Performance of Ad-hoc Retrieval with Automatic Speech Recognition Transcribed Queries

Max Robinson max.robinson@jhu.edu
Johns Hopkins University,
Baltimore, MD 21218 USA

Abstract

Ad-hoc retrieval with automatic speech recognition (ASR) transcribed queries presents a new set of challenges. The types of errors in transcribed queries are not the same as queries that were typed. This paper looks at the FIRE 2010 dataset with transcribed queries and investigates which simple indexing schemes provide the most benefit to mitigate the errors that arise from ASR. The results showed that while the performance of ad-hoc retrieval is significantly degraded by poor transcription, using an indexing scheme of character 4-grams provided the best mitigation for errors introduced by ASR compared to character 3-grams and word tokenization.

1. Introduction

Ad-hoc retrieval is a staple in modern day technology. With the world wide web and services such as Google and Microsoft's Bing, searching for relevant information has common practice. Now as technology has continued to advance, automatic speech recognition is again at the forefront of technology through voice user interfaces for information retrieval.

Much of where ASR is used today is to enable easier search for information. Many smart devices, including smart phones, have some sort of on board or built in ASR that is used for search. Everything from searching the Internet to looking up movies can have integration with ASR for these searches.

The goal of ASR is to have perfect accuracy in transcription of what a user says to what the system outputs as text. The ability accurately transcribe text in an every day scenario similar to a human has shown to be a difficult task. This provides an interesting problem to information retrieval systems to still provide relevant information even though the query could have errors.

Providing accurate document retrieval with errors like spelling mistakes in queries is an active area of research. However, ASR transcribed text presents a new problem with the possibility of a transcribed word being entirely different than the intended word by the speaker. While, more accurate ASR would help fix this problem, it is not clear that ASR will reach 100% accuracy soon.

This paper explores how different indexing schemes for ad-hoc document retrieval affect the performance retrieval using queries that are transcribed with ASR. To do so, this paper uses the English FIRE 2010 corpra (https://www.isical.ac.in/ clia/2010/) to test character 3-gram, 4-gram, and word based indexing for retrieval using the open source search engine Elasticsearch for retrieval of documents. Mozilla's DeepSpeech implementation (Mozilla, 2014) was used to transcribe audio versions of the FIRE 2010 queries used for testing.

2. Previous Work and Background

Automatic speech recognition saw research into the domain with working systems as early as 1952 with a single speaker digit recognition system built by Bell Labs (Juang & Rabiner, 2005).

Since then, ASR has come a long ways. Services from companies like Google and Micrsoft are reaching close to 95% accuracy (Meeker, 2018) (Xiong, Wu, Alleva, Droppo, Huang, & Stolcke, 2017). Human accuracy is considered to be about 95% (Xiong et al., 2017) as well, meaning that ASR is reaching a human level of accuracy very quickly.

However, not all ASR is this accurate and not all uses of ASR can use tools provided by companies like Google or Microsoft. Alternative non-proprietary based solutions exist for this reason. A stand alone recurrent neural network called Deep Speech was developed by researchers at Baidu in 2014 (Hannun, Case, Casper, Catanzaro, Diamos, Elsen, Prenger, Satheesh, Sengupta, Coates, & Ng, 2014). They claimed a 16% error rate on a widely studied dataset, Switchboard Hub5'00. An open source implementation of Deep Speech by Mozilla (Mozilla, 2014) was later created with publicly available language models to enable ASR without the need to use commercial services.

Errors in queries for information retrieval systems have also been studied. In a typical query system, errors in the query are typically caused by a human mistyping a word resulting in a spelling mistake. Spelling mistakes have been studied for sometime and there are existing solutions such as edit distance, context sensitive spelling corrections, or phonetic corrections shown in books like Manning's "Introduction to Information Retrieval" (Manning, Raghavan, & Schtze, 2008).

Other techniques such as relevance feedback and query expansion have also been used to combat non-specific queries and enhance retrieval performance. Query expansion attempts to add relevant words to a query in the hopes that the expanded query provides better results. Relevance feedback aims to adjust a query based on the top documents initially retrieved to improve results. Relevance feedback can also be used as a corrective measure to suggest to a user a more likely query if a mistake was made.

3. Experiment

To test the effect of different indexing methods on retrieval performance for ASR transcribed queries the Forum for Information Retrieval Evaluation (FIRE) 2010 (https://www.isical.ac.in/~clia/2010/) English dataset was selected. The FIRE 2010 dataset is a collection of documents focused on news articles in the South Asia region. The documents are available in a few different languages, but for the purpose of this experiment the English version of these documents will be used.

FIRE 2010 follows the TREC standard convention for document, topic, and qrels encodings. This allows the application of the standard TREC_EVAL calculations and metrics to be used. The primary metric used in this experiment is the mean average precision (MAP) for each set of queries.

In order to test ASR transcription, the queries for FIRE 2010 had to be transformed into audio and then the audio must have ASR applied to it to obtain a transcribed query. To transform the queries, the queries were spoken out loud and recorded with an inexpensive

headset microphone. The audio was recorded at 16KHz with 1 channel and a chunk size of 1024.

Each recording was 20 seconds in length, even if speaking the query did not take 20 seconds. Queries were spoken with roughly the same cadence and speed to as much of a degree as possible by the speaker. The speaker was a native English speaker and words were spoken close to dictionary pronunciations typically (i.e. little to know accent). The speaker was not familiar with many of the named entities in the query text so pronunciations of non-English words were done to the best of the speakers ability, following English phonetic rules.

Mozilla's DeepSpeech implementation was chosen to transcribe the audio queries back to text. Mozilla's DeepSpeech was chosen because of its free availability and its popularity in the open source community. In addition, DeepSpeech is free and able to be run on a single computer with middling specifications. The model used for DeepSpeech is the freely available v0.3.0 model retrievable via the Mozilla DeepSpeech repository.

Elasticsearch (www.elastic.co) was used as the search engine and document storage. Elasticsearch is an open source and commercial search engine built on top of Apache Lucene (lucene.apache.org) which is an open source text search engine written in Java. Elasticsearch allows documents to be placed into different indices and each index can specify a mapping, text analyzer, and tokenizer for indexing and querying on specific fields.

Three indices were created and used for the experiment. One index specified characters to be indexed using character 3-grams, another using character 4-grams, and the last used the standard Elasticsearch word indexing with stopword removal. For all indices, text was translated into all lowercase characters prior to being indexed. An example of the Elasticsearch indexing settings can be seen in Appendix A.

The documents for FIRE 2010 were indexed into Elasticsearch using a companion tool Logstash (www.elastic.co/products/logstash), after first being translated into tsv files. Logstash was used primarily to quickly index data into Elasticsearch. For each document, the entirety of the document was indexed, and the document ID was also stored with the document.

Queries were made to Elasticsearch using both the transcribed queries and the original queries for comparison. The query used was a standard Elasticsearch "match" query. Elasticsearch preprocessed the query the same way as if indexing a document according to that index, and then executes a boolean "or" query. The format of the query can be seen in Appendix B. Only the document ID was retrieved for each query and the score for each document was captured.

In an attempt to retrieve as many relevant documents as possible for each query, the top 1000 documents were retrieved for each query. There were 50 queries total and the top 1000 documents were retrieved for each of those, resulting in a total of 50,000 documents retrieved.

The results of each query were output in the standard TREC evaluation format for a trec_top_file. TREC_EVAL was run on the results from executing all 50 queries transcribed queries against all indices and executing all 50 original queries against all indices.

4. Results and Analysis

5. Future Work

Maybe?

6. Conclusion

References

Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. *CoRR*, *abs/1412.5567*.

Juang, B. H., & Rabiner, L. R. (2005). Automatic speech recognition - A brief history of the technology development. *Elsevier Encyclopedia of Language and Linguistics*.

Manning, C. D., Raghavan, P., & Schtze, H. (2008). *Introduction to Information Retrieval* (1 edition). Cambridge University Press.

Meeker, M. (2018). Mary meeker's 2018 internet trends report. electronic. Published at Code Conference 2018.

Mozilla (2014). https://github.com/mozilla/deepspeech..

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., & Stolcke, A. (2017). The microsoft 2017 conversational speech recognition system. Tech. rep..

Appendix A: Elasticsearch Index 4-gram Configuration

```
{
  "settings":{
    "number_of_shards" :   1,
    "number_of_replicas" : 0,
    "analysis":{
      "analyzer":{
        "4gram":{
           "type":"custom",
           "tokenizer":"4grams",
           "filter":[
              "lowercase"
           ]
        }
       },
      "tokenizer": {
        "4grams": {
          "type": "ngram",
          "min_gram": 4,
          "max_gram": 4,
          "token_chars": []
        }
      }
    }
  },
  "mappings": {
    "doc": {
      "properties": {
        "doc": {
          "type": "text",
          "analyzer":"4gram",
          "search_analyzer":"4gram"
        }
      }
    }
  }
}
```

Appendix B: Elasticsearch Query

```
{
    "_source": ["docID"],
    "query": {
        "match": {
            "doc": "<query text>"
        }
    },
    "size": 1000
}
```