Survey of Intelligent Chatbots and Conversational Interfaces

January 10, 2017

0.1 Three New Graphical Models for Statistical Language Modelling [1]

This paper shows how real-valued distributed representations for words can be learned at the same time as learning a large set of stochastic binary hidden features that are used to predict the distributed representation of the next word from previous distributed representations. One of the proposed models significantly outperforms the best n-gram models.

The first model is called the *Factored Restricted Boltzmann Machine Language Model (FRBM)*. Each word is represented using a real-valued feature vector of length N_f. Let R be an $N_w \times N_f$ matrix with row i being the feature vector for the i-th word. The *joint energy* of a sequence of words $w_1, ..., w_n$ is defined as:

$$E(w_n, h; w_{1:n-1}) = -(\sum_{i=1}^{n} v_i^T R W_i) h.$$

Here matrix W_i specifies the interaction between the vector of hidden variables and the feature vector.

The joint conditional distribution of the next word and the hidden configuration h is defined in terms of the energy function as:

$$P(w_n, h | w_{1:n-1}) = \frac{1}{Z_c} \exp(-E(w_n, h; w_{1:n-1})),$$

where Z_c is a context-dependent normalization term. The conditional distribution of the next word can be obtained by marginalizing over the hidden variables.

The second model is called the *Temporal Factored RBM (TFRBM)*. The basic idea is to make a simple extension to the factored RBM language model. Suppose we want to predict word w_{t+n} from $w_1, ..., w_{t+n-1}$ for some large t. The method applies a separate instance of the model to words $w_\tau, ..., w_{\tau+n-1}$ for each τ in $\{1, ..., t\}$. In order to propagate context information forward, it further introduces directed connections from h^τ to $h^{\tau+1}$, and computes the hidden state of model $\tau + 1$ using the inputs from the hidden state of model τ as well as its visible units. Figure 0.1 (a) shows the diagram for the temporal FRBM.

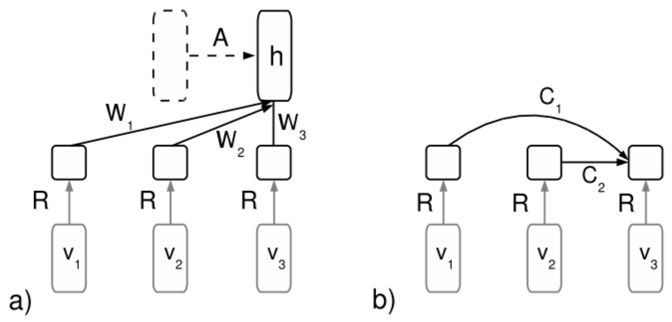


Figure 0.1: a) The diagram for FRBM and TFRMB. The dashed part is included only for the TFRBM; b) The diagram for the log-bilinear model.

The third model is called the *log-bilinear language model.* The energy function of this model is specified as:

$$E(w_n; w_{1:n-1}) = -(\sum_{i=1}^{n-1} v_i^T R C_i) R^T v_n - b_r^T R^T v_n - b_v^T v_n.$$

In the FRBM energy function the interaction is between the word feature vectors and the hidden variables, whereas in this model the interaction is between the feature vectors for the context words and the feature vector for the predicted word. Intuitively, the model predicts a feature vector for the next word by computing a linear function of the context word feature vectors. Then it assigns probabilities to all words in the vocabulary based on the similarity. This model is similar to the energy-based model proposed in [2]. However, the model proposed here uses a bilinear energy function, while the energy function in [2] is a one-hidden-layer neural network. Figure 0.1 (b) shows a diagram of the log-bilinear language model.

In the experimental study, the paper evaluates the proposed models using the *Associated Press News (APNews)* dataset consisting of a text stream of about 16 million words. In the first experiment, it is shown that three of the four network models are competitive with n-gram models. The best results were obtained by averaging with the temporal network model, resulting in 21% reduction in perplexity over the best n-gram model. In the second experiment, it is also shown that the log-bilinear models clearly outperform the n-gram models.

0.2 Limited Domain Synthesis [3]

This paper presents a reliable and efficient method for building limited domain speech synthesis voices. By constructing databases close to the target domain of the speech application, it uses *unit selection* synthesis techniques to reliably give high quality synthesis within domain.

The task of building a voice consists of the following processes:

- Design the corpus
  The first step is to design a prompt list that adequately covers the domain. In general, prompts should have at least one occurrence of each word in the vocabulary in each prosodic context.

- Synthesize each utterance
  The prompts are synthesized for a number of reasons: 1) ensure that all the tokens are expanded properly (e.g. flight numbers and dates); 2) estimate the time required for recording; 3) use the synthesized prompt in labeling the human spoken utterance.

- Record the voice talent
  Recording with studio quality equipment gives better results, but the paper is also interested in making the process more accessible. It uses a laptop in a quiet room. The recording quality shows to be acceptable once audio devices are set up appropriately.
- Label the recordings
  After recording, it labels the text using a simple but effective technique based on [4]: it uses DTW to align between the mel-scale cepstral coefficients of the synthesized and recorded waveforms.
- Extract pitchmarks
- Extract pitch-synchronous parameters
- Build a cluster unit selection synthesizer
  The unit selection technique used in this paper is an updated version of that more fully described in [5]. The general algorithm takes all units of the same type and calculates an acoustic distance between each. Selected features including phonetic and prosodic context are used to build a decision tree that minimizes acoustic distance in each partition. At synthesis time, it selects the appropriate cluster using the decision tree, and then finds the best path through the candidates.
- Test and tune, repeating as necessary

The proposed technique is tested on three domains: a talking clock, a weather report system, and the CMU Communicator system.

The original demonstration of this technique was a simple talking clock. The prompts consist of 24 simple utterances of the form: “The time is now, a little after quarter past two in the afternoon.” Not counting recording time, this takes around 3 minutes to build. Such clocks have also been built in languages other than English, such as Chinese.

The most difficult example is the CMU Communicator system. At first it appears that the domain is not closed, as it includes greeting to registered users by name, and allows reference to any airport in the world. For the words like cities and airports, which are essentially open classes, it used the frequency information in the logs to select which set to include in the recordings. For the more frequently mentioned cities it includes more than one occurrence in the prompts. The final voice was built in under one-man week. After the version was running, it made some changes to the language generation system, and constructed further 50 utterances and added them into the system in another morning’s work.

0.3 Multilingual Distributed Representations without Word Alignment [6]

This paper proposes a method for learning distributed representations in a multiligual setup. The model learns to assign similar embeddings to aligned sentences and dissimilar ones to sentences which are not aligned. It shows that the representations are semantically informative and applies them to a cross-lingual document classification task.

The basic idea is that, given enough parallel data, a shared representation would be forced to capture the common elements between sentences from different languages. What two parallel sentences have in common, of course, is the semantics of those two sentences. Using such parallel data, it proposes a novel method for learning vector representations at the word level and beyond.

The first step is to define a bilingual error function as follows: given a *compositional sentence model (CVM)* $\mathcal{M}_A$, which maps a sentence to a vector, it trains a second CVM $\mathcal{M}_B$ using a parallel corpus $\mathcal{C}_{A,B}$. For each pair of parallel sentences $(a, b) \in \mathcal{C}_{A,B}$, it attempts to minimize

$$E_{dist}(a, b) = ||a_{root} - b_{root}||^2,$$

where a_{root} (b_{root}) is the vector representing sentence a (b).

The CVM used in this paper is a simple additive composition function:

$$a_{root} = \sum_{i=0}^{|a|} a_i.$$

Due to the use of parallel data, we know that a and b are semantically equivalent. Hence the goal is to jointly train both models $\mathcal{M}_A$ and $\mathcal{M}_B$, so that

$$E_{bi}(\mathcal{C}_{A,B}) = \sum_{(a,b)\in\mathcal{C}_{A,B}} E_{dist}(a,b)$$

is minimized.

However, the models could learn to reduce all embeddings and composition weights to zeros and thereby minimize the objective function. This problem is addressed by penalizing small distances between non-parallel sentence pairs. For every pair of parallel sentences (a,b) it samples a number of additional sentences $n \in \mathcal{C}_B$, which are not exact translation of a:

$$E_{noise}(a,b,n) = [1 + E_{dist}(a,b) - E_{dist}(a,n)]_+,$$

where $[x]_+ = \max(x,0)$ denotes the standard hinge loss. Thus, the final objective function is defined to be:

$$J(\theta_{bi}) = \sum_{(a,b)\in\mathcal{C}_{A,B}} (\sum_{i=1}^{k} E_{noise}(a,b,n_i)) + \frac{\lambda}{2}||\theta_{bi}||^2.$$

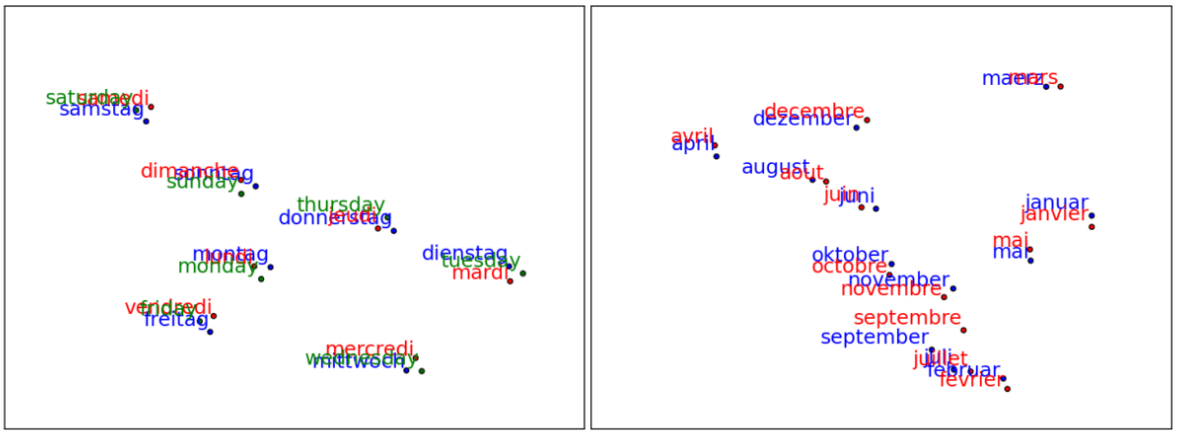


Figure 0.2: The left scatter plot shows t-SNE projections for a weekdays in three language. The right plots shows months using only the German and French words.

In the experimental study, the paper evaluates the proposed model in to experiments: the *BiCVM* model was trained on 500k sentence pairs of a English-German parallel corpus. The *BiCVM+* used this dataset in combination with anther 500k parallel sentences form the English-French section. The motivation behind BiCVM+ is to investigate whether it can learn better embeddings by introducing additional data in a different language. The models are evaluated on the *cross-lingual document classification (CLDC)* task. It is shown that both models outperform all prior work on this task. Further, BiCVM+ outperforms BiCVM, indicating the usefulness of adding training data from a separate pair. Figure 0.2 shows the t-SNE projections for a number of English, French and German words.

0.4 Energy-Based Models and Boltzmann Machines [7]

In this survey we present the fifth chapter of the book *Learning Deep Architectures for AI* by Bengio [7]. This chapter introduces the main mathematical concepts helpful to understand *Restricted Boltzmann Machines (RBMs)*, which are particular energy-based models.

Energy-based models associate a scalar energy to each configuration of the variables of interest. For example, we would like plausible or desirable configurations to have low energy. The probability distribution of an energy-based model may be defined as follows:

$$P(x) = \frac{e^{-Energy(x)}}{Z}. \tag{0.1}$$

Here Z is a normalization term called the *partition function* by analogy with physical systems.

In the *products of experts* formulation, the energy function is a sum of terms, each one associated with an "expert" f_i:

$$Energy(x) = \sum_i f_i(x). \tag{0.2}$$

Each expert can thus be seen as a detector of implausible configurations of x, or equivalently, as enforcing constraints on x.

In many cases we do not observe all the components simultaneously, or we want to introduce some non-observed variables to increase the expressive power of the model. So we consider an observed part x and a hidden part h:

$$P(x,h) = \frac{e^{-Energy(x,h)}}{Z}, \tag{0.3}$$

$$P(x) = \sum_h P(x,h). \tag{0.4}$$

In order to map this formulation to the standard form, we introduce the notation of *free energy*:

$$P(x) = \frac{e^{-FreeEnergy(x)}}{Z}, \tag{0.5}$$

$$FreeEnergy(x) = -\log \sum_h e^{-Energy(x,h)}. \tag{0.6}$$

The average log-likelihood gradient over the training set is

$$E_{\hat{P}}[\frac{\partial \log P(x)}{\partial \theta}] = -E_{\hat{P}}[\frac{\partial FreeEnergy(x)}{\partial \theta}] + E_P[\frac{\partial FreeEnergy(x)}{\partial \theta}], \tag{0.7}$$

where expectations are over x, with $\hat{P}$ the training set empirical distribution and E_P the expectation under the model's distribution.

If the energy can be written as a sum of terms associated with at most one hidden unit:

$$Energy(x,h) = -\beta(x) + \sum_i \gamma_i(x,h_i), \tag{0.8}$$

then the free energy and numerator of the likelihood can be computed tractably:

$$P(x) = \frac{e^{\beta}(x)}{Z} \prod_i \sum_{h_i} e^{-\gamma_i(x,h_i)}. \tag{0.9}$$

The *Boltzmann machine* is a particular type of energy-based model with hidden variables. In a Boltzmann machine, the energy function is a general second-order polynomial:

$$Energy(x,h) = -b'x - c'h - h'Wx - x'Ux - h'Vh. \tag{0.10}$$

The gradient of the log-likelihood can be written as:

$$\frac{\partial \log P(x)}{\partial \theta} = -\sum_h P(h|x)\frac{\partial Energy(x,h)}{\partial \theta} + \sum_{\hat{x},h} P(\hat{x},h)\frac{\partial Energy(\hat{x},h)}{\partial \theta}. \tag{0.11}$$

This gradient can be computed, if we have a procedure to sample from $P(h|x)$ and from $P(x,h)$. The basic idea is to use *Gibbs sampling* in two phases: in the *positive phase*, x is clamped to the observed input vector, and we sample h given x; in the *negative phase* both x and h are sampled. Since two *MCMCs (Monte Carlo Markov Chain)* (one for the positive phase and one for the negative phase) are needed for each example x, the computation can be very expensive.

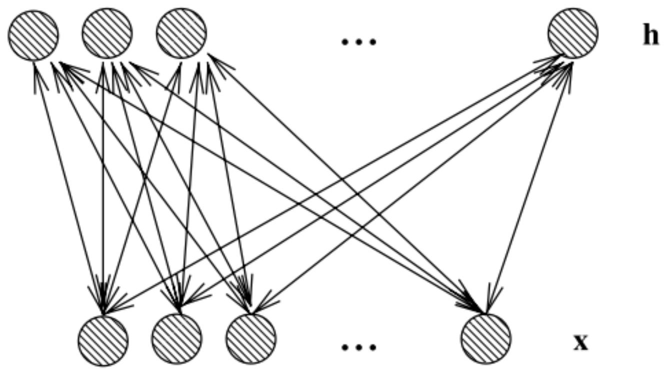


Figure 0.3: Undirected graphical model of a RBM

In the *Restricted Boltzmann Machine (RBM)*, the h_i are independent of each other when conditioning on x, and the x_j are independent of each other when conditioning on h (cf. Figure 0.3). The energy function is bilinear:

$$Energy(x,h) = -b'x - c'h - h'Wx. \tag{0.12}$$

The free energy of the input can be computed efficiently:

$$FreeEnergy(x,h) = -b'x - \sum_i \log \sum_{h_i} e^{h_i(ci+W_i x)}. \tag{0.13}$$

Using the same factorization trick, we can obtain a tractable expression for $P(h|x)$ and $P(x|h)$.

Gibbs sampling in fully connected Boltzmann Machines is slow because there are as many sub-steps in the Gibbs chain as there are units in the network. The Factorization enjoyed by RBMs bring two benefits: 1) we do not need to sample in the positive phase because the free energy is computed analytically; 2) the set of variables (x,h) can be sampled in two sub-steps in each step of the Gibbs chain: first we sample h given x, and then a new x given h.

Contrastive Divergence (CD) is an approximation of the log-likelihood gradient that has been found to be a successful update rule for training RBMs. The basic idea of k-step CD is simple, and involves a second approximation, which introduces some bias in the gradient: run the MCMC chain $x_1, ..., x_{k+1}$ for only k steps starting from the observed example $x_1 = x$. The surprising empirical result is that even $k = 1$ often gives good results.

0.5 Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech [8]

This paper describes a statistical approach for modeling *dialogue acts (DAs)* on conversational speech, i.e., speech-act-like units such as *Statement*, *Question*, *Backchannel*, etc. The proposed model detects and predicts dialogue acts based on lexical, collocational, and prosodic cues, as well as on the discourse coherence. It develops a probabilistic integration of speech recognition with dialogue modeling, to improve both speech recognition and dialogue act classification accuracy.

The goal of this paper is twofold: 1) presets a comprehensive framework for modeling and automatic classification of DAs, and proposes a framework that provides a mathematically principled way to condition the speech recognizer on conversation context through dialogue structure; 2) presents results obtained with this approach on a large, widely available corpus of spontaneous conversational speech.

This paper follows a recent standard for shallow discourse structure annotation, the *Dialogue Act Markup in Several Layers (DAMSL)* tag set, which aims to provide a domain-independent framework for dialogue annotation.

The mathematical and computational framework used in the paper is the *hidden Markov model (HMM)*. Given all available evidence E about a conversation, the goal is to find the DA sequence U that has the highest posterior probability $P(U|E)$ given that evidence.

$$U^* = \arg\max_U P(U|E) = \arg\max_U P(U)P(E|U).$$

Next we will discuss the prior probability $P(U)$, and the posterior probability $P(E|U)$ respectively.

In modelling the prior distribution $P(U)$, the paper assumes that the distribution is Markovian, i.e., each U_i depends only on a fixed number k of preceding DA labels:

$$P(U_i|U_1, ..., U_{i-1}) = P(U_i|U_{i-k}, ..., U_{i-1}).$$

The paper trains standard backoff N-gram models, using the frequency smoothing approach. The authors also tried several alternative approaches, but there was no significant improvement over the simple N-gram method.

The computation of $P(E|U)$ depends on the types of evidence use. The paper used the following three types of sources of evidence, either alone or in combination:

- **Transcribed words**
  $P(W|U)$, where W refers to the true (hand-transcribed) words spoken in a conversation.
- **Recognized words**
  A standard approach is to use the 1-best hypothesis. A more thorough approach is to compute $P(A|U)$ (where A is the acoustics input) by decomposing it into an acoustic likelihood $P(A|W)$ and a word-based likelihood $P(W|U)$:

  $$P(A|U) = \sum_W P(A|W)P(W|U).$$
- **Prosodic features**
  $P(F|U)$, where F are the features capturing various aspects of pitch, duration, energy, etc., of the speech signal. For the prosodic classifiers, the paper used CART-style decision trees. One example in the corpus was the distinction between backchannels and agreement. The prosodic tree trained on this task is shown in Figure 0.4. The paper also tried to use neural network classifiers on this task, and tested various network structures. However, there was no encouraging result.

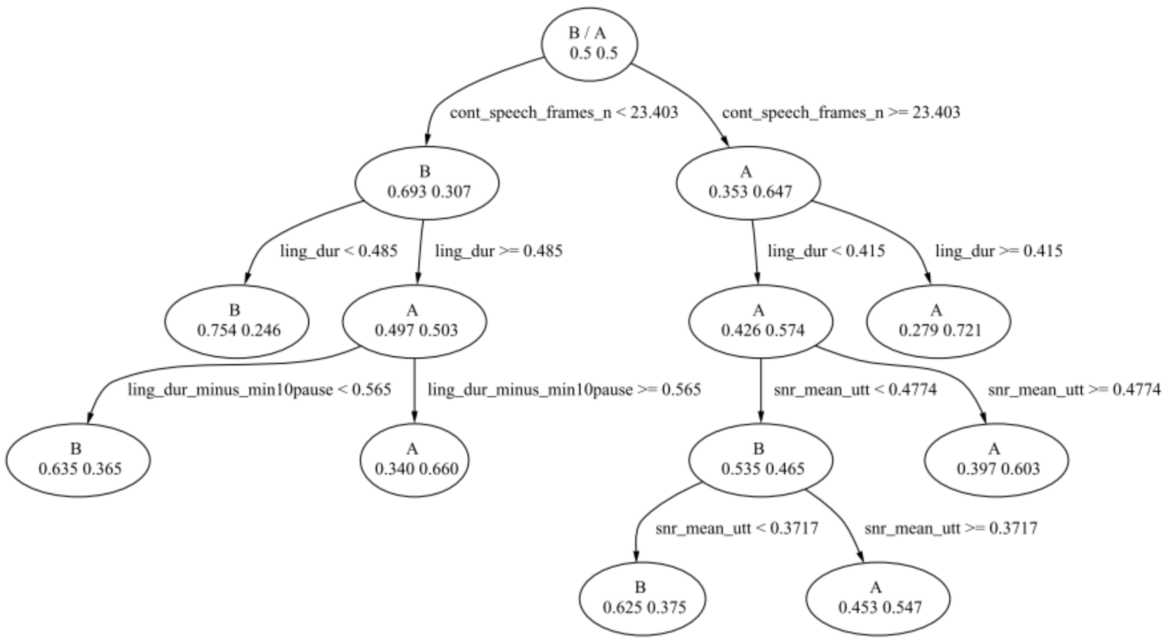


Figure 0.4: Decision tree for the classification of backchannels and agreements.

The paper proposes a method to combine multiple knowledge source, by using the following approximation:

$$\begin{aligned} P(A_i, W_i, F_i|U_i) &= P(A_i, W_i|U_i)P(F_i|A_i, W_i, U_i) \\ &\approx P(A_i, W_i|U_i)P(F_i|U_i). \end{aligned}$$

The HMM representation allows using efficient dynamic programming algorithms to compute relevant aspects of the model, such as: 1) the most probable DA sequence (the *Viterbi algorithm*); 2) the posterior probability of various DAs for a given utterance (the *forward-backward algorithm*).

The paper considers ways to use DA modeling to enhance automatic speech recognition (ASR). The first method is called *mixture-of posteriors*, which yields:

$$P(W_i|A_i, E) = \sum_{U_i} \frac{P(W_i|U_i)P(A_i|W_i)}{P(A_i|U_i)} P(U_i|E).$$

The second method, called *mixture-of-LM*, results in:

$$P(W_i|A_i, E) \approx (\sum_{U_i} P(W_i|U_i)P(U_i|E)) \frac{P(A_i|W_i)}{P(A_i)}.$$

The experiments confirmed that DA modeling can improve word recognition accuracy quite substantially in principle, at least for certain DA types.But the skewed distribution of DAs limits the usefulness of the approach on the Switchboard corpus. The paper suggests that the benefits of DA modeling might be more pronounced on corpora with more even DA distribution, which is typically the case for task-oriented dialogues.

0.6 Recurrent Convolutional Neural Networks for Discourse Compositionality [9]

This paper introduces both a sentence model and a discourse model corresponding to the two levels of compositionality: cords combine to form the meaning of sentences, and sentences combine to form the meaning of dialogues. The sentence model adopts convolution as the central operation, and is based on a novel *hierarchical convolutional neural network (HCNN)*. The discourse model extends the sentence model, and is based on a *recurrent neural network (RNN)*.

The basic kernel operation used in HCNN is described as follows. Given a sentence s and its paired matrix $\mathbf{M}^s$, let $\mathbf{m}$ be a feature that is a row in $\mathbf{M}^s$. Let $w_1, ..., w_k$ be a sequence of k weights. The local weighted addition over the first k values is:

$$y = w_1\mathbf{m}_1 + ... + w_k\mathbf{m}_k.$$

Then a kernel simply defines the value of k, and the one-dimensional convolution applies local weighted addition to each subsequence of values of $\mathbf{m}$. The one-dimensional convolution $(\mathbf{k} * \mathbf{m})$ is defined as:

$$(\mathbf{k} * \mathbf{m})_i = \sum_{j=1}^{k} \mathbf{k}_j \cdot \mathbf{m}_{k+i-j}.$$

To define the hierarchical architecture of the model, the paper next defines a sequence of kernel sizes and associated weights. Let l be the number of words in the sentence s. The sequence of kernel sizes is defined as

$$k_1^l = 2, k_{i+1}^l = k_i^l + 1, k_t^l = l - \sum_{j=1}^{t-1}(k_j^l - 1).$$

That is, kernel sizes increase by one until the resulting convolved vector is smaller or equal to the last kernel size (cf. Figure 0.5).

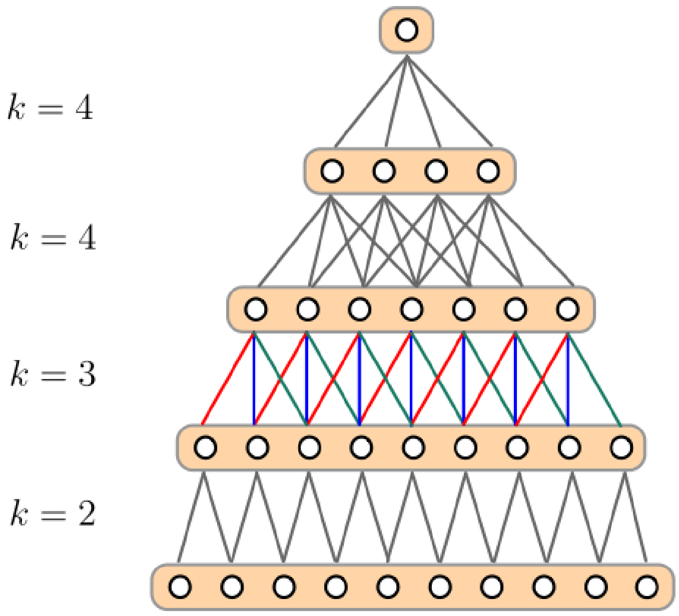


Figure 0.5: A hierarchical convolutional neural network for sentential compositionality.

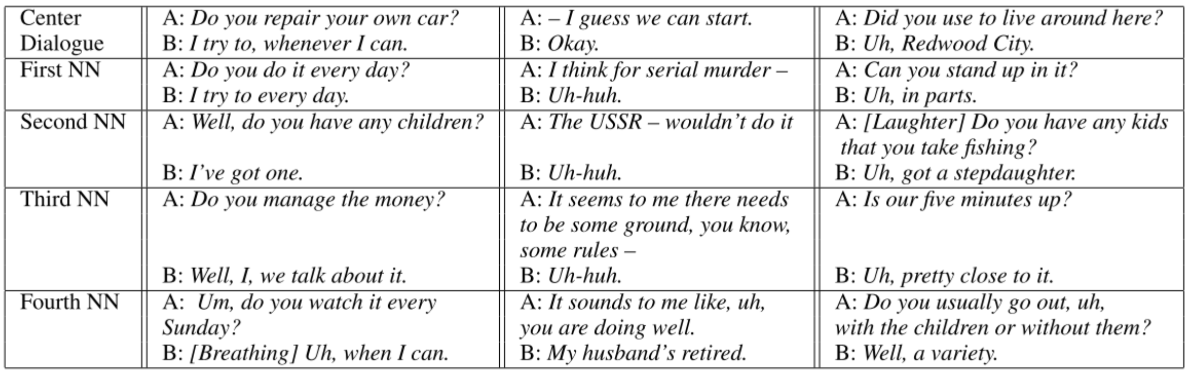
The composition operation in HCNN is formally defined as follows. For each feature f, let $\mathbf{K}_i^f$ be a sequence of t kernels, where the size of the kernel $|\mathbf{K}_i^f| = k_i^l$. The hierarchy of matrices and corresponding features are defined as:

$$\mathbf{M}_{f,:}^{1} = \mathbf{M}_{f,:}^{s},$$
$$\mathbf{M}_{f,:}^{i+1} = \sigma(\mathbf{K}_i^f * \mathbf{M}_{f,:}^{i} + b_f^i).$$

The discourse model adapts a RNN architecture in order to capture central properties of discourse. The proposed method aims to capture at least two of the most prominent properties: the sequentiality of the utterances, and the interactions between the speakers.

The RNN architecture takes inputs from a HCNN conditioned on the respective speakers. Let $s_1, ..., s_T$ be a sequence of sentences, each in turn being a sequence of words $s_i = y_1...y_l^i$. Let $x_1, ..., x_T$ be a sequence of labels, and $a_1, ..., a_T$ be a sequence of speaker. The RNN computes probability distribution by iterating the following equations:

$$\mathbf{h}_i = \sigma(\mathbf{I}x_{i-1} + \mathbf{H}^{i-1}\mathbf{h}_{i-1} + \mathbf{S}\mathbf{s}_i + b_h)$$
$$p_i = softmax(\mathbf{O}^i\mathbf{h}_i + b_o)$$



Center Dialogue	A: *Do you repair your own car?* B: *I try to, whenever I can.*	A: *– I guess we can start.* B: *Okay.*	A: *Did you use to live around here?* B: *Uh, Redwood City.*
First NN	A: *Do you do it every day?* B: *I try to every day.*	A: *I think for serial murder –* B: *Uh-huh.*	A: *Can you stand up in it?* B: *Uh, in parts.*
Second NN	A: *Well, do you have any children?* B: *I've got one.*	A: *The USSR – wouldn't do it* B: *Uh-huh.*	A: *[Laughter] Do you have any kids that you take fishing?* B: *Uh, got a stepdaughter.*
Third NN	A: *Do you manage the money?* B: *Well, I, we talk about it.*	A: *It seems to me there needs to be some ground, you know, some rules –* B: *Uh-huh.*	A: *Is our five minutes up?* B: *Uh, pretty close to it.*
Fourth NN	A: *Um, do you watch it every Sunday?* B: *[Breathing] Uh, when I can.*	A: *It sounds to me like, uh, you are doing well.* B: *My husband's retired.*	A: *Do you usually go out, uh, with the children or without them?* B: *Well, a variety.*

Figure 0.6: Short dialogues and nearest neighbours.

In the experimental study, the paper evaluates the proposed model with the prediction of dialogue acts within the *Switchboard Dialogue Act corpus.* The RCNN model achieves prediction accuracy of 73.9%, while the best previous result was 71.0%. Figure 0.6 shows three randomly chosen dialogues, and the four nearest neighbours of each.

0.7 Recurrent Neural Network based Language Model [10]

This paper proposes a new *recurrent neural network based language model (RNN LM)* with applications to speech recognition. Experimental results indicate that it reduces 50% perplexity by using mixture of several RNN LMs, and speech recognition experiments show around 18% reduction of word error rate on the Wall Street Journal task.

This paper uses an architecture that is usually called a *simple recurrent neural network*, which is probably the simplest possible version of RNN, and very easy to implement and train. Let the input to the network at time t be $x(t)$, output is denoted as $y(t)$, and $s(t)$ is the state of the network (hidden layer). Let $\oplus$ denote the concatenating operator. The simple recurrent neural network works by iterating the following equations:

$$\begin{aligned} x(t) &= w(t) \oplus s(t-1) \\ s_j(t) &= \sigma(\sum_i x_i(t) u_{ji}) \\ y_k(t) &= softmax(\sum_j s_j(t) v_{kj}) \end{aligned}$$

At each training step, error vector is computed according to cross entropy criterion and weighs are updated with the standard backpropagation algorithm.

The paper further suggests that the network should continue training even during testing phase, and refers to such model as *dynamic*. While in training phase all data are presented to network several times in epochs, dynamic model gets updated just once as it processes testing data. It is shown that such simple technique is enough to obtain large perplexity reductions against static models. Dynamically updated models can thus automatically adapt to new domains.

In the experimental study, the paper evaluates the proposed method on several standard speech recognition tasks. On the WSJ corpus, the proposed model reduces WER by 18% against 5-gram model with modified Kneser-Ney smoothing. Perplexity reductions are one of the largest ever reported (almost 50%). In another experiment on the NIST RT05 corpus, it is shown that the proposed model trained on just 5.4M words can outperform backoff models that are trained on hundreds times more data.

0.8 Unit Selection in Speech Synthesis [5]

This paper describes a new method for synthesizing speech by concatenating sub-word units from a database of labelled speech. The method is implemented within a full text-to-speech system, called the *Festival*, offering efficient natural sounding speech synthesis.

In the selection based synthesis, there is a large database of speech with a variable number of units from a particular class. The goal of these algorithms is to select the best sequence of units from all the possibilities in the database, and concatenate them to produce the final speech.

The basic idea of the unit selection technique proposed in the paper is summarized as follows: 1) A large units inventory is created by automatically clustering units of the same phone class based on their phonetic and prosodic context; 2) The appropriate cluster is then selected for a target unit; 3) Finally an optimal path is found through the candidate units based on their distance from the cluster center and an acoustically based join cost.

The first step is to cluster the units. The paper defines an acoustic measure to measure the distance between two units of the same phone type, by using an acoustic vector which comprises Mel frequency cepstrum coefficients, F_0, power, etc. The acoustic distance between two units is simply the average distance for the vectors (including some frames in the previous units). The distance measure is used by the CART algorithm, which builds a decision tree whose questions best minimise the impurity of the sub-clusters.

To join consecutive candidate units from clusters selected by the decision trees, it uses an optimal coupling technique to measure the concatenation costs between two units. This technique offers two results: the cost of a join, and a position for the join. Optimal coupling allows it to select more stable positions towards the center of the phone.

At synthesis time the input is a stream of target segments to be synthesized. For each target it uses the CART for that unit type, and ask the questions to find the appropriate cluster which provides a set of candidate units. Let $Tdist(U)$ be the distance of a unit U to its cluster center, and the function $Jdist(U_i, U_{i-1})$ be the join cost. The proposed method uses a Viterbi search to find the optimal path through the candidate units that minimizes the following expression:

$$\sum_{i=1}^{N} Tdist(U_i) + W * Jdist(U_i, U_{i-1}).$$

The paper further discusses the pruning technique, which can reduce the size of the distributed database. Pruning has two effects: 1) remove spurious atypical units which may have been caused by mislabelling or poor articulation in the original recording; 2) remove those units which are so common that there is no significant distinction.

In the experimental study, the paper tries the proposed method with two databases: 1) TIMIT, a male British English speaker (about 14,000 units); 2) Boston University FM Radio corpus, a female American news reader (about 37,000 units). In the test it generated 20 sentences for each evaluated model. The sentences from each model were played against those from another set, and the subjects decide which ones are better. In this way, the author conducted comparison study by varying cluster size, F_0 weight in the acoustic cost, and the amount to prune final clusters. The empirically optimal parameters were respectively reported.

References

[1] a. Mnih and G. Hinton, "Three New Graphical Models for Statistical Language Modelling," *Proceedings of the 24th International Conference on Machine Learning (2007)*, vol. 62, pp. 641–648, 2007.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A Neural Probabilistic Language Model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[3] A. W. Black and K. A. Lenzo, "Limited Domain Synthesis," *International Conference on Spoken Language Processing - ICSLP*, pp. 411–414, 2000.

[4] F. Malfrere and T. Dutoit, "High-Quality Speech Synthesis for Phonetic Speech Segmentation," *Eurospeech*, no. January 1997, pp. 2631–2634, 1997.

[5] A. W. Black and P. Taylor, "Unit Selection in Speech Synthesis," in *Eurospeech97*, pp. 601–604, 1997.

[6] K. M. Hermann and P. Blunsom, "Multilingual Distributed Representations without Word Alignment," *Iclr 2014*, pp. 1–9, 2013.

[7] Y. Bengio, *Learning Deep Architectures for AI*, vol. 2. 2009.

[8] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Ess-Dykema, and M. Meteer, "Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech," *Computational linguistics*, vol. 26, no. 3, pp. 339–373, 2000.

[9] N. Kalchbrenner and P. Blunsom, "Recurrent Convolutional Neural Networks for Discourse Compositionality," *ACL WS on Continuous Vector Space Models and their Compositionality*, pp. 119–126, 2013.

[10] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent Neural Network based Language Model," *Interspeech*, no. September, pp. 1045–1048, 2010.