Interactively Exploring a Machine Translation Model

Steve DeNeefe, Kevin Knight, and Hayward H. Chan
Information Sciences Institute and Department of Computer Science
The Viterbi School of Engineering, University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{sdeneefe,knight}@isi.edu, hhchan@umich.edu

Abstract

This paper describes a method of interactively visualizing and directing the process of translating a sentence. The method allows a user to explore a model of syntax-based statistical machine translation (MT), to understand the model's strengths and weaknesses, and to compare it to other MT systems. Using this visualization method, we can find and address conceptual and practical problems in an MT system. In our demonstration at ACL, new users of our tool will drive a syntax-based decoder for themselves.

1 Introduction

There are many new approaches to statistical machine translation, and more ideas are being suggested all the time. However, it is difficult to determine how well a model will actually perform. Experienced researchers have been surprised by the capability of unintuitive word-for-word models; at the same time, seemingly capable models often have serious hidden problems — intuition is no substitute for experimentation. With translation ideas growing more complex, capturing aspects of linguistic structure in different ways, it becomes difficult to try out a new idea without a large-scale software development effort.

Anyone who builds a full-scale, trainable translation system using syntactic information faces this problem. We know that syntactic models often do not fit the data. For example, the syntactic system described in Yamada and Knight (2001) cannot translate n-to-m-word phrases and does not allow for multi-level syntactic transformations; both phenomena are frequently observed in real data. In building a new syntax-based MT system which addresses these flaws, we wanted to find problems in our framework as early as possible. So we decided to create a tool that could help us answer questions like:

1. Does our framework allow good translations for real data, and if not, where does it get stuck?

2. How does our framework compare to existing state-of-the-art phrase-based statistical MT systems such as Och and Ney (2004)?

The result is DerivTool, an interactive translation visualization tool. It allows a user to build up a translation from one language to another, step by step, presenting the user with the myriad of choices available to the decoder at each point in the process. DerivTool simplifies the user's experience of exploring these choices by presenting only the decisions relevant to the context in which the user is working, and allowing the user to search for choices that fit a particular set of conditions. Some previous tools have allowed the user to visualize word alignment information (Callison-Burch et al., 2004; Smith and Jahr, 2000), but there has been no corresponding deep effort into visualizing the decoding experience itself. Other tools use visualization to aid the user in manually developing a grammar (Copestake and Flickinger, 2000), while our tool visualizes

Starting with:	被 警方 击毙
and applying the rule:	`NPB(DT(the) NNS(police))` ↔ 警方
we get:	被 `NPB(DT(the) NNS(police))` 击毙
If we then apply the rule:	`VBN(killed)` ↔ 击毙
we get:	被 `NPB(DT(the) NNS(police)) VBN(killed)`
Applying the next rule:	`NP-C(x0:NPB)` ↔ `x0`
results in:	被 `NP-C(NPB(DT(the) NNS(police))) VBN(killed)`
Finally, applying the rule:	`VP(VBD(was) VP-C(x0:VBN PP(IN(by) x1:NP-C)))` ↔ 被 `x1 x0`
results in the final phrase:	`VP(VBD(was) VP-C(VBN(killed) PP(IN(by) NP-C(NPB(DT(the) NNS(police))))))`

Table 1: By applying applying four rules, a Chinese verb phrase is translated to English.

the translation process itself, using rules from very large, automatically learned rule sets. DerivTool can be adapted to visualize other syntax-based MT models, other tree-to-tree or tree-to-string MT models, or models for paraphrasing.

2 Translation Framework

It is useful at this point to give a brief description of the syntax-based framework that we work with, which is based on translating Chinese sentences into English syntax trees. Galley et al. (2004) describe how to learn hundreds of millions of tree-transformation rules from a parsed, aligned Chinese/English corpus, and Galley et al. (submitted) describe probability estimators for those rules. We decode a new Chinese sentence with a method similar to parsing, where we apply learned rules to build up a complete English tree hypothesis from the Chinese string.

The rule extractor learns rules for many situations. Some are simple phrase-to-phrase rules such as:

`NPB(DT(the) NNS(police))` ↔ 警方

This rule should be read as follows: replace the Chinese word 警方 with the noun phrase "the police". Others rules can take existing tree fragments and build upon them. For example, the rule

`S(x0:NP-C x1:VP x2:.)` ↔ `x0 x1 x2`

takes three parts of a sentence, a noun phrase (x0), a verb phrase (x1), and a period (x2) and ties them together to build a complete sentence. Rules also can involve phrase re-ordering, as in

`NPB(x0:JJ x1:NN)` ↔ `x1 x0`

This rule builds an English noun phrase out of an adjective (x0) and a noun (x1), but in the Chinese, the order is reversed. Multilevel rules can tie several of these concepts together; the rule

`VP(VBD(was) VP-C(x0:VBN PP(IN(by) x1:NP-C)))`
↔ 被 `x1 x0`

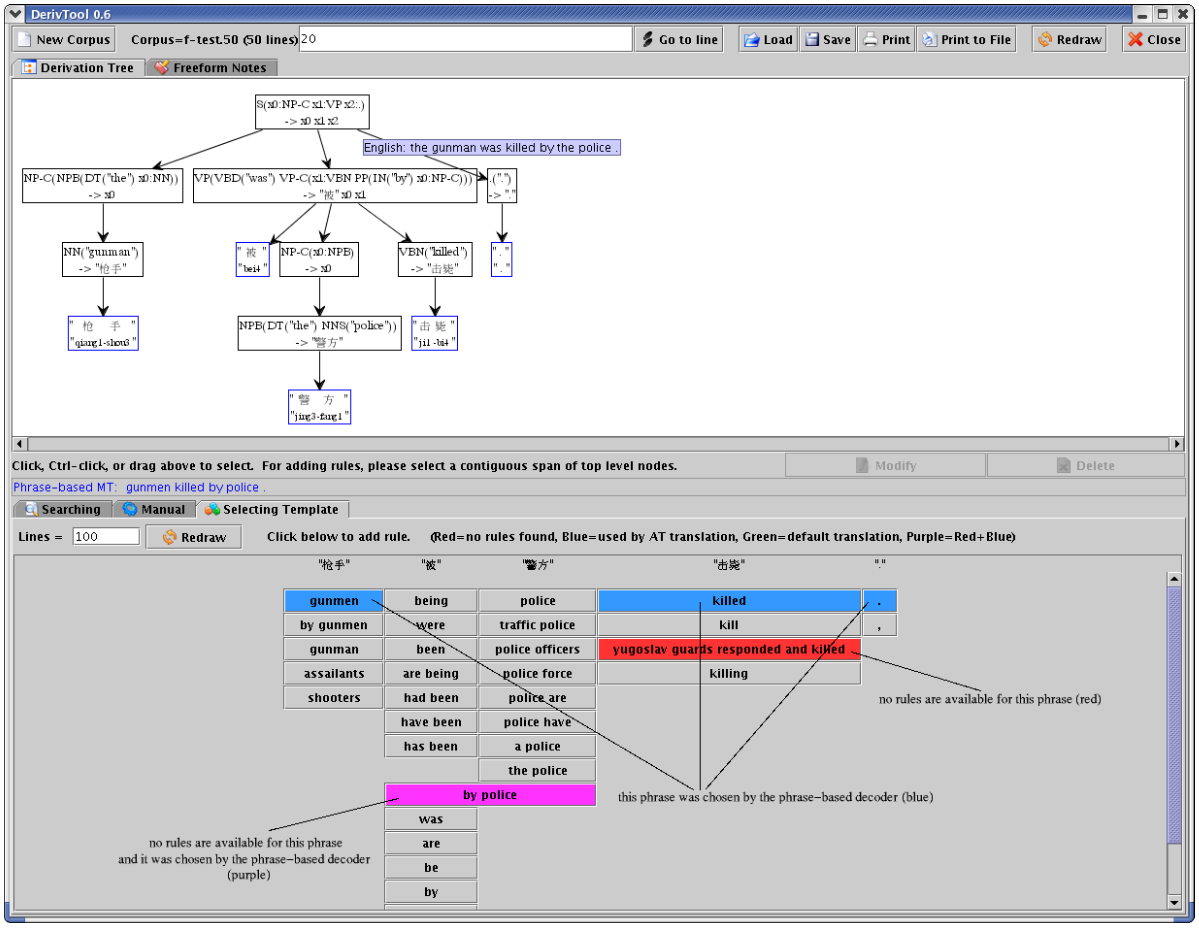
takes a Chinese word 被 and two English constituents — x1, a noun phrase, and x0, a past-participle verb — and translates them into a phrase of the form "was [verb] by [noun-phrase]". Notice that the order of the constituents has been reversed in the resulting English phrase, and that English function words have been generated.

The decoder builds up a translation from the Chinese sentence into an English tree by applying these rules. It follows the decoding-as-parsing idea exemplified by Wu (1996) and Yamada and Knight (2002). For example, the Chinese verb phrase 被 警方 击毙 (literally, "[passive] police kill") can be translated to English via four rules (see Table 1).

3 DerivTool

In order to test whether good translations can be generated with rules learned by Galley et al. (2004), we created DerivTool as an environment for interactively using these rules as a decoder would. A user starts with a Chinese sentence and applies rules one after another, building up a translation from Chinese to English. After finishing the translation, the user can save the trace of rule-applications (the *derivation tree*) for later analysis.

We now outline the typical procedure for a user to translate a sentence with DerivTool. To start, the user loads a set of sentences to translate and chooses a particular one to work with. The tool then presents the user with a window split halfway up. The top



English: the gunman was killed by the police .

Click, Ctrl-click, or drag above to select. For adding rules, please select a contiguous span of top level nodes.

Phrase-based MT: gunmen killed by police .

Click below to add rule. (Red=no rules found, Blue=used by AT translation, Green=default translation, Purple=Red+Blue)

"枪手"	"被"	"警方"	"击毙"	"."
gunmen	being	police	killed	.
by gunmen	were	traffic police	kill	,
gunman	been	police officers	yugoslav guards responded and killed	
assailants	are being	police force	killing	
shooters	had been	police are		
	have been	police have		
	has been	a police		
		the police		
	by police			
	was			
	are			
	be			
	by			

no rules are available for this phrase (red)

this phrase was chosen by the phrase–based decoder (blue)

no rules are available for this phrase and it was chosen by the phrase–based decoder (purple)

Figure 1: DerivTool with a completed derivation.

half is the workspace where the user builds a translation. It initially displays only the Chinese sentence, with each word as a separate node. The bottom half presents a set of tabbed panels which allow the user to select rules to build up the translation. See Figure 1 for a picture of the interface showing a completed derivation tree.

The most immediately useful panel is called *Selecting Template*, which shows a grid of possible English phrasal translations for Chinese phrases from the sentence. This phrase grid contains both phrases learned in our extracted rules (e.g., “the police” from earlier) and phrases learned by the phrase-based translation system (Och and Ney, 2004)[1]. The user presses a grid button to choose a phrase to include in the translation. At this point, a frequency-ordered list of rules will appear; these rules translate the Chinese phrase into the button-selected English phrase, and the user specifies which one to use. Often there will be more than one rule (e.g., 击毙 may translate via the rule `VBD(killed)` ↔ 击毙 or `VBN(killed)` ↔ 击毙), and sometimes there are no rules available. When there are no rules, the buttons are marked in red, telling us that the phrase-based system has access to this phrasal translation but our learned syntactic rules did not capture it. Other buttons are marked green to represent translations from the specialized number/name/date system, and others are blue, indicating the phrases in the phrase-based decoder’s best output. A purple button indicates both red and blue, i.e., the phrase was chosen by the phrase-based decoder but is unavailable in our syntactic framework. This is a bad combination, showing us where rule learning is weak. The

[1]The phrase-based system serves as a sparring partner. We display its best decoding in the center of the screen. Note that in Figure 1 its output lacks an auxiliary verb and an article.

remaining buttons are gray.

Once the user has chosen the phrasal rules required for translating the sentence, the next step is to stitch these phrases together into a complete English syntax tree using more general rules. These are found in another panel called *Searching*. This panel allows a user to select a set of adjacent, top-level nodes in the tree and find a rule that will connect them together. It is commonly used for building up larger constituents from smaller ones. For example, if one has a noun-phrase, a verb-phrase, and a period, the user can search for the rule that connects them and builds an “S” on top, completing the sentence. The results of a search are presented in a list, again ordered by frequency.

A few more features to note are: 1) loading and saving your work at any point, 2) adding free-form notes to the document (e.g. “I couldn’t find a rule that...”), and 3) manually typing rules if one cannot be found by the above methods. This allows us to see deficiencies in the framework.

4 How DerivTool Helps

First, DerivTool has given us confidence that our syntax-based framework can work, and that the rules we are learning are good. We have been able to manually build a good translation for each sentence we tried, both for short and long sentences. In fact, there are multiple good ways to translate sentences using these rules, because different DerivTool users translate sentences differently. Ordering rules by frequency and/or probability helps us determine if the rules we want are also frequent and favored by our model.

DerivTool has also helped us to find problems with the framework and to see clearly how to fix them. For example, in one of our first sentences we realized that there was no rule for translating a date — likewise for numbers, names, currency values, and times of day. Our phrase-based system solves these problems with a specialized date/name/number translator. Through the process of manually typing syntactic transformation rules for dates and numbers in DerivTool, it became clear that our current date/name/number translator did not provide enough information to create such syntactic rules automatically. This sparked a new area of research before we had a fully-functional decoder.

We also found that multi-word noun phrases, such as “Israeli Prime Minister Sharon” and “the French Ambassador’s visit” were often parsed in a way that did not allow us to learn good translation rules. The flat structure of the constituents in the syntax tree makes it difficult to learn rules that are general enough to be useful. Phrases with possessives also gave particular difficulty due to the awkward multilevel structure of the parser’s output. We are researching solutions to these problems involving restructuring the syntax trees before training.

Finally, our tool has helped us find bugs in our system. We found many cases where rules we wanted to use were unexpectedly absent. We eventually traced these bugs to our rule extraction system. Our decoder would have simply worked around this problem, producing less desirable translations, but DerivTool allowed us to quickly spot the missing rules.

5 Conclusion

We created DerivTool to test our MT framework against real-world data before building a fully-functional decoder. By allowing us to play the role of a decoder and translate sentences manually, it has given us insight into how well our framework fits the data, what some of its weaknesses are, and how it compares to other systems. We continue to use it as we try out new rule-extraction techniques and finish the decoding system.

References

Chris Callison-Burch, Colin Bannard and Josh Schroeder. 2004. Improved statistical translation through editing. *EAMT-2004 Workshop.*

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. *Proc. of LREC 2000.*

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What’s in a translation rule? *Proc. of NAACL-HLT 2004.*

Franz Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

Noah A. Smith and Michael E. Jahr. 2000. Cairo: An Alignment Visualization Tool. *Proc. of LREC 2000.*

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. *Proc. of ACL.*

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. *Proc. of ACL.*

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. *Proc. of ACL.*