Joint Morphological-Lexical Language Modeling for Processing Morphologically Rich Languages with Application to Dialectal Arabic

Ruhi Sarikaya, Mohamed Afify, Yonggang Deng, Hakan Erdogan, Yuqing Gao

Abstract— **Language modeling for an inflected language such as Arabic poses new challenges for speech recognition and machine translation due to its rich morphology. Rich morphology results in large increases in out-of-vocabulary (OOV) rate and poor language model parameter estimation in the absence of large quantities of data. In this study, we present a joint morphological-lexical language model (JMLLM) that takes advantage of Arabic morphology. JMLLM combines morphological segments with the underlying lexical items and additional available information sources with regards to morphological segments and lexical items in a single joint model. Joint representation and modeling of morphological and lexical items reduces the OOV rate and provides smooth probability estimates while keeping the predictive power of whole words. Speech recognition and machine translation experiments in dialectal-Arabic show improvements over word and morpheme based trigram language models. We also show that as the tightness of integration between different information sources increases, both speech recognition and machine translation performances improve.**

Index Terms— **Language Modeling, Maximum Entropy Modeling, Morphological Analysis, Joint Modeling.**

I. Introduction

There are numerous widely spoken inflected languages. Arabic is one of these highly inflected languages. In Arabic, affixes are appended to the beginning or end of a stem to generate new words. Affixes indicate case, gender, tense, number, and many other attributes that can be associated with the stem. Most natural language processing applications use word based vocabularies that are unaware of the morphological relationships between words. For inflected languages this leads to a rapid growth of the vocabulary size. For example, a parallel corpus (pairwise sentence translations) of 337K utterances between English and dialectal Iraqi Arabic has about 24K and 80K unique words for English and Iraqi Arabic, respectively.

A standard *n*-gram language model computes the probability of a word sequence, $W = \{w_1, \ldots, w_K\}$, as a product of conditional probabilities of each word given its history. This probability is typically approximated by *n-1* most recent words,

$$P(W) \approx \prod_{i=1}^{K} P(w_i | w_{i-1}, \ldots w_{i-n+1})$$

There is an inverse relationship between the predictive power and robust parameter estimation of *n*-grams. As *n* increases the predictive power increases, however due to data sparsity language model parameters may not be robustly estimated. Therefore, setting *n* to 2 or 3 appears to be a reasonable compromise between these competing goals. Robust parameter estimation problem is however more pronounced for Arabic due to its rich morphology compared to non-inflected languages. One would suspect that words may not be the best lexical units in this case and, perhaps, morphological units would be a better choice.

In addition to its morphological structure, Arabic has certain lexical rules for gender and number matching. For example, the adjective in *ftAh gydh* (*good girl* in English) differs from the same adjective in *wld gyd* (*good boy* in English) to match the gender, and also in *ftAtAn gydtAn* (*two good girls* in English) to match the number. By examining error patterns in speech recognition and machine translation outputs we observed that many sentences contain lexical mismatch errors between words. Using the above example, the utterance *wld gyd* might in some cases be recognized as *wld gydh*, where the correct adjective is replaced by the adjective of the wrong gender. This makes a lot of sense in speech recognition because many of the lexically mismatched items differ in only one phone and are thus acoustically confusable. Adding gender information in the language model could help in reducing these errors. This motivated us to introduce lexical attributes in the language model. Lexical attributes of the vocabulary, e.g. number, gender, and type are manually marked. These attributes will be discussed in more detail in Section V.

In this work, we present a new language modeling technique called Joint Morphological-Lexical Language Model (JMLLM) for inflected languages in general and Arabic

R. Sarikaya is the contact author and he is with IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA. (Tel: 914-945-3919, Fax: 914-945-4490; e-mail: sarikaya@us.ibm.com).

M. Afify was with IBM T.J. Watson Research Center when the work was carried out. He is now with Cairo University, Egypt.

Y. Deng and Y. Gao are with IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA.

H. Erdogan is with Sabanci University, Istanbul Turkey.

in particular and apply it to speech recognition and machine translation tasks. JMLLM models the dependencies between morpheme and word *n*-grams, attribute information associated with morphological segments[1] and words. These dependencies are represented via a tree structure called Morphological-Lexical Parse Tree (MLPT). MLPT is used by JMLLM to tightly integrate information sources provided by a morphological analyzer with the lexical information in a single joint language model. MLPT is a generic structure and it can include, if available, other information sources about the lexical items (i.e. lexical attributes, syntactic/semantic information), or the sentence (i.e. dialog state).

JMLLM is simply a joint distribution defined over the vocabularies of the leaves (morphemes in our case) and non-terminal nodes of the MLPT. In our implementation, we use a maximum entropy model to represent this joint distribution with a set of features given in Section V-D. Loosely speaking, this maximum entropy model can be viewed as an interpolation of distributions of the nodes of the tree and hence provides a desirable smoothing effect on the final distribution. JMLLM also improves the dictionary's coverage of the domain and reduces the out-of-vocabulary (OOV) rate by predicting morphemes while keeping the predictive power of whole words. This model statistically estimates the joint probability of a sentence and its morphological analysis.

In the above presentation of the model and also in the model description in Section V, to be precise, we restrict our discussion to a certain configuration of the tree. For example, we associate morphemes to leaves, and limit the internal tree nodes to the morphological attributes, lexical items and their attributes. However any sensible choice of the leaf nodes or internal tree nodes can be covered by the presented model. Even though in our implementation we use a deterministic parse provided by a rule based segmentation method, the proposed model also accommodates the case of probabilistic parses.

The rest of the paper is organized as follows. Section II provides an overview of prior work addressing language modeling for morphologically rich languages. Section III describes our morphological segmentation method. A short overview of maximum entropy modeling is given in Section IV. The proposed JMLLM is presented in Section V. Section VI describes the speech recognition and statistical machine translation (SMT) architecture. Experimental results and discussions are provided in Section VII, followed by the conclusions.

II. Relevant previous work

Recently, there has been a number of new studies aimed at addressing robust parameter estimation and rapid vocabulary growth problems for morphologically rich languages by using the morphological units to represent the lexical items [1,2,3,4,34]. Even though Arabic is receiving much of the attention, there are many other morphologically rich languages facing the same language modeling issues [25, 26, 30, 34]. In all of the mentioned studies above the use of morphological knowledge at the modeling stage is limited to only segmenting the words into shorter morphemes. In these models the relationship between the lexical items and morphemes is not modeled explicitly. Instead, two separate language models are built on the word based original corpus and segmented corpus and they are interpolated. However, in most of these studies morpheme sequence generation process in speech recognition or machine translation decoding is further constrained [4] by some rule based mechanisms exploiting the knowledge of the morphological segmentation algorithm. For example, if lexical items are segmented into one or more prefixes followed by a stem, which is also followed by one or more suffixes, then a suffix cannot follow a prefix without having a stem coming before it.

[1] We use "morphological segment" and morpheme interchangeably.

TABLE I
PREFIX AND SUFFIX LIST FOR DIALECTAL IRAQI ARABIC IN BUCKWALTER REPRESENTATION

Prefix List	chAl, bhAl, lhAl, whAl, wbAl, wAl, bAl, hAl, EAl, fAl, Al, cd, ll, b, f, c, d, w
Suffix List	thmA, tynA, hmA, thA, thm, tkm, tnA, tny, whA, whm, wkm, wnA, wny, An, hA, hm, hn, km, kn, nA, ny, tm, wA, wh, wk, wn, yn, tk, th, h, k, t, y

Factored Language Models (FLMs) [5, 14] are different than the previous methods and are similar to JMLLM to some extent. Unlike other approaches, in both FLM and JMLLM the relationship between lexical and morphological items are explicitly modeled within a single model. In an FLM words are decomposed into a number of features and the resulting representation is used in a generalized back-off scheme to improve robustness of probability estimates for rarely observed word *n*-grams. In an FLM, each word is viewed as a vector of k factors: $w_i = \{f_i^1, ..., f_i^K\}$. An FLM provides the probabilistic model $P(f \mid f_1, ..., f_N)$ where the prediction of factor *f* is based on *N* parents $\{f_1, ..., f_N\}$. For example, if *w* represents a word token and *t* represents a part-of-speech (POS) tag, the model, $P(w_i \mid w_{i-2}, w_{i-1}, t_{i-1})$, predicts the current word based on traditional *n*-gram model as well as POS tag of the previous word. The main advantage of FLMs compared to previous methods is that they allow users to put in linguistic knowledge to explicitly model the relationship between word tokens and POS, or morphological information. Like *n*-gram models, smoothing techniques are necessary in parameter estimation. In particular, a generalized back-off scheme is used in training an FLM. Our approach uses maximum entropy modeling as opposed to direct maximum likelihood modeling used in FLMs.

III. Morphological analysis

Applying morphological segmentation to data improves the domain coverage of the dictionary used for speech recognition or machine translation and reduces the OOV rate. Even though there is a large volume of segmented data available for Modern Standard Arabic (MSA), we do not know of any such data for training a statistical morphological analyzer to segment Iraqi Arabic language. In fact, Iraqi Arabic is so different than MSA, we are not aware of any study leveraging the MSA text resources to improve Iraqi Arabic language modeling or machine translation.

In this section, we present a word segmentation algorithm that is used to generate the morphological decomposition needed by the proposed language models. This algorithm was initially proposed in [4]. Starting from predefined lists of prefixes and suffixes (affixes) the segmentation algorithm decomposes each word in a given vocabulary into one of three possible forms: {prefix+stem, stem+suffix, prefix+stem+suffix}, or leaves it unchanged. Although affixes in Arabic are composite, i.e. a word can start (end) with multiple prefixes (suffixes), we found in preliminary experiments that allowing multiple affixes leads to a large insertion rate in the decoded output and results in worse overall performance. For this reason, we decided to only allow a single prefix and/or suffix for each stem. In our implementation, we use the sets of prefixes and suffixes given in Table I, in Buckwalter transliteration [6], for dialectal Iraqi Arabic.

The most straightforward way to perform the decomposition is to do blind segmentation using the longest matching prefix and/or suffix in the list. However, the difficulty with blind segmentation is that sometimes the beginning (ending) part of a word agrees with a prefix (suffix). This leads to illegitimate Arabic stems. For example, the word *AlqY*[2] (*threw* in English), a verb that should not be decomposed, has its initial part agreeing with the popular prefix *Al*. In this case blind segmentation leads to the decomposition *Al*-*qY* and hence to the invalid stem *qY*. In order to avoid this situation we employ the following segmentation algorithm. The algorithm still relies on blind segmentation but accepts a segmentation only if the following three rules apply:

(1) The resulting stem has more than two characters.

(2) The resulting stem is accepted by the Buckwalter morphological analyzer [6].

(3) The resulting stem exists in the original dictionary.

The first rule eliminates many of the illegitimate segmentations. The second rule ensures that the word is a valid stem in the Buckwalter morphological analyzer list. The Buckwalter morphological analyzer provides a decent coverage of Modern Standard Arabic (MSA). It was found experimentally that for a news corpus it only misses about 5% of the most frequent 64K words and that most of the missed words are typos and foreign names. Unfortunately, the fact that the stem is a valid Arabic stem does not always imply that the segmentation is valid. The third rule, while still not offering such guarantee, simply prefers keeping the word intact if its stem does not occur in the lexicon. The rationale is that we

[2] Using Buckwalter Arabic transliteration.

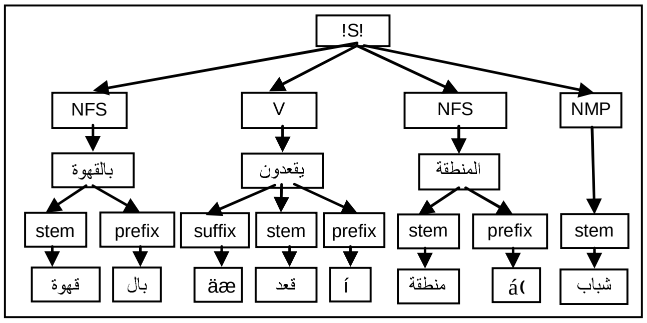

Fig. 1. Morphological Lexical Parse Tree (MLPT) for a dialectal-Arabic sentence.

should not allow a segmentation that may cause an error, if it is not going to reduce the size of the lexicon.

Even after applying the above rules there could still be some erroneous decompositions, and we indeed found a very small number of them by visual inspection of the decomposed lexicon. However, we do not provide a formal "error rate" of the segmentation because this would require a manually segmented reference lexicon. A useful heuristic that can mitigate the effect of these residual errors is to keep the top-N frequent decomposable words intact. A value of N=5000 was experimentally found to work well in practice.

Using a morphological segmentation algorithm will produce affixes in the speech recognition and machine translation outputs. These affixes should be glued to the following or previous word to form meaningful words. To facilitate such gluing each prefix and suffix is marked with a - (e.g. we have prefix Al- or suffix –yn). Two gluing schemes are used. The first is very simple and just sticks any word that starts(ends) with a - to the previous(following) word. The second tries to apply some constraints to prevent sequences of affixes and to ensure that these affixes are not attached to words that start(end) with a prefix(suffix). No noticeable difference is seen between the two approaches.

A few words about the morphological decomposition algorithm are worth mentioning here. First, this is more of a word segmentation algorithm than a morphological decomposition algorithm in a strict linguistic sense. However, it is very simple to apply and all it needs is a list of affixes and a lexicon. In previous work [4], we found that using this algorithm to tokenize the lexicon and the language model data leads to significant reduction in word error rate. This was a major motivation in using it in more elaborate language model schemes as discussed in the rest of this paper.

IV. Maximum entropy modeling

The Maximum Entropy method is a flexible statistical modeling tool that has been widely used in many areas of natural language processing [9, 12, 27]. Maximum entropy modeling produces a probability model that is as uniform as possible while matching empirical feature expectations exactly. This can be interpreted as making as few assumptions as possible in the model. Maximum entropy modeling combines multiple overlapping information sources (features). For an observation *o* (e.g. a morpheme or word) and a history (context) *h*, the probability model is given by:

$$P(o \mid h) = \frac{\exp(\sum_i \lambda_i f_i(o,h))}{\sum_{o'} \exp(\sum_j \lambda_j f_j(o',h))}$$

Notice that the denominator includes a sum over all possible outcomes, *o'*, which is essentially a normalization factor for probabilities to sum to 1. The functions f_i are usually referred to as feature functions or simply features. In the context of natural language processing using binary feature functions is very popular. These binary feature functions are given as:

$$f_i(o,h) = \begin{cases} 1, & \text{if } o{=}o_i \text{ and } q_i(h){=}1 \\ 0, & \text{otherwise} \end{cases}$$

where o_i is the outcome associated with feature f_i and $q_i(h)$ is an indicator function on history.

For example, a bigram feature f_i representing the word sequence “ARABIC LANGUAGE" in maximum entropy modeling would have o_i = “LANGUAGE" and $q_i(h)$ would be the question “Does the context *h* contain the word “ARABIC" as the previous word of the current word ?". The model parameters are denoted by λ_i, which can be considered as weights associated with feature functions. There are several methods to smooth maximum entropy models to avoid overtraining [9]. The most effective smoothing method, as shown in [9], is an instance of fuzzy maximum entropy smoothing. This type of smoothing amounts to adding a zero-mean Gaussian prior to each parameter. The only smoothing parameters to be determined are variance terms for each Gaussian. In our experiments, we used the same variance value for all model parameters. This fixed value was optimized on a held-out set using Powell's algorithm [31].

Besides Maximum Entropy method, another alternative machine learning approach that can be used for our task is memory based learning (MBL) [28]. MBL can represent exceptions that are crucial for linguistics. Similar to MBL, each instance, including exceptions is represented as a feature in maximum entropy modeling. However, unlike MBL, maximum entropy method may forget about individual instances if there is feature selection/pruning during model training. In this study, we did not perform any feature selection. If there is any exception represented in the form of a feature, they will not be lost. However, maximum entropy method weighs a set of features (evidences) to prefer one outcome over the other. If the contribution of the feature belonging to an exception is not sufficiently high, then the exception may not be predicted correctly. This phenomenon may look like a disadvantage at first, but it can also show the strength of maximum entropy modeling. That is, a set of evidences related to an outcome are weighted to assign a probability to that outcome. The weights are learned via improved iterative scaling (IIS) algorithm [9]. The main reason for using the maximum entropy method is its flexibility in integrating overlapping information sources into the model. This is a desirable feature for integrating morphological and lexical attributes in the language model.

Because of the aforementioned advantages we use the maximum entropy method for implementing JMLLM. The maximum entropy method allows JMLLM to incorporate lexical items, morphemes as well as attributes associated with these lexical items and morphemes into the language model. The maximum entropy method has been used in language modeling before, in the context of *n*-gram models [9], whole sentence models [13], syntactic structured language models [7] and semantic structured language models [8]. So far, the use of morphology for language modeling has been largely limited to segmenting words into morphemes to build a morpheme based language model. Language specific information such as morphological and lexical attributes is overlooked. Additionally, joint modeling of these information sources rather than using them as sources to derive features has not been considered. Integrating all available information sources such as morphological and language specific features in a single model could be very important to improve both speech recognition and machine translation performance. Next, we present the maximum entropy based Joint Morphological-Lexical Language Modeling (JMLLM) method.

V. Joint Morphological-Lexical Language Modeling

This section describes in detail the JMLLM models. Before discussing the models we will present the morphological-lexical parse tree (MLPT) which represents the information sources and their dependencies used in the model. We will also discuss two implementations of the JMLLM which we refer to as *JMLLM-leaf* and *JMLLM-tree*.

A. Morphological-Lexical Parse Tree

The MLPT consists of a tree structured joint representation of the lexical and morphological items in a sentence and their associated attribute information. An example of an MLPT for an Arabic sentence is given in Fig. 1. The leaves of the tree are morphemes that are predicted by the language model. Each morpheme has one of the three attributes: {prefix, stem, suffix} as generated by the morphological analysis mentioned in Section III. In addition to the morphological attributes, each word can take three sets of attributes: {type, gender, number}. Word type can be considered as POS, but here we consider only nouns (N), verbs (V) and remaining words are labeled as “other” (O). Gender can be masculine (M) or feminine (F). Number can be singular (S), plural (P) or double (D) (this is specific to Arabic). For example, the label “NMP” for the first word, شباب, shows that this word is a noun (N), male (M), and plural (P).

The MLPT given in Fig. 1 is built by starting with a sequence of decomposable words, which is in the middle row. Then, a morphological analysis is applied to the word sequence to generate the morpheme sequence along with their morphological attributes. We have a lexical attribute table prepared by human annotators for all the words in the training data. This table contains lexical attributes mentioned above. The result of the morphological analysis together with the lexical attributes is used to fill the corresponding nodes in the tree.

The dependencies represented in MLPT are integrated in JMLLM. We hypothesize that as we increase the amount of information represented in the MLPT and the tightness of integration, the JMLLM performance should improve. Applying morphological segmentation to data improves the dictionary's coverage of the domain and reduces the OOV rate. For example, splitting the word, بالقهوة as بال (prefix) and قهوة (stem) as in Fig. 1, allows to decode other combinations of this stem with the prefixes and suffixes provided in Table I. These additional combinations will hopefully cover those words in the test data that have not been seen in the unsegmented training data.

B. *JMLLM Basics*

In language modeling, we are interested in estimating the probability of the morpheme sequence, *P(M)*. Formally, we can compute *P(M)* by summing over all possible MLPT parses:

$$P(M) = \sum_{C} P(M,C)$$

where *C* denotes a parse tree that includes all the information in the non-terminal nodes of an MLPT. Note that any MLPT is composed of two parts, *M* and *C*. Here, C_M is the most likely parse tree (in statistical parsing) or the proposed single parse of the morpheme sequence (in rule based segmentation). Note that we do not need to specify the way the parsing is done, whether it is deterministic, as used in this paper, or statistical. Given a proposed parse tree C_M, we can calculate $P(C_M)$ or $P(M,C_M)$ based on all possible parses seen in the training data. The reasoning behind using $P(M,C_M)$ as the language model score is that it relies not only on the morphological history, but also on lexical, and attribute history in the sentence and can be more indicative of the meaningfulness of the morpheme sequence *M*. Using the joint probability of the word sequence and syntactic parse tree [35] or semantic parse tree [8] as the language model score yielded encouraging improvements. We also adopt the same approach in this paper by estimating the probability of MLPT for the language model score.

Another reasonable choice for language model score is to consider the parse tree C_M as given information and calculate the conditional probability $P(M \mid C_M)$ as the language model score. The relation between the conditional and joint probabilities is given as:

$$P(M,C_M) = P(M \mid C_M)P(C_M)$$

Here, we interpret $P(C_M)$ as the probability of a parse among all possible parses in the language of interest, and calculating $P(C_M)$ is possible regardless of the method of generating the parse tree C_M, whether the parsing is deterministic or probabilistic. However in this paper, we do not need to calculate $P(C_M)$ separately, since we either calculate $P(M \mid C_M)$ or $P(M,C_M)$ directly in our models.

We refer to the model predicting $P(M \mid C_M)$ as *JMLLM-leaf* since it predicts the morpheme sequence (at the leaves of MLPT) given the parse information. *JMLLM-leaf* represents a "loose integration" of information between morpheme sequence and its parse tree since it assumes the parse tree C_M as part of the "world" information. Another interpretation of *JMLLM-leaf* is that the parse probability $P(C_M)$ is assumed to be 1 in the expression for the joint probability $P(M,C_M)$, thus it is assumed that $P(C_M)$ does not affect the computation of $P(M,C_M)$.

The model predicting the joint probability $P(M,C_M)$ is called *JMLLM-tree* since all the information in the MLPT is used directly to calculate the joint probability. The joint probability is estimated by multiplying the probability of the non-terminal nodes with the probability of the morpheme sequence. This model represents a "tight integration" of all available information sources in the MLPT.

The first step in building the JMLLM is to represent MLPT as a sequence of morphemes, morphological attributes, words, and word attributes using a bracket notation [8]. Converting the MLPT into a text sequence allows us to group lexically related morphological segments and their attributes. In this notation, each morpheme is associated (association is denoted by "=") with an attribute (i.e. prefix/stem/suffix) and the lexical items are represented by opening and closing tokens, [WORD and WORD] respectively. Lexical attributes are represented as an additional layer of labels over the words. The parse tree given in Fig. 1 can be converted into a token sequence in text format as shown below. Note that Arabic is read from right to left.

[!S! [NMP شباب=stem NMP] [NFS [منطقة المنطقة=stem áÇ=prefix المنطقة] NFS] [V [يقعدون äæ=suffix قعد =stem í =prefix يقعدون] V] [NFS [بالقهوة بالقهوة=stem بال=prefix بالقهوة] NFS] !S!]

This representation uniquely defines the MLPT given in Fig. 1. Here, lexical attributes can be used as joint labels as in "NFS" or three separate labels: "N, F, S".

Next, we explain how the bracket representation can be used to train two different JMLLM models and determine features.

C. JMLLM for Morphological-Lexical Parse Tree Leaf Prediction: JMLLM-leaf

In this model, we decompose the conditional probability expression $P(M \mid C_M)$ as follows:

$$P(M \mid C_M) = \prod_{i=1}^{N} P(m_i | h_i)$$

Here, m_i denotes the *i*th morpheme in the morpheme sequence *M*, where *M* has *N* morphemes. h_i represents the history for the morpheme m_i and includes all tokens appearing before m_i in the bracket notation given above. Thus, in the history part, we can use the non-terminal nodes of the MLPT parse tree along with the previous morphemes. This model loosely integrates the parse and the morpheme sequence by assuming a conditional dependence of *M* on the non-terminal nodes of the parse tree.

Although we may use all parse tree information in our history, since C_M is assumed to be given, we only use a subset, corresponding to the tokens appearing before m_i in the bracket notation. This enables the models we develop to be used in real-time decoding (if real-time parsing can be done as well) or lattice rescoring. We explain features used in *JMLLM-leaf* and *JMLLM-tree* in Section V.E.

D. JMLLM for Entire Morphological-Lexical Parse Tree Prediction: JMLLM-tree

In the previous section, we decomposed the probability computation into two parts. However, it is possible to jointly calculate the probability of the morpheme sequence and the C_M within a single model. *JMLLM-tree* directly calculates $P(M, C_M)$ and, thus "tightly integrates" the parse and language model probabilities. To facilitate the computation of the joint probability, we use the bracket notation introduced earlier to express an MLPT. This representation makes it easy to define a joint statistical model since it enables the computation of the probability of both morpheme and word tokens using similar context information. Unlike loose-integration, tight-integration requires every token in the bracket representation to be an outcome of the joint model. Thus, the model outcome vocabulary, $\Re = V_M \cup V_W \cup V_{MA} \cup V_{LA}$, is the union of morpheme, word, morphological attribute and lexical attribute vocabulary. Note that for each item in the word and lexical attribute vocabularies there is an opening and closing bracket version. We represent the joint probability $P(M, C_M)$ as:

$$P(M, C_M) = \prod_{i=1}^{T} P(t_i | t_1, ..., t_{i-1})$$

where t_i is a token in the bracket notation and *T* is the total number of tokens. We note that the feature set for training the JMLLM models stays the same and is independent of the "tightness of integration".

E. Features Used for JMLLM

JMLLM can employ any type of questions one can derive from MLPT to predict the next morpheme. In addition to trigram questions about previous morphemes, questions about the attributes of the previous morphemes, parent lexical item and attributes of the parent lexical item can be used. The set of questions used in the model are as follows:

- Unigram history (empty history).
- Previous morpheme: m_{i-1} (bigram feature)
- Previous two morphemes: m_{i-1}, m_{i-2} (trigram feature).
- Immediate parent word (w_i) for the current morpheme (m_i).
- Previous parent word (w_{i-1})
- Morphological attributes for the previous two morphemes (ma_{i-1}, ma_{i-2}).
- Lexical attributes for the current parent word (wa_i).
- Lexical attributes for the previous parent word (wa_{i-1}).
- Previous token: t_{i-1} (token bigram feature).
- Previous two tokens: t_{i-1}, t_{i-2} (trigram token features).
- Previous morpheme and its parent word (m_{i-1}, w_{i-1}).

The history given in $P(o \mid h)$ consists of answers to these questions. Clearly, there are numerous questions one can ask from the MLPT in addition to the list given above. The "best" feature set depends on the task, information sources and the amount of data. In our experiments, we have not exhaustively searched for the best feature set but rather used a small subset of these features (listed above) which we believe are helpful for predicting the next morpheme. It is also worth noting that we did not use morpheme 4-gram features nor word 3-gram features. Therefore, morpheme trigram language model can be considered as a fair baseline to compare JMLLMs to.

The language model score for a given morpheme using JMLLM is conditioned not only on the previous morphemes but also on their attributes, the lexical items and their morphological and lexical attributes. Therefore, the language model scores are expected to be smoother compared to *n*-gram models especially for unseen morpheme *n*-grams. For example, during decoding we want to estimate the probability of "*P(estimate | probability, smooth)*". However, assume that we observe neither "smooth probability estimate" nor "probability estimate" in the training data. In *n*-gram modeling we back off to unigram probability for "estimate". On the other hand, in JMLLM, the *n*-gram features (trigram, bigram and unigram) are only 3 of the 11 features we listed above. Typically, in addition to unigram feature there will be several features that are active (e.g., lexical attributes, morphological attributes, or parent lexical item for the current word or previous word). The probabilities of these features are added to the unigram probability, which may result in a smoother probability estimate than the unigram probability alone. However, we do not know of a way to quantify this smoothness.

VI. System Architectures

A. *Speech Recognition Architecture*

The speech recognition experiments are conducted on an Iraqi Arabic speech recognition task, which covers the military and medical domains. The acoustic training data consist of about 200 hours of speech collected in the context of IBM's DARPA supported speech-to-speech (S2S) translation project [10].

The speech data is sampled at 16kHz and the feature vectors are computed every 10ms. First, 24-dimensional MFCC features are extracted and appended with the frame energy. The feature vector is then mean and energy normalized. Nine vectors, including the current vector and four vectors from its right and left contexts, are stacked leading to a 216-dimensional parameter space. The feature space is finally reduced from 216 to 40 dimensions using a combination of linear discriminant analysis (LDA) and maximum likelihood linear transformation (MLLT). This 40-dimensional vector is used in both training and decoding.

We use 33 graphemes representing speech and silence for acoustic modeling. These graphemes correspond to letters in Arabic plus silence and short pause models. Short vowels are implicitly modeled in the neighboring graphemes. The reason for using grapheme models instead of the more popular phone models is as follows. Arabic transcripts are usually written without short vowels, and hence using phone models requires restoring these short vowels; a process known as vowelization. Doing this manually is very tedious, and automatic vowelization is error-prone especially for dialectal Arabic. In numerous experiments with vowelization of the training data and hence building phone models we were not able to outperform the grapheme system. This is in contrast to MSA where it was found that phone models are better than the graphemes [33]. This was achieved largely because of an accurate vowelization process supplied by the Buckwalter analysis. Each grapheme is modeled with a 3-state left-to-right hidden Markov model (HMM).

Acoustic model training proceeds as follows. Feature vectors are first aligned, using initial models, to model states. A decision tree is then built for each state using the aligned feature vectors by asking questions about the phonetic context; quinphone questions are used in this case. The resulting tree has about 2K leaves. Each leaf is then modeled using a Gaussian mixture model. These models are first bootstrapped and then refined using three iterations of forward-backward training. The current system has about 75K Gaussians.

The language model training data has 2.8M words with 98K unique words and it includes acoustic model training data as a subset. The pronunciation lexicon consists of the grapheme mappings of these unique words. The mapping to graphemes is one-to-one and there are very few pronunciation variants that are supplied manually mainly for numbers. A statistical trigram language model using Modified Kneser-Ney smoothing [23, 29] has been built for both the unsegmented data, which is referred to as Word-3gr, and the morphologically analyzed data, which is called Morph-3gr.

A static decoding graph is compiled by composing the language model, the pronunciation lexicon, the decision tree, and the HMM graphs. This static decoding scheme, which compiles the recognition network off-line before decoding, is becoming very popular in speech recognition [32]. The resulting graph is further optimized using determinization and minimization to achieve a relatively compact structure. Decoding is performed on this graph using a Viterbi beam search.

B. *Statistical Machine Translation System*

Statistical machine translation training starts with a collection of parallel sentences. We train 10 iterations of IBM Model-1 followed by 5 iterations of word-to-word HMM [11]. Models of two translation directions, from English to Iraqi Arabic and from Iraqi Arabic to English, are trained simultaneously for both Model-1 and HMM. More specifically, let $C^{(n)}_{e\to f}(e, f)$ be the number of times (soft count, collected in the E-step of the Expectation Maximization (EM) algorithm) that the English word *e* generates the foreign word *f* in the direction from English to Arabic at iteration *n*. Similarly let $C^{(n)}_{f\to e}(f, e)$ be the corresponding number of times that *f* generates *e* in the other direction. To estimate the translation lexicon from English to foreign language in the M-step of the EM algorithm, we linearly combine counts from two directions and use that to re-estimate the word-to-word translation probability $t(f \mid e)$ at iteration *(n+1)*:

$$C^{(n)}(e \to f) = \alpha \cdot C^{(n)}_{e\to f}(e, f) + (1-\alpha) C^{(n)}_{f\to e}(f, e)$$

$$t^{(n+1)}(f \mid e) = \frac{C^{(n)}(e \to f)}{\sum_{f'} C^{(n)}(e \to f')}$$

where $\alpha \in [0,1]$ is a scalar controlling the contribution of statistics from the other direction. A higher value of α indicates less proportion of soft counts borrowed from the other direction. We fix α value to be 0.5 for a balanced lexicon. Similarly, we can re-estimate word-to-word translation probability $t(e \mid f)$ at iteration (*n+1*).

After HMM word alignment models are trained, we perform a Viterbi word alignment procedure in two directions independently. By combining word alignments in two directions using heuristics [17], a single set of static word alignments is then formed. Phrase translation candidates are derived from word alignments. All phrase pairs which respect the word alignment boundary constraint are identified and pooled together to build phrase translation tables in two directions using the maximum likelihood criterion with pruning. We set the maximum number of words in Arabic phrases to be 5. This will finish the phrase translation training part.

The translation engine is a phrase based multi-stack implementation of log-linear models similar to Pharaoh [15]. Given an English input ***e***, the decoder is formulated as a statistical decision making process that aims to find the optimal foreign word sequence ***f**** by integrating multiple feature functions:

$$f^* = \underset{f}{\operatorname{argmax}} \sum_{k=1}^{K} \lambda_k h_k(f, e)$$

where λ_k is the weight of feature function h_k. Like most other maximum entropy based translation engines, active features in our decoder include translation models in two directions, IBM Model-1 style lexicon weights in two directions, language model, distortion model, and sentence length penalty. These feature weights (λ_k) are tuned discriminatively on the development set to directly maximize the translation performance measured by an automatic error metric (such as BLEU [18]) using the downhill simplex method [16]. The decoder generates an N-best list, which can be re-scored using a different model, such as an improved language model, in a post-processing stage to generate the final translation output.

VII. Experimental results

A. Speech Recognition Experiments

We mentioned that the language model training data has 2.8M words with 98K unique lexical items. The morphologically analyzed training data has 58K unique vocabulary items. The test data consists of 2719 utterances spoken by 19 speakers. It has 3522 unsegmented lexical items, and morphological analysis reduces this figure to 3315.

In order to evaluate the performance of JMLLM, a lattice with a low lattice error rate is generated by a Viterbi decoder using the word trigram model (Word-3gr) language model. From the lattice at most 200 (N=200) sentences are extracted for each utterance to form an N-best list. These utterances are rescored using the JMLLM and the morpheme trigram language model (Morph-3gr). The language model rescoring experiments are performed for the entire corpus, which has 460K utterances and half the corpus, which has 230K utterances. The last column in Table II presents results for the 460K corpus. The first entry (18.4%) is the oracle error rate of the N-best list. Morph-3gr error rate is 0.9% better than that of the Word-3gr. Log-linear interpolation of these language models provides a small improvement (0.3%) over Morph-3gr.

TABLE II
Speech Recognition Language Model Rescoring Experiments With the 460K Sentence Complete corpus and 230K Sentence Half The Corpus

Language Models	Half The Corpus WER (%)	Complete Corpus WER (%)
N-best Oracle	22.1	18.4
Word Trigram (Word-3gr)	38.7	32.2
Morpheme Trigram (Morph-3gr)	37.7	31.3
Word-3gr + Morph-3gr	37.6	31.0
JMLLM-leaf	37.1	30.5
JMLLM-leaf + Morph-3gr	36.4	30.1
JMLLM-leaf + Word-3gr	36.6	29.8
JMLLM-tree	**36.9**	**29.9**
JMLLM-tree + Morph-3gr	**35.9**	29.4
JMLLM-tree + Word-3gr	36.1	**29.2**

In a previous study [20], we reported results for “loosely integrated” JMLLM (*JMLLM-leaf*) which are provided here. *JMLLM-leaf* obtains 30.5%, which is 1.7% and 0.8% better than Word-3gr and Morph-3gr, respectively. Interpolating *JMLLM-leaf* with Word-3gr improves the WER to 29.8%, which is 1.2% better than that of the interpolation of Word-3gr and Morph-3gr. The interpolation weights are set equally to 0.5 for each LM. Adding the Morph-3gr in a three way interpolation does not provide further improvement.

In this study, we also provide results for “tightly integrated” JMLLM (*JMLLM-tree*). *JMLLM-tree* provides an additional 0.6% improvement over *JMLLM-leaf*. Interpolating *JMLLM-tree* with Morph-3gr and Word-3gr improves the WER by 0.5% and 0.7%, respectively compared to *JMLLM-tree*. Again three-way interpolation does not provide additional improvement. Even though JMLLMs are not built using 4-gram morpheme features, it is valuable to report the Morph-4gr results. The Morph-4gr language model achieves 30.6% WER.

In order to investigate the impact of different amounts of training data on the proposed methods, the experiments described above are repeated with 230K utterance corpus. The results are provided in the middle column of Table II. Morph-3gr still outperformed Word-3gr. However, the results with half the data reveal that Morph-3gr becomes more effective than the Word-3gr, when interpolated with both *JMLLM-leaf* and *JMLLM-tree*. We believe this is because of the fact that data sparseness has a more severe impact on Word-3gr than it has on Morph-3gr. Interpolating *JMLLM-tree* with Morph-3gr provided the best result (35.9%), which is 1.7% better than Word-3gr + Morph-3gr.

In summary, for the complete training corpus, *JMLLM-tree* alone achieves a 2.3% and 1.4% absolute error reductions compared to Word-3gr and Morph-3gr, respectively. When interpolated with Word-3gr, *JMLLM-tree* obtains 1.8% absolute error reduction compared to interpolated Word-3gr and Morph-3gr. Standard p-test[3] shows that these improvements are significant at $p<0.001$ level.

[3] We used the Matched Pairs Sentence-Segment Word Error (MAPSSWE) test, available in standard SCLITE’s statistical system comparison program from NIST with the option “mapsswe”.

B. Machine Translation Experiments

The machine translation task considered here is about translating English sentences into Iraqi Arabic. The parallel corpus has 430K utterance pairs with 90K words (50K morphemes). The Iraqi Arabic language model training data includes the Iraqi Arabic side of the parallel corpus as a subset. A statistical trigram language model using modified Knesser-Ney smoothing [23] has been built for the morphologically segmented data. A development set (DevSet) of 2.2K sentences is used to tune the feature weights. A separate test set (TestSet) of 2.2K utterances are used to evaluate the language models for machine translation.

The translation performance is measured by the BLEU score [18] with one reference for each hypothesis. In order to evaluate the performance of the JMLLM, a translation N-best list (N=10) is generated using the baseline Morph-3gr language model. First, on the DevSet all feature weights including the language model weight are optimized to maximize the BLEU score using the downhill simplex method [16]. These weights are fixed when the language models are used on the TestSet. In a previous study [21], we applied *JMLLM-leaf* on a different TestData. In this study, in addition to the results for JMLLM-*leaf,* we also provide results for *JMLLM-tree*. The translation BLEU (%) scores for the DevSet are given in the first column of Table III. The first row (37.89 and 38.27) provides the oracle BLEU scores for the N-best list generated for both DevSet and TestSet. Given the tuned weights for Morph-3gr and other translation scores, the N-best list is used to tune the weight for JMLLMs. On the DevSet, the baseline Morph-3gr achieves 29.74, and word-trigram rescoring improves the BLEU score to 30.71. Interpolating the Morph-3gr and Word-3gr does not provide additional improvement. *JMLLM-leaf* achieves 30.63 by itself and interpolating it with Morph-3gr and Word-3gr improves the BLEU score marginally. On the other hand, *JMLLM-tree* achieves 30.80 and interpolation with Morph-3gr improves the result to 31.10. Interpolation with Word-3gr improves the score to 31.28, which is about 1.5 points better than that of the Morph-3gr and 0.6 points better than that of the Word-3gr.

The results on DevSet are encouraging but results on the TestSet are the true assessment of the proposed language models. In the second column of Table III the results are provided for the TestSet by fixing the tuned weights on the DevSet. *JMLLM-leaf* improves the results by 0.8 points and 0.2 points compared to Morph-3gr and Word-3gr, respectively. Interpolating *JMLLM-leaf* with Morph-3gr and Word-3gr improves the results by an additional 0.1 points and 0.3 points, respectively. *JMLLM-tree* improves the result from 31.40 to 31.71 compared to *JMLLM-leaf*. Interpolating *JMLLM-tree* with Morph-3gr and Word-3gr improves the results marginally.

In summary, *JMLLM-tree* improves the results by 1.1 points which is significantly[4] better compared to Morph-3gr and 0.5 points compared Word-3gr on the TestSet.

TABLE III
STATISTICAL MACHINE TRANSLATION NBEST LIST RESCORING WITH JMLLM

Language Models	BLEU (%)	BLEU (%)
	DevSet	TestSet
N-best List Oracle	37.89	38.27
Morpheme Trigram (Morph-3gr)	29.74	30.61
Word Trigram (Word-3gr)	30.71	31.21
Morph-3gr + Word-3gr	30.71	31.25
JMLLM-leaf	30.63	31.40
JMLLM–leaf + Morph-3gr	30.69	31.49
JMLLM–leaf + Word-3gr	30.71	31.67
JMLLM-tree	**30.80**	**31.71**
JMLLM-tree + Morph-3gr	31.10	31.73
JMLLM-tree + Word-3gr	**31.28**	**31.76**

Additionally, *JMLLM-tree* consistently outperforms *JMLLM-leaf* for both DevSet and TestSet.

C. Discussions

When examining the errors from our translation system, we see that some of the poor recognition and translation may be explained by the rather different behavior of the segmentation method on the training and test data. The OOV rate for the unsegmented speech recognition test data is 3.3%, the corresponding number for the morphologically analyzed data is 2.7%. Hence, morphological segmentation reduces the OOV rate by only 0.6%. It is worth comparing the vocabulary reduction on the training data (41%) to the vocabulary reduction on the test set (6%). Even though the OOV rates for both unsegmented and segmented test data are not that high, the characteristics of the test data appear to be different than the training data in its morphological make-up. We believe this is because the training data was collected over several years. In the beginning there was more emphasis on the medical domain but later the emphasis shifted towards the checkpoint, house search, vehicle search types of dialogs in the military domain. However, the test data was set aside from the very first part of the collected data. In other words the test data was not uniformly/randomly sampled from the entire data. Despite this apparent mismatch between training and test data the speech recognition results are encouraging.

For machine translation experiments, the OOV rate for the unsegmented machine translation test data is 8.7%, the corresponding number for the morphologically analyzed data is 7.4%. Hence, morphological segmentation reduces the OOV rate by 1.3% (15% relative), which again, is not as large a reduction as compared to the training data reduction (about 40% relative reduction). We believe this would limit the potential improvement we could get from JMLLM, since JMLLM is expected to be more effective compared to word *n*-gram models, when the OOV rate is significantly reduced after segmentation. Improving the morphological segmentation to cover more words can potentially improve the performance of JMLLMs.

Even though it was not evaluated in this study, one of the benefits of tight-integration using joint modeling becomes apparent when a set of alternatives are generated for a sentence rather than just a single parse. For example, we may have more than one MLPT for a given sentence because of alternative morphological analysis, tagging or

[4] The improvement is significant at the 80% confidence interval. We use the well-known bootstrapping technique to measure the confidence interval for BLEU.

semantic/syntactic parses. Then, tight-integration with joint modeling allows not only to get the best morpheme sequence but also the best morphological analysis and/or tagging and/or semantic/syntactic parses of a sentence.

VIII. Conclusions

We presented a new language modeling technique called Joint Morphological-Lexical Language Modeling (JMLLM) for inflected languages in general and Arabic in particular. JMLLM allows joint modeling of lexical, morphological and additional information sources about morphological segments and lexical items. JMLLM has both the predictive power of the word based language model and the coverage of the morpheme based language model. It is also expected to have smoother probability estimates than both morpheme and word based language models. Two implementations of the JMLLM were proposed. One called *JMLLM-leaf* that loosely integrates the parse information while the other tightly integrates the parse information and is referred to as *JMLLM-tree*. Speech recognition and machine translation experimental results demonstrate that JMLLM provides encouraging improvements over the baseline word and morpheme based trigram language models. Moreover, tight-integration of all available information sources in the MLPT provides additional improvements over the loose-integration.

References

[1] A. Ghaoui, F. Yvon, C. Mokbel, and G. Chollet, “On the use of morphological constraints in N-gram statistical language model,” *Interspeech’05*, Lisbon, Portugal, 2005.

[2] B. Xiang, K. Nguyen, L. Nguyen, R. Schwartz, J. Makhoul, “Morphological decomposition for Arabic broadcast news transcription”, *ICASSP’06*, Toulouse, France, 2006.

[3] G. Choueiter, D. Povey, S.F. Chen, and G. Zweig, “Morpheme-based language modeling for Arabic LVCSR”, *ICASSP’06*, Toulouse, France, 2006.

[4] M. Afify, R. Sarikaya, H-K J. Kuo, L. Besacier, and Y. Gao, “On the Use of Morphological Analysis for Dialectal Arabic Speech Recognition”, *Interspeech’06*, Pittsburgh, PA 2006.

[5] K. Kirchhoff, D. Vergyri, K. Duh, J. Bilmes and A. Stolcke, ``Morphology-based language modeling for Arabic speech recognition", Computer Speech and Language 20(4), 2006, pp. 589-608.

[6] T. Buckwalter, “Buckwalter Arabic morphological analyzer version 1.0”, LDC2002L49 and ISBN 1-58563-257-0, 2002.

[7] S. Khudanpur and J. Wu, "Maximum Entropy Techniques for Exploiting Syntactic, Semantic and Collocational Dependencies in Language Modeling," *Computer Speech and Language*, 14(4):355-372, 2000

[8] H. Erdogan, R. Sarikaya, S.F. Chen, Y. Gao and M. Picheny, “Using Semantic Analysis to Improve Speech Recognition Performance,” *Comp. Speech & Language*,, vol. 19(3), pp: 321-343, July 2005.

[9] S. F. Chen, R. Rosenfeld. “A survey of smoothing techniques for ME models”, *IEEE Trans. Speech and Audio Process*. 8 (1), 37–50, 2000.

[10] Y. Gao, L. Gu, B. Zhou, R. Sarikaya, H.-K. Kuo. A.-V.I. Rosti, M. Afify, W. Zhu, "IBM MASTOR: Multilingual Automatic Speech-to-Speech Translator", *ICASSP’06*, Toulouse, France, 2006.

[11] S. Vogel, H. Ney, and C. Tillmann, “HMM-based word alignment in statistical translation”, *COLING-96*, pages: 836-841, Copenhagen, August, 1996.

[12] A. Berger, S. Della Pietra and V. Della Pietra, "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistic*s, vol. 22, no. 1, March 1996

[13] Ronald Rosenfeld, Stanley F Chen, and Xiaojin Zhu. “Whole sentence exponential language models: a vehicle for linguistic-statistical integration”, *Computer Speech and Language*, 15(1), 2001.

[14] K. Kirchhoff and M. Yang. 2005.“Improved language modeling for statistical machine translation”. *ACL’05 workshop on Building and Using Parallel Text*, pages 125–128.

[15] P. Koehn, F. J. Och, and D. Marcu, “Pharaoh: A beam search decoder for phrase based statistical machine translation models”, *Proc. of 6th Conf. of AMTA*, 2004.

[16] F. J. Och and H. Ney, “Discriminative training and maximum entropy models for statistical machine translation”, *ACL*, pages 295–302, University of Pennsylvania 2002.

[17] F. J. Och and H. Ney, “A Systematic Comparison of Various Statistical Alignment Models”, *Comp. Linguistics*, 29(1):9-51, 2003.

[18] K. Papineni, S. Roukos, T. Ward, and W-J. Zhu, “BLEU: a Method for Automatic Evaluation of machine translation”, *ACL 02*, pages 311–318, 2002.

[19] P. Liang, B. Taskar, and D. Klein, “Alignment by agreement”, *HLT/NAACL*, pages 104–111, 2006.

[20] R. Sarikaya, M .Afify and Y. Gao, “Joint Morphological-Lexical Modeling (JMLLM) for Arabic,” *ICASSP’07*, Honolulu Hawaii, 2007.

[21] R. Sarikaya and Y. Deng, “Joint Morphological-Lexical Modeling for Machine Translation,” *HLT/NAACL’07*, Rochester, NY, 2007.

[22] R. Zens, E. Matusov, and H. Ney, “Improved word alignment using a symmetric lexicon model”, *COLING*, pages 36–42, 2004.

[23] S. Chen, J. Goodman, “An Empirical Study of Smoothing Techniques for Language Modeling", *ACL-96*, Santa Cruz, CA, 1996.

[24] P. Geutner, “Using morphology towards better large-vocabulary speech recognition systems”, *ICASSP’95*, Detroit, MI, 1995.

[25] M. Kurimo, *et.al.*, “Unlimited vocabulary speech recognition for agglutinative languages”, *HLT/NAACL*, pp. 104–111, 2006.

[26] O.W. Kwon, and J. Park, “Korean Large Vocabulary Continuous Speech Recognition with Morpheme-based Recognition Units”, *Speech Communication*, Vol. 39, pp. 287-300, 2003.

[27] K. Toutanova, C. D. Manning, “Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger”, *Joint SIGDAT Conference on EMNLP/VLC*, Hong Kong, 2000.

[28] J. Zavrel, W. Daelemans, “Memory-Based Learning”, *ACL-97*, Madrid, Spain, 1997.

[29] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. *ICASSP*, vol. 1. pp. 181–184, 1995.

[30] E. Arisoy, H. Sak and M. Saraclar. Language Modeling for Automatic Turkish Broadcast News Transcription. *Interspeech-2007*. Antwerp Belgium, 2007.

[31] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1992. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, second edition.

[32] M. Riley, E. Bocchieri, A. Ljolje and M. Saraclar, “The AT&T 1x real-time Switchboard speech-to-text system”, *NIST RT’02 Workshop*, 2002.

[33] M. Afify, L. Nguyen, B. Xiang, S. Abdou, J. Makhoul, "Recent Progress in Arabic Broadcast News Transcription at BBN", *Interspeech’05*, Lisbon, Portugal, 2005.

[34] H Erdogan, O Buyuk, K Oflazer, "Incorporating language constraints in sub-word based speech recognition," *IEEE ASRU Workshop*, San Juan, Puerto Rico, Dec. 2005.

[35] Chelba, Ciprian, Jelinek, Frederick, “Structured language modeling,” *Computer Speech and Language* 14 (4), 283–332, 2000.