Proceedings of the 1st International Workshop on Language Technology Platforms (IWLTP 2020), pages 112–120
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

From Linguistic Research Projects to Language Technology Platforms : A Case Study in Learner Data

Annanda Sousa[1], Nicolas Ballier[2], Thomas Gaillat[3], Bernardo Stearns[1], Manel Zarrouk[4], Andrew Simpkin[1], Manon Bouyé[2]
National University of Ireland Galway[1], Université de Paris[2], Universités de Rennes 1&2[3], Université Sorbonne Paris Nord[4]
Insight Centre for Data analytics[1] , CLILLAC-ARP F-75013[2], LIDILE[3] , LIPN [4]
{a.defreitassousa1, bernardo.stearns, andrew.simpkin}@nuigalway.ie,
nicolas.ballier@u-paris.fr, thomas.gaillat@univ-rennes2.fr, mbouye@eila.univ-paris-diderot.fr, zarrouk@lipn.univ-paris13.fr

Abstract
This paper describes the workflow and architecture adopted by a linguistic research project on learner data. We report our experience and present the research outputs turned into resources that we wish to share with the community. We discuss the current limitations and the next steps that could be taken for the scaling and development of our research project. Allying NLP and language-centric AI, we discuss similar projects and possible ways to start collaborating towards potential platform interoperability.

Keywords: NLP text processing, Machine Learning, Linguistic Data workflow, User Interface (UI)

1. Introduction

This paper illustrates the current shift from language sciences to linguistic data science. We intend to describe the prototype of an Automatic Essay Scoring system (AES) user interface predicting proficiency levels in English and discuss scalability, interoperability and some development issues when sharing our models and other research outputs of our project.

Automating language assessment is a task conducted with Automatic Essay Scoring systems (AES). Initially based on rule-based approaches (Page, 1968), more modern systems now rely on probabilistic models. Some of these models depend on the identification of features that are used as predictors of writing quality. Some of these features operationalise complexity and act as criterial features in L2 language (Hawkins and Filipović, 2012). They help build computer models for error detection and automated assessment and, by using model explanation procedures, their significance and effect can be measured. Recent work on identifying criterial features has been fruitful, as many studies have addressed many types of features. However, most of the studies (one notable exception is found in (Volodina et al., 2016), with a system designed for Swedish) are experimental and do not include any automated pipeline that can handle user data from input to output. In other words, pre-processing and data analysis are not necessarily connected to any machine learning module and a user interface. Most experiments include several experimental stages of data modeling, which impedes any real-life exploitation of the models such as a student typing a text to have it graded.

The work on criterial features has also raised the need to build systems dedicated to linguistic feature extraction. The purpose of such a task is to build datasets reflecting the multi-dimensionality of language. Several tools have been developed to suit the needs of specific projects in the extraction of linguistic complexity features (Lu, 2014; Crossley et al., 2014; Crossley et al., 2019; Kyle et al., 2018). These tools provide features of different dimensions of language. However, it is not possible to apply them to a single data set in one operation. The researcher who wants to weigh the significance of all these features would benefit from a single tool applied uniformly to any data set.

Our proposal stems from a project dedicated to predicting proficiency levels in learner English. This system is made up of a user interface in which learners of English can type in a text and immediately be prompted with an assessment of their proficiency level after submission. The system was designed following a modular approach which provides room for other researchers' models. We show that it is possible to use what we have called 'the DemoIT infrastructure' to implement other models dedicated to processing texts with a view to classify them according to predetermined classes. In addition, we have derived a feature extraction pipeline from the demo and it enables researchers to build datasets by applying several state-of-the-art tools for further analysis.

In an effort to contributing to the FAIR paradigm, we have made available the code of the interface, the initial dataset and our statistical model (the .sav file). This how-to paper guides the computationally literate linguist from the data modelling to the actual web-interface for the deliverables of her linguistic project. Our case study is at the crossroads of

- research projects in applied linguistics,
- containerisation and virtualisation technologies for Language Technology Platforms,
- development of Language Technology platform interoperability: we present our web application and our workflow, as well as the exchange models, data and metadata...

The rest of the paper is organised as follows: section 2 presents some comparable tools for the analysis of complexity. Section 3. describes the context that triggered the need of the implementation of the infrastructure, which is a project aiming to automatically predict the CEFR level of a language learner. Section 4 details the tool we are proposing in this paper as well as some suggestions of improvements. The solution we adopted, "DemoIt", is a web-based infrastructure that allows users to demo their text processing systems easily in a scheduled asynchronous way. Section 5 presents the infrastructure we have adopted and the sub-components that can be re-used. Section 6 details the resulting resources we make available as deliverables of the project. Section 7 discusses our next steps in relation to other similar infrastructures, taking into account interoperability, multilingualism, scalability and legal restrictions (GDPR and copyright).

2. Existing tools for linguistic complexity feature extraction

A number of projects already exist in the domain of complexity feature extraction. Specific tools are dedicated to a specific dimension of complexity. A number of tools focus on lexical complexity, e.g. LCA (Lu, 2012) and TAALES (Kyle et al., 2018). Other tools focus on syntactic complexity, e.g. L2SCA (Lu, 2010) and TAASC (Kyle et al., 2018). Other tools focus on pragmatic dimensions, e.g. cohesion with TAACO (Crossley et al., 2019) and Coh-Metrix (McNamara et al., 2014). All these tools provide many metrics of one dimension to build datasets for further analysis.
More recently, work has been invested in developing common frameworks to support data interoperability with shared tools. CTAP (Chen and Meurers, 2016) is such a tool and allows a researcher to select various types of linguistic features to extract prior to building a customised data set. This approach provides the benefit of letting researchers choose and apply complexity analyses from a broad set of available features.

3. The Project: a Machine Learning Driven Language Proficiency Level Prediction

This section presents the experimental setup, the components of our project.

3.1. Aims of the Project

Our project aims to investigate criterial features in learner English and to build a proof-of-concept system for language proficiency level assessment in English. Our research focus is to identify linguistic features and to integrate them within a system with a machine learning component. The purpose is to create a system to analyse learner English essay writings and map them to specific language levels of the Common European Framework of Reference (CEFR) for Languages (European Council, 2001).
The proposal is a supervised learning approach in which we build several models designed to assign levels of the CEFR, which, to the best of our knowledge is novel. The system is trained on a database of more than 40,000 texts (approx. 3,298,343 tokens) that have already been labeled and grammatically annotated (Geertzen et al., 2013; Huang et al., 2018). The model relies on error-independent features of English to build a multi-dimensional feature representation of written essays. Figure 1 recaps the pipeline of the project.

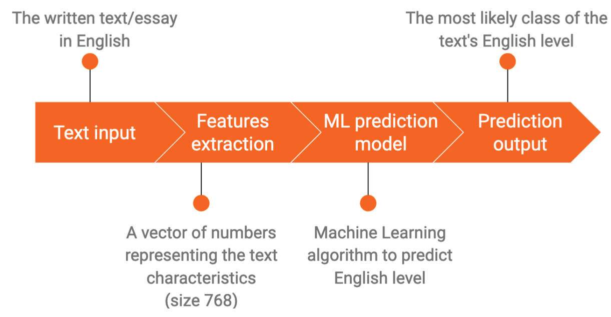

The written text/essay in English
The most likely class of the text's English level
Text input
Features extraction
ML prediction model
Prediction output
A vector of numbers representing the text characteristics (size 768)
Machine Learning algorithm to predict English level

Figure 1: The pipeline for our project

3.2. Experimental Setup

3.2.1. Corpora and Dataset

The model was trained and tested on the Spanish and French L1 subsets of the Education First-Cambridge Open Language Database (EFCAMDAT) (Geertzen et al., 2013), an 83-million-word corpus collected and made available by Cambridge University and its partner, the organization Education First. Data [1] from 49,813 texts written by 8,851 learners were extracted. The model was also tested[2] on the CEFR ASAG corpus (Tack et al., 2017), another collection of learner texts made up of short answers to open-ended questions and written by French L1 learners of English. The texts were graded with CEFR levels by three experts.
By using the aforementioned corpus subsets, we implemented a program pipeline which is designed to convert the texts into series of values, subsequently used as features. Several state-of-the-art tools are exploited to extract features of several linguistic dimensions and create three datasets. Two internal sets are created from the 49,813 observations, i.e. the training set (75% randomly extracted from the EFCAMDAT corpus) and a test set (25% randomly extracted). One external data set was created from the ASAG-CEFR corpus including 299 observations. The internal dataset will be made available online for the research community on the EFCAMDAT website. Programming scripts will also be made available via an online software development platform. In order to ensure compliance with the General Data Protection Regulation (GDPR), data will be anonymised and no personal identification of learners will be used and published.

3.2.2. Feature Extraction

The model relies on a dataset of linguistic complexity metrics of different dimensions: syntactic, lexical, semantic, accuracy and pragmatic. These metrics form numeric feature vectors of values and characterise the learner texts. The vectors are matched with the CEFR levels assigned to the texts. We use several tools to compute the met-

[1]The University of Cambridge and English First took no part in the data manipulation. The dataset including the EFCAMDAT texts will be hosted by Cambridge, in accordance with the corpus regulations. Access is free for academic non-commercial uses, provided potential users request permission using an academic email address.

[2]Evaluation results are discussed in (Gaillat et al., submitted)

rics. Syntactic complexity measures are computed with the L2 Syntactic Complexity Analyzer (L2SCA). These tools rely on the Tregex module of Stanford CoreNLP (Manning et al., 2014) for phrase constituent retrieval. The Tool for the Automatic analysis of Syntactic Sophistication and Complexity (TAASC) is also used to compute ratios and scores of syntactic complexity such as prepositions per nominal, adjectival modifiers per object of the preposition, and also probability that two items occur together. For lexico-semantic features, the pipeline implements the Lexical Complexity Analyzer (LCA) relying on Treetagger (Schmid, 1994) to compute lexical diversity metrics. The Automatic Assessment of Lexical Proficiency (TAALES) tool computes includes 130 lexical indices with classic lexical complexity metrics and psycholinguistic properties of words. These properties are based on judgments of concreteness, familiarity, imageability, or supposed age of first exposure. The TAALES indices include frequencies, ratios of lexical words and n-grams as well as comparative metrics sourced from reference corpora. The textstat Python library[3] was used to compute readability metrics that indicate the level of difficulty of texts. Accuracy features are computed with the pyenchant Python library (Kelly, 2016) [4] for misspelt words. Regarding pragmatic features, the pipeline includes the Tool for the Automatic Analysis of Cohesion (TAACO) which computes metrics based on referential and discourse characteristics such as pronouns, lexical overlaps and connectives. In total, 768 different features were extracted and merged into one dataset to input into the classification models.

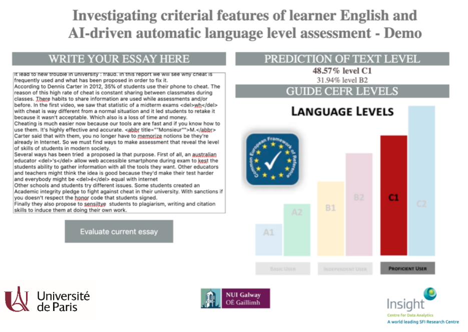
3.2.3. CEFR Level Classifier

The aim was to construct a classification model of learner CEFR levels (A1, A2, B1, B2, C1, C2, ie from beginners to advanced speakers). Among several model types tested, the optimal classification performance in the testing data set was found using multinomial logistic regression . The classifier using all features reached 82% accuracy (0.80 mean F1-score) on a six-point scale classification.

Given that the levels are ordinals, one of the reviewers suggested misclassifications B1 for B2 should incur different costs than, say, misclassifying B1 for C2. We did not resort to cost matrix, a system [5] used in the Cap2018 data challenge (Ballier et al., 2020) with the same classification task. The cost matrix used in this data challenge penalised a default assignment to A1 (because of a skewed dataset) and rewarded B1 versus B2 distinction (a sensitive boundary for some educational institutions like engineer schools). With hindsight, the robustness of such a cost matrix should be tested among several candidates to assess the consequences of penalisation weights.

3.3. The Infrastructure

To be able to demonstrate and test our model in order to have a proof-of-concept, we decided to create a web-based infrastructure that i) handles the Input/Output from and to the user and ii) schedules the tasks to and from the classifier. The infrastructure's primary requirement was to process new texts for on-the-fly metric computation followed by classification. As a result, a web interface outputs the CEFR level predicted for the new texts. This infrastructure is composed of Docker modules (Merkel, 2014), which are interconnected to handle data ingestion, processing and model classification. The infrastructure is built to allow model switching. The system can be modified in three points. The feature extraction pipeline can be modified so as to compute different metrics. The classification model can be changed to match the extracted features. The User Interface (UI, cf. Figure 2) can be modified according to the task at hand.



It lead to new trouble in university : fraud. In this report we will see why cheat is frequently used and what has been proposed in order to fix it.
According to Dennis Carter in 2012, 35% of students use their phone to cheat. The reason of this high rate of cheat is constant sharing between classmates during classes. There habits to share information are used while assessments and/or before. In the first video, we saw that statistic of a midterm exams <del>wh</del> with cheat is way different from a normal situation and it led students to retake it because it wasn't acceptable. Which also is a loss of time and money.
Cheating is much easier now because our tools are are fast and if you know how to use them. It's highly effective and accurate. <abbr title=""Monsieur"">M.</abbr> Carter said that with them, you no longer have to memorize notions be they're already in Internet. So we must find ways to make assessment that reveal the level of skills of students in modern society.
Several ways has been tried a proposed la that purpose. First of all, an australian educator <del>'s</del> allow web accessible smartphone during exam to kest the students ability to gather information with all the tools they want. Other educators and teachers might think the idea is good because they'd make their test harder and everybody might be <del>é</del> equal with internet
Other schools and students try different issues. Some students created an Academic integrity pledge to fight against cheat in their university. With sanctions if you doesn't respect the honor code that students signed.
Finally they also propose to sensiltye students to plagiarism, writing and citation skills to induce them at doing their own work.

Figure 2: The user interface prototype

4. A Web-based Scheduling Infrastructure for Easy Text-Processing System Demonstration

4.1. Description of the Infrastructure Flow

Front-end. The UI demo has two main components (Figure 3), i) the web app component that interacts with the user on the web browser by receiving and responding text, and ii) the background component that manages the text processing system (in our case, extracting features and using them to classify the text's English level). These two components communicate with each other using a message broker - Redis, the third component of the UI demo. We created these different components in order to decouple the web app from the processing part, so the web app interface would not be blocked by a given request. This architecture provides the benefit of availability for new input, while processing feature extraction and classification to get a response for the current user.

Back-end. The environment of the web app and the processor are separated into two different docker containers, both of them having an instance of a Celery app. Celery works in an asynchronous way, from one side the Celery scheduler is responsible to create tasks, and from the other side the Celery worker is responsible to process the Celery scheduler created before. The web app is an instance of a Celery scheduler while the processor is an instance of a celery worker. When the web app receives a request with a text to process, it uses Celery to create a task and put it into the Celery queue by sending the text to be processed together

[3]see https://pypi.org/project/textstat/
[4]see https://pypi.org/project/pyenchant/
[5]http://cap2018.litislab.fr/competition_en.pdf

with the task ID through Redis. The processor docker, a Celery worker, communicates with Redis polling for tasks to process. When it receives a task from Redis, the worker processes it and uses Redis again to communicate the processed prediction level of the text. In the end, the web app can consult Redis, via the ID task, to get the result of the processing.

4.2. Plugging into the Model

Prior to plugging in a new model, it is necessary to conduct a supervised learning method on some data in order to fit a model. Once the model is tuned, a .sav file can be retrieved and placed into the architecture as shown in Figure 3. This .sav format stores models as a binary file. It is also essential to match the input features of the new model with the output features of the processing pipeline. The features created by the tools need to be filtered in order to pass only the data with the required features into the model.

4.3. Tests and Redeployment

We deployed our docker on an a web platform hosted by the French Linguistic Platform infrastructure HUMA-NUM [6]. The deployment of our UI was tested on HUMA-NUMm for a deployment on a Virtual machine using our docker virtualization. The system is fast and the HUMA-NUM team is very reactive. We needed to change the port 80 as the web interface required a different access port. [7] The feature extraction tool was also tested on this virtual machine. You need at least 32 CPUs 32 Gigabytes. We processed a 2019 version of the entire EFCAMDAT and it processes 100 files every five minutes. This confirms our desire to optimise the feature extraction. The feature extractor is optimised in the sense that each repertory is turned into an individual .csv files for the output data, so that the calculated features are regularly saved in case of crash (in the case of non-utf8 files, or unexpected sequences of commas, as experience seem to show).

4.4. Infrastructure Requirements

The system relies on the open-source Docker technology[8]. It is made up of three Docker containers, each designed for a specific purpose as detailed in 4.1. In order to run the system, docker must be installed. Docker-compose[9] is also required to run the three container Docker applications. Note that the infrastructure relies on a number of data handling technologies and a framework that do not require installation as they are located within Docker containers. This includes Redis[10], a database management system, as well as the Flask web application framework[11].

5. Available Resources

This section describes the current state of our platform-related project and research outputs.

[6]https://www.huma-num.fr/about-us

[7]http://linguisticdataprocessing.huma-num.fr/

[8]see https://www.docker.com/

[9]see https://docs.docker.com/compose/install/

[10]see https://redislabs.com/

[11]see https://www.palletsprojects.com/p/flask/

5.1. Research Output

The project has yielded a number of resources and tools presented in Table 1 and all referenced in the project's web page [12]. The first tool is dedicated to end users, i.e. learners of English. A User Interface (UI) demo program provides a prototype for real-time proficiency assessment of new texts. Texts in English of more than 70 tokens are assigned probabilities of belonging to A1, A2, B1, B2, C1, C2 of CEFR levels. A documentation is available and it can be installed from NUIG Insight's Gitlab repository on request. Once installed, the demo acts as a web server, handles the text processing pipeline and interfaces with the results sent from the classification model as explained in Figure 3.
The second tool aims at the research community and focused on criterial features and AES systems. Researchers in this domain need to test different types of linguistic features to assess whether they are potentially criterial in determining CEFR levels (Hawkins and Filipović, 2012). We provide a tool that allows researchers to process batches of learner texts and output data sets ready for analysis. This tool includes the same processing pipeline as the UI demo and is only operated with the shell or command line. Users only have to place texts in a directory used as program input. It is available from NUIG Insight's Gitlab repository on request.
In the course of developing these tools, a number of data resources were created. First, the data set used for modeling in the UI demo system is available and can be exploited for other types of analyses grounded in linguistic complexity. The data set is composed of a list of metric values matched with CEFR levels. Secondly, the classification model implemented in the UI demo program is also available for download. Following the latest version of Nakala, the dataset warehouse hosted and managed by HUMA-NUM, we will upload our datasets with the corresponding permanent handles and DOIs to be attributed by HUMA-NUM [13].

5.2. Technical Aspects

For the web architecture, we have adopted Celery to avoid bottlenecks and nevertheless allow several users to query the system at the same time. As one of the reviewers pointed out, this decision has some consequences as opposed to an event-based approach. We intended to create the UI demo as a proof-of-concept for the project with time limitation. Celery seemed to be the best choice for us considering i) we were using a Python web application, ii) our team did not include specialists of event-based technologies, so the learning curve to implement using Celery was shorter than using an event-based approach, iii) our provisional goal for the project's scope was to make the UI demo available for more than one user, although we did not have the budget to support a massive number of users connecting to the system at the same time. We feel we can recommend this solution for a small-scale implementation but definitely, because of scalability limitation , future developments of our project could include an

[12]see www.clillac-arp.univ-paris-diderot.fr/projets/ulysse2019

[13]https://humanum.hypotheses.org/5989

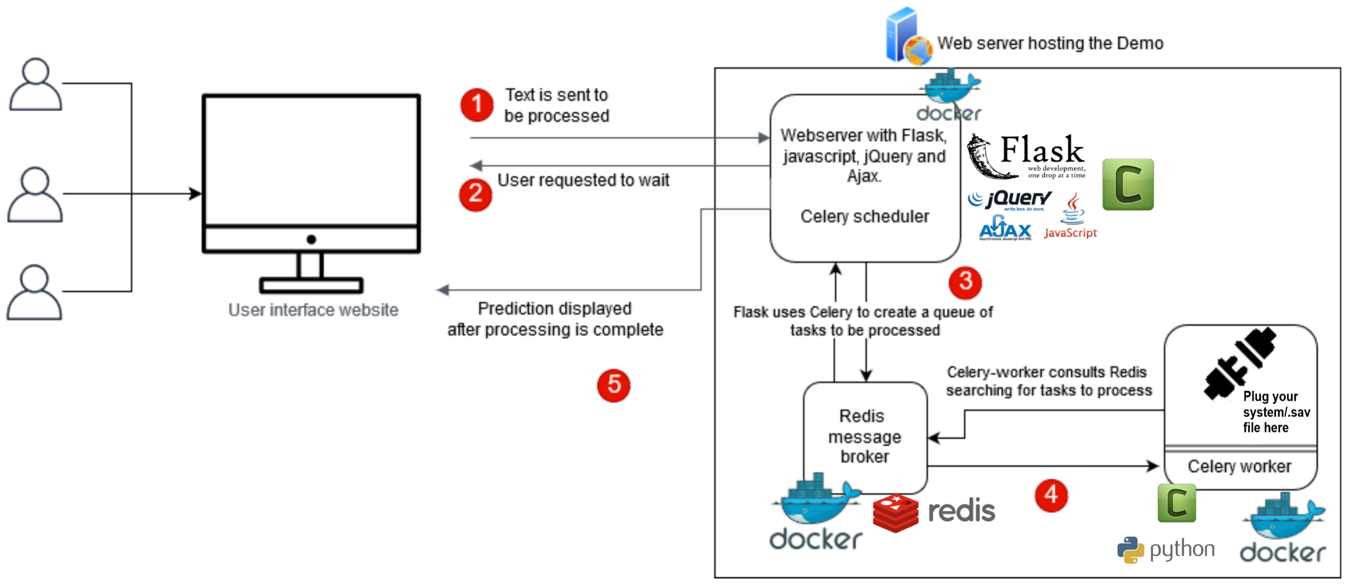

Web server hosting the Demo
Text is sent to be processed
User requested to wait
User interface website
Prediction displayed after processing is complete
Webserver with Flask, javascript, jQuery and Ajax.
Celery scheduler
Flask uses Celery to create a queue of tasks to be processed
Celery-worker consults Redis searching for tasks to process
Redis message broker
Plug your system/.sav file here
Celery worker
1
2
3
4
5

Figure 3: The customisable infrastructure

Resources	Availability
DemoUI (Web-based interface) http://ulysses.datascienceinstitute.ie:8080	Universal
Fully functional batch feature extraction tool (with pipeline modification access)	Restricted private Gitlab
Numerical Dataset metrics.csv file	Universal
Hybrid Dataset Texts_metrics.csv file	Universal (registration)
Statistical model	Universal model.sav file
"DemoIt" insfrastructure's source code https://gitlab.com/ulysses2019/ulysses	Universal

Table 1: The granularity of availability of the resources

event-based distributed approach for services communication/scheduler. This would make our system more robust and able to deal with more users at the same time.

For the feature extractor and individual distribution of the UI for researchers, we tested the Docker setup on MS Windows 10, iOS (Mojave) and Linux (Ubuntu 18.06). Because TAALES loads greedy frequency inventories and parsing the data is also memory-demanding, 16 Go of RAM are necessary for the execution (3 CPUs and 7 Gigas recommended for preferences in the docker). We wrote a user's guide in English for the installation of each tool. The whole annotation pipeline is in python, building from the former blocks from the tools. We have opted for a 3.6 version of python.

5.3. Governance and Uptake

The project was funded by the two partner countries and partners abode by the standard legal framework in use for this binational scheme [14]. Research papers were signed by the members of the projects of the two teams. Though we acknowledged respective percentages of ownership in relation to the input of the programming team, we agreed on Creative Commons Non Commercial Share Alike Licence for all our research outputs.

In terms of social aspects and community, we targeted two types of audiences. Our on-line prototype aims at learners of English and teachers in classroom environments. It is maintained until end of 2020 on the Irish partner infrastructure. To ensure sustainability afterwards, we have adopted the HUMA-NUM infrastructure [15] to host our project. As a prototype, it may experience scalability issues. The feature extraction tool was designed to be of interest to a potential consortium of researchers sharing a similar aim. Researchers working on linguistic complexity and conducting feature extraction tasks (for instance, for text classification purposes) may find the tool a useful assistant as it avoids coding. Specialists in Second Language Acquisition or Learner Corpus Research may benefit from the customisable micro-systems implemented in LS2CA_MS. For this tool parsing English data based on LS2CA (Lu, 2014), simple Tregex syntax (Levy and Andrew, 2006) can be used to create new features for the analysis of micro-systems (Gaillat et al., submitted).

6. Perspectives and Improvements

This section discusses developments in the making.

[14] A Guideline (in French) of recommended good practices for holders of this grant is available here: https://www.campusfrance.org/sites/default/files/medias/documents/2017-11/guide_bonnes_pratiques_ulysses.pdf

[15] https://www.huma-num.fr/about-us

6.1. Crowdsourcing Derived Applications and GDPR

To improve our model, we would like to store users' input to exploit it in further analyses. For the time being, the essays /texts submitted for assessment are not stored. We would like to make sure further crowdsourcing developments are compliant with GDPR and consider publishing general conditions of use warning the user that the texts used as queries will be stored to fine-tune the models. Anonymising the data for crowdsourcing will be carried out in accordance with (Klavan et al., 2012).
More generally, and more theoretically, the question is the compatibility of the models with the features when the system needs to evolve. If we collect more data (and possibly add other features), we will probably need to fine-tune the statistical models. We do not know whether this should lead to a standardization of (linguistic) features and interfaces as one of the reviewers suggested, but we suggest that an adaptation of the models can be reimplemented in the architecture by modifying the .sav file. We are not aware of any interface or standards to cater for this need.

6.2. Engineering and Interoperability

This section sums up current (and future) developments in the making of our project, with a view to offer more interoperability with existing platforms or similar projects.

6.2.1. Feature Engineering and Dimensionality Reduction

We spent much time integrating the various tools, and necessarily more time on feature collection and extraction than on dimensionality reduction. We are currently processing a complete dump of the EFCAMDAT dataset to address these issues and (Gaillat et al., submitted) reports preliminary findings on the French and Spanish datasets. The project was to build a proof-of-concept for the automation of proficiency level assessment. Further developments are required to improve the system. In its current version the model relies on a broad set of features (over 750) which makes it prone to overfitting. It is thus necessary to find a simpler model based on less features. This, in turn, will impact the data processing pipeline as only those tools related to selected features will be kept. We have obtained preliminary results with a new model based on the elastic net regression method (Zou and Hastie, 2005) trained on the EFCAMDAT training set and tested on another data set extracted from a totally different corpus, i.e. the CEFR-ASAG corpus (Tack et al., 2017). This method comes with the benefit of including feature dimensionality reduction. Using just 44 features classification showed 75.0% accuracy (CI [74.3, 75.8], $p<0.001$) and 59.2% (CI [53.4, 64.8], $p <0.001$) on the EFCAMDAT and CEFR-ASAG test sets respectively
For more generic linguistic feature processing and analysis, it would be relevant to design a tool and feature selection assistant for the batch feature extraction tool. This would enable researchers to select features and tools as needed very much like CTAP (Chen and Meurers, 2016). The latter tool supports a modular approach to feature extraction allowing for reusability. Users can compose their dataset variables prior to running an NLP pipeline that processes texts to produced the desired variable values. As the authors mention, additional functionality including machine learning modules is required to combine the collected evidence with specific outcome variables such as CEFR levels. An interface between CTAP and our Web demo UI could be developed in order to allow data exchange between the CTAP output and our CEFR classifier. Another advantage of a feature selection assistant would be to support multilingual processing. The current pipeline makes use of a number of tools that are language agnostic for the computation of some of the metrics. By allowing researchers to select language-agnostic metrics, it would be possible to build data sets used for modeling CEFR classification in other languages than English. Conversely, some metrics are language-dependent, as in the case of many lexical sophistication metrics which are based on lexical frequency inventories extracted from reference corpora of English. One line of research is to adapt some metrics to French as a Foreign Language, especially readability metrics (François, 2015) or to Dutch (Tack et al., 2018). Developing the interoperability of our feature extracting tool in the sense of multilingualism is also made possible by adapting our microsystem features to other languages, probably French as a Foreign Language for the next phase.

6.2.2. Pre-processing

One of the reviewers enquired about the implications of spoken data for our system. The short answer is that some written-based metrics may not be adequate for spoken data but speech data could be pre-processed to be fully tested by our system. The team discussed implementing a speech-to-text system, with the proviso that a single acoustic model should be chosen (eg preferably French speakers with available data). (Mariko and Kondo, 2019) reported successful use of IBM Watson Speech-to-text technology to transcribe learner speech for Japanese learners of English and give examples of the output. They reported Word Error Rates on 50 randomised speech samples and concluded that the automatic procedure was worth it. An important caveat for the calculation of the metrics is the absence of punctuation marks and the potentially useful insertion of "%HESITATION" for filled pauses (no threshold reported for the duration of filled pauses, though). The Watson system runs on Python 3.4. but is in the cloud and is not free. They do not seem to indicate whether the quality of the voice recognition improves over time for their longitudinal data. Following a uniformly python pipeline, we would try to use SpeechRecognition (Zhang, 2017) as a pre-processing stage of spoken data. We would have the added benefit of analysing spoken production, but this would probably imply a semi-automatic solution as the speech-to-text outputs would probably need to be manually edited. We also have initial reservations as to the applicability of written-designed metrics to spoken data (Ballier and Gaillat, 2016), in particular the transferability of the T-unit (a crucial concept for some complexity metrics) for spoken utterances, but we could experiment a speech-to-text module to pre-process learner recordings in order to test our model on spoken data, at least for fluency.

6.2.3. Post-processing

Collecting metrics to assess learner performance could be used for didactic purposes. It sounds plausible to select some features to guide learners in ICALL systems for self-assessment of their performance. A member of our team has begun applying some of the features to produce immediate feedback for learners, elaborating on a prototype (Ballier et al., 2019). This data visualisation application of our feature extractor takes the form of a dashboard where learner scores are compared to means of students of the same cohort and to native scores on similar essays.

6.3. Sustainability and the FAIR Paradigm

We have reached the final stage of our one-year project, and have tried to work in line with the FAIR paradigm (Mons, 2018) for our resources to be:

- **Findable** : the tools and resources are available from the project's web page, possibly from the LREC resource map and our datasets are to be linked with permanent handles and DOIs thanks to the Nakala HUMA-NUM services.
- **Accessible**: Some copyright restrictions apply to the corpus we used and to some of the tools. Access is either universal or restricted. There are copyright restrictions to the TAALES tool and to the initial EFCAMDAT corpus data.
- **Interoperable**: our project is multi-platform and our UI infrastructure could be compared in terms of interoperability with similar existing language platforms. A project is in the making with the curators of the REALEC corpus (Vinogradova, 2016) and of the REALEC-inspector web interface [16] to analyse the relevance of the automatically extracted features for Russian learners of English.
- **Re-usable** : The UI interface can be customised for on-the-fly processing of texts. For example, to improve comparability with other language platforms, a three-point scale of learner levels could be reimplemented on our system with a different statistical model (.sav file). Maybe our datasets will be reused as well. Data Management Plans were not required for this level of funding, but we tried to produce comparable information for our datasets. Following the DMP Template of the EU Recommended practises (European Research Council, 2017) and few examples or guidelines for EU-funded research projects (Reymonet et al., 2018), we documented the Dataset reference and name, Data set description, Standards and metadata, Data Sharing, Archiving and preservation for our two datasets.

7. Conclusion

This paper has presented two tools and a set of of resources implemented in a Language Technology project. These tools rely on a modular implementation of a Docker architecture. As a result, this architecture is reusable in other LT contexts such as L1 identification or Text Classification. We provide a web-based user demo tool and a linguistic complexity metric extraction tool. These tools can be modified to accommodate other projects relying on text features and classification. Our idea was to showcase the full workflow from the linguistic modelling to the web-based user interface to help linguists to disseminate their research projects. In this sense, this paper was intended as a 'how-to' for corpus linguistics to possibly publish web-based interfaces exploiting their data modelling.

Our project is a case study for linguistics as a cumulative data science. We showcase the data life cycle and some of its uses. We were able to reuse part of the EFCAMDAT data collection, we were able to concatenate several existing tools in a single workflow, we added our own micro-system features based on our analysis of learners' issues (the LS2CA_MS component of our pipeline) and our modelling (the .sav file), we shared the demoIT infrastructure we designed to exploit it. More data production can be expected with the UI and the feature extraction tool. Customisation is expected for our demoIT UI infrastructure and micro-system features. Our collaboration between corpus linguists, computational linguists, statisticians and computational scientists pertake of the current shift towards linguistic data science.

8. Acknowledgements

This work was undertaken with a PHC Hubert Currien Ulysse 2019 grant (project ref. 43121RJ) with the financial support of:

- Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund.
- The French Ministry of Higher Education, Research and Innovation (Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation, MESRI)
- The French Ministry for Europe and Foreign Affairs (Ministères de l'Europe et des affaires étrangères, MEAE)

We wish to thank the three anonymous reviewers for their comments. We are grateful to Kristopher Kyle and Scott Crossley for providing us with the source code of the TAALES tool and to Xiaofei Lu for showing us his L2SCA as visiting scholar and allowing us to implement the micro-system features. We thank Theodora Alexopoulou and Carlos Balhana for their willingness to help us in our project by hosting our resulting dataset on the EFCAMDAT website, in accordance with the corpus maintenance regulations. We thank Detmar Meurers for his enthusiasm and for letting us access an early stage version of the CTAP annotation platform. We also thank Felipe Arruda who provided us with substantial software development advice. We also thank the audience of the Paris workshop (30 Oct.2019) for their feedback on the presentation of the deliverables of the project.

[16]https://linghub.ru/inspector/

9. Bibliographical References

Ballier, N. and Gaillat, T. (2016). Classification d'apprenants francophones de l'anglais sur la base des métriques de complexité lexicale et syntaxique. In *JEP-TALN-RECITAL 2016*, volume 9, pages 1–14.

Ballier, N., Gaillat, T., and Pacquetet, E. (2019). Prototype de feedback visuel des productions écrites d'apprenants francophones de l'anglais sous moodle. In Julien Broisin, et al., editors, *Actes de la 9ème Conférence sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH2019)*, pages 395–398.

Ballier, N., Canu, S., Petitjean, C., Gasso, G., Balhana, C., Alexopoulou, T., and Gaillat, T. (2020). Machine learning for learner English. *International Journal of Learner Corpus Research*, 6(1):72–103.

Chen, X. and Meurers, D. (2016). CTAP: A web-based tool supporting automatic complexity analysis. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, pages 113–119.

Crossley, S. A., Kyle, K., Allen, L. K., Guo, L., and McNamara, D. S. (2014). Linguistic Microfeatures to Predict L2 Writing Proficiency: A Case Study in Automated Writing Evaluation. *The Journal of Writing Assessment*, 7(1):1–34.

Crossley, S. A., Kyle, K., and Dascalu, M. (2019). The Tool for the Automatic Analysis of Cohesion 2.0: Integrating semantic similarity and text overlap. *Behavior Research Methods*, 51(1):14–27.

European Council. (2001). *Common European Framework of Reference for Languages: Learning, teaching, assessment*. Cambridge University Press, Cambridge.

European Research Council. (2017). *Guidelines on the implementation of Open Access to scientific publications and research data in projects supported by the European Research Council under Horizon 2020*.

François, T. (2015). When readability meets computational linguistics: a new paradigm in readability. *Revue française de linguistique appliquée*, 20(2):79–97.

Gaillat, T., Simpkin, A., Ballier, N., Stearns, B., Sousa, A., Bouyé, M., and Zarrouk, M. (submitted). Predicting CEFR levels in learners of English: the use of microsystem criterial features in a machine learning approach. *Journal With Anonymous Submission*.

Geertzen, J., Alexopoulou, T., and Korhonen, A. (2013). Automatic Linguistic Annotation of Large Scale L2 Databases: The EF-Cambridge Open Language Database (EFCamDat). In R. T. Miller, et al., editors, *Proceeedings of the 31st Second Language Research Forum*, Carnegie Mellon. Cascadilla Press.

Hawkins, J. A. and Filipović, L. (2012). *Criterial Features in L2 English: Specifying the Reference Levels of the Common European Framework*. Cambridge University Press, United Kingdom.

Huang, Y., Murakami, A., Alexopoulou, T., and Korhonen, A.-L. (2018). Dependency parsing of learner English.

Kelly, R. (2016). Pyenchant a spellchecking library for python. *available: https://pythonhosted. org/pyenchant*.

Klavan, J., Tavast, A., and Kelli, A. (2012). The legal aspects of using data from linguistic experiments for creating language resources. In Arvi Tavast, et al., editors, *Human Language Technologies The Baltic Perspective: Proceedings of the Fifth International Conference Baltic HLT 2012*, pages 71–78. IOS Press.

Kyle, K., Crossley, S., and Berger, C. (2018). The tool for the automatic analysis of lexical sophistication (TAALES): version 2.0. *Behavior Research Methods*, 50(3):1030–1046.

Levy, R. and Andrew, G. (2006). Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of LREC2006*, pages 2231–2234.

Lu, X. (2010). Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15(4):474–496.

Lu, X. (2012). The Relationship of Lexical Richness to the Quality of ESL Learners' Oral Narratives. *The Modern Language Journal*, 96(2):190–208.

Lu, X. (2014). *Computational Methods for Corpus Annotation and Analysis*. Springer, Dordrecht.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Mariko, A. and Kondo, Y. (2019). Constructing a longitudinal learner corpus to track l2 spoken English. *Journal of Modern Languages*, 29:23–44.

McNamara, D. S., Graesser, A. C., McCarthy, P. M., and Cai, Z. (2014). *Automated Evaluation of Text and Discourse with Coh-Metrix*. Cambridge University Press, USA.

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2.

Mons, B. (2018). *Data stewardship for open science: Implementing FAIR principles*. Chapman and Hall/CRC.

Page, E. B. (1968). The Use of the Computer in Analyzing Student Essays. *International Review of Education / Internationale Zeitschrift für Erziehungswissenschaft / Revue Internationale de l'Education*, 14(2):210–225.

Reymonet, N., Moysan, M., Cartier, A., and Délémontez, R. (2018). Réaliser un plan de gestion de données "fair" : modèle 2018.

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 14–16, Manchester: UK.

Tack, A., François, T., Roekhaut, S., and Fairon, C. (2017). Human and Automated CEFR-based Grading of Short Answers. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 169–179, Copenhagen, Denmark, September. Association for Computational Linguistics.

Tack, A., François, T., Desmet, P., and Fairon, C. (2018). NT2Lex: A CEFR-graded lexical resource for Dutch as a foreign language linked to open Dutch WordNet. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 137–146, New Orleans, Louisiana, June. Association for Computational Linguistics.

Vinogradova, O. (2016). The role and applications of expert error annotation in a corpus of English learner texts. *Computational Linguisitics and Intellectual Technologies. Proceedings of Dialog 2016*, 15:740–751.

Volodina, E., Pilán, I., and Alfter, D. (2016). Classification of Swedish learner essays by CEFR levels. In Salomi Papadima-Sophocleous, et al., editors, *CALL communities and culture ; short papers from EUROCALL 2016*, pages 456–461. Research-publishing.net.

Zhang, A. (2017). Speech recognition (version 3.8) python library. *Available from https://pypi.org/project/SpeechRecognition/*.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320.