* 154(1-2):127–143.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2009 Demos*. Prague, Czech Republic, pages 177–180.

Riedel, Sebastian and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP 2006*. pages 129–137.