Semi-Supervised Keyword Spotting in Arabic Speech Using Self-Training Ensembles

Mohamed Mahmoud
Computer Science Department
Stanford University
elgeish@stanford.edu

***Abstract*—Arabic speech recognition suffers from the scarcity of properly labeled data. In this project, we introduce a pipeline that performs semi-supervised segmentation of audio then—after hand-labeling a small dataset—feeds labeled segments to a supervised learning framework to select, through many rounds of hyperparameter optimization, an ensemble of models to infer labels for a larger dataset; using which we improved the keyword spotter's F_1 score from 75.85% (using a baseline model) to 90.91% on a ground-truth test set. We picked the keyword na'am (yes) to spot; we defined the system's input as an audio file of an utterance and the output as a binary label: keyword or filler.**

Keywords*—*Arabic; Acoustic Model; Ensemble Learning; Extra Trees; Gradient Boosting; K-Nearest Neighbors; Keyword Spotting; King Saud University Arabic Speech Database; Random Forests; Segmentation; Self-Training; Semi-Supervised Learning; Speech Recognition; Support Vector Machines; West Point Arabic Speech

I. Introduction

The Internet of things brought about low-cost connected devices, like Google Home, that can connect users in developing countries to the wealth of information on the Internet with little to no training using natural language. The barrier to use voice interfaces with natural language is — by design — low; however, the vast majority of users in developing countries don't speak the lingua franca of voice interfaces on connected devices: English. A few languages, that are commonly spoken in developed countries, have been the focus of the overwhelming majority of speech recognition applications we have today. In this project, we tackle an application of speech recognition — keyword spotting — in Arabic; it's the native language for 420 million people and a de jure or de facto language of 27 nations [1] — most of them are developing countries [2].

We picked the keyword *yes* to spot because most speech applications need to spot it in order to identify a confirmation uttered by the user. In Arabic, *yes* is *na'am* —written here in ArabTeX orthography— and it maps to the phone string *n ah C ah m* in West Point's non-standardized labeling scheme [3] (e.g., *C* represents the voiced pharyngeal fricative). Spotting of said keyword has to be accurate and fast for a speech application to be useful: keyword detection errors (either false positives of false negatives) are vexatious; slow responses are torturous. Improving the accuracy of a speech recognition system usually implies increasing its latency. One challenge of speech recognition is finding a reasonable trade-off between the two key metrics: accuracy and performance [4]. Other challenges include the scarcity of ground-truth data, variable environment settings (e.g., noisy environments), imperfect inductive transfer using cross-language models [5], etc. In addition to the above, speech recognition of Arabic presents its own set of challenges: dialects vary drastically from one region to another (across countries and within the same country), a much smaller number of corpora, and a consonantal system that's substantially different from that of the English language.

The input to our KWS system is an audio file of an utterance; the system then uses a semi-supervised SVM to segment the input and feeds each segment to a classifier — obtained from self-training ensembles — to output a predicted binary label: keyword or filler.

II. Related Work

One approach that attempted to compensate for the scarcity of ground-truth Arabic speech data is cross-language modeling (using Arabic and English corpora); it showed promising results [5] that may not be satisfactory for commercial applications: 54.02% was the best accuracy reported using acoustic models learned from the West Point corpus [3] then gradually replaced with models learned from TIMIT [6] for phonemes common in both Arabic and English. The same paper reported better results when bigram language models were added to constrain the allowed sequences of words in an utterance; however, we claim that the results reported without the assist of the bigram language models models are closer to what users of connected devices would observe since we expect the majority of input utterances to be brief (e.g., in our case, we're trying to spot a unigram).

Another paper that was published in the International Journal of Advanced Computer Science and Applications [7] used recitations of the Quran — the holy book of Islam — as a corpus for KWS experiments as it's accurately labeled and phonetically rich; however, Quran recitations have limited sound variations [8] and the lexis of Quranic Arabic is different from that of Modern Standard Arabic (MSA) [9]. The discrepancy between the training corpus and a test dataset (that represents normal speech in MSA) is possibly the reason why accuracy was low — only 87.8% — for said test dataset.

On the other hand, regarding approaches for KWS in English, Google showed a 45% relative improvement in confidence score when predicting keywords using Deep Neural Networks (DNNs) compared to competitive Hidden Markov Model-based systems [10] even though Keyword/Filler HMM is a common technique for speech tasks [11]. In 2015, Google showed a state-of-the-art method with even better results using Convolutional Neural Networks (CNNs) with fewer parameters than DNNs: a 27-44% relative improvement in false reject rate while keeping other metrics in an acceptable range [12].

III. Dataset and Features

A. Corpora

We used subsets of two corpora to train and test the system: West Point's Arabic speech corpus [3] and King Saud University's Arabic speech database [13]; both are published by the Linguistic Data Consortium (LDC), of which Stanford University is a member. Both corpora are phonetically rich and include recordings of the keyword (and filler words) by a diverse set of speakers: male and female, native and non-native, many dialects, etc. Speakers were recorded in various environments with different noise levels: a soundproof room, offices, a cafeteria, etc. Both datasets lack speech frame

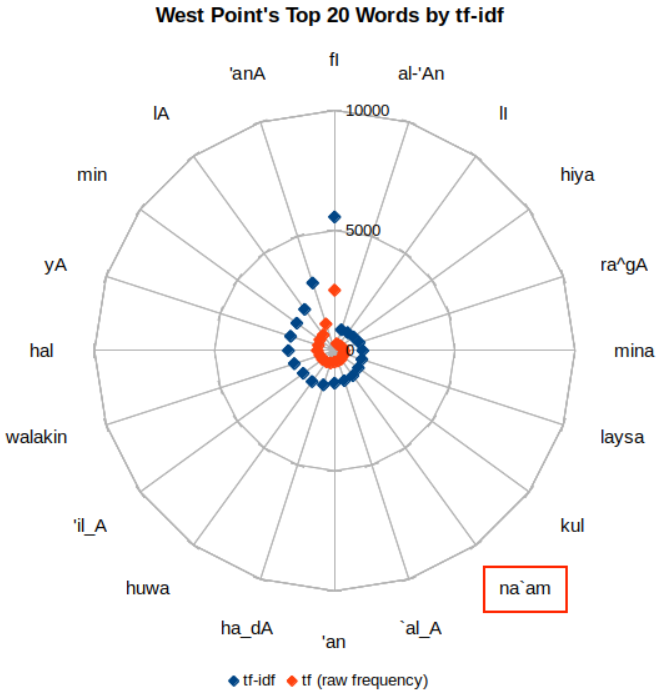

Fig. 1. A net chart of West Point's top 20 words using tf-idf [14] analysis.

labels. King Saud University's dataset only includes Arabic transcripts; we had to label the monophones of the selected utterances manually (to obtain a phonetic distribution). West Point's corpus includes monophone- and word- level master label file transcription in Hidden Markov Model Toolkit (HTK) format. However, upon inspection, we found an example of a recording, identified by the key M05ARABIC1/S1_091, that had no speech and wasn't labeled as such (i.e., it had a respective transcript); such mislabeling is easily missed in speech datasets; the fact that it wasn't reported to — and corrected by — the LDC could be seen as a sign of the apathy towards Arabic speech recognition. We used the Google Speech API to transcribe and validate West Point's corpus; post validation, we established a goal using Google's transcriptions for KWS; it achieved 100% precision and 84.25% recall (an F_1 score of 91.45%), which shows that achieving zero false positives is possible when testing using examples recorded in a controlled environment (in addition to using a plethora of data and computational power at Google's scale).

B. Preprocessing

1) Segmentation: We selected all utterances that contained the keyword from both corpora (450 from West Point's and 6940 from King Saud University's) and converted them to waveform files using the Sound eXchange (SoX) utility [15]. Since neither corpus is properly labeled with speech frame data, we used a semi-supervised SVM segmentation technique from the pyAudioAnalysis library [16] with hand-tuned parameters to generate a dataset for the classification stage. The parameters used were short-term window and step size (10 ms), size of window used to smooth the SVM probabilistic sequence (0 ms), and a strictness threshold (0.6 and 0.5 for the sentences of West Point's dataset and the word lists of King Saud University's dataset, respectively). We chose said parameters after comparing to the results of many configurations as recommended by the library. The semi-supervised SVM model was trained on a dataset provided by the library to detect silence by separating high- and low- energy short-term frames. Then we apply it on our unlabeled input files to generate a probabilistic output for high-energy frames, which is then compared against a dynamic threshold to detect active (non-silent) segments.

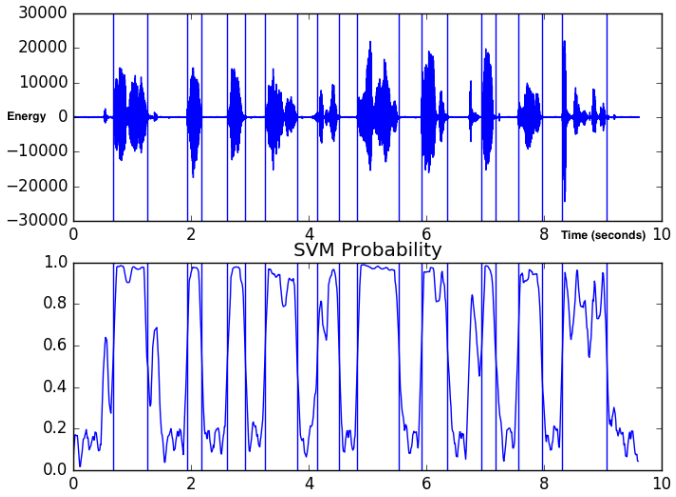

Fig. 2. Segmentation of a word-list utterance that contains the keyword (the third word) from King Saud University's dataset using silence removal.

2) Manual Labeling: Using heuristics like the index of the keyword in each utterance, we established a baseline labeling for our dataset and proceeded to label a small subset manually to use in supervised learning. We noticed that the longest duration of our keyword is about 1 second, so we examined the duration of the initially labeled positive examples (keywords) and detected outliers that were the result of a bad segmentation. We created a script to listen to each positive example and either trim it so that it only includes the keyword relabel it as a negative example if it didn't include the keyword. We then examined the corresponding negative examples that were the results of segmenting utterances that included the keyword (to make sure the keyword is never labeled as a negative example); all other negative examples came from utterances that didn't include the keyword so they were not manually checked one-by-one; we only examined a small sample and looked at a histogram of their durations. Histograms of duration for

both classes were used then for feature extraction parameter selection (mid-term window size) later on. The result of this process was a ground-truth dataset of 34,722 examples (381 keywords and 34,341 fillers) from West Point's corpus and 119 examples (51 keywords and 68 fillers) out of 62,733 segments from King Saud University's corpus; later on, we extended the latter dataset to 587 examples (234 keywords and 353 fillers).

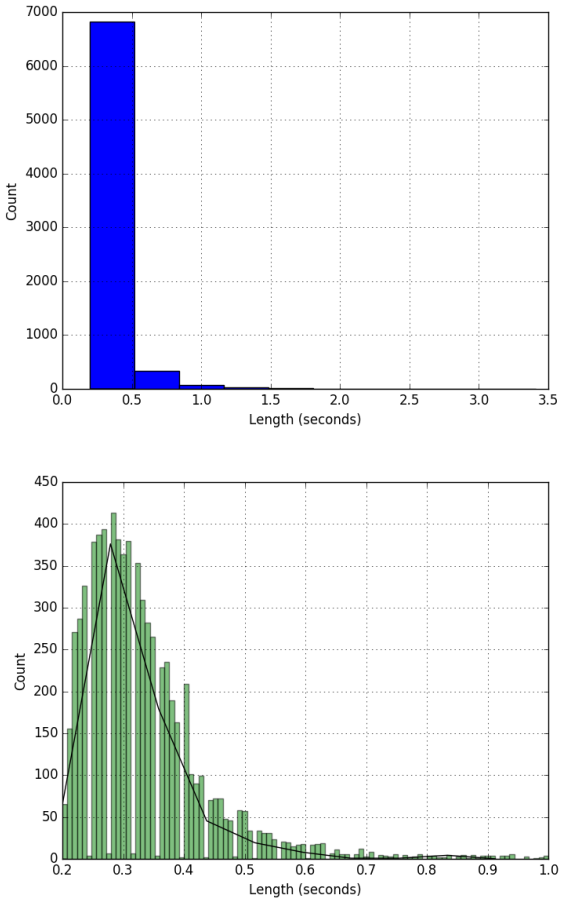

Fig. 3. Histograms (using matplotlib [17]) of length of initially labeled keywords before removing outliers (on the top) and after (on the bottom).

3) Normalization: Audio files from King Saud University's dataset (two-channel with a sampling rate of 48 KHz and a bit rate of 633 kbits/s) were normalized using the FFmpeg utility [18] to match the settings of West Point's files: a single channel (mono) with a sampling rate of 22.05 KHz and a bit rate of 354 kbits/s. Both dataset had a 16-bit precision so that wasn't changed.

4) Synthesis by Superposition: At one stage we doubled the size of a subset of West Point's data by adding synthesized white noise to generate more examples and generalize better; training using noisy data may help improve performance in noisy environments as well [19]. The added white noise is scaled down — to prevent clipping — using a gain multiplier of 5% of the maximum possible amplitude while speech is scaled up to take the rest of the amplitude [20].

C. Features

Each audio file is divided into short-term frames with sliding windows (frame shift); we extract 34 features for each frame, from which we derive features for mid-term sliding windows (mean and standard deviation of short-term frames that fit in the mid-term window) to create a 68-element feature vector. We used the pyAudioAnalysis [16] library to extract the following features (from the library's documentation):

- **Zero Crossing Rate:** The rate of sign-changes of the signal during the duration of a particular frame.
- **Energy:** The sum of squares of the signal values, normalized by the respective frame length.
- **Entropy of Energy:** The entropy of sub-frames' normalized energies It can be interpreted as a measure of abrupt changes.
- **Spectral Centroid:** The center of gravity of the spectrum.
- **Spectral Spread:** The second central moment of the spectrum.
- **Spectral Entropy:** Entropy of the normalized spectral energies for a set of sub-frames.
- **Spectral Flux:** The squared difference between the normalized magnitudes of the spectra of the two successive frames.
- **Spectral Rolloff:** The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
- **MFCCs:** 13 Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
- **Chroma Vector:** A 12-element representation of the spectral energy.
- **Chroma Deviation:** The standard deviation of the 12 chroma coefficients.

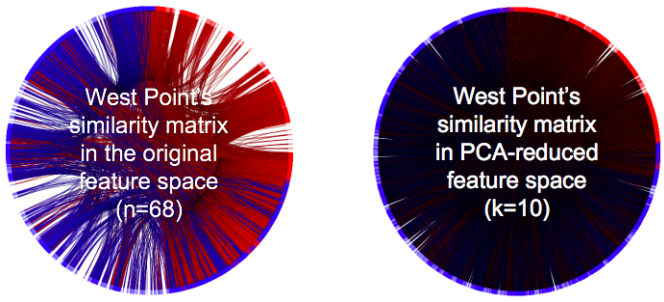

Fig. 4. Chordial charts of similarity matrices calculated from a subset of West Point's examples; red signifies 381 keywords and blue signifies 500 fillers. Reducing the number of features increases the chances of confusing keywords and fillers as per the above similarity analysis.

We forked the pyAudioAnalysis library to add persistent (on-disk) caching of extracted features to improve the efficiency of the system by extracting audio features once then reusing them many times in parallel processes.

IV. Key Metrics and Goals

We defined the input as a phone string (*n ah C ah m*) that represents the keyword to spot (*na'am*) and an audio file that represents the utterance, which may or may not contain said keyword; the output is a binary label indicating whether or not the application spotted the given keyword in said utterance. Since the keyword we want to spot is used for confirmation, we aim to maximize true positives (hits) while minimizing false positives (false alarms) even if the latter objective leads to some — tolerable — increase in false negatives (misses). Both types of error are annoying to users; however, false positives cause interruptions and unintended consequences while false negatives happen when the user is already engaged with the system and can say the keyword again. With the above in mind, we aim at maximizing precision at the expense of decreasing recall if need be. To represent the performance of our proposed system using a single score, we use the F_1 score, which incorporates both precision and recall (and unlike Figure-of-Merit, it doesn't consider a time window in evaluation):

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

V. Methods and Results

A. Parameter Optimization

Four parameters govern how the feature values are extracted are: the duration of mid- and short- term windows; and the step size for each. Optimizing said parameters for KWS in Arabic (while maximizing its F_1 score) is a major part of this project and we use multiple learning methods while doing so: Extra Trees, Gradient Boosting, K-Nearest Neighbors (KNNs), Random Forests, and Support Vector Machines (SVMs). Said learning methods were used by our fork of pyAudioAnalysis to train models using 10-fold cross-validation; each method was tuned using a single parameter as follows:

- **KNNs:** k in {1, 3, 5, 7, 9, 11, 13, 15}
- **SVMs (with linear kernels):** C in {0.001, 0.01, 0.5, 1.0, 5.0, 10.0}
- **Random Forests:** number of trees in {10, 25, 50, 100, 200, 500}
- **Extra Trees:** number of trees in {10, 25, 50, 100, 200, 500}
- **Gradient Boosting:** number of boosting stages in {10, 25, 50, 100, 200, 500}

With the exception of KNNs, which is implemented within pyAudioAnalysis [16], the library calls the respective scikit-learn [21] modules with the above parameter values. We beam-search through the parameter space (learning method, its parameter, and feature extraction parameters) using a hand-tuned parameter sweep.

Segmentation parameters (short-term window size, short-term step size, smoothing window size, and a strictness threshold) were hand-tuned to find an optimal combination on a random test sample. The West Point corpus (34,722 examples) was manually verified after segmentation (and wrong labels were fixed) while only a small test set from King Saud's corpus was manually labeled (119 out of 62,733 examples).

Then we trained various supervised classifiers using 80% (27,779 examples: 306 keywords and 27,473 fillers) of the West Point corpus exploring 872 unique models. After that, we exploited the top 135 models with the highest training F_1 score (for each learning method and feature extraction parameters tuple) and tested them using a hand-labeled test set from King Saud's corpus (119 examples: 51 keywords and 68 fillers).

And then we selected 9 feature extraction parameters tuples: from the top 4 models with the best training F_1 score, top 4 models with the best test F_1 score, and from a randomly selected model for A/B testing from. Then we expanded the search to include various learning methods and their parameters (9×8 KNN models $+9 \times 6$ models per each of the other 4 learning methods $= 288$ models). After that, we retrained the selected 228 models using a downsampled dataset from the West Point corpus (881 examples: all 381 keywords and randomly selected 500 fillers). Then tested them using the same test set from King Saud's corpus (119 examples: 51 keywords and 68 fillers), which represents 11.9% of the data.

After that stage, we added white noise to double the size of both the training data and to generalize better; then selected the top 5 models with the highest test F_1 score. Said models had 2 unique learning methods (Gradient Boosting and SVM), with 6 learning method parameters—each—to explore, and 4 unique feature extraction parameters tuples; so the beam search explored 48 unique models at this round. The learning method parameters with the best training F_1 scores (8 models) were tested against two datasets: the 119 examples from before (now 6.3% of the data) and an extended — also manually labeled — one that includes 587 examples (24.7% of the data) with 234 keywords and 353 fillers.

B. Ensemble Learning

We selected the following models as tournament winners (all scores were calculated using the larger test set of 587 examples):

Method	Parameter	MT Window (ms)	MT Step (ms)	ST Window (ms)	ST Step (ms)	Test F_1 Score	Notes
SVM	$C = 0.01$	500	200	25	10	80.53	1st (before noise)
SVM	$C = 0.01$	250	100	25	10	79.91	2nd (before noise)
G.B.	$N = 500$	125	50	25	10	79.38	1st (after noise)
RFs	$N = 200$	125	50	25	10	77.68	best tree model
KNNs	$k = 11$	400	100	40	10	72.28	best True Positives

All explored learning methods are represented above except for Extra Trees whose model was too slow to score examples online (it took 27 hours to score 30,000 labels), which we redeemed impracticla for online applications.

We then formulated two techniques to create ensembles that score a label of $+1$ (keyword) or -1 (filler) as the following:

- **F_1-Weighted:** $sign(\sum_i (prediction_i \times F_{1i}))$

- **Simple Majority:** $sign(\sum_i prediction_i)$

Votes from each predictor that's a member i of the ensemble are cast according to the strategies listed above to calculate a final verdict for each example. Testing both strategies on the ground-truth dataset showed that simple majority (uniform) method outperforms the F_1-weighted one as the threshold (minimum required F_1 score) of membership increased, which is the case for our winners.

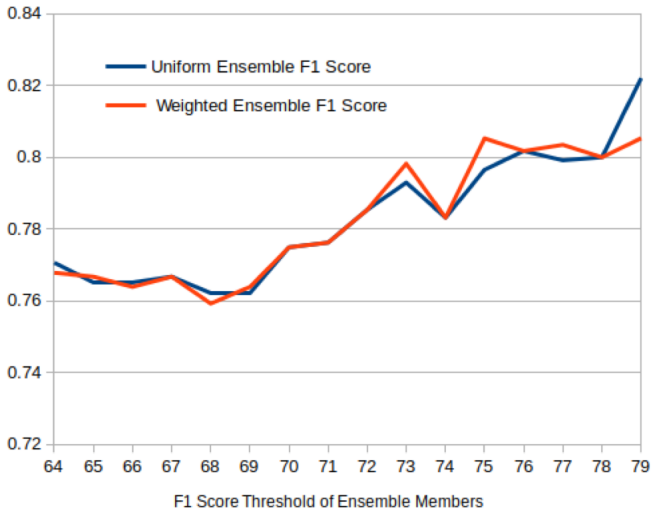

Fig. 5. Graph of ensemble performance vs. the threshold (minimum required F_1 score) of membership using test results from evaluating ensembles of a maximum of 100 best models in training before adding noise.

The following variations of ensembles made using the winners above were evaluated using the ground-truth dataset of 587 examples:

Ensemble	F_1 Score
Simple majority of top SVM and gradient boosting	90.02
Weighted five votes from all winners	79.21
Simple majority of four votes (all but RFs)	85.40
Weighted two votes from top SVM and gradient boosting	80.53
Weighted four votes (all but RFs)	80.62
Simple majority of five votes from all members	84.52

Using the best ensemble above, labels for examples in King Saud University's (unlabeled) corpus were predicted for self-training [22]. In this stage, we trained our winning ensemble's models using 15680 examples of King Saud University's corpus and tested using the hand-labeled examples from West Point's corpus; and compared the results with a similar setup that only differs in the training labels (obtained from the baseline using the heuristics of segmentation and indexes of the keyword in utterances). The best obtained model (using gradient boosting as the above winner) had a training F_1 score of 93.8 and a test F_1 score of 90.91; compared to a training F_1 score of 82.7 and a test F_1 score of 75.85 for the baseline model. In total, we trained 1220 unique models to obtain that winner.

In many examples where the model failed, we found that the classifier confused the keyword with words of similar length (e.g. *lah* in Arabic, which means *no*). The confusion could also be caused by the mislabeled training data that our ensemble couldn't classify correctly.

VI. Discussion

- Ensemble learning has a boosting effect for weak learners; self-training helped bootstrap the supervised learning framework; combining them was a big plus.
- KNNs tend to overfit; Extra Trees are slow at scoring.
- Downsampling the negative examples had the biggest impact on improving the results.
- The best performing frame and window parameters are within the literature's recommended range (even for Arabic): 25ms with a step size of 10ms.
- Beam searching through the parameter space proved very effective compared to a greedy approach.
- Running multiple model training processes in parallel in the cloud saved us a lot of time.

VII. Future Work

Attempts to train neural networks yielded an F_1 score of 41% on average; there are too many setups to test that we'd like to explore. Also, for various parameter optimization tactics, we'd like to try random, instead of grid, search for parameters.

Acknowledgment

The author would like to thank François Germain, project TA and doctoral student at the Center for Computer Research in Music and Acoustics at Stanford University, for his help with this project; and Sebastian Schuster, corpus TA doctoral student in the Department of Linguistics at Stanford University, for his recommendations and help in obtaining the corpora used in this project.

REFERENCES

[1] *The world factbook*, 2016. [Online]. Available: https://www.cia.gov/library/publications/the-world-factbook/fields/2098.html.

[2] "World economic outlook: Uneven growth—short- and long-term factors.," p. 153, Apr. 2015, ISSN: 1564-5215. [Online]. Available: http://www.imf.org/external/pubs/ft/weo/2015/01/pdf/text.pdf.

[3] *West point arabic speech ldc2002s02*, 2002. [Online]. Available: https://catalog.ldc.upenn.edu/LDC2002S02.

[4] J. Benesty, M. M. Sondhi, and Y. Huang, *Springer Handbook of Speech Processing*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007, ISBN: 3540491252.

[5] Y. A. Alotaibi, S. A. Selouani, M. M. Alghamdi, and A. H. Meftah, "Arabic and english speech recognition using cross-language acoustic models," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, Jul. 2012, pp. 40–44. DOI: 10.1109/ISSPA.2012.6310585.

[6] *Timit acoustic-phonetic continuous speech corpus*, 1993. [Online]. Available: https://catalog.ldc.upenn.edu/ldc93s1.

[7] "Audio search based on keyword spotting in arabic language," vol. 5, 2 2014. [Online]. Available: http://thesai.org/Downloads/Volume5No2/Paper_19-Audio_Search_Based_on_Keyword_Spotting_in_Arabic_Language.pdf.

[8] R. Yuwan and D. P. Lestari, "Automatic extraction phonetically rich and balanced verses for speaker-dependent quranic speech recognition system," in *International Conference of the Pacific Association for Computational Linguistics*, Springer, 2015, pp. 65–75.

[9] J. C. Watson, *The phonology and morphology of Arabic*. Oxford University Press on Demand, 2002, p. 8.

[10] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 4087–4091.

[11] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden markov modeling for speaker-independent word spotting," in *International Conference on Acoustics, Speech, and Signal Processing,*, May 1989, 627–630 vol.1. DOI: 10.1109/ICASSP.1989.266505.

[12] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, ISCA, 2015, pp. 1478–1482. DOI: http://www.isca-speech.org/archive/interspeech_2015/i15_1478.html.

[13] *King saud university arabic speech database*, 2014. [Online]. Available: https://catalog.ldc.upenn.edu/LDC2014S02.

[14] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge: Cambridge University Press, Oct. 2011, p. 8, ISBN: 9781139058452. [Online]. Available: https://www.cambridge.org/core/books/mining-of-massive-datasets/A06D57FC616AE3FD10007D89E73F8B92.

[15] (2015). Sox - sound exchange, [Online]. Available: http://sox.sourceforge.net/ (visited on 12/16/2016).

[16] T. Giannakopoulos, "Pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, 2015.

[17] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[18] (2016). Ffmpeg, [Online]. Available: http://http://www.ffmpeg.org/ (visited on 12/16/2016).

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

[20] (2016). Sox in phonetic research, [Online]. Available: http://linguistics.berkeley.edu/plab/guestwiki/index.php?title=Sox_in_phonetic_research (visited on 12/16/2016).

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] N. Fazakis, S. Karlos, S. Kotsiantis, and K. Sgarbas, "Self-trained lmt for semisupervised learning," *Computational intelligence and neuroscience*, vol. 2016, p. 10, 2016.