ISBN: 978-628-95207-4-3. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2023.1.1.1269

Analyzing and Filtering Multimedia Digital Evidence with Artificial Intelligence

Bruno Constanzo[12], Eng., Santiago Trigo[12], Eng., Valentina Fernández[1], Eng. Student,
Enzo Nogueira[1], Eng. Student, Ana Haydée Di Iorio[12], Eng.
[1] InFo-Lab, Laboratorio de Investigación y Desarrollo de Tecnología en Informática Forense
[2] Universidad FASTA, Argentina, {bconstanzo, santiagotrigo, diana}@ufasta.edu.ar

Abstract*– Digital evidence has been growing exponentially over the years, not just in size but also in quantity: we now store more files than ever, both on computers and smart devices. The use of multimedia formats has also increased, with audio and video recordings being more prevalent than they previously were. This poses a problem for digital forensic examiners, who now must sift through these files listening to what is said in them. Artificial Intelligence models and applications that can be run locally provide us with a novel way to work with these challenging forms of evidence, drastically reducing the time required to find important information in these ever more present formats, while meeting the stringent requirements of data privacy that laws impose upon investigations.

Keywords*-- digital forensics, artificial intelligence, digital evidence analysis.

Analyzing and Filtering Multimedia Digital Evidence with Artificial Intelligence

Bruno Constanzo[12], Eng., Santiago Trigo[12], Eng., Valentina Fernández[1], Eng. Student,
Enzo Nogueira[1], Eng. Student, Ana Haydée Di Iorio[12], Eng.
[1] InFo-Lab, Laboratorio de Investigación y Desarrollo de Tecnología en Informática Forense
[2] Universidad FASTA, Argentina, {bconstanzo, santiagotrigo, diana}@ufasta.edu.ar

***Abstract*– Digital evidence has been growing exponentially over the years, not just in size but also in quantity: we now store more files than ever, both on computers and smart devices. The use of multimedia formats has also increased, with audio and video recordings being more prevalent than they previously were. This poses a problem for digital forensic examiners, who now must sift through these files listening to what is said in them. Artificial Intelligence models and applications that can be run locally provide us with a novel way to work with these challenging forms of evidence, drastically reducing the time required to find important information in these ever more present formats, while meeting the stringent requirements of data privacy that laws impose upon investigations.**

***Keywords*-- digital forensics, artificial intelligence, digital evidence analysis.**

I. Introduction

Digital Forensics is the branch of forensic sciences that deals with the acquisition, preservation, extraction, and analysis of evidence obtained from computers and other digital media – it is the forensic application of Computer Sciences. In the past years digital evidence has been increasing both in size and in quantity, that is the raw number of files and digital artifacts that are involved be it instant messages, emails, location data points, rows in a specific application database, or other kind of data. As technology advances, this trend will continue, as new forms of devices and new sources of digital evidence appear.

Previously it was thought that this was mainly a problem of scale that would affect the time required to process a case [1, 2], stalling investigations as storage capacity grew larger and larger, with CPU processing speeds being unable to catch up. Largely that has not been as much of an issue as it was believed back then[1], but now data and media types are becoming a new problem: whereas in the past most forms of digital communication used to be text-based mediums (email, SMS, instant messaging), it is now more common to find audio and video evidence. “Classic” tools can usually work only over the metadata contained in these formats but provide little insight into the content itself. With instant messaging platforms having added audio messages support, a part of their users opts for this form of communication.

While the media files and content can be obtained through standard acquisition methodologies, processing their content is something that can only be done manually by the digital forensics expert, listening to each file, and transcribing it if the contents are of importance to the case. This is a very time-consuming task and can place a heavy burden upon an investigation.

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

Artificial Intelligence (AI) in general, and Deep Learning methods in particular, can help us build new tools that are capable of extracting content from these multimedia formats, and they can also be used to process and analyze the content, both audio and text, in novel ways. While speech-to-text and language processing models have been available for a time in cloud environments and services, these were not usable because law enforcement agents are required to work with evidence under strict secrecy and are forbidden from sharing any information from the cases they work on while they are still open.

In this paper Section II presents a brief historical and theoretical background for AI, speech-to-text models, and Natural Language Processing (NLP) models. Section III focuses on the tests we have carried out and proof-of-concepts of these technologies that we are building. Section IV closes with our conclusions and proposals for future work that can improve the use of AI in the analysis of digital evidence.

II. Background

A. Artificial Intelligence and Deep Learning

Artificial Intelligence (AI) is the branch of Computer Sciences that deals with the development and training of algorithms that mimic some form of intelligence [3, 4]. Machine Learning (ML) algorithms are a large branch of AI, in which data is presented to the computer with some training algorithm, which then adjusts the parameters of a model, and in this process the model "learns" (for some definition of learning). After the training is complete, the model can be queried with new data, that it has never seen before during training. Ideally, the answers it gives will be satisfactory, and the trained model will meet certain requirements for accuracy, precision, recall, F-score, and/or some other metrics [5, 6].

Artificial Neural Networks (ANN) are systems inspired by biology, that use a simplified model for the neurons where matrices and tensors represent the different parameters for them. Initially proposed in the first half of the 20th century by

McCulloch and Pitts [7], the first developments were carried by Hebb [8], Farley, Clark [9], and Rosenblatt [10], to name a few.

Werbos's backpropagation [11] and Rumelhart, Hinton and Williams multilayer perceptron [12] helped regain interest in ANNs by the 80s, solving non-linear problems that had been challenging for the previous generation of neural networks. Improvements in hardware manufacturing which increased the compute power available also helped this new era, though during the 90's interest in neural networks was waning once again.

By the start of the 2000s once again hardware and algorithmic improvements gave new traction to neural networks. General purpose computing on graphics processing units (GPGPU) provided the compute power needed for the new generation of ANNs. On the theory and algorithmic side, works by LeCun [13], Hinton [14], Ciresan [15], Krizhevsky [16], Ng and Dean [17], amongst others, laid the foundation for what would later be called Deep Learning.

Deep Learning can be summarized as the use of many-layered neural networks, interconnected in different ways, involving the use of multiple kinds of layers (convolutional layers, dropout, pooling, etc) to build networks that are more capable than feed forward networks, can attain better generalization properties, are more stable in their training, and achieve improved learning metrics. Simpler feed forward networks tend to have numerical stability issues when training, such as the vanishing and exploding gradients problems, a long-standing issue [18].

Software also developed on par with the theory and practice of machine learning, and deep learning, with notable frameworks such as scikit-learn [19], TensorFlow [20], PyTorch [21], Keras [22], and pandas [23], amongst others, being part of the basic toolbox for ML and DL practitioners.

With their improved capabilities, deep learning methods have been applied to solve a wider range of problems over the past 10 years, eventually reaching speech-to-text and natural language processing applications, topics which will be briefly discussed in the following sub-sections.

B. *Speech-to-text*

Speech recognition, also called speech-to-text (STT) is the problem of processing an audio recording of a person (or persons) speaking and transcribing the words they speak into text. It should not be confused with *voice recognition* or *speaker identification* that aims to separate and identify people speaking in a recording.

There is a rich history of speech-to-text systems, starting in the 1960's, culminating, at the moment of writing, in deep learning Long-Short Term Memory (LSTM) and Transformer based models achieving the best performance. A main characteristic of modern STT systems is that most have achieved what is referred to as *speaker independence*, which means that the system can work without any training or fine tuning with the speakers voice, something that was a staple of speech-to-text systems in the 90s and early 2000s.

As speech recognition is not the authors main research topic, we refer the reader to look for further details in the Whisper model paper [24], of which we will give a very brief overview.

Whisper is a transformer-based model that achieves close-to-human level robustness and accuracy on English speech recognition, that has been trained on multiple tasks, including the transcription of English and non-English languages and the translation from any[2] language to English. One of the main focuses of their work was to drastically extend the scale of the data that the model is trained on, using a weakly supervised training dataset. They trained on 680,000 hours of audio, in contrast with other more strongly supervised speech recognition datasets that are in the order of a few thousand hours of labeled data. Even though the labeled data obtained with their scheme is not of the same quality as other supervised datasets, this large expansion of weaker quality data allows for the training of a system with much improved performance.

OpenAI, the company that funded the research and development of Whisper, has released it under an MIT License, which is a permissive software license that allows others to build upon their efforts.

In this work, we use the models provided by OpenAI to transcribe audio extracted on digital investigations (from any source that is relevant, but mainly audio messages sent or received through instant messaging platforms) to transcribe them and then search on the text obtained, either with classical methods as string search and regular expressions, or also using natural language processing techniques.

C. *Natural Language Processing*

Natural Language Processing is the branch of AI that deals with the processing and understanding human language [25]. Even though it can work with audio and even video formats (e.g. sign languages), NLP is usually understood in the context of text processing.

Common tasks in natural language processing include the tokenization, part-of-speech tagging, stemming, parsing, named entity recognition (NER), sentiment analysis, and text classification, to name a few. NLP systems can be built to perform one or more of these, usually following a pipeline architecture where different components (now mostly based on machine learning approaches) perform a part of the processing and the input of each component is fed to the next one until the full processing is completed.

In our work we use spaCy [26], an NLP library for Python that builds on latest research techniques, and provides fast implementations, support and pretrained models for many languages, is extensible through custom components, and has a fairly easy process to train and deploy user-trained language models.

[2] Any of the languages on which Whisper was trained, in particular we make use of the Spanish transcription capabilities that their models provide.

One of the key features to choose spaCy from other alternatives was the availability of very good quality Spanish language models provided by the developers. These models can be used as given, or as basis for custom models that are trained on your own datasets.

III. Tests, Results & Discussion

For the past months we have been performing tests to utilize speech recognition and natural language processing on digital investigations. Whisper was used on test cases whenever there was one with enough audio files that it was reasonable to transcribe them into text form. That text was later loaded into the Autopsy forensic environment [27] and processed using string-based or regex-based search, or through ad-hoc scripts that utilize spaCy for NLP if the case requires a more powerful approach.

All the audio recordings evaluated were in Spanish, and the obtained text was processed with the spaCy Spanish language models, specifically their “es_core_news” family, in small, medium, and large variants. Accuracy and precision of these models will be different in other languages. Whisper in general performs better when processing English audio, since it was trained for English first, with other languages as a “nice to have”. We recommend performing tests on your own to validate the accuracy and general performance of this approach to evaluate if it is applicable in your organization.

A. *Whisper tests*

The time saved when using Whisper is substantial. Without it, digital forensic experts either had to discard the audio files as sources of evidence, or dedicate enormous amounts of time to listening to the recordings, and transcribing them if anything of interest was mentioned in them. That is not just a tedious task but is also prone to error due to loss of attention and mental fatigue after having listened to the audios for a while.

Using speech recognition, this kind of tasks can be dramatically sped up. Time estimates for human listening placed each case in the range of 3 to 6 weeks’ time for a standard case (and more in larger cases). Delegating the transcription task to the computer takes a few hours, and can be programmed to be performed as a batch-style job from one day to the next. After the transcription is ready, sifting through the results is done in a couple of hours. This workflow converts a task that used to take several weeks into something that can be done in the lapse of one or two days.

An Nvidia Quadro RTX 4000 achieves faster than real-time processing when using the “medium” Whisper model, resulting in good accuracy in the transcription of Spanish speakers. The “large” model was not tested on this GPU because it needs about 10 GB of VRAM to run, and this part has 8 GB.

Further tests were performed on a computer with an AMD Ryzen 5 4600H processor, 24GB of RAM, and an Nvidia GeForce 1650Ti 4 GB VRAM video card. In this setup, we tested both “tiny” and “small” models running on GPU, and “tiny”, “small”, “medium” and “large” models running on CPU. Timings for these models were run against a selection of benchmark files, in particular Table 1 details the timings obtained when analyzing a 43 seconds of audio of a person reading text in a noisy environment.

TABLE I
Timings Benchmark against a 43 second recording

CPU AMD Ryzen 5 4600H		
Model	Time	Times Slower
tiny	21.82 s	0.51x
small	120.61 s	2.81x
medium	454.10 s	10.58x
large	637.81 s	14.86x
GPU Nvidia GeForce 1650Ti		
Model	Time	Times Faster
tiny	8.18 s	5.25x
small	18.53 s	2.32x

The timings for the GPU on Table 1 must be considered with caution, as the audio is only a bit longer than Whisper’s “listening” window (30 seconds). In this situation, model loading times start to take a significant portion of run time, as these measurements were independent from each other. When working with larger files, load times are not as significant. It is also possible to load the model only once, either by passing the command line tool multiple files as arguments, or using an ad-hoc script that loads the model only once and runs the transcription task over them.

Regarding accuracy, the models were tested on a variety of audios, ranging from good quality recordings in noise-free environments, to very noisy environments, recordings over the telephone, and audio recorded on the street with multiple speakers. General results and findings are as follow:

- Larger models are more precise with the timings for the spoken text, and split the text over shorter lines. This is generally good for subtitling, and getting a precise excerpt of text, but not particularly important for the problem we evaluated.
- Smaller models have a reduced vocabulary and are more prone to word errors, confusing one word for another or a mix of similar-sounding words.
- In general, all models can tackle good quality recordings, but “medium” and “large” are recommended if there is significant noise.
- In general, for Spanish, we consider it is best to avoid the “tiny” model unless the recordings are of very good quality, and the hardware available cannot run any of the other models.

On few occasions, Whisper’s model “hallucinated” text that was not present in the audio:

- Smaller models (“tiny”, “small”) had a tendency to repeat text over and over, usually at the end of an audio file.
  - This could be detected and mitigated by post-processing the results with timing information

(either raw output or .srt/.ytt files) to detect overlapping regions with identical text.

- Larger would only hallucinate results when the speaker can barely be heard against a noisy background. On some occasions a person can understand what is being said, on others it was not possible.
  - Adjusting different parameters for the model could potentially help on this scenario, but further tests are needed.
  - The programmatic output of Whisper can be post-processed and use the different probabilities that are associated with each "chunk" of text extracted to determine if the model is confident on the transcription or not. A user interface to call attention on the low-score chunks would be ideal to get a human operator to listen to these regions and fix the transcription if necessary.

The majority of the hallucinations in our testing happened when working with lower precision floating point (*float16* or *fp16*, a numeric data type that allows for faster processing on certain GPUs). Not only did it give much worse results, than standard 32-bit floating point precision, but also GPU usage was not significantly lower with respect to using 32-bit precision in our tests, neither on the Quadro nor the GeForce card.

When running on CPU, PyTorch runtime immediately fell back to 32-bit floats due to lack of support for the lower precision mode, avoiding this problem entirely when CPUs are used. We did not perform any tests on quantized weights (*int8* or *int4*), but based on our experience those could potentially be even more problematic, at least for Spanish audio.

On our tests we did not find any need to adjust the parameters that Whisper lets the user configure: *temperature*, *compression ratio*, *logprob threshold*, and the *no speech threshold* parameters were used on their default values. Under these conditions, the "whisper" command line tool can be used, simply passing to it the path of the files to be processed, the model, float16 preference (defaults to True, so we would set it to False), and indicating an output path if necessary. If no output path is given, .srt, .ytt and .txt files will be created on the same path as the input audio. The subtitle files (.srt and. ytt) have timestamps in their corresponding format, whereas the .txt will have the full transcription of the text with no timing information.

B. SpaCy tests

After transcription was performed, on most of the test-cases performing a string search or a regular-expression search was enough to find the information of interest. On few occasions, more complex search and processing was needed, and spaCy was used either through an interactive python session or through scripts that would perform a specific analysis kind of search. Note that NLP over text allows the use to work over any source of text, not just the results form speech recognition, so these techniques can be applied on their own.

Working with spaCy on an interactive python shell is simple, as long as the user is accustomed to that environment. Loading a spaCy model usually takes a few seconds, and once it is in memory, text is processed by passing it to the model via a function call, which then returns a *document* object. This *document* is a sequence of tokens, and holds all the information and processing results that the model applied over the text: tokenization, part-of-speech tagging, parsing, NER results, etc.

One of the most interesting analysis that we found exploring spaCy's models capabilities is the use of lemmatization to find any occurrence of a verb, despite the many was it could be conjugated in a sentence. Iterating over the tokens of a document, the user can access the `.pos_` attribute (the part-of-speech tags) to find all verbs, and then the `.lemma_` attribute and compare it to the infinitive form of the verb (or verbs) that they're interested in.

This way, the investigator or analyst can simply build a list of verbs that could be of interest in a specific type of investigation, for example, credit card fraud or Ponzi schemes, and then analyze all the text sources from a case looking for these. When doing this type of analysis on the test cases, potential results were easily found in seconds, later requiring further consideration by the analyst.

This capability is not only limited to verbs, but we found that limiting it to them reduced the amount of false positives, as many words can share the same root.

Named entity recognition was also found to be an interesting feature, as it can help detect many things. Most spaCy standard models come with a NER component that can recognize persons, organizations, locations, and a miscellaneous category. Some of the language models, like the Dutch or English models, include many other categories such as time, money, and quantity (among others).

Using the NER component it is possible to query every document to get all the instances where different entity types are being used, aggregating them, and processing them. For example, using NER it is possible to find all the persons, locations, quantities and/or monetary values mentioned in a set of documents. Finding all these mentions can be used to quickly contrast that information with relevant persons, locations, or amounts known from the case, or find new information.

If the models provided by spaCy developers are not sufficient, it is possible to train custom models, provided there is enough labeled data available. To label the training data for our custom spaCy models, we use Label Studio [28], since it provides for easy-to-use user interfaces for multiple kinds of labeling modes and supports exporting the labeled datasets in formats that are compatible with spaCy versions 3.x.

On a test model we trained, just with 210 examples labeled on NER categories of dates, locations, miscellaneous, money, organizations, persons, quantities, and time, we obtained fairly good results in detecting these kinds of entities on real world cases. However, this model had more than a few quirks and issues, achieving lower accuracy than the standard spaCy models on the same categories, and it had a tendency to mislabel new things that it had not seen during training under any of its categories. Despite these issues, it showed the potential for improvement, and functioned as a proof of concept.

Finally, a text classification model was also trained that had the ability to detect threats of physical harm, and instances of gender-based violence. This model was trained on a custom dataset derived from publicly available anonymized data, which required data clean up and deduplication. The text categorizer component was trained on a bit less than 340 labeled examples and achieved close to 80% accuracy on the limited tests that were run. We did not have access to a test case where this model could be used, though the potential use for it, or an improved model, is evident.

Processing speeds with spaCy models were in the range of 4,000 words per second when running on CPU, on a single core, for the standard models (with many components in their pipeline) and up to 10,000 for the custom trained models, that only consisted of a tokenizer and an additional component (either NER or text categorizer).

IV. Conclusions and Future Work

The advantages that Whisper provides for the cases where it is usable are substantial. Mainly, it allows to process the audio files present in the case without having to dedicate a person to the task. The implications for that are farther reaching than it seems at first. Audio messages, same as text messages, can cover a wide variety of topics and information that is simply not relevant to the investigation. Prior to the use of speech recognition, when this was a task that could only be carried out by humans, it was largely considered a waste of time, with very few instances where one specific piece of evidence had been found listening to audio messages. With the current proof of concept application of Whisper, the transcription is carried out by computers, and the results can also be searched at blazingly fast speeds.

While the accuracy of Spanish transcriptions is not perfect, it is already better than “good enough”, even when using the “small” model. If possible, it is recommended to use the largest model that can be run on the available hardware, prioritizing accuracy rather than speed. From our testing, both the “medium” and “large” models are on the same order of magnitude in timings, but the biggest model has a lower error rate.

As mentioned before, a batch-like use can be orchestrated so that compute time is dedicated to speech recognition tasks when investigators are away from the office. If spare hardware is available, even old equipment, it could also be repurposed to perform these tasks, as long as it has enough memory and a reasonably fast CPU.

With regards to NLP, the main advantage that we have found is similar to that of speech recognition: compute time can be dedicated to performing a task that would be infeasible to perform by humans, due to the sheer scale alone. From our limited tests it is evident spaCy models are able to “read” at speeds far larger than any person can, albeit with a simpler “understanding” of the text. That means the results will still require more analysis and processing by a person, and that is a good place for these technologies, simply being tools that can be used to augment the limited human capacity, and accelerate the work of digital forensic experts and analysts.

The work presented in this paper is just a starting point, and many improvements can be made. A few ideas and opportunities, some of which we are already pursuing, are:

- Fine-tuning the speech recognition models to improve their performance on non-English languages. There is already ongoing discussion on OpenAIs online GitHub repository for Whisper about efforts in this direction, but no high-quality Spanish model has been released so far.
  - Mozilla Common Voice datasets [29] could potentially be used for this task.
- Quantization techniques could be used to accelerate the inference time on larger Whisper models. Again, there is already discussion online in this direction hinting that PyTorchs Dynamic Quantization process works “out of the box” with the models. While we are not confident on quantization from our already bad results using *fp16*, it could still be interesting to pursue for languages other than Spanish.
- With regards to spaCy (and NLP in general), it would certainly help to have a large corpus of high-quality data for Named Entity Recognition in Spanish. Many datasets and models follow the CoNLL2002 tags (also used by spaCy on their standard Spanish models), or have tags that are either too general, or too specific, and not particularly useful for the analysis that digital forensic experts would analyze.
- Finally, while the results obtained so far show promise, it is certainly uncomfortable for digital forensic experts and analysts to work outside their usual tools and environments. We are currently working to integrate our proof-of-concept scripts and programs with popular digital forensic environments through plugins.

Acknowledgment

We would like to thank Universidad FASTA and Ministerio Público Fiscal de la Provincia de Buenos Aires for their continued support to InFo-Lab.

In particular we want to thank to Roberto Giordano Lerena for his support and commitment with our labs work, and Juan Iturriaga for always having an interesting suggestion.

References

[1] S. Garfinkel, “Digital forensics research: the next 10 years” *Digital Investigations*, vol. 7, Supplement, pp. S64-S73, August 2010.
[2] G. Richard, V. Roussev, “Next generation digital forensics”, *Communications of the ACM*, vol. 49, issue 2, pp. 76-80, February 2006.

[3] F. Chollet, *Deep Learning with Python*, 2nd ed., Manning Publications, 2021.

[4] I. H. Sarker, “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions”, *SN Computer Science.* 2, Art. No. 420, August 2021. DOI: https://doi.org/10.1007/s42979-021-00815-1.

[5] scikit-learn developers, “Metrics and scoring: quantifying the quality of predictions”, *scikit-learn official documentation*, available online: https://scikit-learn.org/stable/modules/model_evaluation.html.

[6] Y. Sasaki, “The truth of the F-measure”, 2007, available online: https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf.

[7] W. S. McCulloch, W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *Bulletin of Mathematical Biophysics 5*, 115-133, December 1943.

[8] D. O. Hebb, *The Organization of Behavior*, Psychology Press, 2005 (reprint), 1949 (original publication).

[9] B. Farley, W. Clark, “Simulation of self-organizing systems by digital computer”, *Transactions of the IRE Professional Group on Information Theory*, vol. 4 issue 4, pp. 76-84, September 1954.

[10] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain”, *Psychological Review 65(6)*, pp. 386–408, 1958.

[11] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Harvard University, 1975.

[12] R. E. Rumelhart, G. Hinton, R. J. Williams, “Learning representations by back-propagating errors”, *Nature 323*, pp. 533-534, October 1986.

[13] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, November 1998.

[14] G. E. Hinton, S. Osindero, Y. W. Teh, “A fast learning algorithm for deep belief nets”, *Neural Computation 18 (7)*, pp. 1527-1554, July 2006.

[15] D. C. Cireşan, U. Meier, L. M. Gambardella, J. Schmidhuber, “Deep, Big, Simple Neural Nets for Handwritten Digit Recognition”, *Neural Computation 22 (12)*, pp. 3207–3220, December 2010.

[16] A. Krizhevsky, I. Sutskever, G. E. Hinton, “ImageNet classification with deep convolutional neural networks”, *Communications of the ACM*, vol. 60 no. 6 pp. 84–90, June 2017.

[17] Q. V. Le, M. A. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, A. Ng, “Building High-level Features Using Large Scale Unsupervised Learning”, *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[18] Y. Bengio, P. Simard, P. Frasconi, “Learning long-term dependencies with gradient descent is difficult” *IEEE Transactions on Neural Networks*, vol. 5, issue 2, pp. 157–166, March 1994

[19] F. Pedregosa et al, “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems”, 2015. Software available from tensorflow.org.Pytorch

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library”, *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019.

[22] F. Chollet & others, "Keras", *GitHub*, 2015, available online: https://github.com/keras-team/keras.

[23] W. McKinney, “Data Structures for Statistical Computing in Python”, *Proceedings of the 9th Python in Science Conference*, pp. 56-61, 2010.

[24] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision”, *arxiv eess.AS Electrical Engineering and Systems Science > Audio and Speech Processing*, December 2022.

[25] H. Lane, C. Howard, H. M. Hapke, *Natural Language Processing in Action*, Manning Publications, 2019.

[26] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, “spaCy: Industrial-strength Natural Language Processing in Python”, 2020.

[27] BasisTech, *Autopsy Digital Forensics Software*, https://www.sleuthkit.org/autopsy/.

[28] M. Tkachenko, M. Malyuk, A. Holmanyuk, N. Liubimov, *Label Studio: Data labeling software*, 2020, available online: https://github.com/heartexlabs/label-studio.

[29] Mozilla Common Voice, publicly available voice dataset, https://commonvoice.mozilla.org/.