

SPEECH RECOGNITION.....	2
1. OVERVIEW	2
2. HIDDEN MARKOV MODEL	5
3. ALGORITHM FOR HMM	7
4. APPROACH TO SPEECH RECOGNITION	7
4.1. <i>Introduction</i>	7
4.2. <i>String identifying</i>	10
4.3. <i>Criterion of reliability</i>	11
4.4. <i>Results</i>	12
5. ADDITIONAL REMARKS	13
5.1. <i>Multi-criterion evaluation</i>	13
5.2. <i>Phonetic uncertainty</i>	14
6. SOME THEORETICAL REMARKS	15
6.4. <i>Lexical Segmentation</i>	15
6.5. <i>Bigram and Trigram Specialty Language Models</i>	15
6.6. <i>Perplexity per word</i>	16
REFERENCES.....	17

Speech Recognition

Automatic Speech Recognition (ASR) is very useful for a lot of applications. Due to ASR it is possible to develop systems of control, decision-making, information exchange. ASR system converts speech (acoustic signal) into a text and then this text could be represented as a string of characters (letters) or utilized as a base to achieve text formatting or for taking certain action. In the latter case necessity of automatic speech understanding arises. On the other hand speech understanding helps to restore losses of information (restore input) and to transform speech to text successfully.

1. Overview

Speech processing can be divided into the following six categories:

1. Speech recognition, which deals with analysis of the linguistic content of a speech signal.
2. Speaker recognition, where the aim is to recognize the identity of the speaker.
3. Enhancement of speech signals, e.g. noise reduction,
4. Speech coding for compression and transmission of speech.
5. Voice analysis for medical purposes, such as analysis of vocal loading and dysfunction of the vocal cords.
6. Speech synthesis: the artificial synthesis of speech, which usually means computer generated speech.

The first of these categories, i.e. "Speech recognition" is under consideration. Research into speech recognition started in 1938, but the technology did not become sufficiently developed for commercial applications until the late 1980s. The early language-recognition systems had to make compromises: they were "tuned" to

be dependent on a particular speaker, or had small vocabulary, or used a very stylized and rigid syntax.

It has been said (in 1994) that computers will need to be something like 1000 times faster before large vocabulary (a few thousand words), speaker-independent, connected speech voice recognition will be feasible. But we can say now that the processing speed of speech recognition is sufficient and other problems are actual (though computers are not as fast as they said in 1994).

Most of researchers were attempting to recognize spoken language by providing contextual information, such as the speaker's identity, what the speaker knew, and what the speaker might be trying to say, in addition to rules of language.

Other approach was based purely on statistical relationships, such as the probability that any two or three words would appear one after another in spoken language. They created a phonetic dictionary with the sounds of different word groups and then set to work on an algorithm to decipher a string of spoken words based on phonetic sound matches and the probability that someone would speak the words in that order.

According to "Survey of the State of the Art in Human Language Technology (1997) by Ron Cole et al." *Speech recognition* is the process of converting an acoustic signal, captured by a microphone or a telephone, to a set of words. The recognized words can be the final results, for such applications as commands and control, data entry, and document preparation. They can also serve as the input to further linguistic processing in order to achieve text formatting or speech understanding. For example, speech recognition allows a user to use his/her voice as an input device; speech recognition may be used to dictate text into the computer or to give commands to the computer (such as opening application programs, pulling down menus, or saving work). In telephony, interactive voice response, or IVR, is a phone technology that allows a computer to detect voice and touch tones using a normal phone call. And speech recognition for IVR is normally used to carry out more complex transactions and simplifies the application menu structure.

According to other sources *Speech Recognition* (Speech Recognition, ASR, speech-to-text) is automated recognition of the content of speech for the purposes of representing it as text or taking an appropriate action or simply *identification of spoken words by a machine* and even *any technique by which a computer can understand ordinary speech*.

Sometimes they say about NLSR - Natural Language Speech Recognition. It is an advanced type of speech recognition which is intended to recognize particular words and phrases, but it can also interpret and assign meaning to those words and phrases.

Speaking modes are the following:

- Isolated Words,
- Connected Words,
- Continuous Speech.

Speech recognition systems can be characterized by many parameters as in the table below.

<i>Parameters</i>	<i>Range</i>
Speaking Mode	Isolated words to continuous speech
Speaking Style	Read speech to spontaneous speech
Enrollment	Speaker-dependent to Speaker-independent
Vocabulary	Small (< 20 words) to large (> 20,000 words)
Language Model	Finite-state to context-sensitive
Perplexity	Small (< 10) to large (> 100)
SNR (Signal-to-Noise Ratio)	High (> 30 dB) to low (< 10 dB)
Transducer	Voice-cancelling microphone to telephone

An isolated-word speech recognition system requires that the speaker pause briefly between words. This allows the machine to determine where one word begins and the next stops. This style of dictation is called discrete speech. Many people (especially those with learning disabilities) prefer these systems.

Connect word systems (or more correctly 'connected utterances') are similar to Isolated words, but allow separate utterances to be 'run-together' with a minimal pause between them. An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

There are continuous speech recognition applications which allow a user to dictate text fluently into the computer. While these systems do give the user system control they are not yet hands-free.

Spontaneous, or extemporaneously generated, speech contains disfluencies and is much more difficult to recognize than speech read from script. Some systems require speaker enrollment (a user must provide samples of his or her speech before using them) whereas other systems are said to be speaker-independent, in that no enrollment is necessary. Some of the other parameters depend on the specific task. Recognition is generally more difficult when vocabularies are large or have many similar-sounding words. When speech is produced in a sequence of words, language models or artificial grammars are used to restrict the combination of words. The simplest language model can be specified as a finite-state network, where the permissible words following each word are explicitly given. More general language models approximating natural language are specified in terms of a context-sensitive grammar. Many programs establish context through *trigram analysis*, a method based on a database of frequent three-word clusters in which probabilities are assigned that any two words will be followed by a given third word.

Speaker-dependent means that the system demands some patterns of individual speech for tuning in interactive initial mode. Then parameters might be corrected in process of usage for improvement of recognition.

Speaker-independent systems are intended for recognition of speech of any user. Thus influences of specific nature of individual speech is hardly might been kept in mind.

Two key problems of recognition of speech - achievement of absolute accuracy on the limited set of commands even for one announcer's voice and recognition of spontaneous speech independent of the announcer with comprehensible quality

One popular measure of the difficulty of the task, combining the vocabulary size and the language model, is perplexity, loosely defined as the geometric mean of the number of words that can follow a word after the language model has been applied. In addition, there are some external parameters that can affect speech recognition system performance, including the characteristics of the environmental noise and the type and the placement of the microphone.

High perplexity tasks with a vocabulary of thousands of words are intended primarily for the dictation application.

Speech recognition is a difficult problem, largely because of the many sources of variability associated with the signal.

1. The acoustic realizations of phonemes, the smallest sound units of which words are composed, are highly dependent on the context in which they appear. At word boundaries, contextual variations can be quite dramatic.
2. Acoustic variability can result from changes in the environment as well as in the position and characteristics of the transducer.
3. Within speaker variability can result from changes in the speaker's physical and emotional state, speaking rate, or voice quality. Finally, differences in sociolinguistic background, dialect, and vocal tract size and shape can contribute to across speaker variability.

Performance of speech recognition systems is typically described in terms of word error rate, E , defined as:

$$E = \frac{S + I + D}{N},$$

where N is the total number of words in the test set, and S , I , D are the total number of substitutions, insertions and deletions respectively.

They can consider also phoneme error rate (for transcriptions), defined as

$$P = \frac{E}{\bar{m}},$$

where \bar{m} is the average number of phonemes per word in the test set.

They can consider also character error rate, defined as

$$C_E = \frac{E}{\bar{m}},$$

where \bar{m} is the average number of letters per word in the test set.

2. Hidden Markov Model

Hidden Markov Model (HMM) is the underlying technology for speech recognition in the past years.

HMM is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. The extracted model parameters

can then be used to perform further analysis, for example for speech and other pattern recognition applications.

An HMM is a doubly stochastic model, in which the generation of the underlying phoneme string and the frame-by-frame, surface acoustic realizations are *both* represented probabilistically as Markov processes.

An interesting feature of frame-based HMM systems is that speech segments are identified during the search process, rather than explicitly. An alternate approach is to first identify speech segments, then classify the segments and use the segment scores to recognize words. This approach has produced competitive recognition performance in several tasks.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a HMM, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states.

In mathematics, a Markov chain is a discrete-time stochastic process with the Markov property. In such a process, only the current state is necessary for predicting a subsequent state or states — states prior to the current state are not needed if the current state is known.

A Markov chain describes at successive times the states of a system. At these times the system may have changed from the state it was in the moment before to another or stayed in the same state. The changes of state are called transitions. The Markov property means the system is *memoryless*, i.e. it does not "remember" the states it was in before, just "knows" its present state, and hence bases its "decision" to which future state it will transit purely on the present, not considering the past.

If the state space is finite, the transition probability distribution can be represented as a matrix, called the transition matrix, with the (i, j)'th element is equal to

$$P_{ij} = P(X_{ij} = j / X_n = i)$$

For a discrete state space, the integrations in the k-step transition probability are summations, and can be computed as the k 'th power of the transition matrix. That is, if P is the one-step transition matrix, then P^k is the transition matrix for the k-step transition.

A transition matrix which is positive (that is, every element of the matrix is positive) is irreducible and aperiodic. A matrix is a stochastic matrix if and only if it is the matrix of transition probabilities of some Markov chain.

A left stochastic matrix is square matrixes whose columns are probability vectors, i.e., the entries in each column are nonnegative real numbers whose sum is 1. Likewise, a right stochastic matrix is a square matrix whose rows are probability vectors. In a doubly stochastic matrix all rows and all columns are probability vectors. Stochastic matrices can be considered representations of the transition probabilities of a finite Markov chain.

Systems assume a sequence of input frames which are treated as if they were independent. But it is known that perceptual cues for words and phonemes require the integration of features that reflect the movements of the articulators, which are dynamic in nature. How to model dynamics and incorporate this information into recognition systems is an unsolved problem.

To improve recognition HMMs for triphones and biphones, i.e. contextually dependent manifestations of acoustic units could be useful.

3. Algorithm for HMM

Input for algorithm currently discussed is phoneme sequences, transition matrices and vocabulary; output means that words are corresponding for the sequences.

The Markov chain is one-step (first-order) as in our case, which means we have only two states. The first one is phoneme sequences of input for algorithm and the second one is state with genuine phoneme sequences (before transducer properties and other influences transform them). Therefore, computing of the most likely hidden sequence up to a certain point t must depend only on the observed event at point t , and the most likely sequence at point $t - 1$.

The **Viterbi algorithm** is a dynamic programming algorithm for finding the most likely sequence of hidden states and is used to find the most likely string of text given the acoustic signal and can be seen as an application of dynamic programming for finding a maximum probability path in a graph with weighted arcs. It is possible to use not only transition probabilities but also start and emission ones too.

HMMs can be classified according to the nature of the elements of the B matrix, which are distribution functions.

Distributions are defined on finite spaces in the so called *discrete HMMs*. In this case, observations are vectors of symbols in a finite alphabet of N different elements. For each one of the Q vector components, a discrete density $\{w(k) \mid k = 1, \dots, N\}$ is defined, and the distribution is obtained by multiplying the probabilities of each component. Notice that this definition assumes that the different components are independent. Computation of probabilities with discrete models is faster than with continuous models (continuous and semi continuous HMMs are beyond our consideration). For large vocabularies, search is performed in two steps. The first generates a word lattice of the n -best word sequences with simple models to compute approximate likelihoods in real-time. In the second step more accurate likelihoods are compared with a limited number of hypotheses. Some systems generate a single word sequence hypothesis with a single step. The search produces a hypothesized word sequence if the task is dictation.

4. Approach to speech recognition

4.1. Introduction

The task of design could be written down as follows:

$$F(V_l, V_c, r, t, I_q) \rightarrow \min_D,$$

$F(\cdot)$ – cost of development,
 V_l – current vocabulary length,
 V_c – current vocabulary content,
 C – complexity of common vocabulary V for creating a current vocabulary content, $C = C(l, c)$ – function of length l and content c of common vocabulary,
 $r = r(V_l, V_c, I_q)$ – reliability of recognition,
 $t = t(V_l)$ – recognition time,
 I_q – input quality,
 D – range of $F(\cdot)$ definition .

We can point that:

$$\begin{aligned}
 \frac{\partial F}{\partial V_l} &\geq 0, & \frac{\partial F}{\partial C} &\leq 0, \\
 \frac{\partial F}{\partial r} &> 0, & \frac{\partial F}{\partial t} &\leq 0, \\
 & & \frac{\partial F}{\partial I_q} &\leq 0.
 \end{aligned}$$

There are two equivalent tasks below:

$$\begin{aligned}
 &r(V_l, V_c, I_q) \rightarrow \max, \\
 &V_l \geq l, \\
 &V_c \subseteq V, & (1) \\
 &t \leq t_0, \\
 &I_q \geq I_0, \\
 &F(V_l, V_c, t, I_q) \leq f_0
 \end{aligned}$$

$$\begin{aligned}
 &V_l \rightarrow \max, \\
 &r \geq r_0 \\
 &V_c \subseteq V, & (2) \\
 &t \leq t_0, \\
 &I_q \geq I_0,
 \end{aligned}$$

$$F(V_c, r, t, I_q) \leq f_0$$

Let's assume that speech recognition parameters are the following:

<i>Parameters</i>	<i>Value</i>
Speaking Mode	Connected utterances
Speaking Style	Read speech
Enrollment	Speaker-independent
Vocabulary	Large (> 20,000 words)
Language Model	Finite-state
Perplexity	Small (< 10)
SNR (Signal-to-Noise Ratio)	High (> 30 dB)
Transducer	Telephone

Let

$$X = \{x_i\}, i = 1, 2, \dots, m$$

- is phoneme sequence (word) on input and

$$Y = \{y_j\}, j = 1, 2, \dots, n$$

- is phoneme sequence on output.

The problem is reconstruction of unknown X with the use of Y and vocabulary V . A vocabulary V is a set of words (utterances)

$$V = \{V_k\}, k = 1, 2, \dots, K$$

and

$$X \in V$$

To solve the problem we should find a distance D between set V and word Y .

A distance between two sets A with elements A_i and B with elements B_j is defined as

$$D = \min_{i,j} d(A_i, B_j),$$

where d is a distance between elements A_i and B_j

In our case

$$D = \min_k d(Y, V_k),$$

and

$$W = \operatorname{argmin}_k d(Y, V_k)$$

is the word we tried to find.

W will be equal to X with probability not equals to 1 because of errors of initial values, that is Y .

Let r to be maximal length of V_k and $s = 2r$.

For identifying X we should choose a way to compute a distance d . There are two ways for computing distance of two strings with different length:

- Levenstein distance, which puts in correspondence substitution, insertion and deletion a cost 1 and
- editor distance, which evaluates insertion and deletion as 1 and substitution as 2.

We see now that:

1. Distance is integer;
2. Maximal possible distance value is not greater than s (editor distance) and is not greater than r (Levenstein distance).

Thus, we can divide a vocabulary V on either s or r groups G_d of words corresponding with a distance $d = 0, 1, \dots, r$ or s from Y to V . Let G_M is a group, which consists of one or more words and d is minimal. Quantity of words in groups varies in wide range and depends on length of the output string Y , a distance d , and a number of groups. As a rule, the longer Y , the fewer words' quantity in G_1 .

One of the ways to improve differentiation is to use weighted distance: quantity of possible values of distance sufficiently increases. Therefore, a quantity of words in G_M -group might be less then in non-weighted distance case.

Weights should be built with the use of experiments and then done statistical data analysis.

Example. For phoneme sequence *xet* there are 18 clauses in vocabulary for the only substitution of the first phoneme for the non-weighted distance and 5 for the weighted one, such as *set, let, met, get, jet*.

For *zet* we have 18 and 1 (*set*) respectively.

4.2. String identifying

After a number of experiments, they obtain the following formula for weights:

$$w = (1 - k_1)^p (1 - k_2),$$

where w - weight (value, cost) of distance between phoneme a and b ,
 k_1 - frequency of a -to- b transition,
 k_2 - frequency of event when b was obtained from a .

If a / b is null-phoneme then we deal with insertion / deletion.

To compute weighted distances a dynamic programming algorithm of Vagner-Fisher was applied:

$$d_{i,j} = \min\{d_{i-1,j} + w(x_i, \delta), d_{i,j-1} + w(v, y_j), d_{i-1,j-1} + w(x_i, y_j)\},$$

$$d_{0,0} = 0$$

$$d_{i,0} = \sum_{k=1}^i w(x_k, \delta), \text{ if } 1 < i < m$$

$$d_{0,j} = \sum_{k=1}^j w(v, x_k), \text{ if } 1 < j < n$$

where

- $w(a, \delta)$ – a cost of deletion,
- $w(v, b)$ – a cost of insertion,
- $w(a, b)$ – a cost of substitution,
- $d_{i,j} = d(x(1, i), y(1, j))$ – a distance between substrings $x_1x_2\dots x_m$ and $y_1y_2\dots y_n$

The process of getting transition frequencies was similar to the one for **HMM** construction, but we could not speak about probabilities because of lack of statistical data.

Let there are transition matrices P with probabilities p_{ij} of phoneme transition obtained with **HMM**.

Let Q - is matrices with elements $q_{ij}=1-p_{ij}$.

Let q_{ij} - are weights, then we deal with distance as it was described above and develop approach with use of transition matrices.

It is assumed that we know a precision of calculating probabilities (weights). We suppose that this precision is no more than two digits after decimal point.

For weighted distance, precision is no more than two significant digits.

Thus, a quantity K of groups might be approximately 100 as for our evaluation.

4.3. Criterion of reliability

There is no loss of generality in supposing that weights are integer. The assumption allows us to use distance values as was done in Levinstein / editor distance case.

Let a word detected to be X and $X = V_p$.

We take it that detecting of X is **reliable** if $d(X, V_k) > h$, where $h > 0$, for all $k = 1, 2, \dots, K, k \neq p$.

If $h = 0$, i.e. G_M consists of more than one member, detecting is **non-reliable**, but could be treated as *good-enough* if quantity S of members of groups $G_M, G_{M+1}, \dots, G_{M+s}$ is small, s – is the integer positive number.

We can consider h as a level of reliability. The level is low if $h = 1$. The value of S is to be taken into account for all cases of small h .

Error rate E is an average evaluation of system quality.

For error rate of single word, we could suppose that for $E > 0.2$ the reliability is very low in most cases, especially for $N < 6$.

4.4. Results

The task of diminishing of large vocabulary (over 20000) to small (under 500 words) for utterance of continuous speech was successfully solved with use of the next approach. In other words this approach allows generating a word lattice of the *n-best* word sequences.

An algorithm of Vagner – Fisher for calculation of Levenshtein distance and an algorithm of fuzzy search of Landau-Vishkin were modified for calculation of maximal probability. Let us say that we obtain the 1st and the 2nd algorithm respectively. The results of the 2nd algorithm application were recalculated with formula which takes in account a result of statistical evaluation of probabilities of phonemes transformation. Parameters of the 1st algorithm application are defined after statistical evaluation of a large vocabulary.

Let us consider all steps of diminishing process.

1. Do statistical analysis of big dictionary to obtain a number n – a length of interval inside phoneme string for the 1st algorithm application.
2. Apply the 1st algorithm. Let m_1 is a quantity of intervals for 1st algorithm application; m_2 is length of a list of the best candidates for inclusion in small dictionary. Thus a list of $m_1 \times m_2$ words will be obtained.
3. Analyze the IDS-matrix (the matrix with probabilities of deletion, insertion, substitutions; it is the right stochastic matrix) and detect a coefficient for a recalculating formula.
4. Apply the 2nd algorithm to input string and a large dictionary and recalculate all results with the recalculating formula. Choose m_3 best words.
5. Join $m_1 \times m_2$ words which were obtained after Step 2 with the m_3 ones of the 4th step and delete repetitions. The rest is a small dictionary.

Let t_1 is duration of the 1st algorithm application to one interval, l_1 is a length of phoneme string and l_2 is a length of interval. Then duration of the 2nd algorithm application is

$$t_2 \approx t_1 \frac{l_1}{l_2}$$

and $t \approx t_1 \left(m_1 + \frac{l_1}{l_2} \right)$ is total duration of diminishing process.

Let U is intersection of small dictionary and speech test, q is length of U , and Q is quantity of different words of speech test.

Now we can put a task of optimization:

$$S(m_1, m_2, m_3, t, a) \rightarrow \min,$$

$$\begin{aligned} t &\leq T, \\ a &\geq a_0, \end{aligned}$$

where S – is a small dictionary length,
 $a = q/Q$ - is accuracy or coverage.

5. Additional remarks

5.1. Multi-criterion evaluation

The need of non-standard decision making arises when decision criterion indicates low reliability (see 4.3.). In that case we should use more than one criterion to make a choice among some words with close or equal values of decision criterion.

Let the problem be the comparison of some similar objects O_k , $k = 1, 2, \dots, m$ with characteristics Y_{ki} , $i = 1, 2, \dots, n$.

Without loss of generality we can assume that the less a value of characteristic the better a quality of an object.

We need to make an optimal choice. But how should we come to determination?

If there is an object O_M , $Y_{Mi} \leq Y_{ki}$ for every k and i , $k \neq M$ and there is at least one i_k for every k ,

$$Y_{Mi_{i_k}} < Y_{ki_{i_k}},$$

then the answer is evident: this point is the best. But obviously the situation is not as described above.

We can separate all characteristic vectors (points in characteristic space) into two groups: Pareto points (or points of agreed optimum) and non-Pareto ones.

Definition. If Y_p is *Pareto point* then there are no points Y_k , $k \neq M$, $Y_{ki} \leq Y_{pi}$ for every i and one of inequalities is strong.

Thus each point of the first group is neither better nor worse when compared between each other.

Example. $X = (1;2)$, $Y = (0;3)$, $Z = (2,3)$. X and Y – are Pareto points, Z – non-Pareto one.

If we have only one Pareto point then this point is Y_M – the point of global optimum.

On the other hand each point of the second group is worse than at least one point of the first group. So, the second group possesses no interest and only the Pareto points would be under our consideration.

There are some ways of comparing Pareto points.

1. Hierarchy: they assign a place in hierarchy for each criterion. For example, the first criterion is the main one, the second is main one among all criteria except the first one, and the third criterion is main one after the first and the second and so on.

Then we should compare consequently all Pareto points, choose points with minimal value of the first criterion and then choose among these points the ones with minimal value of second criterion and so on until only one point is left or all criteria were considered.

2. Convolution: they consider a generalized additive criterion

$$\Phi_{\Sigma}(Y_k) = \sum_{i=1}^n \alpha_i Y_{k i}$$

or generalized multiplicative criterion

$$\Phi_{\Pi}(Y_k) = \prod_{i=1}^n \alpha_i Y_{k i}$$

We should minimize additive or multiplicative criterion with choice of Y_k .

Weight coefficients α_i might be obtained during an experiment or after expertise. In addition to evaluations' aims coefficients α_i are needed for reducing ingredients to dimensionless form.

5.2. Phonetic uncertainty

Let us consider some words and their transcription (for English):

1. English.
 - a. Two – [tu];
 - b. To – [tu];
 - c. Too – [tu:].
2. Hebrew.
 - a. אני
 - b. עני
3. Hebrew.
 - a. אושר
 - b. עושר

It is examples of phonetic uncertainty. There no methods to differentiate words 1a, 1b, 1c; 2a, 2b; 3a, 3b by use of phoneme sequences (to reconstruct a word notation only on base of its phoneme representation). It is possible to determine a real notation with use of context analysis which might be very sophisticated often. Sometimes even the context analysis hardly attains its goal (for example, case 3 for many sentences).

For our opinion it would be useful to suggest to advanced users an option to read reconstructed phoneme sequences.

Some theoretical remarks

The basic problems worth on a way of development of systems recognition of speech:

- Great volumes of dictionaries.
- Patterns of continuous speech.
- Various accents and pronunciations.
- A problem of branch of a speech signal from a background.

5.4. Lexical Segmentation

In all natural languages, the meaning of a complex spoken sentence (which often has never been heard or uttered before) can be understood only by decomposing it into smaller *lexical segments* (roughly, the words of the language), associating a meaning to each segment, and then combining those meanings according to the grammar rules of the language. The recognition of each lexical segment in turn requires its decomposition into a sequence of discrete *phonetic segments* and mapping each segment to one element of a finite set of elementary sounds (roughly, the phonemes of the language); the meaning then can be found by standard *table lookup* algorithms.

For most spoken languages, the boundaries between lexical units are surprisingly difficult to identify. In normal speech, one typically finds many consecutive words being said with no pauses between them, and often the final sounds of one word blend smoothly or fuses with the initial sounds of the next word.

Moreover, an utterance can have different meanings depending on how it is split into words. A popular example, often quoted in the field, is the phrase *How to wreck a nice peach*, which sounds very similar to *How to recognize speech*. As this example shows, proper lexical segmentation depends on context and semantics which draws on the whole of human knowledge and experience, and would thus require advanced pattern recognition and artificial intelligence technologies to be implemented on a computer.

5.5. Bigram and Trigram Specialty Language Models

For languages with poor morphology, such as English, word n-gram models are the most common statistical language models used in speech recognition today. And the most widely used statistical model is the trigram model, which estimates the likelihood of a word solely on the identity of the preceding two words in an utterance. Speech recognition programs are really just playing the odds to predict what you actually said. They narrow down what you say based on the sounds you make and how high on the vocabulary scale the words are that match the sounds. A short list of possible words is then compared to a data set (the language model) and the speech program further narrows down what it *thinks you meant* using trigrams based on the context of the language model.

For example, there are trigram vocabularies for Dragon NS Medical.

To estimate the bigram probabilities from a text we must count the number of occurrences of the word pair (w_{n-1}, w_n) and divide that by the number of occurrences of the preceding word w_{n-1} . This is a relatively easy computation and accurate

estimates can be obtained from transcriptions of language similar to that expected as input to the system. In general each application requires the language model to be fine tuned to the language input expected.

To estimate the trigram probabilities we must count the number of times the triple of (w_{n-2}, w_{n-1}, w_n) is observed and divide this by the number of times the pair (w_{n-2}, w_{n-1}) occurs. The problem here is clearly that for many triples the number of occurrences is likely to be very low and so reliable estimates of the trigram probability are unlikely.

To overcome this paucity of data the technique of language model smoothing is used. Here the overall trigram probability is derived by interpolating trigram, bigram and unigram probabilities:

$$P(w_n|w_{n-2}, w_{n-1}) = k_1 * f(w_n|w_{n-2}, w_{n-1}) + k_2 * f(w_n|w_{n-1}) + k_3 * f(w_n)$$

Where the functions $f()$ are the unsmoothed estimates of trigram, bigram and unigram probabilities. This means that for a triple which does not occur in the training text, the estimated probability will be derived from the bigram model and the unigram model; the estimate will be non-zero for every word included in the lexicon (i.e. every word for which there is an estimate of $P(w)$). The choice of the parameters k_1 , k_2 , k_3 is another optimization problem.

Another approach for solving a problem of predicting the next word on words already seen and data sparseness is connected with use of smoothing technique based on randomly grown Decision Trees and interpolated Kneser-Ney smoothing.

For agglutinative languages, they use a method based on the Minimum Description Length principle to split words statistically into subword units.

5.6. Perplexity per word

In natural language processing, perplexity is a common way of evaluating language models. A language model is a probability distribution over entire sentences or texts.

Using the definition above, one might find that the average sentence x_i in the test sample could be coded in 190 bits (i.e., the test sentences had an average log-probability of -190). This would give an enormous model perplexity of 2^{190} per sentence. However, it is more common to normalize for sentence length and consider only the number of bits per word. Thus, if the test sample's sentences comprised a total of 1,000 words, and could be coded using a total of 7,950 bits, one could report a model perplexity of $2^{7.95} = 247$ per word. In other words, the model is as confused on test data as if it had to choose uniformly and independently among 247 possibilities for each word.

The lowest perplexity that has been published on the Brown Corpus (1 million words of American English of varying topics and genres) is indeed about 247 per word, corresponding to a cross-entropy of $\log_2 247 = 7.95$ bits per word or 1.75 bits per letter. It is often possible to achieve lower perplexity on more specialized corpora, as they are more predictable.

The error rate of a speech recognizer is no less than the percentage of spoken words that are not in its vocabulary V . So a major part of building a language model is to select a vocabulary that will have maximal coverage on new text spoken to the rec-

ognizer. A corpus of text is used in conjunction with dictionaries to determine appropriate vocabularies. A tokenizer (a system that segments text into words) is needed. Then a unigram count for all of the spellings that occur in a corpus is determined. Those words that also occur in the dictionary are included. In addition a human screens the most frequent subset of new spellings to determine if they are words.

* * *

No word has ever been said exactly the same way twice, so the recognizer is never going to find an exact match. And for any given segment of sound, there are very many things the speaker could potentially be saying. The quality of a recognizer is determined by how good it is at refining its search, eliminating the poor matches, and selecting the more likely matches. A recognizer's accuracy relies on it having good language and acoustic models, and good algorithms both for processing sound and for searching across the models.

Experiments show, that in a continuous speech signal at a usual level of noise (offices, apartments) only 30 percent of the information of an initial signal reaches a listener. 70 percent are physically lost.

It is not enough dictionary information and knowledge of morphology for restoration of a signal. Nevertheless, the listener-person restores the told words almost completely. He takes the lost 70 percent from so-called "knowledge a priori".

Semantic technologies (model of knowledge of the world) integrated with qualitative technologies of recognition of a sound signal will really allow to promote cardinally in a problem of recognition of spontaneous speech.