Procesamiento del Lenguaje Natural, Revista nº 65, septiembre de 2020, pp. 119-122 recibido 30-03-2020 revisado 24-05-2020 aceptado 12-06-2020

Nalytics: Natural Speech and Text Analytics

Nalytics: Analíticas del Habla y Texto Naturales

Ander González-Docasal[1], Naiara Pérez[1], Aitor Álvarez[1], Manex Serras[1]
Laura García-Sardiña[1], Haritz Arzelus[1], Aitor García-Pablos[1], Montse Cuadros[1]
Paz Delgado[2], Ane Lazpiur[2], Blanca Romero[2]
[1]Vicomtech Foundation, Basque Research and Technology Alliance (BRTA),
Mikeletegi 57, 20009 Donostia-San Sebastián (Spain) +34 943 309 230
{agonzalezd, nperez, aalvarez, mserras, lgarcias, harzelus, agarciap, mcuadros}@vicomtech.org
[2]Natural Vox S.A.U., Bulevar de Salburua Kalea 8, 01002 Vitoria-Gasteiz (Spain)
+34 945 227 200 {pdelgado, ane, bromero}@naturalvox.com

Abstract: Call centres have long demanded technology for analysing the data they manage. In this context, we present Nalytics, a platform that integrates Speech and Text Analytics in Spanish in a modular design, and capable of customising its models to the users' demands.
Keywords: Speech Analytics, NLP, Deep Learning, Call Centres

Resumen: Los centros de llamadas han demandado durante años soluciones tecnológicas para analizar los datos que gestionan. En este contexto, presentamos Nalytics, una plataforma que integra el análisis de voz y texto en español en un diseño modular capaz de personalizar sus modelos a demanda del cliente.
Palabras clave: Analítica del Habla, PLN, Aprendizaje Profundo, Centros de Llamadas

1 Introduction

Traditionally, call centres have addressed their conversational quality audits manually, by means of agents who manage to analyse $5 - 10\%$ of the recorded material at best. Therefore, the market has long demanded Speech Analytics tools for the automatic analysis of conversational content.

Nowadays, companies which offer Speech Analytics services can be categorised into two groups: *a)* those who offer a black box flow that handles the client directly and comes with numerous functionalities and advanced technology, but through solutions in the cloud; and *b)* those who offer more adapted services but rely on more traditional search technology, such as word spotting, and work on batch mode.

Nalytics combines the best of both cases. It includes advanced technology based on machine and deep learning techniques, operates in both batch and online modes, it can be deployed as a Software as a Service (SaaS) or on premise, and offers the possibility of adapting its technological modules to the application domain of the customers. Nalytics integrates technological modules to perform Speech Analytics, including speaker diarisation and rich transcription as its main components; and Text Analytics, including segmentation and normalisation, sentiment analysis, and text classification. Within its modular design, the solution allows different combinations of these modules considering the users' needs, resulting on a powerful platform for the automatic analysis of call centres' content in Spanish.

The rest of the paper is structured as follows: Section 2 introduces a general overview of the platform; Section 3 describes the developed front-end application for the use of the platform; Sections 4 and 5 explain the technological modules for speech and text analytics, respectively; Section 6 describes the process of including new models to the solution; and finally, Section 7 adds some conclusions to this work.

2 General architecture

Nalytics is a back-end service built in Python and hosted on a Linux system. It is integrated as a web service inspired on the REST architecture, in which the communication with the REST client is performed through HTTP requests and, if necessary, JSON objects sent to a specific address (i.e., IP and

ISSN 1135-5948. DOI 10.26342/2020-65-17

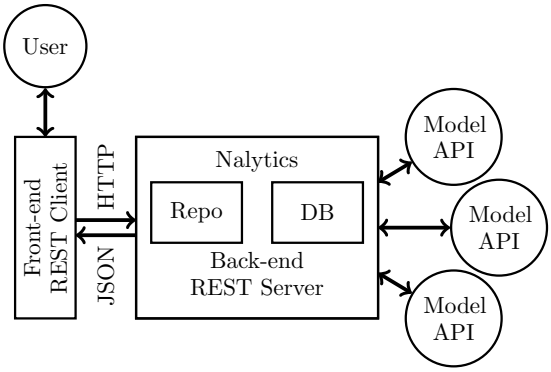

Figure 1: Minimal schematic diagram of the architecture of Nalytics

port). Once a request is received, the REST server responds with an informative JSON object and an HTTP status code. The platform is complemented by a database and a physical repository where all the information, data, and results are stored. In addition, informative web services retrieve information such as previously sent requests or status of ongoing processes. A minimal schematic diagram of the architecture of Nalytics is shown in Figure 1.

In order to service multiple customers that may have contracted different licences, all the HTTP requests are put in a priority queue inside Nalytics, where the management of the individual priorities is left to the main REST client.

Nalytics supports multiple request types for data analysis, including *offline* and *online* decoding over *synchronous* and *asynchronous* methods. For *offline* decoding, the platform loads the corresponding model in memory and frees it once the result is returned. On the contrary, in *online* decoding, the model is previously loaded into an API, which is assigned a free port and remains deployed to attend the incoming requests. It allows to operate in a real-time scenario, with a very low latency.

Additionally, in *asynchronous* methods the tasks are executed in the background depending on the available hardware resources in the server. If the request is successful, the result may be retrieved later using an informative service. In the *synchronous* requests, the master server waits for the result and returns it directly to the REST client as part of the HTTP response. However, if the task takes more than a previously defined and configurable time, it returns a timeout error. If so, the request is treated as *asynchronous* and the client is able to retrieve the result from the repository once it is completed.

Finally, the technological modules can be applied individually or as a pipeline, in what is known as a *combo* decoding. The longest *combo* decoding involves the execution of all the modules in chain from the audio preprocessing and speech transcription to text analysis (segmentation and normalisation, sentiment analysis or classification). In Figure 2, the accepted request module sequences are shown.

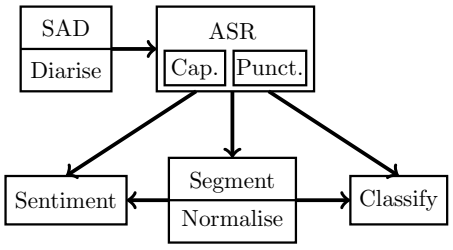

Figure 2: Accepted request submodule sequences in inference. Every module in the graph is a starting and ending node

3 *Front-end application*

Thanks to the RESTful API of Nalytics, it can be easily integrated with any front-end client. This application can serve as the main interface for communication between the user and the back-end. In this project, the client application was developed by the company Natural Vox and it is currently being served as a SaaS (Software as a Service) solution to a number of user clients. In the Figure 3, a screenshot of the current front-end application is presented.

The current front-end solution from Natural Vox is a web application that the users

Figure 3: The Natural Vox front-end application for Nalytics

can access through any web browser. The main objective is to provide a powerful tool to analyse speech and text data from call centres. The users can analyse audios by means of the rich transcription and speaker diarisation modules, whilst input texts or draft transcriptions can be classified or analysed at sentiment level. In addition, new categories for classification can be currently incorporated, adapting the modules to their domain and contents. It is worth mentioning that a rule-based anonymisation module is also included to deal with personal data. Finally, the results are generated in CSV format which are later converted into formal reports using Qlik[1].

4 Speech Analytics

This section presents the speech analysis modules in more detail, grouped as *Audio preprocessing* and *Speech transcription*.

4.1 Audio preprocessing

Firstly, the incoming audio data from the telephonic domain (e.g., raw format compressed with A-law, μ-law or G.729 codecs) is transcoded to a known audio format for Nalytics (PCM WAV 16 kHz and 16 bits). Then, the audio can be optionally processed by the Speech Activity Detection (SAD) and Speaker Diarisation modules, with the aim of discarding non-speech segments and performing speakers segmentation and clustering, respectively. Both modules have been developed using the Kaldi toolkit (Povey et al., 2011).

4.2 Speech transcription

The core of any Speech Analyitcs solution relies on the Automatic Speech Recognition (ASR) engine. Two different ASR architectures have been integrated into Nalytics. The first was estimated with the Kaldi toolkit and is composed of a hybrid Long-Short Term Memory-Hidden Markov Model (LSTM-HMM) acoustic model, where unidirectional LSTMs were trained to provide posterior probability estimates for the HMM states. In addition, a modified Kneser-Ney smoothed 3-gram language model, trained with the KenLM toolkit (Heafield, 2011), is used for decoding. The second system corresponds to an E2E neural network based on the Baidu's Deep Speech 2 architecture (Amodei et al., 2016), with an Kneser-Ney smoothed 5-gram used to rescore the hypothesis. Both acoustic and language models were estimated with the SAVAS corpus (del Pozo et al., 2014) transferred through land- and mobile-lines and augmented, as explained in (Bernath et al., 2018).

The automatic transcription can then be optionally sent through the capitalisation and punctuation modules, which take use of bidirectional Recurrent Neural Networks (RNNs) with an attention mechanism (Tilk and Alumäe, 2016) and a *recasing* model trained with the Moses toolkit (Koehn et al., 2007), respectively.

5 Text Analytics

This section introduces the different modules for text analysis available in Nalytics: *Text preprocessing*, *Text classification* and *Sentiment analysis*.

5.1 Text preprocessing

The text preprocessing module can be subdivided in two different tasks: *Normalising* the text, that is, correcting misspelling errors; and *Segmenting* or dividing each sentence into spans with potentially different polarities (e.g., "I loved the service but my problem was not solved" starts with positive polarity but ends with a negative remark).

The normalisation module is built on Hunspell[2]. Through configurable dictionaries and morphotactic rules, Hunspell detects possible typographic and orthographic errors and suggests correction candidates for each. Then, the actual correction is chosen using heuristics that involve Levenshtein's distance and knowledge about frequent orthographic errors in Spanish.

With respect to segmentation, the module first locates in the input text adversative conjunctions and phrases that may indicate contrast (e.g., "but", "on the contrary", etc.). Then, it obtains the dependency tree of the text using spaCy[3]. Finally, the text is segmented via heuristics based on these two pieces of information: each segment is a phrase or clause headed by the token an adversative phrase depends on. This module's behaviour can also be configured by incorporating new segmentation rules through configuration files.

[1] `https://www.qlik.com`

[2] `https://hunspell.github.io`

[3] `https://spacy.io/api/dependencyparser`

5.2 Text classification

Nalytics offers a diverse set of classification algorithms in an attempt to cover a wide range of application scenarios –read data availability, computational power, and so on–: *a)* Multinomial Naive Bayes (MNB) and Support Vector Machines (SVM), as implemented in scikit-learn[4]; *b)* spaCy's ensemble implementation[5] of Convolutional Neural Networks (CNN); *c)* finally, a learner for sequence-labelling tasks based on Conditional Random Fields (CRF), from sklearn-crfsuite[6].

5.3 Sentiment analysis

The sentiment analysis module is a specialisation of the Text classification module; that is, it offers the same text classification model learners and decoding functionalities, but oriented towards the sentiment analysis task. Specifically, the classifiers benefit from a rich set of features well-known for improving sentiment classification results, such as Brown clusters and gazetteer lookup. In addition, this module takes special advantage of the Text preprocessing module, which divides clauses with potentially different polarity.

6 Adapted models acquisition

As previously stated, Nalytics includes mechanisms for incorporating new models to the internal database. In the scope of speech recognition, externally created models can be transferred by placing them in the repository and executing a single HTTP request that registers the model in the database. This is implemented both for Kaldi and E2E models, whilst it is not supported for the SAD and diarisation models yet. New rules for the text preprocessing segmentation submodule can be registered in a similar way.

In the case of Text Analytics, new models can be trained directly from Nalytics. If the training process is successful, the resulting model is automatically registered in the database and placed in the repository for future use. Nalytics accepts all the training parameters exposed by the aforementioned libraries (Section 5.2).

These interesting functionalities allow Natural Vox to offer customised models to their user clients, considering that all the technological modules included in Nalytics perform better if they are adapted to the application domain.

7 Conclusions

This work presents a new solution for Speech and Text Analytics. Nalytics takes advantage of the last modelling paradigms to construct a back-end platform that can be easily integrated with any REST based client. Its operation modes, the available request methods, its modularity, powerful functionalities along with the variety of its technological modules turns Nalytics into an attractive solution that is already being exploited in the market.

References

Amodei, D., S. Ananthanarayanan, R. Anubhai, et al. 2016. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In *Proceedings of ICML 2016*, volume 48, pages 173–182.

Bernath, C., A. Alvarez, H. Arzelus, and C. D. Martínez. 2018. Exploring E2E speech recognition systems for new languages. In *Proceedings of IberSPEECH 2018*, pages 102–106.

del Pozo, A., C. Aliprandi, A. Álvarez, et al. 2014. Savas: Collecting, annotating and sharing audiovisual language resources for automatic subtitling. In *Proceedings of LREC 2014*, pages 432–436.

Heafield, K. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of WMT 2011*, pages 187–197.

Koehn, P., H. Hoang, A. Birch, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL*, pages 177–180.

Povey, D., A. Ghoshal, G. Boulianne, et al. 2011. The Kaldi Speech Recognition Toolkit. In *Proceedings of ASRU 2011*.

Tilk, O. and T. Alumäe. 2016. Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration. In *Proceedings of INTERSPEECH 2016*.

[4] `https://scikit-learn.org`

[5] `https://spacy.io/api/textcategorizer#architectures`

[6] `https://sklearn-crfsuite.readthedocs.io`