This paper is an assignment in the course 729G11 and gives a brief introduction to what speech recognition is. It brings up a few different ways of modeling speech, as well as the ideas of two algorithms necessary for the models. It explains prosody in general, as well as its use in modern speech recognition systems.
# Table of contents

INTRODUCTION .................................................................................................................. 1

TERMINOLOGY .................................................................................................................. 2

PROSODY BASICS ............................................................................................................. 3
  A General Definition ......................................................................................................... 3
  Fundamental Frequency ................................................................................................... 3
  Energy ............................................................................................................................... 3
  Duration ............................................................................................................................ 4
  Shortly on Differences between Languages .................................................................. 4

GENERAL CONCEPT OF SPEECH RECOGNITION ..................................................... 5
  Frame Representation ..................................................................................................... 5
  Templates .......................................................................................................................... 6
  Issues in Pattern Recognition .......................................................................................... 7

MODELING SPEECH RECOGNITION ............................................................................ 9
  Probabilistic Model .......................................................................................................... 9
  Language Models .............................................................................................................. 10
  Markov Models ................................................................................................................. 11
  The Forward Algorithm ................................................................................................... 12
  Hidden Markov Models (HMMs) .................................................................................... 13
  Briefly on the Viterbi Algorithm ..................................................................................... 14

USE OF PROSODY IN SPEECH RECOGNITION ...................................................... 17

DISCUSSION ..................................................................................................................... 19

WORKS CITED .................................................................................................................. 21
INTRODUCTION

Trying to talk sense to synthesized speech agents over customer service has at some point got many of us screaming with impatience over their evident incapability to understand what we are saying. Seemingly oblivious to our distress, they ask us once more “Did you mean 'heating system’?” when we for minutes have been trying to tell them we want to speak with someone about the dripping shower in the bathroom. Last time I called to have someone from my internet deliverer look at the wall sockets, the synthesized speech agent surprised me by asking could I repeat that again, she had problem hearing what I said. It sounded so human, that I for a moment forgot I was talking to a program.

The aim of this paper is to introduce attempts at making communication easier between humans and machines, bridging the gap where machines have a hard time reading, as well as conveying, emotions and cues given by intonation in human speech. The concept of melody and rhythm in speech, called prosody, is being implemented in many speech recognition systems today, enhancing the ability to recognize ambiguous meanings, accent and so on in verbal interaction. We will explore prosody and its role in speech recognition, as well as automatic speech recognition (ASR) in general, and discuss in short, what the future may hold for prosody in artificial intelligence.
TERMINOLOGY

Below is a brief explanation of the terms used in this paper.

*corpus*  A large and structured set of texts (Wikipedia, 2012-09-29). Commonly sampled from books, blogs or websites, among other types of media sources

*phone*  The smallest distinguishable unit in a stream of spoken language, pronounced in a defined way

*phoneme*  A small unit to distinguish between meanings, pronounced in one or more ways

*semantics*  Broadly defined, the study of meaning of linguistic expressions

*syllable*  A sound unit consisting of a sonority peak, usually a vowel, sometimes with consonants around, e.g. syllable

*syntax*  A pattern or structure for a language. Words can be validly organized into it to form sentences

The definitions above were collected from online linguistic dictionaries which can be found under *Cited Works.*
PROSODY BASICS

A General Definition
Prosody is the way we alternate melody and rhythm to add different meanings and emotional tone to our verbal utterances. It is the complementary to syntax and semantics – it is what gives meaning beyond structural and lexical meaning. Prosody also gives us other information about the speaker, such as gender, age and overall condition (slurred drunken talk, for example). I will bring up phones quite a few times in this paper, and want to begin by saying that prosody does not run over singles phones, but longer linguistic units – words, phrases and sentences, for example. It is also called the study of suprasegmental phenomena. (Schmandt, 1994) In most literature, there are three main prosodic features: fundamental frequency, energy and duration.

Fundamental Frequency
The sound of speech starts in our vocal chords. The vocal chords vibrate with a fundamental frequency, which rises and falls as we speak, causing pitch and intonation. We can use intonation to set a mood for a sentence, separate statements from questions. For short, we call this frequency $F_0$. A rising $F_0$ in “see you next Monday” suggests a question, whereas a falling $F_0$ implies a statement. (Ladd, 1996)

Energy
Energy can also be expressed as relative prominence and means stressing certain words. It can change the meaning within the context and so can phrase grouping, that is grouping words together to include or exclude them to an attribute. Wang gives the example of “old men and women” where “old men | and women” exclude the women from the attribute “old”, while “old | men and women” include both groups. (Wang, 2001)
**Duration**

Duration refers to the relative length and stress of segments in an utterance. It can be the length of a spoken segment or a pause. To measure $F_0$ and energy, one has first to measure the duration of a segment up for analysis.

**Shortly on Differences between Languages**

Intonation, relative prominence and phrase grouping varies in importance between languages. In English, stress is crucial in providing meaning to a sentence, but non-native speakers might not have the same feeling for it as it is not used in their tongue. Understanding Mandarin, which is a tonal language, requires analysis of $F_0$ patterns in syllables. (Wang, 2001)
GENERAL CONCEPT OF SPEECH RECOGNITION

To analyze speech automatically, we first have to separate the meaningful signals from other auditory disturbances. We can do this by setting a sound threshold, excluding from our analysis the acoustic elements which are not discernible from background noise. When we have done this, it is time to break the remainders into manageable pieces for our system to match against word templates saved in a vocabulary. The representation, templates and pattern matching are the three cornerstones of all speech recognition systems.

In figure 1, Jurafsky and Martin shows us a simple visualization of this process.

![Diagram](image)

*Figure 1. The noisy channel could among other things be different pronunciations or an interfering medium, e.g. a bad telephone connection. (Jurafsky & Martin, 2000)*

**Frame Representation**

The incoming acoustic signal is received and digitalized by the recognizer. It then goes through a process where frames, containing distinct articulatory features (information about resonance and voice) are in many cases extracted every 20 millisecond, which makes 50 frames every second, each 20 milliseconds long. The recognizer starts extracting frames when the signal exceeds a threshold where it is distinguishable from background noise, and it stops when the signal again fades away. It is important to wait for a short time before stopping the extracting process, as some words have short periods of silence or low energy in them (e.g. a stop consonant). Capturing these short pieces of the signal reduces the amount of information to only the most essential. The word templates in our system vocabulary are also stored as frames. (Schmandt, 1994)
Templates

To store a word template, we must simply have people say the word, and let that acoustic signal go through the process described under Frame Representation. Simpler recognizers use templates where the word is pronounced by only one user, whereas more robust systems want to train their templates to cover as many different pronunciations as possible. Some recognizers allow for retraining templates. This comes in handy if the vocabulary should change, or the first round of pronunciations was badly recorded. The old information in the template is then replaced by the new. (Schmandt, 1994)

One of the most common ways to store different pronunciations of the same word is to model it as a weighted automaton. Conceptually, it is a succession of nodes with weighted arcs leading from the first node of the word to the last. Each node then represents a phone. The succession is weighted, because all possible ways to pronounce the word are included in the model, and the weighted arcs represent the probability of one node leading to another. (Jurafsky & Martin, 2000) See figure 2.

![Figure 2](image-url)

*Figure 2. A weighted automaton as shown in (Jurafsky & Martin, 2000).*

This kind of model is also known as a Markov Model and is one of the more robust ways of representing words in a vocabulary. I will elaborate more on this in *Markov Models and HMMs.*
**Issues in Pattern Recognition**

Before we dive into the actual explanations of the pattern matching process, let us have a look some issues in the matching process, so as to get some perspective on the difficulties that must be solved. Schmandt states three main problems:

1. **Substitution**: an observation (word) is wrongly interpreted as a similar-sounding word, e.g. “on” is interpreted as “an”.
2. **Deletion**: an observation does not pass the amplitude threshold and is excluded from the analytical process. Usual for the first word of the sentence, e.g. “the mat”.
3. **Insertion**: a word is included in the sentence, though it was not originally in the observation, e.g. “the a mat”. (Schmandt, 1994)

Ellis and Mandel illustrates this as shown below in Figure 3:

<table>
<thead>
<tr>
<th>Reference:</th>
<th>THE</th>
<th>CAT</th>
<th>SAT</th>
<th>ON</th>
<th>THE</th>
<th>MAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognized:</td>
<td>-</td>
<td>CAT</td>
<td>SAT</td>
<td>AN</td>
<td>THE</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MAT</td>
</tr>
</tbody>
</table>

![Diagram](image)

*Figure 3. A visual representation of common ASR mistakes. (Ellis & Mandel, 2006)*

The error quota grows with the volume of speech being analyzed. Ellis and Mandel describe in this formula the error percentage of a speech recognition analysis:

\[ WER = \frac{(D + S + I)}{N} \]

where \( WER = \) word error rate consists of \( D = \) deletion, \( S = \) substitution, \( I = \) insertion and \( N = \) number of analyzed words. (Ellis & Mandel, 2006)
Russel and Norvig also define three main issues in ASR. These relate to, although are not equal to, the issues above.

1. **Segmentation:** we rarely pause between every word in fluent speech, rather we let the words tumble out more like one long word than several short ones. Of course, this causes problems in speech analysis, because it is difficult to recognize the end of one word and the beginning of next. A classic sentence is “How to recognize speech” which easily becomes “How to wreck a nice beach”.

2. **Coarticulation:** the phenomena when we let one syllable influence the pronunciation of the next. Consider the word “coarticulation” – when you say it fast, the o and a become one sound, although they are to separate syllables in the word.

3. **Homophones:** words, or parts of words, that sound similar. The words to, too and two are good examples of this – without context, it is impossible to tell them apart. So naturally, we want to implement a consideration for context in our system. This is useful to prevent wrong interpretations of ambiguous words as well. (Russel & Norvig, 2009)

None of these issues are easy to solve, but there have been more or less successful attempts to provide solutions for all of them. Next section will explain the aim of the robust, probabilistic way of modeling ASR.
MODELING SPEECH RECOGNITION

**Probabilistic Model**

Pattern matching is easiest performed by matching probabilities. We want to calculate the probability $P$ of our observation $O$ corresponding to a certain word $w$ from our vocabulary $V$. Remember the observation is a vector of articulatory features which we have extracted from the audio signal, perhaps the phone [ni]. When we have compared probabilities of different words, we decide on the most probable word $\hat{w}$ to correspond to our observation. If all probabilities are very low, we might not find a credible correspondence.

We can express this as

$$\hat{w} = \arg\max_{w \in V} P(w|O)$$

where $\arg\max$ stands for the maximized probability. The formula looks intuitive enough, but $P(w|O)$ is difficult to compute. If we however calculate it as a product of two probabilities, we can apply Bayes' theorem and get

$$P(w|O) = \frac{P(O|w)P(w)}{P(O)}$$

We can estimate the probability of each word $w$ through its frequency in our vocabulary. This is called the **language model**, which I will explain in further detail later on. And before we concern ourselves with computing $P(O)$, we observe that no such calculation is necessary. As we maximize over all words, we will always have the same observation $O$ when we ask for the most probable word string and the probability of $O$ remains constant. So we scratch that, and end up with the formula

$$\hat{w} = \arg\max_{w \in V} P(O|w)P(w)$$
and are rather happy with the simplicity of it. The only part left to estimate is \( P(O|w) \) which we can compute through a probabilistic acoustic model. (Jurafsky & Martin, 2000)

Acoustic models are of great importance in all speech recognition. They “describe how a sequence or multi-sequences of fundamental speech units (such as phones or phonetic feature) are used to represent larger speech units such as words or phrases which are the object of speech recognition” (Microsoft Research, Acoustic Modeling) and we use them to find the correct word given an input. We will have a closer look at the most broadly used types, the Markov Models and the Hidden Markov Models or HMMs, after we learn what a language model really is.

**Language Models**

The aforementioned language model is simply a listing of frequencies in which the words in our vocabulary appear in everyday language use, that is in the context of the corpus we are using. Of course, we want the corpus to be as large in volume and variety as possible, so as to have realistic probability values even for uncommon words. There are language models for one, two or more words in a succession. When simply checking how frequently a word appears in a corpus, we call the model a unigram. When we want to know the probability of “book” following “text” (as in “text book”), we use a bigram – checking the following word for every “text” we can find in our vocabulary. Surely, “text book” must occur more often than “text cook”, for example. This applies for three words in succession (trigram) or any number of words likely to follow each other (n-gram) and comes in handy when we have trouble distinguishing between phones. Remember we combine the language model and the acoustic model in our probabilistic model - if our acoustic model is confused as to whether the observation sequence is to be interpreted as “cook” or “book”, the system checks which is more likely to occur in the given
context, and it chooses the one scoring highest in probability. To prevent problems from occurring when some words in our vocabulary do not at all exist in the corpus, we simply assign all non-existing words a very small probability, just to have numbers for the algorithms to work with. (Jurafsky & Martin, 2000)

**Markov Models**

To estimate the probability of a phone string \( O \) given a word \( w \), in other words the acoustic model \( P(O|w) \) above, most modern systems apply a Hidden Markov Model (HMM) – a two-step probabilistic model. I mentioned weighted automata earlier, and in order to understand what HMMs do, I will begin with explaining how the kind of weighted automata known as an ordinary Markov Model guesses at words in the recognition process.

We define weighted automata before moving on. They consist of:

1. A sequence of states (which each corresponds to a phone)
2. Transition probabilities between the states (probability between 0 and 1)

In a Markov Model, there is an initial state from which the process sets out towards the final state, jumping from one state to the next. In ASR, the way leads in one direction, it is impossible to go back to a previous state in the process. To make use of this as a comparison, a Markov Model uses the forward algorithm, that in every state lets us compare that state to the corresponding phone, as seen in figure 4. Either it is a match \( (P=1) \) or not \( (P=0) \).
The Forward Algorithm

For each word, the forward algorithm takes as input the observation sequence as well as the pronunciation network of that word, and returns $P(O|w)P(w)$. It builds up the probability of the observation sequence by storing the intermediate values in a table. It creates a matrix where each cell $\text{forward}[t, s]$ is the likelihood of being in state $s$ after having seen the $t$ observations, given the automaton $\lambda$. The value of each cell is the sum of probabilities of every path that could lead to this particular cell.

Such an extension of a path can be computed by multiplying three factors:

1. The probability of the previous path for the previous cell $\text{forward}[t-1, s]$,
2. the transition probability from the previous state to the current, and
3. the observation probability that the current state matches the observation, that is the phone. For HMMs, this value can be between 0 and 1.

Running the forward algorithm on all words that are candidates for producing the acoustic features in our observation sequence, we can choose the word with the highest likelihood.

Although a good way of modeling ASR, a Markov Model needs to be more nuanced to deal better with continuous speech, as it is only really useful for
analyzing one word at a time. A more robust model is then a HMM, which is more detailed in it its analysis and is equipped to locate word boundaries.

**Hidden Markov Models (HMMs)**

A Hidden Markov Model is in several ways similar to an ordinary Markov Model. In Figure 5, we see the states and arcs representing transition probabilities, and we see a comparison with a set of observations.

![Hidden Markov Model](image)

*Figure 5. A Hidden Markov Model for the word “need”. (Jurafsky & Martin, 2000)*

What differs mainly here, is that we do not know the transition probabilities during the analysis process, because we use a different algorithm. We also have transition probability arcs going back to their own states. This is the case for long acoustic segments, such as a long [n]. The duration of an acoustic segment corresponds to the number of steps the model remains in the same state. Moreover, the observations do not belong to the same alphabet as the phones in the language model. Lastly, the new algorithm allows us to calculate probabilities *between* 0 and 1 for each state corresponding to a specific observation, unlike the ordinary Markov Model.

A simple model consists of two (sometimes three) probability functions. These sets of probabilities that define an HMM are:
1. For each state $s$, the probability of observing each output, $o$ (each phone) at $s$,
2. from each state $s$, the probability of transition to each of the other states $s'$ in one time step, and
3. sometimes, a probability distribution over the initial state. (Shapire)

The second type of probability describes for example the probability of the phone [n] leading to [iy] in need ([n iy d]), while the third probability intends to make sure the process does not start from completely the wrong state. (Jurafsky & Martin, 2000)

The Hidden Markov Model generates at each step an observable output through which we can guess at which state the process is currently in. Recall the first type of probability mentioned above. This is how we combine the states and the phones.

An HMM-based ASR relies on the Viterbi algorithm, which basically is a more complex version of the forward algorithm, to determine which set of phonemes is most likely to produce the observed acoustic sequence in continuous speech, where we have to consider things such as word structure (likelihood of a certain succession of words) and word boundaries. (Schmandt, 1994)

**Briefly on the Viterbi Algorithm**

As we only get a long row of phones stored as frames from our extracting process, it is difficult to determine where one word ends and another starts, if we have more than one word in a succession. **Segmenting** words using the n-gram language model is one of the things that puts the Viterbi algorithm one step ahead of the Forward algorithm. Another thing, is that the Viterbi algorithm considers all word candidates at once in its calculation, instead of running each word at a time, which soon becomes very inefficient.
The Viterbi algorithm solves this by combining the words into one large automaton. We can use the bigram model to add transition arcs with probabilities, between each of the words, in addition to the arcs that already exist within each word. This will help us later on.

The Viterbi algorithm works recursively – this is easiest understood by describing the matrix it generates. For the state sequence \( q = (q_1, q_2, q_3 \ldots q_t) \) and the set of observed phones \( o = (o_1, o_1, o_3 \ldots o_t) \) each cell \( \text{viterbi}[i, t] \) in the generated matrix will contain 1) the probability of the best way up to that cell, accounting for the \( t \) first observations, ending in the current state \( i \), and 2) the probability of the state being pronounced as the current observation.

If we take a look at Figure 6, we see a matrix consisting of an observation sequence along the x-axis (where each observation has a column representing a time step \( t \)) and the words of a combined automaton, with the states of each word given its own row along the y-axis. For simplicity, the observation vectors are represented as phones, although that would not be the case in practice. The Viterbi algorithm has taken as input the phones [aa n iy dh ax] which we know should be “I need the”. In the matrix, the words “I”, “need”, “the” and “on” are considered, and the goal is to determine word boundaries and correct word succession. The algorithm works its way from left to right along the x-axis and then back again. On the first way, it calculates the probability of each cell leading to another (here is where the probability arcs between words are useful!), column after column, and on the way back it tries to find the most reasonable word succession – that is what the arrows represent. There are two possible full-length paths and one which ends midway. It ends, because the vocabulary did not contain a word or set of words pronounced iy dh ax, which then theoretically would follow “on” in the matrix. (Jurafsky & Martin, 2000) At column dh, there are two possible paths to go back, either “need” or “the”. Since “need” contained the highest
likelihood, it is chosen. Looping back one more step to “I”, the correct path is found. That is the general idea of the Viterbi algorithm.

Figure 6. A matrix showing how the Viterbi algorithm has reached an end state and backtracks to find the most likely path among the states on the y-axis, to correspond to the phones along the x-axis. (Jurafsky & Martin, 2000)
USE OF PROSODY IN SPEECH RECOGNITION

Measuring prosody in ASR is still at an early stage of research, but there has been extensive testing of pitch detecting algorithms and emotion estimation over the years. Advanced measuring of prosodic features in speech in combination with a good knowledge of syntax provides for easier interpretation of ambiguous meanings, emotions and turn-taking in dialogues, among many other things.

I began this paper with definitions of $F_0$, energy and duration. We have already seen in the HMM model, that duration is important (the long acoustic segments causing state loops) in a segment-based recognizer.

Recall the fundamental frequency as being the base for tracking pitch and intonation. Here is where prosody in ASR begins – by tracking the pitch in an utterance, we can get information regarding emotion, tone and relative prominence. It is, however, not easy to measure $F_0$ features, often due to an interfering medium such as a bad telephone connection.

I also mentioned the importance of lexical stress in English – measuring energy can provide lexical constraints, which helps to improve the acoustic modeling. They can reduce the number of candidate words discussed in the forward algorithm.

Prosodic feature also helps in defining word boundaries – it is easier to find the beginning and end of a word if you consider the intonation and energy. (Wang, 2001) Shriberg and Stolcke described how they used prosodic labeling in dialogue act modeling. On a sentence level, rather than only word boundaries, they aimed to classify each utterance into types, e.g. questions, statements and acknowledgements. They saw greatest success in ambiguous
dialogue acts, such as deciding whether a “right” is an acknowledgement or a backchannel. (Shriberg & Stolcke, 2002)
DISCUSSION

I think applied prosody will be immensely useful implemented in speech recognition systems. Not only will it make acoustic models more robust, but also bring new depths to ASR, such as recognition of emotion in speech. This would be helpful, for example in dialogue systems such as automatic customer services. In the introduction, I mentioned how I was surprised by the agent sounding to human to me. And yet, had I kept on failing to pronounce correctly what I was saying, she probably would have kept asking me the same thing, eventually making me angry for always responding in the same way. Imagine if ASR systems could base their responses not only on what we say, but how we say it. Communication would be much easier. In the field on linguistics, we might find new, interesting rules for prosodic features corresponding to syntax, if we can develop models that record that kind of information when decoding prosody in speech.

Given more time, I would have liked to further explore the possibilities of prosodic implementations, as it still remains a relatively untouched area in ASR.

It was difficult to know how to limit the study. Originally, I wanted to focus more on prosody itself, but then realized I would leave too much of the concept of speech recognition unexplained. Trying to elaborate on the HMMs, there was always another algorithm, and another, and then yet another, to precede the one I wanted to explain. It seems as though the simplest algorithms evolved into more complicated ones, and they are implemented in different ways, hence the difficulty to know whether it was relevant or not, to include the preceding algorithms. Also, I found the literature sometimes incoherent and therefore difficult to compare and weave together. That might depend on the writer’s style, the time of writing or simply on different ways of
implementing the same things. Naturally, all articles and books had different highlights on which they focused, so while some elaborated much on certain details, others completely ignored them. For another deep-dive into the workings of an algorithm or a system, I hope to be able to actually see or use it myself, to get a genuine and for me relevant understanding of it.
WORKS CITED

**Books**


**Lecture notes**


**Articles and reports**


**Websites**


**Online dictionaries**
