HUMAN MACHINE INTERFACES

SPEECH RECOGNITION

MADE BY

MARIJN STAM

University of Applied Sciences
Amsterdam

DEC 18, 2019

Contents

1 Abstract

Software as a Service has allowed anyone to use complex algorithms based on deep-learning AI models for their own use. In this paper we test one of those services on its speed and accuracy. IBM Watson offers a speech-to-text service. With a standardized test set containing many tongue twisters and homonyms, we will test this service. The results received from Watson will be compared to the test set to obtain metrics which describe speed and accuracy. It is found that the service is fast enough to perform real-time and is somewhat accurate (Word Error Rate = 24.8%).

2 Introduction

Many companies have exposed their machine-learning and AI algorithms to the public through a SaaS (Software as a Service) architecture. The consumer can make use of extensive and well-tested algorithms of many different purposes from big companies for small prices. These prices are usually payed per request.
Many different applications can be used from many different companies. Examples hereof are; speech recognition, text-to-speech, visual object detection, tone analyzer, etc.. All these algorithms offered by companies like IBM or Google, promise a far more accurate and robust result than something built by a small company or individual, given that the large companies have invested much time and effort into developing and teaching these algorithms.
In this paper, the speech recognition service offered by IBM, also known as IBM Watson, will be tested on several metrics. These methods will be later discussed and described.

3 Description of Watson Speech Recognition

Watson is a cloud-based AI solution offered by IBM. Watson Speech to Text is a cloud-native solution that uses deep-learning AI algorithms to apply knowledge about grammar, language structure, and audio/voice signal composition to create customizable speech recognition for optimal text transcription [1]. It allows for custom learning to enable domain-specific knowledge so that every use-case can be met.

4 Experimental set-up

Our experiment exists of feeding the Watson API several .WAV files as input. These .WAV files come from the standardized test set supplied by the students. The test set containts many tongue twisters and homonyms to truly evaluate the strength and robustness of the algorithm. Watson will return its results in a neatly packed JSON file, which we can then simply map to a Python object. Each word is individually seperated with its own confidence rate. This makes it trivial to compare each individual word with the words from the test set, since those are also given as a .CSV file. With these two inputs, we can compare the results to the actual words in the sentence. We do this for every sentence defined in the CSV file, which makes for a total of 18 sentences. To ensure accurate results, we only use the .WAV files from Roland, this will reduce variable between results. For each word recognized, we check the confidence rate and if the word matches with the CSV, we update our confusion matrix based on the result as following:

- Word match AND confidence > 0.50 = True Positive (TP)
- Word match AND confidence < 0.50 = False Negative (FN)
- No word match AND confidence > 0.50 = False Positive (FP)
- No word match AND confidence < 0.50 = True Negative (TN)

Along with that, we also check for deletion and insertions. Watson has a habit of missing small words like "the" or "it", when it does so, our application will be trying to match all the words in the CSV to the result with an offset, leading to all negatives. To prevent this, a small look-forward and look-behind check is built-in. When a negative is found, check one word ahead or behind in the CSV and see if that word matches, if it does, the index is reset and a deletion or insertion is registered.
For each sentence, a complete confusion-matrix is collected. With this matrix we can find several metrics, namely: recall, precision and F-score. These will be given for each sentence but also as an average over the whole test set. Precision describes the proportion of predicted positives which is truly positive. Recall describes the proportion of actual positives which is correctly classified. The F1-score offers a more general performance indication, given by the harmonic mean of the precision and the recall [2]. Other than that, we also measure the Word Error Rate, which is given by $\frac{FP+TN+Deletions+Insertions}{Sentencelength}$. This gives us a general indication of the correctness of the algortihm. To measure the speed of the algorithm, we use the metric Real Time Factor. We retrieve this by $\frac{ProcessingTime}{AudioLength}$. If this value is 1, the algorithm can keep up with input in real-time, if it is higher, it can not.
The application which was built to obtain these metrics and run the tests can be openly found as a attachment to this paper.

5 Results

All 18 sentences were fed to the API and the metrics were collected. The results can be seen in figure 1. The micro-metrics are displayed per sentence and the macro averages are displayed in the bottom row.

	Recall	Precision	F1	RTF	WER	
fort_fight,	96,6	100	98,2	0,572	9,68	
dear_deer,	60	85,7	70,6	0,674	28,6	
weak_week,	100	100	100	0,58	20	
scene_scene,	100	94,1	97	0,453	5,56	
meat_meet,	100	91,7	95,7	0,522	26,7	
mail_male,	88,2	100	93,8	0,584	0	
short_story,	90,2	91	90,6	0,524	12,6	
polling,	82,6	92,3	87,3	0,699	19,4	
priority_inversion,	95,7	95,7	95,7	0,454	15,4	
sempahore,	82,6	95	88,4	0,698	19,2	
Caffaro,	100	76,9	87	0,636	53,3	
submissive_inferior,	83,3	62,5	71,4	0,573	41,4	
Palestine,	93,3	82,4	87,5	0,676	30,4	
canner,	100	85,7	92,3	0,74	50	
fanny,	66,6	80	72,7	0,723	14,3	
pest,	75	75	75	0,753	16,7	
coffee,	66,6	75	70,6	0,794	41,7	
seashells,	100	100	100	0,711	0	
wristwatches,	100	80	88,9	0,661	66,7	
	88,45789474	**87,52632**	**87,51053**	**0,633**	**24,82316**	**Macro**

Figure 1: Found results from comparing the results with the test set

6 Discussion

In general, the results seem to be fairly positive. There is some large sway in some of the samples, but this could be in variance of difficulty of the sample. Some of the audio samples are purposely made as hard as possible to interpret, understandably Watson has some difficulty with these samples. It also important to understand that the current application only supports look-forward and look-behind with index 1, meaning that if Watson misses two or more words or interprets a single word as three words, the application fails to retrieve usable metrics. This error has not occurred with the current test sample.

7 Conclusion

Based on the results, we can conclude that Watson is real-time compatible. It has always returned its results within the length of the audio clip that was sent. On average it only took Watson 63,3% of the audio length to return its result. This is an important metric, as it can function in real time, without lagging behind.
The other results are less telling without other frameworks or services to compare them to. We see that with an input string of 100 words, Watson gets 24,8 of these words wrong. Seeing as many of these sentences are tongue twisters or homonyms, I believe this to be fairly impressive. I expect the results to be much more accurate if the test set supplied were to contain more everyday sentences.
Further conclusions can be drawn when the results in this paper are to be compared with that of other students who used different frameworks or services.

References

[1] IBM. *Watson Speech to Text*. URL: https://www.ibm.com/cloud/watson-speech-to-text. (accessed: 20.12.2019).

[2] Boaz Shmueli. *Multi-Class Metrics Made Simple, Part I: Precision and Recall*. URL: https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2. (accessed: 20.12.2019).