Syntax-based and Factored Language Models
Advanced MT Seminar (11-734)

Rashmi Gangadharaiah

1 Introduction

In Statistical language modeling, we create probability models over words and sentences. Although there is a large amount of data available, statistical language modeling is a tough task in many languages that have rich morphology. Such languages have a large number of word types in relation to word tokens which leads to high perplexity, a large number of unseen word contexts and a large number of out-of-vocabulary (OOV) words. Many models are simply smoothed conditional probability distributions for a word given its preceding history. Although smoothing methods have significantly improved the performance of n-gram language models, these models have a few limitations like, n-gram language models cannot handle long range information.

The sequential nature of the surface form of sentences does not reflect the deep structure of their meaning and this weakens conventional n-gram models. Experiments on Long-distance bigrams were performed in [Rosenfeld 1994]. His work began as an attempt to capture some of the information present in the longer distance history. The training-set (1 million words of the Brown Corpus) perplexity of long-distance bigrams for various distances were obtained. The perplexity was the least when the distance(d) was 1 and increased significantly as the value of d was increased to 2,3,4 and 5. However, for d=6,....10, training set perplexity remained at almost the same level. This level was consistently below the perplexity of the d=1000 case. Hence, some information exists in the more distant past but it is spread thinly across the entire history.

The idea of incorporating some form of syntax in language models to deal with data sparseness and to generalize well to unseen word sequences was proposed by [Brown et al. 1990] in the form of class-based models. The classes extracted were used in a conditional probabilistic model which predicted the class sequence for a word sequence and then used it to predict the word sequence, w_1^N

$$p(w_1^N) \approx \sum_{t_1,\dots t_N} \prod_{i=1}^{N} p(t_i|t_1^{i-1}) p(w_i|t_i) \tag{1}$$

The above model is less effective at predicting word candidates than an n-gram word-based LM as it removes important lexical information for predicting the next word. POS-based (parts of speech based) LM suggested in [Heeman 1998] achieved a reduction in perplexity in the Speech recognition domain where the above problem was redefined as determining

$$W*, T* \approx argmax_{W,T} p(W,T) p(A|W) \tag{2}$$

where, T is the POS sequence associated with the word sequence W given the speech utterance A. From the above equations, it is clear that both class-based LMs and word-based LMs blindly discard relevant words(or classes) that lie n positions in the past and retain words of little or no value.

A different approach to language modeling is based on Probabilistic Context-free Grammar (PCFG) and this takes syntax into account. PCFGs are used to first generate parse hypotheses for the source sentence, then to prune and rearrange the hypotheses to a single tree. The leaf nodes are finally translated. These PCFGs act as both the translation model and the language model. It has been shown that syntax-based language models improve translation quality over n-gram based language models.

Models that incorporate both syntax and semantics have also been proposed where headword and/or non-headword dependencies are modeled using dependency grammars. These models evolved based on the fact that word dependencies are structurally related rather than syntactically related. These models incorporate the predictive power of words that lie outside of bigram or trigram range.

Structure-based models are not as generally applicable as an n-gram based language model as these models require a parser to parse the source language. A newer approach called Factored language modeling (FLM) tackles long range interactions differently. These models do not require all possible higher n-grams and provide a richer set of conditioning probabilities to improve performance. These models represent words as bundles of features and include a probability model covering sequences of bundles rather than words. These models are also capable of incorporating some amount of linguistic information like semantic classes or parts of speech. In these models, it is not a necessity to have parse trees and linguistic information can be added to the model when available.

This report explains many of the structured LMs and Factored LMs and reports their performance in Machine Translation (MT) and Automatic Speech Recognition(ASR) domains.

2 Structure-based Language models

Section 2.1 presents LMs that incorporate Syntax in the MT domain and Section 2.2 presents LMs that incorporate both Syntax and Semantics in ASR and MT. Section 2.2 presents structure-based models that were applied in ASR in the belief that these methods can somehow be extended to MT.

2.1 Incorporating Syntax into LMs

Recently many MT systems have tried to consider syntax in building their translation models. In [Charniak et al. 2003], a system that combined the syntactic translation model with a syntax-based LM was proposed. The translation model accepts an English parse tree as input and produces a Chinese sentence as output. The equation for decoding in the system is as follows:

$$p(E|F) \approx \sum_{\Pi} p(\pi)p(F|\pi) \quad (3)$$

where, Π is the set of trees with yield E and π varies over this set. The direction in a real translation task is the reverse of the direction of the translation model. The decoder is given a sentence in Chinese and looks for the best parse tree for the English translation according to the language model and translation model. The translation model is a tree-to-string model. The model performs three types of operations:
1. reorder child nodes, eg: change VP $\rightarrow$ VB NP PP into VP $\rightarrow$ NP PP VB
2. insert an optional word at each node
3. translate the leaf English words to Chinese words
The operations are stochastic and their probabilities are assumed to depend only on the node and independent of operations on the node and other nodes. The probability of each operation is obtained by a training algorithm on 780,000 English parse tree-Chinese sentence pairs.

The decoder builds a parse tree from a Chinese sentence using a special CFG grammar which are extracted from a parsed corpus of English. For each non-lexical English CFG rule (eg. VP $\rightarrow$ VB NP PP), possible reordered rules are supplemented (eg. VP $\rightarrow$ NP PP VB, VP $\rightarrow$ PP NP VB, etc.). For insertion operations, rules such as "VP $\rightarrow$ VP X" and "X $\rightarrow$ word" are added. For translation operations, rules such as, "$englishWord \rightarrow chineseWord$" are added. With this special grammar, a decoded tree can be built by parsing a Chinese sentence from which the English parse tree can be extracted by removing leaf Chinese words and by recovering the reordered child nodes into the English order. The decoding process has two steps. The first step builds a forest using a bottom up decoding parser using cost only from the translation model. The second step picks the best tree from the forest using a language model.

Non-lexical PCFG is used to create a large parse forest in the first phase. In the second phase, a more sophisticated lexicalized PCFG is applied. The second phase finds the best parse re-evaluating all the remaining edges. A pruning step is performed before the second phase. $e^k_{i,j}$ denotes a single expansion of a constituent in the parse forest: $n^k_{i,j} \rightarrow n^c_{i,a}$.... $rule(e^k_{i,j})$ is the PCFG rule that underlies the edge, and $rhs(e^k_{i,j})$ is the set of constituents into which the left hand side expands. Edges are removed if $p(e^k_{i,j}|w_{1,n})$ falls below 0.00001.

$$p(e^k_{i,j}|w_{1,n}) = \alpha(n^k_{i,j})p(rule(e^k_{i,j})) \prod_{n^n_{l,m} \in rhs(e^k_{i,j})} \beta(n^n_{l,m})/p(w_{1,n}) \tag{4}$$

The above equation does not take into account how likely these edges are given F (foreign language string). The following equation solves it.

$$p(E|F) = \sum_{\Pi,\Theta} p(\pi) \prod_{e^k_{i,j} \in \pi} \theta(e^k_{i,j}) \tag{5}$$

The PCFG independence assumption breaks down $p(\pi)$ as the product of the probability of each of the edges:

$$p(e^k_{i,j}|w_{1,n}) = \alpha(n^k_{i,j})p(rule(e^k_{i,j})) \prod_{n^n_{l,m}} \in rhs(e^k_{i,j})\beta(n^n_{l,m})/p(w_{1,n}) \tag{6}$$

With the above equations, the figure of merit for pruning the forest is as follows:

$$F(e_i) = p(e_i|w_{1,n})p(\theta(i)|e_i) \tag{7}$$

The first term is computed using eqn. 4 and the second term from the translation model. The method was tested on 347 unseen Chinese newswire sentences. The system was compared against two systems, one of which uses a syntax-based translation model and a standard word-trigram LM and the other system that uses a simple statistical translation model with a standard word-trigram LM. The BLEU score of the proposed method was found to be worse than the scores obtained with the other systems. However, the system was able to obtain more number of perfect translations than the other two systems (45 versus 26 on of the other systems). The system also produced more grammatical output (112 versus 37 on one of the systems).

2.2 Incorporating Syntax and Semantics into LMs

Many of the structured LMs incorporate syntax and semantics via dependency grammar which expresses the relations between words by a directed graph. [Chelba et al. 1997] proposed a headword dependency model where the model turned out to be infeasible and due to the non-availability of a left to right parser, they adopted an N-best rescoring strategy to test the model's performance (Section 2.2.1a). They later proposed a method to perform left to right mannered parsing to decode word lattices [Chelba et al. 1998] (Section 2.2.1b). [Bod 2000] modeled headword and non-headword dependencies to compute the most probable string from a word-graph(lattice) (Section 2.2.2a). A highly lexicalized probabilistic LM was proposed in [Wang et al. 2002] where high levels of word prediction capability was achieved by tightly integrating knowledge of words, structural constraints, morphological and lexical features at the word level (Section

2.2.2b). The model was an almost parsing LM similar to class-based LMs where SuperARV tag sequences were used instead of classes. The method was extended to a full parsing module in [Wang et al.2007] (Section 2.2.2c).

2.2.1 Modeling headword dependencies

a) N-best rescoring strategy

A maximum entropy based language model that incorporated both syntax and semantics via a dependency grammar was proposed by [Chelba et al. 1997]. The model incorporates the predictive power of words that lie outside of a bigram or trigram range. Dependency grammars express the linguistic structure of a sentence in terms of a planar, directed graph where two related words are connected by a graph edge which bears a label that encodes their relationship. The features that were considered as history included a finite context, consisting of 0, 1 or 2 preceding words and a link stack, consisting of the unconnected links at the current position and the identities of the words from which they emerge. The link stack carries the grammar that constrains the disjunct that can appear in the next position, since the left links of the disjunct must match some prefix of the stacked links. This formulation helps in discarding words and grammatical structure that are irrelevant to the prediction at hand.

Since the number of possible futures is too large, they moved the sequence of disjuncts into the model's history. To obtain the required parse K of an utterance S, they used the dependency parser of Michael Collins. The parser required part of speech tags and did not operate in left-to-right fashion, hence, they adopted an N-best rescoring strategy. The maximum entropy tagger of Adwait Ratnaparkhi was used for tagging. Training and testing data were drawn from the Switchboard corpus of spontaneous conversational English Speech and from the Treebank corpus. They trained the tagger using 1 millions words of hand-tagged training data. Then they applied the trained tagger to 226,000 words of hand-parsed training data. The parser was trained with these automatically-tagged, hand-parsed sentences. The trained tagger and parser were finally applied to 1.44 million words which included the tagger and parser training data. This collection of sentences formed the training data for their dependency language models from which they extracted features and the probabilities. 100 best hypotheses for each utterance A were generated using the HTK software. For each hypothesis S, the best tag sequence (T^*) is computed, then from S and T^* the best possible disjunct sequence D^* is found using the parser. $P(S)$ is then computed as a geometrically averaged quantity,

$$P(S) = P(n)^{\alpha} P(T|n,S)^{\beta} P(D|T,n,S)^{\gamma} P(S|D,T,n)^{\delta} \tag{8}$$

the weights α, β, γ and δ are experimentally determined weights. $P(n)$ is an insertion penalty which penalizes the decoding of an utterance as a sequence of many short words. They achieved reduction in WER from 47.5% (using an adjancency bigram model which is the conventional word-based bigram model) to 46.4%

b)Full parsing in left-to-right fashion allowing decoding of word lattices

A language model was developed by [Chelba et al. 1998] that used syntactic structure to model long-distance dependencies. Their model developed a syntactic structure incrementally traversing the sentence from left to right which allowed decoding word lattices. The model operates by means of three modules. Let W be a sentence of length n words to which $< s >$ and $< /s >$ are attached. Let W_k be the word k-prefix $w_0, ...w_k$ of the sentence and W_kT_k is the word-parse k-prefix that contains only those binary subtrees whose span is completely included in the word k-prefix excluding w_0=$< s >$.

(a)WORD-PREDICTOR: Predicts the next word w_{k+1} given the word-parse k-prefix and passes control to the tagger

(b)TAGGER: Predicts the POS tag of the next word t_{k+1} given the word-parse k-prefix and w_{k+1} and passes control to the parser
(c)PARSER: grows the already existing binary branching structure by repeatedly generating the transitions:
(unary,NTlabel): root only tree with the POS tag of the word as the non-terminal label (NTlabel)
(adjoin-left,NTlabel): new headword is inherited from its left daughter and the non-terminal label NTlabel is assigned to the newly built constituent
(adjoin-right,NTlabel): new headword is inherited from its right daughter and the non-terminal label NTlabel is assigned to the newly built constituent
until it passes control to the predictor by taking a null transition
The operations performed by the parser ensure that all possible binary branching parses with all possible headword and non-terminal label assignments for a word sequence are generated. A complete parse is any binary parse of $(w_1, t_1)...(w_n, t_n)(< /s >, SE)$ sequence with the restriction that $(< /s >, TOP')$ is the only allowed head. The probability of a word sequence W and a complete parse T is given as

$$\begin{aligned} P(W,T) &= \prod_{k=1}^{n+1} P(w_k|W_{k-1}T_{k-1}).P(t_k|W_{k-1}Tk-1, w_k) \prod_{i=1}^{N_k} P(p_i^k|W_{k-1}T_{k-1}, w_k, t_k, p_1^k, ...p_{i-1}^k) \\ &= \prod_{k=1}^{n+1} P(w_k|h_0, h_{-1}).P(t_k|w_k, h_0.tag, h_{-1}.tag) \prod_{i=1}^{N_k} P(p_i^k|W_{k-1}T_{k-1}, w_k, t_k, p_1^k, ...p_{i-1}^k) \end{aligned}$$

where, $W_{k-1}T_{k-1}$ is the word-parse $(k-1)$ prefix, w_k is the word predicted by WORD-PREDICTOR, t_k is the tag assigned to w_k by the TAGGER, $N_k - 1$ is the number of operations the PARSER executes before passing control to the WORD-PREDICTOR,p_i^k is the i^th PARSER operation carried out at position k in the word string. The word predictor model predicts the next word based on the preceding 2 exposed heads. The tagger model uses the word to be tagged and the tags of the two most recent exposed heads. The last term in model assigns probability to different parses of word k-prefix by chaining the elementary operations. The parameters are estimated by using an HMM re-estimation technique that works on pruned Nbest-trellises.

The state space of this model is huge and hence they used a synchronous multi-stack search algorithm. Each stack contains hypotheses (or partial parses) that have been constructed by the same number of predictor and the same number of parser operations. The hypotheses in each stack are ranked according to the $ln(P(W,T))$ score, highest on top. The width of the search is controlled by two parameters at a given state, the maximum stack depth (maximum number of hypotheses the stack can contain) and log-probability threshold (the difference between the log-probability score of the top-most hypothesis and the bottom-most hypothesis cannot be larger than a given threshold). An additional pruning is performed after all hypotheses in stage k' have been extended with the null parser transition.

Experiments were performed on the UPenn Treebank corpus. 1M words were used for parameter reestimation. The word vocabulary size was 10k and POS tag vocabulary of 40. The test set size was 82,430 words. A reduction in test set perplexity was achieved from 167.14 (with a trigram word-based model) to 158.28. Interpolating the model with a trigram model resulted in 148.90 (interpolation weight = 0.36 for the trigram model).

2.2.2 Modeling headword and non-headword dependencies

a) Select the best word sequence from a lattice(or word graph)
Rens Bod modeled non-headword dependencies into structured language models called the Data Oriented Parsing model (DOP) and showed that non-headword dependencies contribute significantly to improved

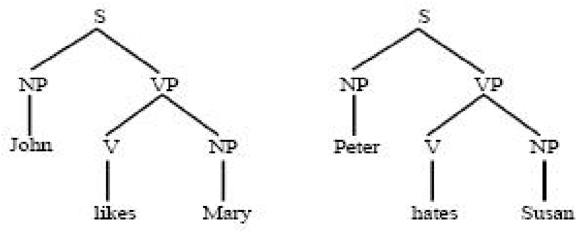


Figure1: Treebank

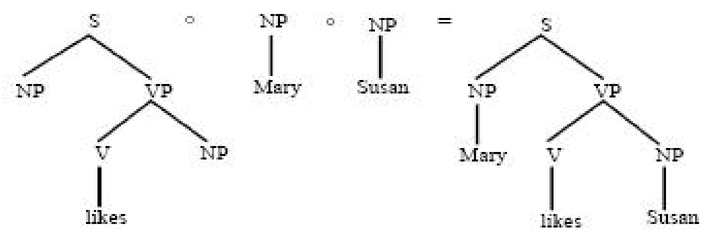


Figure2: Tree Derivation

word error rate. The head-lexicalized grammar model presented in the previous section cannot capture dependencies between non-headwords such as “more” and “than” in phrases like “more people than cargo” or “more couples exchanging rings in 1988 than in the previous year”, where neither more nor than are headwords.

The DOP model learns a stochastic tree-substitution grammar (STSG) from a treebank by extracting all subtrees and assigning probabilities that are proportional to their empirical treebank frequencies. Subtrees are lexicalized at their frontiers with one or more words to take both headword and non-headword dependencies into account. For example, the dependency between “more” and “than” in the previous example can be captured by having a subtree with “more” and “than” as frontier words. A substitution operation is performed to combine subtrees into parse trees. If a treebank contains the trees shown in Fig. 1, new sentences such as “Mary likes Susan” can be derived by combining subtrees from this treebank as in Fig. 2. The probability of a parse tree is the sum of the probabilities of all derivations D that generate T. To compute the probability of a word string W, the probabilities of all parse trees that yielded W were summed. The derivations of a parse tree T is viewed as a state trellis where each state contains a partially constructed tree in the course of a leftmost derivation of T. The subtree probabilities are estimated by a maximum likelihood reestimation procedure related to EM. The EM algorithm applied to this model is similar to the Inside-Outside algorithm for context-free grammars. The initial state s_0 is a tree with depth zero consisting of a root labeled with S. The final state s_T is the given parse tree T. A state s_t is connected forward to all states s_{tf} such that $t_f = tot'$ for some t' (defined by $t_f - t$). A state s_t is connected backward to all states s_{tb} such that $t = t_b o t'$ for some $t'(t - t_b)$. The forward probability αs and backward probability βs of a state are computed recursively,

$$\alpha(s_t) = \sum_{s_{tb}} \alpha s_{tb} P(t - t_b) \text{ and } \beta(s_t) = \sum_{s_{tf}} \alpha s_{tf} P(t_f - t) \tag{9}$$

The update formula for the count of a subtree is as given below. r(t) is the root label of t.

$$ct(t) = \sum_{s_{tb}:\exists s_{tf}, t_b o t = t_f} \beta(s_{tf})\alpha(s_{tb})P(t|r(t))/\alpha(s_{goal}) \tag{10}$$

The update probability distribution is given by,

$$P'(t|r(t)) = ct(t)/ct(r(t)), \text{ with } ct(r(t)) = \sum_{t':r(t')=r(t)} ct(t') \tag{11}$$

Performance was tested on the OVIS corpus which consists of 10,000 user utterances and corresponding word-graphs about Dutch public transport information that are syntactically and semantically annotated. The pre-lexical nodes contain the meanings of the underlying lexical items while the higher nodes contain

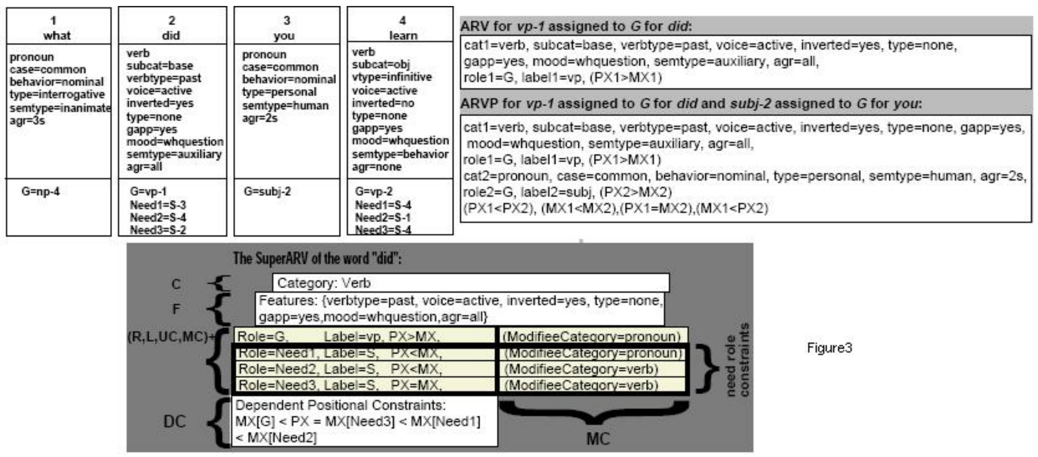

1
what
pronoun
case=common
behavior=nominal
type=interrogative
semtype=inanimate
agr=3s
G=np-4
2
did
verb
subcat=base
verbtype=past
voice=active
inverted=yes
type=none
gapp=yes
mood=whquestion
semtype=auxiliary
agr=all
G=vp-1
Need1=S-3
Need2=S-4
Need3=S-2
3
you
pronoun
case=common
behavior=nominal
type=personal
semtype=human
agr=2s
G=subj-2
4
learn
verb
subcat=obj
vtype=infinitive
voice=active
inverted=no
type=none
gapp=yes
mood=whquestion
semtype=behavior
agr=none
G=vp-2
Need1=S-4
Need2=S-1
Need3=S-4
ARV for vp-1 assigned to G for did:
cat1=verb, subcat=base, verbtype=past, voice=active, inverted=yes, type=none,
gapp=yes, mood=whquestion, semtype=auxiliary, agr=all,
role1=G, label1=vp, (PX1>MX1)
ARVP for vp-1 assigned to G for did and subj-2 assigned to G for you:
cat1=verb, subcat=base, verbtype=past, voice=active, inverted=yes, type=none, gapp=yes,
mood=whquestion, semtype=auxiliary, agr=all,
role1=G, label1=vp, (PX1>MX1)
cat2=pronoun, case=common, behavior=nominal, type=personal, semtype=human, agr=2s,
role2=G, label2=subj, (PX2>MX2)
(PX1<PX2), (MX1<MX2),(PX1=MX2),(MX1<PX2)
The SuperARV of the word "did":
C
Category: Verb
F
Features: {verbtype=past, voice=active, inverted=yes, type=none,
gapp=yes,mood=whquestion,agr=all}
(R,L,UC,MC)+
Role=G, Label=vp, PX>MX, (ModifieeCategory=pronoun)
Role=Need1, Label=S, PX<MX, (ModifieeCategory=pronoun)
Role=Need2, Label=S, PX<MX, (ModifieeCategory=verb)
Role=Need3, Label=S, PX=MX, (ModifieeCategory=verb)
need role constraints
DC
Dependent Positional Constraints:
MX[G] < PX = MX[Need3] < MX[Need1]
< MX[Need2]
MC

Figure3

a formula scheme indicating how the meaning of the constituent is built up out of the meanings of its subconstituents. The most probable string computation employed Viterbi n best search and the most probable string was estimated by the 1000 most probable derivations. If the DOP does not find a derivation for the complete word-graph, the 1000 most probable combinations of subderivations of partial paths is computed. The OVIS corpus was split into 10 random divisions of 90% training and 10% testing data. Subtrees of depth upto 4 were extracted. The word error rate (WER) for each model was averaged over all test sets. The DOP model outperformed a 3-gram model built out of the same training data by 1.7% WER. A version of DOP was tested without semantic annotations and this led to a deterioration in WER by 1.3% which implies that semantic annotations contribute signficantly towards the performance of the model.

b) Lattice rescoring using an almost-parsing algorithm

An almost-parsing LM that incorporated multiple knowledge sources was presented in [Wang et al. 2002] based on the concept of Constraint Dependency Grammars (CDG). Lexical features and syntactic constraints are tightly integrated into a linguistic structure called SuperARV that is associated with a word in the lexicon.

CDG represents a parse as assignments of dependency relations to functional variables (roles) associated with each word in a sentence. Figure. 3 shows a parse for "What did you learn". Each word has a lexical category and a set of feature values. Each word has a governor role (G) assigned a role value which is comprised of a label and a modifiee indicating the position of the word's head. For example, the role value assigned to the governor role of "did" is "vp-1", where, vp is the grammatical function, 1 indicates the position of its head (i.e., "what"). The need roles (N1,N2,N3) are used to ensure the grammatical requirements of a word are met. For example, "did" needs a subject and a base form verb, since the word takes no other complements, the modifiee of the role value assigned to N3 is set equal to its own position. Need roles provide a mechanism for using the non-headword dependencies to constrain the parse structures. The grammaticality of a sentence is determined by applying a set of constraints to the role value assignments. Constraints are derived from CDG annotated sentences, where grammar relations are extracted using information derived from annotated sentences. Using the relationship between a role value's position and its modifiee's position, unary and binary constraints can be represented as finite set of abstract tole values (ARVs) and abstract role value pairs (ARVPs) as shown in Figure. 3. A SuperARV is an abstraction of the joint assignment of dependencies for a word which provides a way for lexicalizing CDG parse rules. SuperARVs encode lexical information as well as syntactic and semantic constraints which makes it more fine-grained than POS. SuperArvs provide admissibility constraints on syntactic and lexical environments in which a word can be used. A SuperARV is defined as a four-tuple, $< C, F, (R, L, UC, MC)+, DC >$, where,

C is the lexical category of the word,
$F = Fname_1 = Fvalue_1, ...FName_f = FValue_f$ is a feature vector, with $Fname_i$ as the name of a feature with $Fvalue_i$ as its value,
$(R, L, UC, MC)+$ is a list of one more tuples, each representing an abstraction of a role value assignment, with R as the role value, L as functionality label, UC as the relative position relation of a word and its dependent.
MC is the lexical category of the modifiee and DC represents the relative ordering of the positions of a word and all its modifiee. MC imposes modifiee constraints hence results in efficient pruning while parsing. Words can have more than one SuperARV to indicate different types of word usage. Verbs have the greatest SuperARV ambiguity due to the variety of feature combinations and variations on complement types and positions. It was found that the number of SuperARVs does not grow significantly as training size increases.

Constituent bracketing found in treebanks are automatically transformed into CDG annotations and dependency structures are generated using headword percolation [Chelba et al. 1998]. The lexical features and need role values are determined by a rule-based approach. SuperARVs are accumulated from the CDG annotations and stored directly with words in a lexicon so that frequency of occurrence for the corresponding word can be learnt. The SuperARV LM estimates the joint probability of words w_1^N and their SuperARV tags t_1^N as given below. The SuperARV LM does not encode the word identity directly as this could cause data sparsity problems.

$$P(w_1^N t_1^N) = \prod_{i=1}^{N} P(t_i | w_{i-2}^{i-1} t_{i-2}^{i-1}).P(w_i | w_{i-2}^{i-1} t_{i-2}^{i}) \quad (12)$$

The probability distributions are estimated using recursive linear interpolation among probability estimations of different orders. Best performace was achieved by using the modified Kneser-Ney smoothing algorithm and adapting it by using a heldout data set to optimize parameters (including cutoffs for rare n-grams). Parameters are chosen to optimize the perplexity on a heldout set. The training set of the WSJ CSR task consists 37,243,300 words. Acoustic model is built out of the speech data. Testing was performed on WSJ CSR 5k and 20k test sets. For each test set sentence, a word lattice was generated. The parameters of the LM were tuned using the lattices of the development sets to minimize WER. Lattices were rescored using Viterbi search. The paper reports reduction in perplexities. The best performance was seen on the 20k test set, with SuperARV achieving 14.28% WER compared to a 3-gram word based model with 14.74% and Chelba's model with 14.36%.

c) Full-parsing for reranking MT hypotheses

Wang et al., used a lingustically motivated and computationally efficient structured language models for reranking N-best hypotheses in a Statistical Machine Translation (SMT) system. Two structured language models were applied for N-best rescoring. One is the almost parsing language model (Wang et al., described in the previous section) and the other is the full parsing language model which utilizes more syntactic features by explicitly modeling syntactic dependencies between words.

The translation system consists of two passes of decoding. The first pass uses a hierarchical phrase decoder developed at SRI to perform integrated decoding with a standard 4-gram LM to generate N-best lists. The phrases and hierarchical rules were extracted from parallel corpora. The second pass rescores the N-best lists using several LMs trained on different corpora and estimated in different ways. The scores are combined in a log-linear modeling framework along with other features like, rule probabilities $p(f|e)$, $p(e|f)$, lexical weights $pw(f|e)$, $pw(e|f)$, sentence length and rule counts. The weights were optimized using minimum error training method to maximize BLEU scores using Amoeba simplex search on N-best

lists.
SuperARV-Almost-parsing LM: The SuperARV (described in Section 2.2.2b) in almost parsing is fundamentally a class-based LM using SuperARVs as classes. The modeling of the SuperARV LM was modified in two ways. Firstly, numbers were mapped to "$number" during preprocessing on the parallel data and other target language model training data. Valid CFG parse trees were generated in the original word formats of numbers and then the numbers were mapped to "$number"' in the parse trees for training the structured LMs. Secondly, unlike in ASR, punctuation is present in the MT N-best hypotheses and since punctuations provide important syntactic information for SMT, punctuation marks were categorized into sentence-final punctuation(period,question mark,exclamation) and intra-sentence punctuation and were discriminated in modeling dependencies between punctuation and word tokens. The root of a sentence was selected to be the headword for sentence-final punctuation marks. For intra-sentence punctuation marks, they were treated similar to coordination by defining the headword of the following phrase as the headword of the punctuation mark.
Parser LM: The parser is a probabilistic generative model and consists of two components: SuperARV tagging and modifiee determination. First, the top N-best SuperARV assignments are generated for an input sentence. Each SuperARV sequence is represented as a sequence of tuples: $< w_1, s_1 >, , , , < w_n, s_n >$ where w_k represents the word and s_k its superARV assignment. The assignments are stored in a stack ranked in non-increasing order by tag assignment probability. In the second step, the modifiees are statistically specified in a left-to-right manner. The algorithm uses modifiee lexical category to filter out candidates with mismatched lexical categories. The algorithm statistically attempts to determine the left dependents of w_k from the closest to the farthest. It also determines whether w_k could be the $(d+1)^{th}$ right dependent of a previously seen word w_p, $p = 1, ...k-1$, where d denotes the number of already assigned right dependents of w_p. After processing word w_k in each partial parse on the stack, the parses are re-ranked according to their updates probabilities. The procedure is iterated until the top parse in the stack covers the entire sentence. Two pruning thresholds are used to control time and memory complexity, maximum stack depth and maximum difference between the log probabilities of the top and bottom partial parses in the stack tuned on a heldout set based on parsing accuracy.

This full parser utilizes statistics of dependencies between all pairs of words and hence the computational complexity is high. Hence, a baseNP model was used to generate a reduced form of a sentence by first marking all baseNPs and then reducing the baseNPs to their headwords. A baseNP is a non-recursive NP such that none of its children constituents are NPs. For example, "Mr. Viken is chairman of the Elsevier N.V., the Dutch publishing group" will be reduced to "Viken is chairman of N.V., group", hence, words internal to baseNPs are not used for training and the efficiency of the LM is significantly improved. The baseNP model is a tagger that tags the boundaries between words using tags from the set S : denoting whether the boundary is at the start of a baseNP, C : continues a baseNP, E : is at the end of a baseNP, B : is between two adjacent baseNPs, or N : is between two words neither of which belongs to any baseNPs. The full-parser is further modified as follows, given T as a dependency annotation for a sentence W, the probability of W is computed as:

$$P(W) = \sum_T P(W, T) = \sum_T P(W, S, B, D) = \sum_T P(D|B, S, W)P(B|S, W)P(S, W) \tag{13}$$

where, S is the SuperARV tag sequence, B is the baseNP model and D is the dependency relations. With Viterbi approximations and independence assumptions,

$$P(W) = P'(D|S')P'(B|W)P'(S, W) \tag{14}$$

Hence, for an input sentence W, the best SuperARV sequence is found and then the best baseNP sequence for it is found. Then based on the reduced sentence W' and the corresponding SuperARV tags S', the best dependency relations that maximize $P(D|S')$ is found using stack decoding.

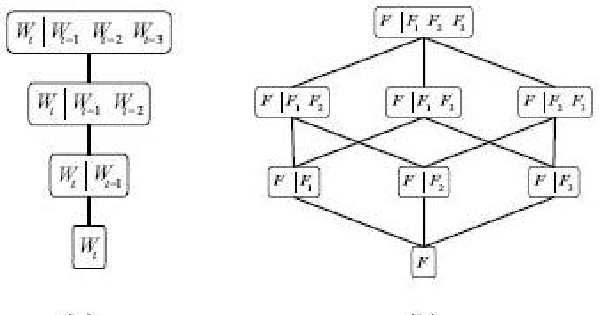
A word-based 4-gram LM was used for searching. The LM was trained on text sources with 5 billion tokens. The text sources were clustered into 4 categories:
1. The English side of Arabic-English(AE) and Chinese-English(CE) parallel data provided by LDC which contains 270 million tokens for CE and 280 million tokens for AE.
2. English BN and BC transcriptions, webtext and translations for mandarin and Arabic BN and BC released under the DARPA EARS and GALE programs which contained 260 million tokens.
3. English Gigaword corpus (LDC2005T12) with 2.5G tokens, Northe American News text corpora (LDC95T21 and LDC98T30) with 620 million tokens.
4. 30 million tokens of English news articles, BBN's web data collected from news websites with 800 million tokens, 300 million tokens by crawling the "internet archive" and downloading past copies from news websites.

For N-best reranking, the word-based 4-gram LM(4g), the almost-parsing LM (sarv), the parser LM (plm), the 5-gram count-LM (during weight estimation and testing, the model needs access to only those n-grams occurring in the respective data sets which allows for fast operation with limited memory) trained on Google N-grams (google), the modified Kneser-Ney smoothed 4-gram LM trained on the Yahoo N-grams (yahoo), the 5-gram count-LM trained on all BBN web data (wlm) were used. The almost parsing LM was trained on 1. and 2., the full parser LM was trained on 2. The N-best list size was set to 3,000. The LM scores are combined in a log-linear framework with weights optimized on the GALE dev07 test set and tested on the blind test set, NIST eval06 GALE subset in the 2006 NIST MT evaluation. Weights for knowledge sources were optimized on the combined set of NW and WT for the text MT and, BN and BC for the audio MT.

Three variations of LMs were trained on the Yahoo n-grams, a 4-gram LM using modified KN smoothing after extrapolation (yahoo-4g) and its 5-gram version (yahoo-5g) and a 5-gram count-LM using all the Yahoo n-grams (yahoo-5g, count-lm). These were used for reranking dev07 and eval06 (blind set) nbests. On dev07 BN, the 5-gram count-LM using Yahoo n-grams produced more improvement than the modified KN smoothed Yahoo 4-gram but it was the opposite on BC. The rescoring schemes for the modified KN smoothed yahoo 4-gram and 5-gram were compared using a dynamic (weight=0.6 for yahoo-4g) and static interpolation of them during rescoring or using them in the log-linear framework. On dev07 BN, the log-linear scheme yields the best improvement while on BC the static interpolation approach was the best. As the gain was mainly from yahoo 4-g (KN), yahoo-4g was used for reranking. The performance of almost-parsing LM was also tested. The reranking effect from statically and dynamically interpolating LMs were compared. Dynamic interpolation produced improvement on dev07 BN and not on BC. The observation was reversed on eval06. BLEU scores from the baseline (no reranking) and using various LMs for reranking on dev07 and eval06 on AE and CE were also compared. On all the experiments, improvements were seen when combined with structured LMs against the baseline.

3 Factored Language models

FLMs allow a larger set of conditioning variables for predicting the current word. Any number of variables like morphological features, syntactic features or semantic features can be included. Hence, a word is



(a) (b)

Standard backoff path for a 4-gram LM over only words (a), backoff graph for factors (b)

equivalent to a fixed number of K factors, $W = f^{1:K}$. The resulting model is as follows,

$$p(f_1^{1:K}, f_2^{1:K}, ...f_T^{1:K}) \approx \prod_{t=3}^{T} p(f_t^{1:K} | f_{t-1}^{1:K}, ...f_{t-2}^{1:K}) \tag{15}$$

If a word is decomposed into lexemes and POS tags, then its factored word representation is as shown below,
Word: Stock prices are rising
Stem: Stock price be rise
Tag: Nsg N3pl V3pl Vpart
This factored representation can be used in two different ways to improve over standard LMs, by using a product model or a backoff model. In a product model, eqn. 15, can be simplified by finding conditional independence assumptions among subsets of conditioning factors and computing the desired probability as a product of individual models over those subsets. [Kirchhoff et al. 2005] uses the factors in a backoff procedure when the word n-gram is not observed in the training data. However, the space of possible models grows extremely large as now there are many ways of choosing subsets of conditioning word features, backoff procedures and discounting methods. This space definitely cannot be searched exhaustively, and optimizing models by linguistic based/knowledge-inspired manual search procedure leads to suboptimal results. [Kevin et al. 2004] investigates the procedure of determining the structure of these models by a data-driven search procedure called Genetic Algorithms.

3.1 Generalized Parallel Backoff

In the standard backoff, the most distant conditioning variable is dropped first then the second most and so on until the unigram in reached. If the only variables in the model are words, such a backoff procedure is simple. Since here the variables occur in parallel, it is not obvious in which order they should be dropped. A fixed predetermined backoff path based on linguistic knowledge can be used or the path can be chosen at run-time based on statistical criteria or we could choose multiple paths and combine their probability estimates.

$$p_{GBO}(f_t|f_{t-1}, f_{t-2}) = \begin{cases} d_c p_{ML}(f_t|f_{t-1}, f_{t-2}), & \text{if count of } (w_t, w_{t-1}, w_{t-2}) > \tau_4 \\ \alpha(f_{t-1}, f_{t-2}) g(f_t, f_{t-1}, f_{t-2}), & \text{otherwise} \end{cases} \tag{16}$$

where,
d_c= discounting factor applied to the higher-order distribution
$g(f_t, f_{t-1}, f_{t-2})$ determines the backoff strategy.
$\alpha(f_t, f_{t-1}, f_{t-2})$= normalization factor ensures that the entire distribution sums to one.

3.2 Learning FLM structure using Genetic Algorithm

Genetic algorithms (GAs) are a class of evolution-inspired search techniques that perform well in problems with complex and poorly understood search spaces. They encode problem solutions as strings and evolve and test successive populations of solutions through genetic operators applied to encoded strings. Solutions are evaluated according to a fitness function which represents the desired optimization criterion. The genetic operators include the "selection" of strings for the next generation, "crossover" (exchanging subparts of different strings to create new strings) and "mutation" (randomly altering individual elements in strings). GAs provide no guarantee of finding the optimal solution but they often find good solutions quickly.

The initial set of conditioning factors are encoded as binary strings. Eg, a trigram for a word representation with 3 factors (A,B,C) has 6 conditioning variables $A_{-1}, B_{-1}, C_{-1}, A_{-2}, B_{-2}, C_{-2}$ which is represented as a 6-bit binary string. The string 10011 would correspond to $F = A_{-1}, B_{-2}, C_{-2}$

The Backoff graph is encoded as a binary string in terms of graph grammar rules. For example, a node with m factors can only backoff to nodes with $m - 1$ factors. For $m = 3$, the following rules can be used:
RULE1:$x_1, x_2, x_3 \rightarrow x_1, x_2$
RULE2:$x_1, x_2, x_3 \rightarrow x_1, x_3$
RULE3:$x_1, x_2, x_3 \rightarrow x_2, x_3$
RULE4:$x_1, x_2 \rightarrow x_1$
RULE5:$x_1, x_2 \rightarrow x_2$
A gene 10110 would indicate applying RULE1, RULE3 and RULE4. The choice of rules used to generate the backoff graph is encoded as a binary string. If two different rules are present at the same level, it corresponds to parallel backoff.

Smoothing options are encoded as tuples of integers. The first integer specifies the discounting method and the second integer specifies th backoff threshold. The string consists of successive concatenated tuples where each represents the smoothing option at a node.

The substring for Conditioning factors, backoff graph and smoothing options are concatenated and GA operators are applied on this concatenated string.

3.3 FLMs applied to MT

In [Kirchhoff et al. 2005] a simple baseline system was trained using standard tools, GIZA++ for alignments and Pharaoh for phrase-based decoding. The training data was from the ACL05 Shared MT task website for 4 different languages pairs- Finnish, Spanish, French into English. Pharaoh was run in N-best mode to produce N-best lists with 2000 hypotheses per sentence. The N-best lists were then rescored with additional LMs. The resulting scores were combined with scores obtained from Pharaoh in a log-linear fashion to obtain the final 1 best hypotheses. Two models were used in the second pass: 4-gram word-based LM and a Factored 3-gram model. Tags and stems were used for the FLM. The FLMs were optimized to achieve a low-perplexity on the oracle 1-best hypotheses. This was done to avoid optimizing the model on word combinations that might never be hypothesized in the first pass. The scores were almost the same with the 4-gram model and with FLM on Finnish to English and French to English but the scores obtained with FLMs were worse than the 4-gram model on Spanish to English. When the two models were combined, a small improvement was noticed (from 22.2 to 22.3 on Finnish to English and from 30.2 to 30.4 on French to English) on the development set of 2000 sentences. No results with the FLMs on the evaluation sets were reported as the improvement on the dev set was not big enough.

[Axelrod 2006] used the same data that was used by [Kirchhoff et al. 2005] to test their system which integrated the factored language models into the Pharaoh Decoder to show the effectiveness of integrated factored language models on Statistical MT. The Pharaoh decoder was modified to use the SRILM FLM. The translation model proposes English translations that look like the factored corpus. The Pharaoh decoder's representation of a word was altered. Pharaoh was also modified to use SRILM tools to assemble feature bundles into a context matrix which contains features from the preceding n-gram arranged into an array that the FLM can use to find the likelihood of a factored n-gram. The performance of an FLM that backed off linearly over the available factors dropping the oldest one in sequence was compared with a word-based trigram model. The model backs off linearly from word to stem to POS tag within each word bundle and drops the oldest bundle first. With minimum error rate training(MERT), the FLMs performed slightly better than the word-based model with FLM scoring 30.72 BLEU and trigram LM scoring 30.59 BLEU.

4 Conclusion

From the results presented in this report, it is clear that Structured LMs and FLMs are promising techniques when applied in the right way. There are a number of reasons for not getting the best improvement possible in many of these systems. Although ASR and MT are very different tasks both use LMs to produce grammatically correct hypotheses. Hence, LMs in the ASR domain are being borrowed and applied to MT without any significant changes. ASR is interested in producing the right text for a given translation however MT looks for a correct translation of a sentence. In cases where the source language and target language have very different word orders, the search space to find the best hypothesis in the MT domain is huge because of which a lot more constraints on the decoder are used while finding the best MT hypothesis. Hence, it is not guaranteed that an LM that helps in boosting the overall score in ASR is truly helpful in the MT domain when used with a highly constrained decoding strategy. Linguistically motivated LMs described in this report tend to be computationally expensive, hence, integrating these LMs into MT is not straightforward and more research is required in developing clever techniques to integrate these computationally expensive LMs into an already expensive MT task.

Research on LMs in the ASR domain is very active and it tends to look at languages like English, French, etc, where large amount of data is available and these languages are not morphologically rich. Recent MT research involves translating languages that are very different from English (Eg. Chinese) and morphologically rich languages (Eg. Arabic). This makes LMs applied in MT sparser making any off-the-shelf LMs less effective. MT should use LMs that are specifically tailored to the the translation task with different language-pairs and with better smoothing techniques.

Due to the non availability(or very few) of easily modifiable publicly available decoders (either due to licensing restrictions or due to poor code documentation) and due to fact that building decoders in MT is expensive, MT research has concentrated more on building better Translation Models (TMs) and good data preprocessing strategies and use publicly available decoders to find the best hypothesis. Without being able to integrate richer LMs into an MT decoder, richer LMs are applied in rescoring N-best lists. Rescoring startegies are helpful but are constrained by the quality of the decoder's initial hypotheses.

LMs are evaluated based on their perplexity on a test set. Lower perplexity could result in lower error rates but it is not true in all cases. In the MT domain, BLEU is used as the standard metric to judge the performance of these systems. Although BLEU has shown to have weak correlations in some cases, it is still used as a standard metric in many MT systems for tuning and scoring. Hence we could blame partly on not having a good scoring technique.

TM and LM scores are combined in a log-linear framework as it is the easiest way to combine scores. May be there is a better technique to combine these scores. The system parameter tuning occurs after the TM and LM have been seperately built and trained. Tuning repeatedly tests model weight combinations

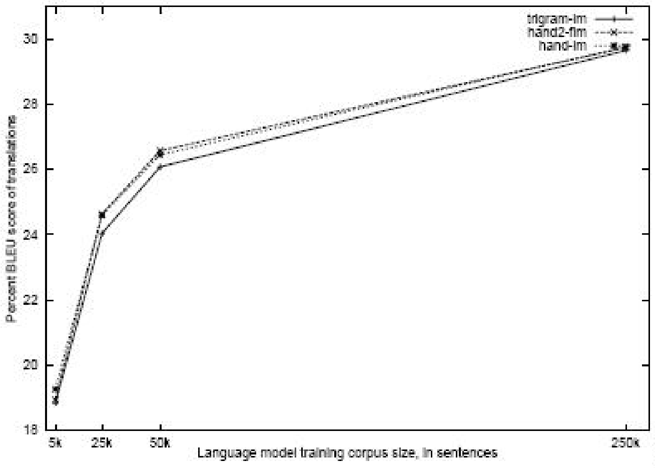


Figure 4

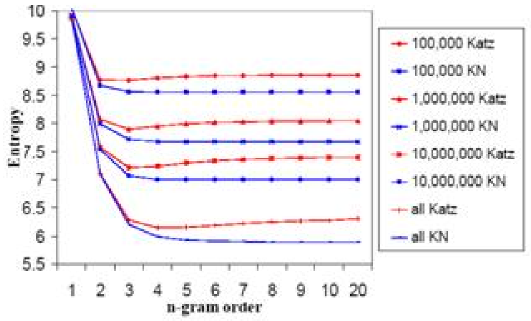


Figure 5

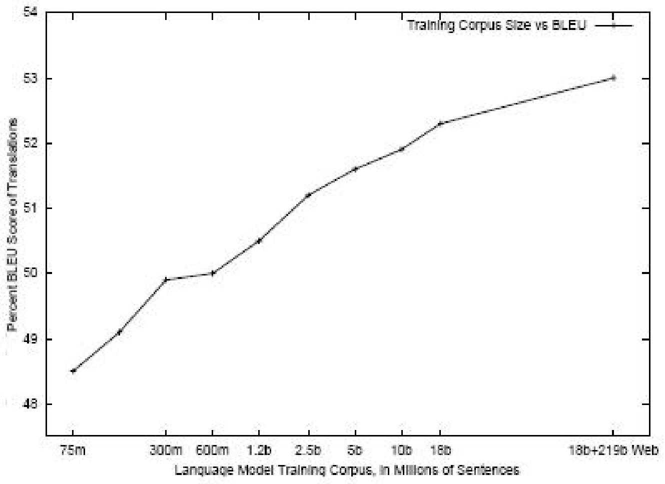


Figure 6

with system output scores and adjusts them accordingly. The tuning step is used to maximize the assembled system's output where it is able to compensate for the weakness of individual components by increasing the weight assigned to the other parameters. The performance at each step is based on its BLEU score. Tuning these systems in a log-linear fashion and finding the best parameters based on BLEU score may not be the best strategy.

[Axelrod 2006] showed that FLMs provide better improvement when translating sparse corpora. He performed experiments with smaller corpora using FLMs (Figure 4). The perplexity scores were found to be significantly better for the FLMs than the word-based trigram models. Three LMs were used for comparison, word-based trigram-lm, handlm and hand2-flm. Handlm model differs from the word-based trigram in that it uses generalized parallel backoff to estimate the probability of previously unseen trigram. Hand2-flm is an FLM with generalized parallel backoff constructed to mimic the structure of three automatically selected models (using GAs based on their perplexity scores) with linguistic knowledge and generalized parallel backoff, while fixing their shared failure to use all the factors available. Translation score were also improved by almost 0.6 percent BLEU with FLMs over the word-based models. These experiments clearly show that linguistic models are more effective on smaller corpora.

N-gram LMs are simple and have better smoothing techniques for probability estimations. Figure 5 shows the entropy values for different n-grams with different amounts of data and different smoothing strategies. From this report it was seen that many of the systems performed almost the same or slightly better with linguistically based models than the n-gram models. To achieve more improvement in BLEU score, the training size for building the word-based n-gram LM has to be increased by almost double the size. Figure 6 shows the improvement in BLEU score with different amounts of data. At some point, these models reach saturation where there is not much improvement even after doubling the amount of data. Improvement can only be seen by using higher order n-grams in such situations. However these models require more time and huge computing requirements. Using higher order n-grams may not be a good strategy. By using complicated linguistic models on large amounts of data is even more computationally expensive. The question still remains if the valuable time spent on improving and building complex and computationally expensive linguistically oriented language models beyond the current n-gram approach would be better spent on improving other parts of the MT system.

References

[Charniak et al. 2003] Eugen Charniak, Kevin Knight and Kenji Yamada. 2003. *Syntax-based Language Models for Statistical Machine Translation*. In Proc. of MT Summit IX.

[Rosenfeld 1994] Ronald Rosenfeld 1994 *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD Thesis.

[Kirchhoff et al. 2005] Katrin Kirchhoff and Mei Yang *Improved Language Modeling for Statistical Machine Translation*. In ACL.

[Kevin et al. 2004] Kevin Duh and Katrin Kirchhoff *Automatic Learning of Language Model Structure*. In Proc. of the 20th international conference on Computational Linguistics.

[Bilmes et al. 2003] J.A. Bilmes and K. Kirchhoff 2003. *Factored Language models and generalized parallel backoff*. In Proc. of HLT/NAACL, pages 4-6.

[Axelrod 2006] Amittai E. Axelrod 2006. *Factored Language models for Statistical Machine Translation*. Master of Science Thesis.

[Brown et al. 1990] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, Jenifer C. Lei 1990. *Class-Based n-gram models of Natural Language*. Computational Linguistics.

[Heeman 1998] Peter A. Heeman 1998 *POS Tagging versus Classes in Language Modeling*. In Sixth Workshop on Very Large Corpora, pg. 179-187.

[Chelba et al. 1997] Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, Dekai Wu. 1997 *Structure and performance of a Dependency Language Model*. In Proc. Eurospeech '97.

[Bod 2000] Bod, R., 2000. *Combining semantic and syntactic structure for language modeling.* In: Proc. of the International Conference on Spoken Language Processing, vol. 3, Beijing, China, pp. 106109.

[Chelba et al. 1998] Ciprian Chelba and Frederick Jelinek 1998 *Exploiting syntactic structure for language modeling*. In ACL.

[Wang et al. 2002] W. Wang and M. Harper 2002 *The SuperARV Language Model: Investigating the effectiveness of tightly integrating multiple knowledge sources*. In Proc. of Conference of Empirical Methods in Natural Language Processing.

[Wang et al.2007] Wen Wang, Andreas Stolcke, Jing Zheng 2007 *Reranking Machine Translation hypotheses with Structured and Web-based Language Models*. In Proc. IEEE Automatic Speech Recognition and Understanding Workshop, Kyoto.