

Threads White Paper: A Framework for automatically measuring performance of Automatic Speech Recognition Systems

John P Yardley PhD BSc CEng MIEE
Geneviève Masoni BSc
Thomas Michel MSc

Abstract

The demand for, and the provision of, services for the machine transcription of human speech, or Automatic Speech Recognition (ASR), has grown significantly over the last decade and is predicted to grow in the future [1]. With and because of the parallel growth of Internet performance and Cloud computing, ASR system deployment has shifted from private LAN-based servers to shared public (Cloud) servers. This makes good sense because strategies such as Neural Networks commonly deployed for ASR are dependent on significant amounts of computing resources. As such, it is better to share a concentration of powerful resources among a wide number of users. This provides maximum resources when needed while soaking up the latency associated with typical user demand. However, it is difficult for potential subscribers to assess the price/performance of these systems to choose a service best suited to their needs. Service providers rarely provide quantitative performance metrics, and even if they did, they would be subject to so many conditions as to make them almost meaningless. This is thought to be a potential brake on the adoption of ASR.

When in 2018 the authors were deploying third-party Cloud-based ASR services as part of a general message hub service, it became necessary to provide independent guidance to potential users. It was quickly established that this was difficult to provide - except anecdotally. Every potential subscriber had unique operational requirements - for example, language used, speaker pronunciation, accuracy, quality and motivation - that any generic performance measures would be of limited value.

As a result, the authors developed a framework such that potentially any Cloud-based ASR service (which provides an API) could be tested and compared by any user at any time. This paper describes the framework. A later paper will document some of the results obtained.

Table of contents

1 Background	3
2 What do we mean by “best”?	3
3 The ASR Application	8
4 How to test transcription accuracy	10
5 Penalty Matrices Explained	11
5.1 Comparing strings of words	11
5.2 Phonetic word matching	17
6 Test Workflow	20
7 Summary	22
Notes	22
Appendix I Web front-end for Request a test	24
Appendix II: Sample results returned to user	25

1 Background

In 2010, the main author's company embarked on a project to commercialise an internal message sharing system. At the outset, the system combined a bespoke CRM with the company's email server to automate the management of a company-wide address book and distribution of messages, and, as importantly, to ensure that employee private or confidential messages were *not* shared. Such was the interest from customers, that the system was redeveloped from the ground up as a scalable Cloud service.

As part of the redevelopment process, it was decided to abstract the message channel medium, so that the service could handle any type of digital/digitised message including telephone calls. One of the authors had a background in ASR research and the potential to transcribe phone calls and make them fully searchable did not escape him. Indeed, intrinsic in the structure was the ability to incorporate any arbitrary processing of messages, either individually or as a group. As a result, a joint project with Kingston University was initiated to look at the feasibility of applying ASR to VoIP phone calls. The outcome of the project was that it was indeed feasible and useful to routinely transcribe calls within the budgets of an SME operation. The details of the service, called Threads, are described by Yardley, Fox, Michel and Hunter; 2016 [2].

It would have made no commercial sense to internally develop any type of ASR code, since by 2015 there were already several companies providing Cloud based ASR services - each representing perhaps hundreds of man/years programming effort. Indeed, it was quickly appreciated that part of the value of aggregating a company's messages in a single database was the future potential to apply any sort of post-processing - most often from some existing or yet to be developed Cloud-based service. Of course, this presumes that the service is accessible programmatically (ie via an API) and certainly not all ASR service providers provide this. However, enough do provide APIs to be able to offer a choice of ASR service to the message hub subscriber and therein lay the problem. Which ASR service is the "best"?

2 What do we mean by "best"?

To establish what we mean by "best", we first need to examine what we seek to achieve from an ASR service and the factors that affect it.

The application of ASR may be broadly divided into 5 categories:

Command and control

Using human speech to replace physical sensors such as switches, dials, tactile pads or touch screens. The objective is to give hands-free control of some process. Applications include home automation ("Switch on lights please"), and control of motor vehicle appliances.

Multiple choice

A verbal form-filling process. Applications include client telephone access to databases such as bank accounts in which transactions can be reduced to a series of multiple-choice questions.

Natural Language Processing

Answering or responding to natural language questions such as “What is the capital of Bolivia?”.

Key word spotting

Finding key-words within constrained or unconstrained free speech, for example searching telephone conversations for the existence of a specific word or phrase. This is equivalent to doing a search for a text string in an email or document,

Transcription

Converting a stream of human speech into digital text as might a stenographer.

Each of these five categories makes different demands of the ASR process and below are some of the factors that impact the choice of ASR service.

Language to be recognised

Each foreign language contains different words which may be articulated using different vocal *sounds* - for example the “th” sound in English is not used in French. We call these building basic units of human speech “phonemes”. Given the way that speech has evolved, there may be considerable overlap of words and phonemes across different languages (for example similarities in Latin-based languages), but even within a given language, regional accents can account for as much or more variation than occurs in national boundaries.

Size of vocabulary

As one might expect, the size of vocabulary affects the likelihood that a given word is correctly recognised. An ASR system required to discriminate between “yes” and “no” will likely have a greater accuracy than one required to transcribe any word written by Shakespeare.

Speaker dependency

Every speaker has different acoustic characteristics to their voice. Women and children generally speak with higher pitched voices than men, but there are innumerable

variations in the way a word may be uttered yet convey the same meaning. A system which is optimised to provide best performance for one speaker will be likely to recognise words better than one designed to work for multiple speakers

Isolated or continuous speech

Words spoken in isolation often differ acoustically from words spoken in continuous speech. For example, uttering the phrase “six seven” will not be articulated in the same way as it would if “six” and “seven” were uttered in isolation - that is with an intervening pause of silence. When we say “six seven” continuously, we share the final “s” sound of “six” with the initial “s” sound of seven. The ASR process must take account of one or more potentially overlapping phonemes.

Furthermore, we tend to be sloppy in our articulation of continuous speech, often dropping words or inserting filler words like “er”, “um”, “you know”, “like” to allow time for thinking. These filler words are at best redundant and at worst confusing and will be less likely to occur in isolated speech.

For these reasons, the recognition of isolated speech is generally better (more accurate and less speaker dependent) than continuous speech but clearly it would not be practical to expect users to utter words in isolation during a conversation or during dictation. In the command/control and form-filling/multiple-choice situation, it is more likely to be acceptable.

Training

Just as a human is taught to speak, so too ASR systems require training. The issue for the ASR user is whether the ASR system requires training for every new speaker and each new language and dialect. As humans, we do not expect to train every new person we speak to, but in any human interaction, there are often considerably more clues to the meaning of the speech than that available in the pure acoustic signal. We call this *context*. So while the listener may not understand a new pronunciation of a known individual word, he/she may be able to infer its meaning from the context of the conversation - eg the sentence around it or some visual clue like holding an item being discussed. Having established this new variation on the way an existing word is pronounced, the listener will hopefully commit it to memory so that in future the word may be recognised with less context. If ASR can infer the meaning of a word without being explicitly trained, then it is likely to satisfy the user more readily.

The way in which ASR systems *learn* language can vary considerably. One strategy is to define *words* in terms of their *sounds*, that is as a string of phonemes. If we can convert some unknown utterance into a string of phonemes we can then match it to the word in the vocabulary that best matches that set of phonemes. The relatively fixed mechanics of the human mouth and vocal tract means that it is possible to establish the acoustic characteristics on certain phonemes and measure them with some confidence.

Another strategy is to process each utterance and extract features thought to be likely to affect the human perception of meaning. These features could be phonemes, but this implies some expertise on behalf of the designer as to what constitutes a phoneme. More likely, these features would be generic - such as frequency spectra, phase, duration, amplitude, etc. Once the process to extract these features has been developed, the next phase is to process a large number of samples of human speech for each word in the required vocabulary and to let the system work out the relationship, if any, between the features and the words. This, loosely speaking, is called a Neural Network approach because it seeks to emulate the way the human brain learns. These features may not be limited to words, but can infer grammatical constraints, such as the likelihood that one word might be followed by another.

As said, Neural Networks can extract phonemes as well as or instead of generic parameters, so these strategies are by no means mutually exclusive.

The merits of these approaches are beyond our needs here except to say that, in general, Neural Network approaches require a considerable amount of computing power and a large number of samples - particularly where they are required to work with large vocabularies and a large number of speakers, none of which they are explicitly trained with. This has implications for real-time operation and ASR systems with limited computing power (eg cell-phones).

Real-time operation

Given sufficient computing power and “intelligence” there is no limit to the potential of the machine to recognise human speech. Nevertheless, if it takes several hours to yield the result, this may limit the practicability of the system. Some applications require a near instantaneous response, for example, command and control, whereas in others, say keyword spotting and transcription, it may be days or weeks before the result is required. The speed of the result will affect the cost of the service.

Such has been the effect of low-cost computing and fast internet speeds that many ASR services are executed remotely on high-performance Cloud-based computers. Users can collect speech from a cell-phone with insufficient computing power to recognise the speech in real-time. Hence the acoustic speech is pre-processed and sent to a Cloud computer for processing. Clearly, without internet access, this functionality is not available.

Speaker motivation

Speaker motivation has a powerful effect on recognition performance. Speakers reciting texts are often highly motivated to use ASR because they can speak faster than they can type so, if accurate, ASR will save them time. Whereas in a telephone conversation that is being recorded and transcribed, the parties involved will be unlikely to use their

best possible diction. This is because they will probably be unaware the conversation is being transcribed, and if it is totally ad-lib, will likely contain lots of filler utterances and use poor grammar.

Channel

The quality and structure of the speech channel - that is the medium (air, cable, string, water, etc) through which the speech is transmitted - can have an impact on the performance of the ASR. It is easy to assume that if a human can understand a low-fidelity signal, then a machine should be able to also. But this ignores the significant advantage the human gets from context, by which means the human can guess words that were not actually heard (perhaps even not uttered). Limiting the bandwidth of the signal (telephone calls are typically limited in bandwidth) or compressing the signal (removing "redundant" information) can have measurable effects on ASR performance.

A further factor is channel separation. In telephone conversation between two parties, each party is normally contained on separate channels. The ASR system therefore does not need to contend with the issue of two speakers with different voices, accents and articulation. Many telephone systems record calls both by combining channels (stereo to mono) and compressing the signal. The result is the ASR performance can be worse than for processing an uncompressed signal where each speaker has their own channel.

Context

Last but by no means least is the impact context can have on ASR performance. Humans are often dismissive of ASR performance which, at the level of recognising random words, can be better than their own. This is because they do not realise how much context they use in *understanding* the speech as opposed to *recognising* the words.

This context is rarely available to the service, and so no matter how good the intrinsic word recognition, there is a limit to how close it may approach the performance of two individuals who know each other and are discussing a topic with which they are both familiar. However, the *Threads* ASR application often has a significant amount of context available by way of emails between the same parties involved in a verbal discussion. These provide context by way of words and phraseology the parties use.

It is clear from the above that there are many factors which affect the performance of any given ASR service. It is not therefore not surprising that service providers do not provide performance figures on their services. This is unfortunate because ASR is essentially an engineering discipline and not providing any quantitative data, reduces it to something akin to marketing.

Potential ASR users face further complications even if they have very specific requirements. It is not easy to test the relative performance of ASR services without a significant knowledge of ASR and the IT infrastructure to deploy it.

This is of little consolation to potential users of ASR, so part of the object of this paper is to describe one solution to this.

3 The ASR Application

As previously discussed, there are 5 main application areas of ASR, of which Keyword Spotting and Transcription are relevant to our application, *Threads*.

Threads ingests telephone calls by either (a) capturing the telephone call from SIP traffic (SIP is the protocol used to transport packets of telephony data) on the company's Local Area Network (LAN) or (b) capturing directly from the subscriber's telephone system (known as a PBX). Since most commercial PBX systems and services adopt proprietary protocols for recording calls, it would be necessary to devise an ingestion process for every different PBX to be used. Indeed, in some cases, PBX systems do not record calls at sufficient fidelity to get the best possible ASR performance. It is clearly impractical to support all known PBX systems, so where this is not possible or the recording fidelity is too low, method (a) may be used. Once the call is ingested, *Threads* then stores it in digitised form in a Cloud database.

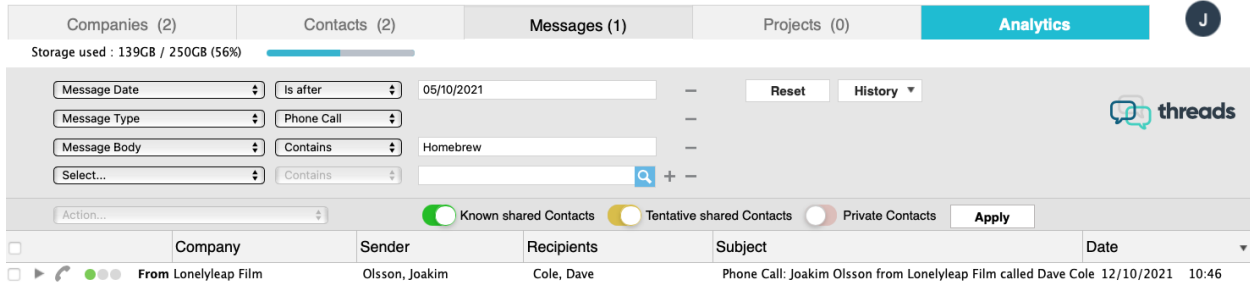
Capturing directly from the LAN SIP traffic is generally preferable because:

1. It is independent of PBX system in use.
2. The speech data is uncompressed.
3. Speaker channels are separate.

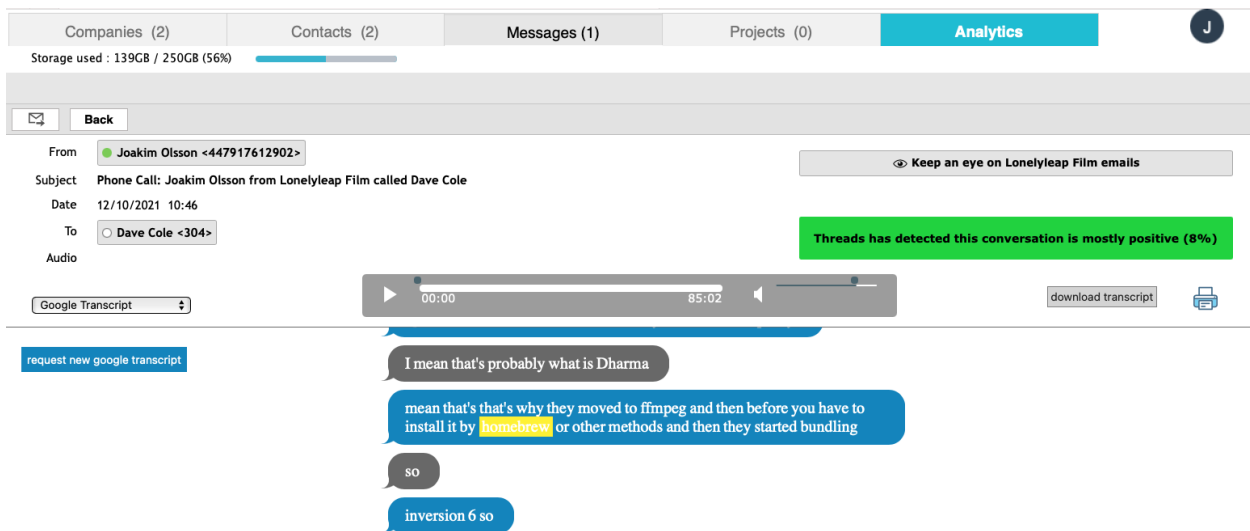
However, this may require the use of specific hardware to intercept SIP traffic, which some subscribers may be reluctant or unable to to deploy. Additionally, this method is portable to Cloud-based PBX systems.

Once the call is captured, the ASR process automatically is initiated by firing off a request to one of several available commercial ASR services which subsequently returns the transcribed call. The transcription is then added to the database record in the same way as the body of an email would be. The metadata associated with the call - for example, caller and called telephone numbers, allows the message to be indexed against the contacts and companies involved.

Having stored the call and transcription in the database, any authorised user can search for the call using some metadata - eg, date, time, contact company, etc - or more significantly some keywords used in the telephone conversation. In the example screen below, the user is searching for any telephone conversation containing the word "homebrew" that occurred in the last week...



Having located the call or interest, the user can view the transcription, scrub through the discussion while listening to the call and viewing the transcription...



For this application, real-time ASR is not important. Just like an email, it is rare the email will be searched for soon after the email is sent or received. It is weeks or months later that locating it may be difficult and a search tool is of most value.

In terms of performance, we wish to measure how accurate is the transcription, how different services compare, and how services change with time. The last requirement is of particular value because Cloud-based services are being constantly improved so it is useful to compare the performance of a given service for an identical recording over time.

It was never intended to provide some industry standard metric for available ASR services but rather, for potential users to decide which service suited them best - using their environment - that is telephone system, language, vocabularies, business focus and staff.

Some ASR service providers offer mechanisms to transcribe user-submitted samples of digitised speech, but these are unlikely to have a high takeup for the following reasons

1. It may be necessary to set up an account - at the investigation stage most prospective users are not keen to create accounts

2. Obtaining a digitised recording requires some degree of technical expertise - simply creating a file can be challenging (selection of microphone, etc) and the format of the digitised recording can significantly affect the performance of the ASR system
3. Once the transcription has been received, how is the performance measured such that it can be compared with other ASR systems.
4. Given that the previous issues were overcome, it needed to be established that the ASR service could be integrated into the prospective user's intended workflow. Some ASR providers do not permit API access, and where they do, this assumes some software development capability to utilise.
5. The prospective user needs to be able to estimate the likely cost of the ASR service in intended operational use. This is not always a trivial calculation.

Although the author's primary goal is to provide potential users an assessment of the value ASR might bring to their (the authors') specific application, it is also intended to offer a generic mechanism for comparing ASR services. There was also significant value in automatically obtaining comparative assessments of various ASR services and the authors' various pre-processing strategies aimed at improving the intrinsic recognition performance of the services adopted - even if the testers had no current requirement for the Threads service itself.

4 How to test transcription accuracy

As one soon discovers when examining a large number of samples, we humans are by no means as good at speech recognition as we sometimes think. We have previously mentioned that one reason for this is the human's implicit use of context which is not available to the machine but which the human assumes is available. Often, once the human sees the machine transcription and listens to the corresponding recorded speech, he/she tends to be more forgiving of the machine and aware of his/her own shortcomings. But nevertheless, it is unreasonable to expect the machine to perform better than the human without that context.

In order to test how well a machine performs ASR on some human speech we need to compare it with a human transcription as this is the only measure relevant to a human.

In an automatic test scenario this is, by definition, not possible except by interactively involving a human at some stage and then assessing that the human transcription is fair. So the approach taken is to require testers to recite a script then compare the machine transcription with the script.

There is, of course, no guarantee that the human tester will recite the text verbatim, and indeed they rarely do but nevertheless, our goal is not to obtain an absolute measurement of performance but rather compare ASR services with one another or with themselves as they evolve.

Recited scripts tend to be far better enunciated than operational conversations and the corresponding accuracy is, not surprisingly, better. Hence, while the use of a script allows the test process to be fully automated and to get a relative measure of different ASR services, it will not necessarily reflect operational use for, say, the transcription of ad-hoc telephone conversations. To provide this test would involve the dialogue first being transcribed by a human so as to provide a script with which the machine transcription could be compared. Clearly, it is impractical to provide this as part of the test service. Obtaining bespoke human transcriptions is expensive, time-consuming and maybe poor if the transcriber is unfamiliar with the language and terms used. Nevertheless, if the adoption of ASR is to be considered, it has to be tested at some stage in an operational environment. To address this, the test framework allows users to upload their own transcriptions of conversations so they may be compared.

Having a script and machine transcription, the next question is how do we measure the accuracy of transcription? The technique the authors have adopted is commonly used in Operational Research and called a penalty matrix,

5 Penalty Matrices Explained

5.1 Comparing strings of words

Let us assume that a human *utters* the following phrase.

a tax on ships.

and that this is *transcribed* by 3 different ASR algorithms to produce the following results:

Algorithm A: *attacks on ships*

Algorithm B: *a tax on chips*

Algorithm C: *a tax on ships*

The objective of the penalty matrix is to compute every permutation of comparing the *uttered* phrase with the *transcribed* phrase so we may decide which transcription fits the “best”.

We may assume that a perfect match of words results in zero penalty, but we must decide on the penalty we apply whenever the words do not match, ie where a word is not recognised at all or where an erroneous word that did not exist has been inserted.

A simple scheme might be as follows:

Match	penalty
Words are identical	0
Words are different	1.9*
Word is removed (ie in the uttered phrase but not in the transcribed phrase)	1
Word is inserted (ie word is not in the uttered phrase but occurs in the transcribed phrase).	1

* Later, we will consider why the penalty of swapping words should be marginally lower than the penalty of removing one and inserting another.

For algorithm A we can guess what the best permutations might be..

Match	penalty	Cumulative penalty
a → attacks	1.9	1.9
tax → <null>	1	2.9
on → on	0	2.9
ships → ships	0	2.9

Match	penalty	Cumulative penalty
a → <null>	1	1
tax → attacks	1.9	2.9
on → on	0	2.9
ships → ships	0	2.9

Match	penalty	Cumulative penalty
a → <null>	1	1

tax → <null>	1	2
<null> → attacks	1	3
on → on	0	3
ships → ships	0	3

The first two of these permutations give a slightly lower cumulative penalty. It is by making the penalty of a substitution lower than that of a removal then insertion, we will favour substitutions. ie that..

a → attacks

Is better than

a → <null>
<null> → attacks

Intuitively, it is more likely the speech algorithm will get a word wrong than miss one word and erroneously insert in another, so it makes sense to favour substitutions.

Rather than favour a substitution with a lower penalty than a deletion and insertion, we could simply give priority to the substitution when the penalty is the same as a deletion or insertion.

For algorithm B we can guess that the best permutation might be..

Match	penalty	Cumulative penalty
a → a	0	0
tax → tax	0	0
on → on	0	0
ships → chips	1.9	1.9

For algorithm C we can guess that the best permutation is..

Match	penalty	Cumulative penalty
a → a	0	0
tax → tax	0	0

on → on	0	0
ships → ships	0	0

Now we need to convert our intuitive guesswork into an algorithm.

We begin by creating a matrix. If there are ROW words in the uttered phrase and COL words in the transcribed phrase, the matrix should have ROW+1 rows and COL+1 columns. If we compare the uttered phrase with the output from Algorithm A, we would have..

$$\text{ROW}+1 = 4 + 1 = 5$$

$$\text{COL}+1 = 3+1 = 4$$

Hence our matrix would look as follows..

	COL	1	2	3	4
ROW			attacks	on	ships
1					
2	a				
3	tax				
4	on				
5	ships				

The white section is the actual stored matrix, and the beige cells are simply the row and column descriptions.

Each cell in the matrix contains the lowest penalty of getting to that cell from the adjacent cell to the left, the adjacent cell diagonally above-left, and the adjacent cell above.

We then proceed by filling in initial values in the top row and leftmost column where cells are indexed as CELL(row, column)

CELL(1,1)=0
 CELL(2,1)=CELL(1,1)+penalty of deleting word in row 2= 0+1 = 1
 CELL(3,1)=CELL(2,1)+penalty of deleting word in row 3= 0+1 = 2
 CELL(4,1)=CELL(3,1)+penalty of deleting word in row 4 = 0+1 = 3
 CELL(5,1)=CELL(4,1)+penalty of deleting word in row 5= 0+1 = 4

CELL(1,2)=CELL(1,1)+penalty of inserting word in col 2= 0+1 = 1
 CELL(1,3)=CELL(1,2)+penalty of inserting word in col 3= 0+1 = 2
 CELL(1,4)=CELL(1,3)+penalty of inserting word in col 4 = 0+1 = 3

This gives..

	COL	1	2	3	4
ROW			attacks	on	ships
1		0	1	2	3
2	a	1			
3	tax	2			
4	on	3			
5	ships	4			

The penalty of getting from an adjacent cell is the value of the adjacent cell plus the penalty of the word comparison, as defined in the first table. The new value is the MINIMUM of the 3 possible routes.

Let

A = CELL(2,1)+penalty of deleting the word in row 2
 B= CELL(1,2)+penalty of inserting the word in col 2
 C= CELL(1,1)+penalty of changing the word in row 2 to the word in col 2

$$\text{CELL}(2,2) = \text{MIN}(A,B,C)$$

This gives [note 1]...

	COL	1	2	3	4
ROW			attacks	on	ships
1		0	1	2	3
2	a	1	1.9	2.9	3.9
3	tax	2	2.9	3.9	3.9
4	on	3	3.9	2.9	3.9
5	ships	4	4.9	3.9	2.9

The matrix for Algorithm B is [note 1]...

	COL	1	2	3	4	
ROW			a	tax	on	chips
1		0	1	2	3	4
2	a	1	0	1	2	3
3	tax	2	1	0	1	2
4	on	3	2	1	0	1
5	ships	4	3	2	1	1.9

Later, we will go on to look at how we might better compare “ships” with “chips”.

The matrix for Algorithm C is [note 1]...

	COL	1	2	3	4	
ROW			a	tax	on	ships
1		0	1	2	3	4
2	a	1	0	1	2	3
3	tax	2	1	0	1	2

4	on	3	2	1	0	1
5	ships	4	3	2	1	0

By tracing the matrix back from the bottom-right cell, following the path through the minimum adjacent cells, the optimal mapping may be established.

5.2 Phonetic word matching

It should be clear that using such a simple word matching rule will provide only a very crude measurement of recognition performance. In the examples used, replacing “ships” with “chips” will give the same result as replacing “ships” with “ship” or “cat”. Clearly “chips” sounds more like “ships” than “cat”. Yet in our simple word matching scheme, unless the word spelling is identical, the penalty takes the same value.

One way to improve the measurement is to compare the alphabetic spelling of the uttered and transcribed words. This could be achieved using a similar penalty matrix method except that, instead of comparing words, we compare letters.

Furthermore, some letters sound more similar than others. For example, “B” sounds more similar to “D” than it does to “J”, hence it would make sense to attribute a penalty that reflects the similarity in sound.

However, if we are going to implement a more sophisticated matching metric, we can use a [phonetic representation of words](#) - which describes how the words sound - rather than how they are spelt.

To illustrate this, let’s look at the comparison of the uttered word “ships” with the alternative transcribed words of “ship” and “cat”. The phonetic transcriptions are as follows:

ships	ʃ	ɪ	p	s
chips	tʃ	ɪ	p	s
cat	k	æ	t	

Before we can fill in our penalty matrix, we must decide on the penalties associated with substituting one phoneme for another, deleting or inserting phonemes. If we use the same simple scheme we used for whole words, that is zero penalty for an exact match, 1.9 for a substitution and 1 for a deletion or insertion, we will get a much better measure of the similarity of the word sounds.

Matching “ships” with “chips” we get [note 1]t...

	COL	1	2	3	4	5
ROW			tʃ	I	p	s
1		0	1	2	3	4
2	ʃ	1	1.9	2.9	3.9	4.9
3	I	2	2.9	1.9	3.9	4.9
4	p	3	3.9	2.9	1.9	2.9
5	s	4	4.9	3.9	2.9	1.9

Matching “ships” with “cat” we get [note 1]...

	COL	1	2	3	4
ROW			k	æ	t
1		0	1	2	3
2	ʃ	1	1.9	2.9	3.9
3	I	2	2.9	3.9	3.8
4	p	3	3.9	3.8	4.9
5	s	4	4.9	3.9	6.7

Clearly, “ship” with a total penalty of 1.9 is a better match than “cat” with a total penalty of 6.7.

However, in our whole word comparison, the maximum penalty we can get for a word substitution is 1.9, no matter how many characters in the word.

We can see that even a simple phonetic matching scheme gives a more accurate comparison than a word matching scheme. However, we can go a step further by using penalties that better represent similar sounds.

Our simple phoneme matching scheme can be represented in a matrix comparing any phoneme with any other. This is a segment of the full matrix...

		Single vowels						
		ɪ	i:	ʊ	u:	e	ɜ:	ə
Example		ship	sheep	book	shoot	left	her	teacher
1	ɪ	ship	0	1	1	1	1	1
2	i:	sheep	1	0	1	1	1	1
3	ʊ	book	1	1	0	1	1	1
4	u:	shoot	1	1	1	0	1	1
5	e	left	1	1	1	1	0	1
6	ɜ:	her	1	1	1	1	1	0

In this matrix, if the phonemes are the same, there is zero penalty and 1 if they are different. Hence the matrix contains all 1s except along the diagonal, which is zeros. The full matrix with all 44 phonemes can be found [here](#).

Acoustically, the “i” as in “ship” is identical to the “ee” sound in “sheep” - the “i” is simply a shortened version of “ee”. If you try to make an “i” sound long, it becomes “ee”.

Therefore, you might reasonably expect that the penalty of substituting an “ee” for an “i” should be less than substituting, say, a “k” sound.

Adopting a more complex penalty scheme for the similarity of phonemes depends on the existence of a quantitative analysis of how humans measure the similarity of phonemes, for example Weber and Smits [3]. This is highly language dependent so the framework allows for a user-specified matrix to be uploaded and applied. However, from the viewpoint of *comparing* speech recognition systems, the choice of penalty scheme is not so important. Provided the same metrics are applied to all the models, then the absolute ASR performance is not an issue.

There are several web resources available for obtaining the phonetic representation of words, and the similarity of phonemes. These include the [Oxford Dictionary](#) and [MediaWiki](#).

Lastly, there will be many words for which a phonetic description is not available. There are likely to be algorithms available for converting English words into phonemes but where a phonetic representation is not available, a letter matching method would provide a more accurate result than a simple word matching method.

6 Test Workflow

We have seen from our discussion of penalty matrixes, that we obtain 3 concise metrics for each transcription

- Words correct
- Words inserted
- Words deleted

We have also discussed the potential issues when performing word comparisons as opposed to phoneme comparisons, so we provide similar metrics by comparing utterances at the letter and phoneme levels. Comparing utterances at the phoneme level favours the correct recognition of sounds rather than alphabetic representation, and may provide the user with a more relevant metric, particularly where the articulation or channel quality is low. Users are more tolerant to words that sound the same even if spelt incorrectly. (eg bow and bough).

There is generally no control of ASR services at the API level so these processes generally only ever return transcriptions at the word level. In many cases, ASR services do break speech into phonemes, or if they do, the phonemes are not available as a resultant stream.

The test process converts words into phoneme strings by using another service to map words to phonemes - for example the Oxford Dictionary API. Needless to say, there are many words whose phonetic spelling is not available, so in these cases, the test process is able to compare alphabetic spellings instead.

The test workflow consists of the following steps..

1 Complete the "Request a test" web form [appendix 1]

Option A - New Test

- Full name - this is the name of the speaker
- User reference - a reference the user may allocate to identify transcription
- Email address - address to receive further instructions

- Last 5 digits of calling number - this enables the test process to identify the appropriate test recording. To participate in the test, the calling number may not be withheld.
- Choose language - At the time of writing, this is one of English, French, Italian, Spanish, Portuguese or German but other languages will be added.
- ASR service to be tested - this is one of the ASR services which has been supported by Threads.
- Transcription type - This may be one of..
 - Generic transcription - this is the recital of a generic script by a single speaker
 - User transcription - the user will need to respond to the confirmation email attaching the digitised speech and transcription

Option B - Repeat Test

- Threads reference - unique reference allocated to a previous new test
- User reference - a reference the user may allocate to identify transcription
- ASR service to be tested - this is one of the ASR services which has been supported by Threads.

2 Await confirmation email with further instructions [appendix 2]

The user will receive an email according to the selections in the “Request a test” form, which contains instructions for either recording the test or submitting a recording and transcript.

3 Call the designated number

4 Record the script and hang up

5 Await email with link to test result [appendix 3]

The test result page will include the following information [appendix 4]

- User name
- Email address
- User reference
- Threads reference
- Last 5 digits of calling number
- Service tested
- Transcription type
- Date and time of test
- Script uttered
- Transcription
- Recording of test utterance
- Penalty Matrix
- Best fit mapping

- Words/letters/phonemes matched/deleted/inserted (9 values)

7 Summary

The framework described distills the complicated and confusing process of testing ASR services into a simple web tool that requires no knowledge of speech recognition processes, no special account setup and which measures performance using any user's voice, any (supported) language, and communication channel.

It is not intended to provide an absolute measure of ASR performance - which for most applications would be irrelevant anyway - but to allow potential users of ASR to establish what they might expect and which service is the best for them.

We hope that this paper will encourage ASR service providers to include their APIs in the framework and while it is not the author's intention provide a league table of ASR service performance, it is hoped that the statistics obtained from the tests will provi

Notes

[1] The actual values in each cell of the penalty matrix are believed to be correct but are illustrative only.


References

[1] [Fortune Business Insights April 2021](#)

[2] Untangling your Threads - a novel cloud computing application: Yardley, Fox, Michel & Hunter; March 2016; IET Engineering and Technology Reference

[3] Consonant And Vowel Confusion Patterns By American English Listeners;
Weber and Smits; 15th ICPHS 2003

Appendix I Web front-end for Request a test

Penalty matrix ASR Test

Request a test

Full name *

Email *

Last 5 digits of calling number (51035, 87218) *

Choose language *

English ▾


ASR service to be tested *

Google Speech ▾

Transcription type *

Generic transcription ▾

Send

©2021 [Threads.cloud](https://threads.cloud) All rights reserved.

Appendix II: Sample results returned to user

General information

User ID	Name	Email	Last 5 digits of calling number
316525	Thomas Michel	thomas.michel@jpy.com	87218

Test ID	Language	ASR service	Transcription type	Date
27	English	None	generic	11 Aug 2021 at 10:35

Metrics

	Matches	Substitutions	Insertions	Deletions
Word level	30.0	197.0	45.0	39.0
Letter level	162.0	288.0	90.0	78.0
Phoneme level	287.0	390.0	135.0	117.0

Machine transcription

add aircraft the brand list of baby boomer musicians and fans for the 70-year Old guitar Legend disclose that he is 10 as a ringing in ear in commonly caused by noise induced hearing loss and then you this week BBC Radio 2 but this Genesis drummer Phil Collins but like Genesis drummer Phil Collins and who guitarist Pete housing have both struggled accepting says he intend taking to you wearing for the foreseeable future I'm still going work I'm doing a few gigs going to do a show at Hyde Park British Summer Time festival in July he says the only thing I'm concerned with Now is being in my 70s to be proficient I mean I'm going yeah I've got 10 in my hand just about work I mean I'm hoping that people will come along and see me from morning

Human transcription

Add Eric Clapton to the growing list of baby boomer rock musicians – and their fans – who are losing their hearing.

The 72-year-old guitar legend disclosed that he is struggling with tinnitus, a ringing in the ear commonly caused by noise-induced hearing loss, in an interview this week on BBC Radio 2.

But like Genesis drummer Phil Collins and Who guitarist Pete Townsend, who have both struggled with hearing loss, Clapton says he intends to continue performing for the foreseeable future.

Word map

