Deep Syntax Language Models and Statistical Machine Translation

Yvette Graham
NCLT
Dublin City University
ygraham@computing.dcu.ie

Josef van Genabith
CNGL
Dublin City University
josef@computing.dcu.ie

Abstract

Hierarchical Models increase the reordering capabilities of MT systems by introducing non-terminal symbols to phrases that map source language (SL) words/phrases to the correct position in the target language (TL) translation. Building translations via discontiguous TL phrases increases the difficulty of language modeling, however, introducing the need for heuristic techniques such as cube pruning (Chiang, 2005), for example. An additional possibility to aid language modeling in hierarchical systems is to use a language model that models fluency of words not using their local context in the string, as in traditional language models, but instead using the deeper context of a word. In this paper, we explore the potential of deep syntax language models providing an interesting comparison with the traditional string-based language model. We include an experimental evaluation that compares the two kinds of models independently of any MT system to investigate the possible potential of integrating a deep syntax language model into Hierarchical SMT systems.

1 Introduction

In Phrase-Based Models of Machine Translation all phrases consistent with the word alignment are extracted (Koehn et al., 2003), with shorter phrases needed for high coverage of unseen data and longer phrases providing improved fluency in target language translations. Hierarchical Models (Chiang, 2007; Chiang, 2005) build on Phrase-Based Models by relaxing the constraint that phrases must be contiguous sequences of words and allow a short phrase (or phrases) nested within a longer phrase to be replaced by a non-terminal symbol forming a new hierarchical phrase. Traditional language models use the local context of words to estimate the probability of the sentence and introducing hierarchical phrases that generate discontiguous sequences of TL words increases the difficulty of computing language model probabilities during decoding and require sophisticated heuristic language modeling techniques (Chiang, 2007; Chiang, 2005).

Leaving aside heuristic language modeling for a moment, the difficulty of integrating a traditional string-based language model into the decoding process in a hierarchical system, highlights a slight incongruity between the translation model and language model in Hierarchical Models. According to the translation model, the best way to build a fluent TL translation is via discontiguous phrases, while the language model can only provide information about the fluency of contiguous sequences of words. Intuitively, a language model that models fluency between discontiguous words may be well-suited to hierarchical models. Deep syntax language models condition the probability of a word on its deep context, i.e. words linked to it via dependency relations, as opposed to preceding words in the string. During decoding in Hierarchical Models, words missing a context in the string due to being preceded by a non-terminal, might however be in a dependency relation with a word that is already present in the string and

Proceedings of SSST-4, Fourth Workshop on Syntax and Structure in Statistical Translation, pages 118–126,
COLING 2010, Beijing, August 2010.

this context could add useful information about the fluency of the hypothesis as its constructed.

In addition, using the deep context of a word provides a deeper notion of fluency than the local context provides on its own and this might be useful to improve such things as lexical choice in SMT systems. Good lexical choice is very important and the deeper context of a word, if available, may provide more meaningful information and result in better lexical choice. Integrating such a model into a Hierarchical SMT system is not straightforward, however, and we believe before embarking on this its worthwhile to evaluate the model independently of any MT system. We therefore provide an experimental evaluation of the model and in order to provide an interesting comparison, we evaluate a traditional string-based language model on the same data.

2 Related Work

The idea of using a language model based on deep syntax is not new to SMT. Shen et al. (2008) use a dependency-based language model in a string to dependency tree SMT system for Chinese-English translation, using information from the deeper structure about dependency relations between words, in addition to the position of the words in the string, including information about whether context words were positioned on the left or right of a word. Bojar and Hajič (2008) use a deep syntax language model in an English-Czech dependency tree-to-tree transfer system, and include three separate bigram language models: a reverse, direct and joint model. The model in our evaluation is similar to their direct bigram model, but is not restricted to bigrams.

Riezler and Maxwell (2006) use a trigram deep syntax language model in German-English dependency tree-to-tree transfer to re-rank decoder output. The language model of Riezler and Maxwell (2006) is similar to the model in our evaluation, but differs in that it is restricted to a trigram model trained on LFG f-structures. In addition, as language modeling is not the main focus of their work, they provide little detail on the language model they use, except to say that it is based on *"log-probability of strings of predicates from root to frontier of target f-structure, estimated from predicate trigrams in English f-structures"* (Riezler and Maxwell, 2006). An important property of LFG f-structures (and deep syntactic structures in general) was possibly overlooked here. F-structures can contain more than one path of predicates from the root to a frontier that include the same ngram, and this occurs when the underlying graph includes unary branching followed by branching with arity greater than one. In such cases, the language model probability as described in Riezler and Maxwell (2006) is incorrect as the probability of these ngrams will be included multiple times. In our definition of a deep syntax language model, we ensure that such duplicate ngrams are omitted in training and testing. In addition, Wu (1998) use a bigram deep syntax language model in a stochastic inversion transduction grammar for English to Chinese. None of the related research we discuss here has included an evaluation of the deep syntax language model they employ in isolation from the MT system, however.

3 Deep Syntax

The deep syntax language model we describe is not restricted to any individual theory of deep syntax. For clarity, however, we restrict our examples to LFG, which is also the deep syntax theory we use for our evaluation. The Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Kaplan, 1995; Bresnan, 2001; Dalrymple, 2001) functional structure (f-structure) is an attribute-value encoding of bi-lexical labeled dependencies, such as *subject*, *object* and *adjunct* for example, with morpho-syntactic atomic attributes encoding information such as *mood* and *tense* of verbs, and *person*, *number* and *case* for nouns. Figure 1 shows the LFG f-structure for English sentence *"Today congress passed Obama's health care bill."*[1]

Encoded within the f-structure is a directed graph and our language model uses a simplified acyclic unlabeled version of this graph. Figure 1(b) shows the graph structure encoded within the f-structure of Figure 1(a). We discuss the simplification procedure later in Section 5.

[1] Morpho-syntactic information/ atomic features are omitted from the diagram.

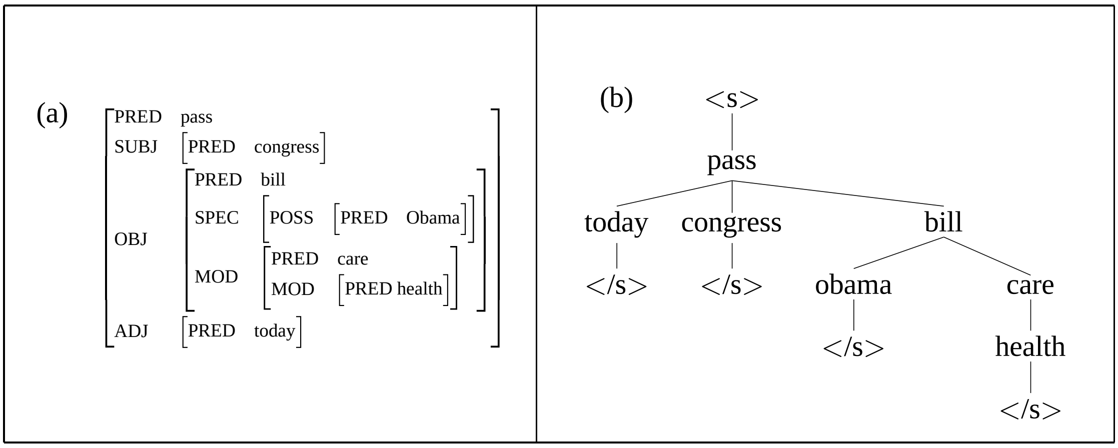

Figure 1: *"Today congress passed Obama's health care bill."*

4 Language Model

We use a simplified approximation of the deep syntactic structure, d_e, that encodes the unlabeled dependencies between the words of the sentence, to estimate a deep syntax language model probability. Traditional string-based language models combine the probability of each word in the sentence, w_i, given its preceding context, the sequence of words from w_1 to w_{i-1}, as shown in Equation 1.

$$p(w_1, w_2, ..., w_l) = \prod_{i=1}^{l} p(w_i|w_1, ..., w_{i-1}) \quad (1)$$

In a similar way, a deep syntax language model probability combines the probability of each word in the structure, w_i, given its context within the structure, the sequence of words from w_r, the head of the sentence, to $w_{m(i)}$, as shown in Equation 2, with function m used to map the index of a word in the structure to the index of its head. [2]

$$p(d_e) = \prod_{i=1}^{l} p(w_i|w_r, ..., w_{m(m(i))} w_{m(i)}) \quad (2)$$

In order to combat data sparseness, we apply the Markov assumption, as is done in traditional string-based language modeling, and simplify the probability by only including a limited length of history when estimating the probability of each word in the structure. For example, a trigram deep syntax language model conditions the probability of each word on the sequence of words consisting of *the head of the head of the word* followed by *the head of the word* as follows:

$$p(d_e) = \prod_{i=1}^{l} P(w_i|w_{m(m(i))}, w_{m(i)}) \quad (3)$$

In addition, similar to string-based language modeling, we add a start symbol, <s>, at the root of the structure and end symbols, </s>, at the leaves to include the probability of a word being the head of the sentence and the probability of words occurring as leaf nodes in the structure. Figure 2(a) shows an example of how a trigram deep syntax language model probability is computed for the example sentence in Figure 1(a).

5 Simplified Approximation of the Deep Syntactic Representation

We describe the deep syntactic structure, d_e, as an approximation since a parser is employed to automatically produce it and there is therefore no certainty that we use the actual/correct deep syntactic representation for the sentence. In addition, the function m requires that each node in the structure has exactly one head, however, structure-sharing can occur within deep syntactic structures resulting in a single word legitimately having two heads. In such cases we use a simplification of the graph in the deep syntactic structure. Figure 3 shows an f-structure in which the *subject*

[2] We refer to the lexicalized nodes in the dependency structure as *words*, alternatively the term *predicate* can be used.

	(a) Deep Syntax LM		(b) Traditional LM
$p(e) \approx$	$p($ pass $\mid$ <s>$)*$	$p(e) \approx$	$p($ passed $\mid$ today congress $)*$
	$p($ today $\mid$ <s> pass $)*$		$p($ today $\mid$ <s>$)*$
	$p($</s> $\mid$ pass today $)*$		
	$p($ congress $\mid$ <s> pass $)*$		$p($ congress $\mid$ <s> today $)*$
	$p($</s> $\mid$ pass congress $)*$		
	$p($ bill $\mid$ <s> pass $)*$		$p($ bill $\mid$ health care $)*$
	$p($ obama $\mid$ pass bill $)*$		$p($ obama $\mid$ congress passed $)*$
	$p($</s> $\mid$ bill obama $)*$		
	$p($ care $\mid$ pass bill $)*$		$p($ care $\mid$ s health $)*$
	$p($ health $\mid$ bill care $)*$		$p($ health $\mid$ ' s $)*$
	$p($</s> $\mid$ care health $)$		
			$p($ ' $\mid$ passed Obama $)*$
			$p($ s $\mid$ obama ' $)*$
			$p($. $\mid$ care bill $)*$
			$p($</s> $\mid$ bill . $)$

Figure 2: Example Comparison of Deep Syntax and Traditional Language Models

of both *like*, *be* and *president* is *hillary*. In our simplified structure, the dependency relations between *be* and *hillary* and *president* and *hillary* are dropped. We discuss how we do this later in Section 6. Similar to our simplification for structure sharing, we also simplify structures that contain cycles by discarding edges that cause loops in the structure.

6 Implementation

SRILM (Stolcke, 2002) can be used to compute a language model from ngram counts (the *-read* option of the *ngram-count* command). Implementation to train the language model, therefore, simply requires accurately extracting counts from the deep syntax parsed training corpus. To simplify the structures to acyclic graphs, nodes are labeled with an increasing index number via a depth first traversal. This allows each arc causing a loop in the graph or argument sharing to be identified by a simple comparison of index numbers, as the index number of its start node will be greater than that of its end node. The algorithm we use to extract ngrams from the dependency structures is straightforward: we simply carry out a depth-first traversal of the graph to construct paths of words that stretch from the root of the graph to words

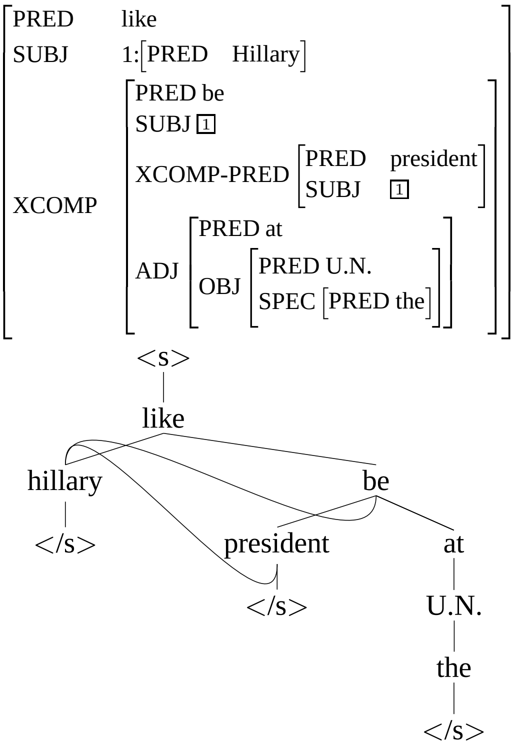

Figure 3: *"Hillary liked being president at the U.N."*

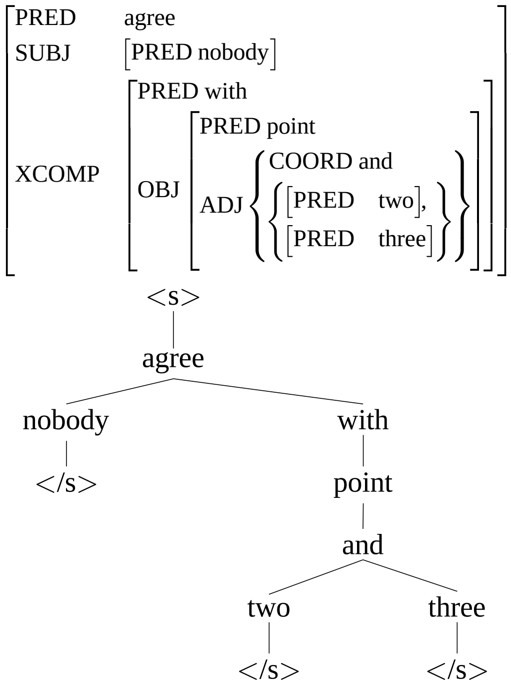

Figure 4: *“Nobody agreed with points two and three.”*

at the leaves and then extract the required order ngrams from each path. As mentioned earlier, some ngrams can belong to more than one path. Figure 4 shows an example structure containing unary branching followed by binary branching in which the sequence of symbols and words “*<s> agree with point and*” belong to the path ending in *two </s>* and *three </s>*. In order to ensure that only distinct ngrams are extracted we assign each word in the structure a unique id number and include this in the extracted ngrams. Paths are split into ngrams and duplicate ngrams resulting from their occurrence in more than one path are discarded. Its also possible for ngrams to legitimately be repeated in a deep structure, and in such cases we do not discard these ngrams. Legitimately repeating ngrams are easily identified as the id numbers attached to words will be different.

7 Deep Syntax and Lexical Choice in SMT

Correct lexical choice in machine translation is extremely important and PB-SMT systems rely on the language model to ensure, that when two phrases are combined with each other, that the model can rank combined phrases that are fluent higher than less fluent combinations. Conditioning the probability of each word on its deep context has the potential to provide a more meaningful context than the local context within the string. A comparison of the probabilities of individual words in the deep syntax model and traditional language model in Figure 2 clearly shows this. For instance, let us consider how the language model in a German to English SMT system is used to help rank the following two translations *today congress passed ...* and *today convention passed ...* (the word *Kongress* in German can be translated into either *congress* or *convention* in English). In the deep syntax model, the important competing probabilities are (i) $p(congress|\text{<s>}pass)$ and (ii) $p(convention|\text{<s>}pass)$, where (i) can be interpreted as the probability of the word *congress* modifying *pass* when *pass* is the head of the entire sentence and, similarly (ii) the probability of the word *convention* modifying *pass* when *pass* is the head of the entire sentence. In the traditional string-based language model, the equivalent competing probabilities are (i) $p(congress|\text{<s>}today)$, the probability of *congress* following *today* when *today* is the start of the sentence and (ii) $p(convention|\text{<s>}today)$, probability of convention following *today* when *today* is the start of the sentence, showing that the deep syntax language model is able to use more meaningful context for good lexical choice when estimating the probability of words *congress* and *convention* compared to the traditional language model.

In addition, the deep syntax language model will encounter less data sparseness problems for some words than a string-based language model. In many languages words occur that can legitimately be moved to different positions within the string without any change to dependencies between words. For example, sentential adverbs in English, can legitimately change position in a sentence, without affecting the underlying dependencies between words. The word *today* in *“Today congress passed Obama’s health bill”* can appear as *"Congress passed Obama's health bill today"* and *"Congress today passed Obama's health bill"*. Any sentence in the training corpus in which the word *pass* is modified by *today* will result in a bigram being counted for the two words, regardless of the position of *today* within each sentence.

In addition, some surface form words such as auxiliary verbs for example, are not represented as predicates in the deep syntactic structure. For lexical choice, its not really the choice of auxiliary verbs that is most important, but rather the choice of an appropriate lexical item for the main verb (that belongs to the auxiliary verb). Omitting auxiliary verbs during language modeling could aid good lexical choice, by focusing on the choice of a main verb without the effect of what auxiliary verb is used with it.

For some words, however, the probability in the string-based language model provides as good if not better context than the deep syntax model, but only for the few words that happen to be preceded by words that are important to its lexical choice, and this reinforces the idea that SMT systems can benefit from using both a deep syntax and string-based language model. For example, the probability of *bill* in Figures 2(a) and 2(b) is computed in the deep syntax model as $p(bill|$ <s> $pass)$ and in the string-based model using $p(bill|health\ care)$, and for this word the local context seems to provide more important information than the deep context when it comes to lexical choice. The deep model nevertheless adds some useful information, as it includes the probability of *bill* being an argument of *pass* when *pass* is the head of a sentence.

In traditional language modeling, the special start symbol is added at the beginning of a sentence so that the probability of the first word appearing as the first word of a sentence can be included when estimating the probability. With similar motivation, we add a start symbol to the deep syntactic representation so that the probability of the head of the sentence occurring as the head of a sentence can be included. For example, $p(be|$ <s>$)$ will have a high probability as the verb *be* is the head of many sentences of English, whereas $p(colorless|$ <s>$)$ will have a low probability since it is unlikely to occur as the head. We also add end symbols at the leaf nodes in the structure to include the probability of these words appearing at that position in a structure. For instance, a noun followed by its determiner such as $p($</s> $|attorney\ a)$ would have a high probability compared to a conjunction followed by a verb $p($</s> $|and\ be)$.

8 Evaluation

We carry out an experimental evaluation to investigate the potential of the deep syntax language model we describe in this paper independently of any machine translation system. We train a 5-gram deep syntax language model on 7M English f-structures, and evaluate it by computing the perplexity and ngram coverage statistics on a held-out test set of parsed fluent English sentences. In order to provide an interesting comparison, we also train a traditional string-based 5-gram language model on the same training data and test it on the same held-out test set of English sentences. A deep syntax language model comes with the obvious disadvantage that any data it is trained on must be in-coverage of the parser, whereas a string-based language model can be trained on any available data of the appropriate language. Since parser coverage is not the focus of our work, we eliminate its effects from the evaluation by selecting the training and test data for both the string-based and deep syntax language models on the basis that they are in fact in-coverage of the parser.

8.1 Language Model Training

Our training data consists of English sentences from the WMT09 monolingual training corpus with sentence length range of 5-20 words that are in coverage of the parsing resources (Kaplan et al., 2004; Riezler et al., 2002) resulting in approximately 7M sentences. Preparation of training and test data for the traditional language model consisted of tokenization and lower casing. Parsing was carried out with XLE (Kaplan et al., 2002) and an English LFG grammar (Kaplan et al., 2004; Riezler et al., 2002). The parser produces a packed representation of all possible parses according to the LFG grammar and we select only the single best parse for language model training by means of a disambiguation model (Kaplan et

Corpus	Tokens	Ave. Tokens per Sent.	Vocab
strings	138.6M	19	345K
LFG lemmas/predicates	118.4M	16	280K

Table 1: Language model statistics for string-based and deep syntax language models, statistics are for string tokens and LFG lemmas for the same set of 7.29M English sentences

al., 2004; Riezler et al., 2002). Ngrams were automatically extracted from the f-structures and lowercased. SRILM (Stolcke, 2002) was used to compute both language models. Table 1 shows statistics on the number of words and lemmas used to train each model.

8.2 Testing

The test set consisted of 789 sentences selected from WMT09 additional development sets[3] containing English Europarl text and again was selected on the basis of sentences being in-coverage of the parsing resources. SRILM (Stolcke, 2002) was used to compute test set perplexity and ngram coverage statistics for each order model.

Since the deep syntax language model adds end of sentence markers to leaf nodes in the structures, the number of (so-called) end of sentence markers in the test set for the deep syntax model is much higher than in the string-based model. We therefore also compute statistics for each model when end of sentence markers are omitted from training and testing. [4] In addition, since the vast majority of punctuation is not represented as predicates in LFG f-structures, we also test the string-based language model when punctuation has been removed.

8.3 Results

Table 2 shows perplexity scores and ngram coverage statistics for each order and type of language model. Note that perplexity scores for the string-based and deep syntax language models are not directly comparable because each model has a different vocabulary. Although both models train on an identical set of sentences, the data is in a different format for each model, as the string-based model is trained on surface form tokens, whereas the deep syntax model uses lemmas. Ngram coverage statistics provide a better comparison.

Unigram coverage for all models is high with all models achieving close to 100% coverage on the held-out test set. Bigram coverage is highest for the deep syntax language model when eos markers are included (94.71%) with next highest coverage achieved by the string-based model that includes eos markers (93.09%). When eos markers are omitted bigram coverage goes down slightly to 92.44% for the deep syntax model and to 92.83% for the string-based model, and when punctuation is also omitted from the string-based model, coverage goes down again to 91.57%.

Trigram coverage statistics for the test set maintain the same rank between models as in the bigram coverage, from highest to lowest as follows: DS+eos at 64.71%, SB+eos at 58.75%, SB-eos at 56.89%, DS-eos at 53.67%, SB-eos-punc at 53.45%. For 4-gram and 5-gram coverage a similar coverage ranking is seen, but with DS-eos (4gram at 17.17%, 5gram at 3.59%) and SB-eos-punc (4gram at 20.24%, 5gram at 5.76%) swapping rank position.

8.4 Discussion

Ngram coverage statistics for the DS-eos and SB-eos-punc models provide the fairest comparison, with the deep syntax model achieving higher coverage than the string-based model for bigrams (+0.87%) and trigrams (+0.22%), marginally lower coverage coverage of unigrams (-0.02%) and lower coverage of 4-grams (-3.07%) and 5-grams (2.17%) compared to the string-based model.

Perplexity scores for the deep syntax model when eos symbols are included are low (79 for the 5gram model) and this is caused by eos markers

[3] test2006.en and test2007.en

[4] When we include end of sentence marker probabilities we also include them for normalization, and omit them from normalization when their probabilities are omitted.

	1-gram		2-gram		3-gram		4-gram		5-gram	
	cov.	ppl	cov.	ppl	cov.	ppl	cov.	ppl	cov.	ppl
SB-eos	99.61%	1045	92.83%	297	56.89%	251	23.32%	268	7.19%	279
SB-eos-punc	99.58%	1357	91.57%	382	53.45%	327	20.24%	348	5.76%	360
DS-eos	99.56%	1005	92.44%	422	53.67%	412	17.17%	446	3.59%	453
SB+eos	99.63%	900	93.09%	227	58.75%	194	25.48%	207	8.35%	215
DS+eos	99.70%	211	94.71%	77	64.71%	73	29.86%	78	8.75%	79

Table 2: Ngram coverage and perplexity (ppl) on held-out test set. Note: DS = deep syntax, SB string-based, eos = end of sentence markers

in the test set in general being assigned relatively high probabilities by the model, and since several occur per sentence, the perplexity increases when the are omitted (453 for the 5gram model).

Tables 3 and 4 show the most frequently encountered trigrams in the test data for each type of model. A comparison shows how different the two models are and highlights the potential of the deep syntax language model to aid lexical choice in SMT systems. Many of the most frequently occurring trigram probabilities for the deep syntax model are for arguments of the main verb of the sentence, conditioned on the main verb, and including such probabilities in a system could improve fluency by using information about which words are in a dependency relation together explicitely in the model. In addition, a frequent trigram in the held-out data is *<s> be also*, where the word *also* is a sentential adverb modifying *be*. Trigrams for sentential adverbs are likely to be less effected by data sparseness in the deep syntax model compared to the string-based model which could result in the deep syntax model improving fluency with respect to combinations of main verbs and their modifying adverbs. The most frequent trigram in the deep syntax test set is *<s> and be*, in which the head of the sentence is the conjunction *and* with argument *be*. In this type of syntactic construction in English, its often the case that the conjunction and verb will be distant from each other in the sentence, for example: *Nobody was there except the old lady and without thinking we quickly left.* (where *was* and *and* are in a dependency relation). Using a deep syntax language model could therefore improve lexical choice for such words, since they are too distant for a string-based model.

3-gram	No. Occ.	Prob.
<s> and be	42	0.1251
<s> be this	21	0.0110
<s> must we	19	0.0347
<s> would i	19	0.0414
<s> be in	17	0.0326
<s> be that	14	0.0122
be debate the	13	0.0947
<s> be debate	13	0.0003
<s> can not	12	0.0348
<s> and president	11	0.0002
<s> would like	11	0.0136
<s> would be	11	0.0835
<s> be also	10	0.0075

Table 3: Most frequent trigrams in test set for deep syntax model

9 Conclusions

We presented a comparison of a deep syntax language and traditional string-based language model. Results showed that the deep syntax language model achieves similar ngram coverage to the string-based model on a held out test set. We highlighted the potential of integrating such a model into SMT systems for improving lexical choice by using a deeper context for probabilities of words compared to a string-based model.

References

Bojar, Ondřej, Jan Hajič. 2008. Phrase-Based and Deep Syntactic English-to-Czech Statistical Machine Translation. In *Proceedings of the third Work-*

3-gram	No. Occ.	Prob.
mr president ,	40	0.5385
<s> this is	25	0.1877
by the european	20	0.0014
the european union	18	0.1096
<s> it is	16	0.1815
the european parliament	15	0.0252
would like to	15	0.4944
<s> i would	15	0.0250
<s> that is	14	0.1094
i would like	14	0.0335
and gentlemen ,	13	0.1005
ladies and gentlemen	13	0.2834
<s> we must	12	0.0120
should like to	12	0.1304
i should like	11	0.0089
, ladies and	11	0.5944
, it is	10	0.1090

Table 4: Most frequent trigrams in test set for string-based model

shop on Statistical Machine Translation, Columbus, Ohio.

Bresnan, Joan. 2001. *Lexical-Functional Syntax.*, Blackwell Oxford.

Chiang, David. 2007. Hierarchical Phrase-based Models of Translation In *Computational Linguistics*, No. 33:2.

Chiang, David. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263-270, Ann Arbor, Michigan.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*, Academic Press, San Diego, CA; London.

Kaplan, Ronald, Stefan Riezler, Tracy H. King, John T. Maxwell, Alexander Vasserman. 2004. Speed and Accuracy in Shallow and Deep Stochastic Parsing. In *Proceedings of Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics Meeting*, Boston, MA.

Kaplan, Ronald M., Tracy H. King, John T. Maxwell. 2002. Adapting Existing Grammars: the XLE Experience. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING) 2002*, Taipei, Taiwan.

Kaplan, Ronald M. 1995. The Formal Architecture of Lexical Functional Grammar. In *Formal Issues in Lexical Functional Grammar*, ed. Mary Dalrymple, pages 7-28, CSLI Publications, Stanford, CA.

Kaplan, Ronald M., Joan Bresnan. 1982. Lexical Functional Grammar, a Formal System for Grammatical Represenation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, 173-281, MIT Press, Cambridge, MA.

Koehn, Philipp, Hieu Hoang. 2007. Factored Translation Models. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 868-876.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicoli Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. *Annual Meeting of the Association for Computational Linguistics, demonstration session*

Koehn, Philipp 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the tenth Machine Translation Summit.*

Koehn, Philipp, Franz Josef Och, Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of Human Language Technology and North American Chapter of the Association for Computational Linguistics Conference*, 48-54.

Riezler, Stefan, John T. Maxwell III. 2006. Grammatical Machine Translation. In *Proceedings of HLT-ACL*, pages 248-255, New York.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, Mark Johnson. 2002. Parsing the Wall Street Journal using Lexical Functional Grammar and Discriminitive Estimation Techniques . (grammar version 2005) In *Proceedings of the 40th ACL*, Philadelphia.

Shen, Libin, Jinxi Xu, Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. *Proceedings of ACL-08: HLT*, pages 577-585.

Stolcke, Andreas. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, Denver, Colorado.

Dekai, Wu, Hongsing Wong. 1998. Machine Translation with a Stochastic Grammatical Channel. In *Proceedings of the 36th ACL and 17th COLING*, Montreal, Quebec.