LinTO Platform: A Smart Open Voice Assistant for Business Environments

Ilyes Rebai, Kate Thompson, Sami Benhamiche, Zied Sellami, Damien Laine, Jean-Pierre Lorre
Linagora R&D Labs
75 Route de Revel, 31400 Toulouse, France
{irebai, cthompson, sbenhamiche, zsellami, dlaine, jplorre}@linagora.com

Abstract

In this paper, we present LinTO, an intelligent voice platform and smart room assistant for improving efficiency and productivity in business. Our objective is to build a Spoken Language Understanding system that maintains high performance in both Automatic Speech Recognition (ASR) and Natural Language Processing while being portable and scalable. In this paper we describe the LinTO architecture and our approach to ASR engine training which takes advantage of recent advances in deep learning while guaranteeing high-performance real-time processing. Unlike the existing solutions, the LinTO platform is open source for commercial and non-commercial use.

Keywords: Smart room assistant, Spoken Language Understanding, Speech recognition, Accuracy and real-time performance

1. Introduction

Speech processing is an active research topic in the signal processing community. There has been a growing interest, both commercial and academic, in creating intelligent machines that interact with humans by voice, suggesting the capability of not only recognizing what is said, but also of extracting the meaning from the spoken language (TURING, 1950).

Advances in machine learning, Automatic Speech Recognition (ASR), and Natural Language Processing (NLP) have lead to dramatic advances in Spoken Language Understanding (SLU) during the last two decades, evolving from relatively simple tasks such as spoken keyword extraction to much more complicated ones involving language understanding in more extended natural language interactions (Serdyuk et al., 2018; Coucke et al., 2018). These achievements have unlocked many practical voice applications, e.g. voice assistants, which are now used in many contexts, including autonomous vehicles (Pfleging et al., 2012), and smart homes[1] (Coucke et al., 2018). Popular commercial solutions for voice assistance include Cortana-Microsoft[2], DialogFlow-Google[3], Watson-IBM[4] or Alexa-Amazon.

SLU is an active research and development field at the intersection of ASR and NLP that focuses on the task of extracting meaning from spoken utterances. Unlike speech recognition, SLU is not a single technology but a combination of technologies: it is a system that requires each of its interdependent components to perform well with respect to speech, speech recognition errors, various characteristics of uttered sentences, and speaker intent detection. This is significantly more difficult to achieve than written language understanding (Tur and De Mori, 2011).

Our objective in this paper is to introduce the LinTO Voice Platform designed for business environments, a competitive solution for voice assistance. Unlike the previous solutions, all the components of our platform are open source[5] for commercial and non-commercial use. All models and components are made available for download for use on our platform.

We outline the design of a SLU system that achieves high performance in terms of response time and accuracy with cloud computing based solutions. This is done by optimizing the trade-off between accuracy and computational efficiency of the two main ASR components: the acoustic model and the language model. While the acoustic model component must be trained on a large corpus using sophisticated deep learning techniques requiring a lot of computational resources in order to achieve high performance, the language model component can be trained on the linguistic domain of the application assistant rather than the domain of the entire spoken language. The total computational cost of the ASR is thus reduced by reducing the cost of the language model while even improving its in-domain accuracy.

2. LinTO Platform

LinTO is an open-source client-server system that enables the conception, deployment and maintenance of software and hardware clients with a voice-activated user interface. The system boosts the ease of use and productivity in both administrative management and customer application contexts, offering hands-free vocal control processes, multiple-speaker speech recognition, and voice access for customer applications.

The LinTO platform features a friendly user console, the LinTO admin, used to design, build and manage specific voice assistants. Each assistant is composed of the models and resources necessary to execute its desired functions, and may include any number of *skills*, or intention-action pairs defined for a particular process, e.g. delivering a verbal weather report if asked about the weather (see section 4.). These are represented in the LinTO admin as an easily manipulable workflow (see Figure 1).

The workflow defines and runs the models of the corresponding SLU pipeline (Figure 2) which is composed of

[1] https://demo.home-assistant.io/

[2] https://www.microsoft.com/en-us/cortana

[3] https://cloud.google.com/dialogflow

[4] https://www.ibm.com/watson

[5] https://doc.linto.ai/#/repos

Proceedings of the 1st International Workshop on Language Technology Platforms (IWLTP 2020), pages 89–95
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

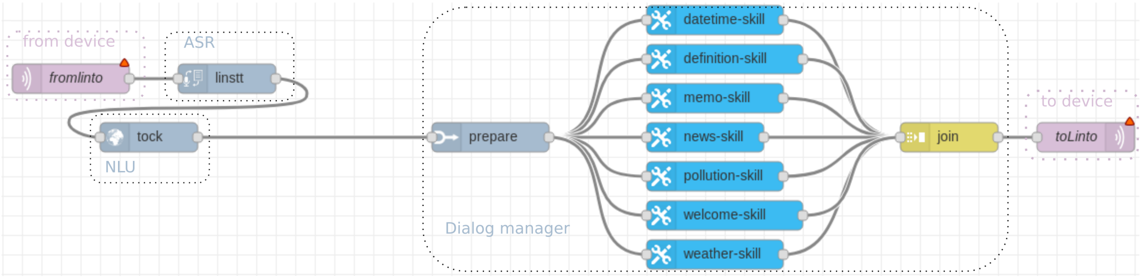


Figure 1: A LinTO Admin workflow describing a voice assistant that incorporates several skills

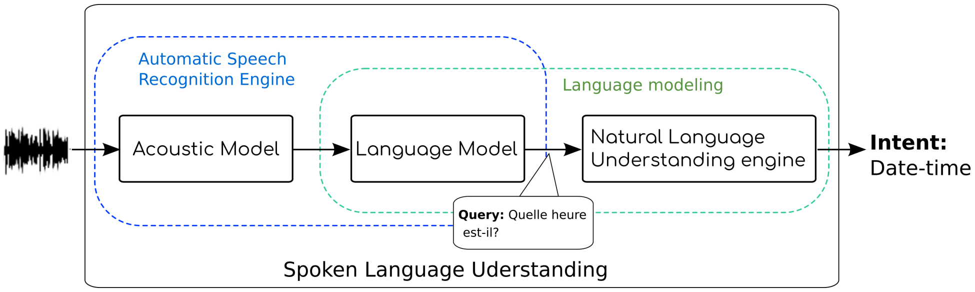


Figure 2: The LinTO SLU architecture

the ASR and NLU engines. The ASR engine recognizes spoken language and transcribes it into text by first mapping the audio stream into a sequence of probabilities over acoustic units (or phones) using an acoustic model (AM), and then converting the outputs of the AM to text using a language model (LM). The NLU engine interprets the language. Specifically, it goes beyond speech recognition to determine a user's intent from her decoded spoken language input (e.g. a query about the weather forecast). Once the user query is interpreted, the final step is to compute a response using a dialogue manager. The response can take multiple forms depending on the request, e.g., an audio response if asked about the weather or an action on a connected device if asked to play a song.

In accordance with our performance, portablity and scalability objectives, we developed an SLU system using cloud-based technologies and offering a service that meets three basic requirements:

1. Can be deployed in any system
2. Can handle a high number of queries
3. Responds in real time with high accuracy

To address the first requirement, we take advantage of Docker technology in order to create "containerized" services that can be used in any OS environment. For the scalability requirement, we implement Docker Swarm, a container orchestration tool that helps manage multiple containers deployed across multiple host machines, such that the service can be scaled up or down depending on the number of queries (see Figure 3). Finally, in order to provide an accurate, real-time response, we design the SLU components to optimize the trade-off between accuracy and computational efficiency. For instance, since the size of the AM architecture has an impact on the computational cost and the accuracy, we determine the final model size by taking into account target resources and the desired accuracy. Similarly for the LM and the NLU components, the models are trained to reduce size and increase in-domain accuracy by restricting the vocabulary as well as the variety of the queries they should model.

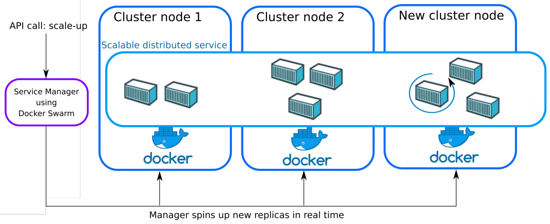

API call: scale-up
Service Manager using Docker Swarm
Cluster node 1
Scalable distributed service
Cluster node 2
New cluster node
docker
docker
docker
Manager spins up new replicas in real time

Figure 3: Service manager using Docker Swarm for container orchestration

The LinTO platform and all its services and components are released publicly[6], at no cost for commercial and non-commercial use. The language currently supported by our platform is French.

3. The Acoustic Model

The AM is the first component of ASR engine and first step of the SLU pipeline, and is therefore crucial to the functioning of the entire system. It is responsible for converting raw audio data to a sequence of acoustic units, or phones. It is

[6] https://doc.linto.ai/

usually trained on large speech corpus using deep networks in order to model context variations, speaker variations, and environmental variations. Given that these variations are the main factors impacting the accuracy of a speech recognition system, robust modeling of them is imperative to successful speech-to-text transcription.

There are two major research directions in acoustic modeling. The first one focuses on collecting a large, in-domain speech corpus adapted to the context of application of the assistant in order to build a robust model. The second one focuses on improving machine learning techniques to better model the long temporal speech contexts. In the next subsections, we will address both directions, first detailing our methods for collecting, processing and validating training data, and then describing the acoustic model architecture.

3.1. Data preparation

A large amount of transcribed data is needed in order to train the deep networks used for speech recognition models, including AMs. A robust AM requires several thousand hours of audio training data with corresponding transcripts. This data must be formatted to include a set of audio extracts and matching transcripts which are cut into segments of lengths suitable for acoustic training (optimally a few tenths a of second).

In recent years there has been a rapid increase in the amount of open source multimedia content on the internet, making collecting a large speech corpus more feasible. We have created a French corpus using various publicly available sources, including those made available by the LibriVox project[7], a community of volunteers from all over the world who record public domain texts. We reformat this data by segmenting and realigning the transcripts, and we also remove any transcription errors. Our corpus preparation process is presented in Figure 4 and detailed in the following subsections.

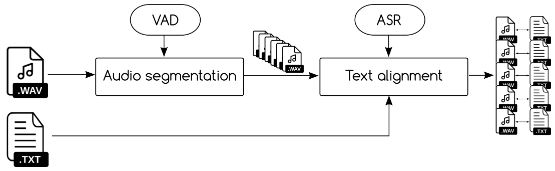


Figure 4: Audio alignment procedure used in the creation of the corpus

3.1.1. Audio segmentation

We segment the audio recordings using Vocal Activity Detection (VAD) technique, which detects the presence or absence of human speech. We use the detected non-speech intervals to split the audio at intervals of 0.5 seconds or less. After segmentation, we must re-align the audio snippets with their transcriptions.

3.1.2. Text alignment

To match the audio segments with the original text, we use a pre-trained acoustic model with a language model biased to the specific vocabulary in order to transcribe each audio segment. The proposed procedure consists of two stages.

During the first stage, we normalize the original text in order to adapt it for language model generation. Additionally, the obtained text is extended by a set of extra-words including numbers, ordinal numbers, most common abbreviations, and punctuation signs in order to anticipate annotation errors. Next, both the normalized original text and the audio segments are split into *n* parts. The text in each part is then converted into a decoding graph. This is done to avoid automatic transcription errors by increasing the likelihood that the audio will be matched to some portion of the text. Finally, we transcribe each audio segment using the pre-trained acoustic model along with the decoding graph that matches the audio.

In order to analyze the quality of the transcription, we develop an alignment algorithm to get the best match between the transcriptions and the normalized original text. We are inspired by Levenshtein alignment algorithm (Levenshtein, 1966). Once the alignment is obtained, the word error rate (WER), a common metric used to compare the accuracy of the transcripts produced by speech recognition, is computed between the reference and the hypothesis. Possible sources of text-audio mismatch include inaccuracies in ASR prediction, as well as in the original text (e.g., normalization errors, annotation errors). At this stage, it is possible to retain the segments that are below a low WER threshold and discard the others.

We then move to the second stage, where we further improve the audio transcription using a custom-generated decoding graph for each audio segment. This is done as follows. First, the decoding graph is formed from a combination of three original text segments. Our aim is to have a decoding graph with a high bias in order to overcome the ASR errors, and to increase the accuracy of the transcription with respect to the first stage. Then, we always use the pre-trained acoustic model with the new decoding graphs to decode the audio segments. Figure 5 shows different examples of errors and the results generated during alignment stage 1 and 2. At the end of this stage, we retain only the segments with a WER of 0.

3.1.3. AM training data

We have collected about 500 hours of read French speech from various freely available sources, including: Commonvoice[8], M-AILabs[9] and librivox. The alignment process produces a set of aligned audio of size approximately 200 hours performed mainly on librivox data. For the first and second decoding pass, we use a pre-trained triphone deep neural network-based AM.

3.2. Acoustic model architecture

Hybrid acoustic models are widely adopted in the current state-of-the-art systems combining a Hidden Markov Model (HMM) with a Deep Neural Network (DNN). The HMM describes temporal variability while the DNN computes emission probabilities from HMM states in order to

[7] https://librivox.org/

[8] https://voice.mozilla.org/

[9] https://www.caito.de/2019/01/the-m-ailabs-speech-dataset/

a- **Reference:** aux artistes ou aux savants dont l' appui aide à percer dans la branche où ils priment
Hypothesis (Stage 1): aux artistes ou aux savants dont l' appui aide à **entre guillemets** percer dans la branche où ils priment
Hypothesis (Stage 2): aux artistes ou aux savants dont l' appui aide à **entre guillemets** percer dans la branche où ils priment

b- **Reference:** pour lesquels ils n' admettent pas la critique qu' ils acceptent aisément **s' il s' agit** de leurs chefs d' oeuvre
Hypothesis (Stage 1): pour lesquels ils n' admettent pas la critique qu' ils acceptent aisément **s' agit** de leurs chefs d' oeuvre
Hypothesis (Stage 2): pour lesquels ils n' admettent pas la critique qu' ils acceptent aisément **s' il s' agit** de leurs chefs d' oeuvre

c- **Reference:** lui préfèreront même des adversaires comme **mm** ribot et deschanel
Hypothesis (Stage 1): lui préfèreront même des adversaires comme **messieurs** ribot et deschanel
Hypothesis (Stage 2): lui préfèreront même des adversaires comme **messieurs** ribot et deschanel

Figure 5: Examples of the audio transcription obtained in the first and second stage. a- Errors in text normalization overcome in the first and second stage. b- Words deleted in the first stage but correctly recognized in the second one. c- 'messieurs' expanded abbreviation in the original text perfectly transcribed by ASR.

model and map acoustic features to phones (Hinton et al., 2012).

Over the last few years, various deep learning techniques have been proposed to improve the emission probabilities estimation (Graves et al., 2013). In fact, modeling long term temporal dependencies is critical in acoustic modeling: by modeling the temporal dynamics in speech, the acoustic model can effectively capture the dependencies between acoustic events and thus improve speech recognition performance (Peddinti et al., 2015). One of the most popular techniques is the Time-Delay Neural Network (TDNN) (Waibel et al., 1989) which has been shown to be effective in modeling long range temporal dependencies (Peddinti et al., 2015).

Our AM is designed to guarantee high transcription accuracy while requiring fewer computational resources. We use a neural network that maps sequences of speech frames to sequences of triphone HMM state probabilities. The speech frames are obtained by computing the mel-frequency cepstral coefficients (MFCC) from the audio signal. The neural network combines a time-delay layers and an attention mechanism that selects the temporal locations over the input frame sequence where the most relevant information is concentrated (Bahdanau et al., 2016). The selection of elements from the input sequence is a weighted sum:

$$c_t = \sum_l \alpha_{tl} \mathbf{h}_l \qquad (1)$$

where α_{tl} are the attention weights. This mechanism helps to improve the acoustic modeling of the network and to subsequently improve speech recognition performance (Bahdanau et al., 2016; Lu, 2019).

In the literature, the acoustic model achieving human parity (Xiong et al., 2016) is a complex neural network, composed of several hundred of million parameters. The size of these models along with the language model, increase the computational resources necessary not only to perform real time decoding, but also to scale them. Models with a variable number of parameters, i.e. different number of layers and neurons, can be trained and evaluated in terms of accuracy and computational cost. This can help to select the most appropriate model that optimizes a trade-off between accuracy and computation time.

We train deep neural AMs using the Kaldi toolkit [10]. Our typical architectures have 7 time-delay layers and an attention layer. The input vector to the model consists of 40 MFCC features, and 100 speaker and channel features computed using an i-vector speaker-adaptive model (Garimella et al., 2015). It is trained with the lattice-free MMI criterion (Povey et al., 2016), using gradient descent with start and final learning rates of 0.001 and 0.0001 respectively.

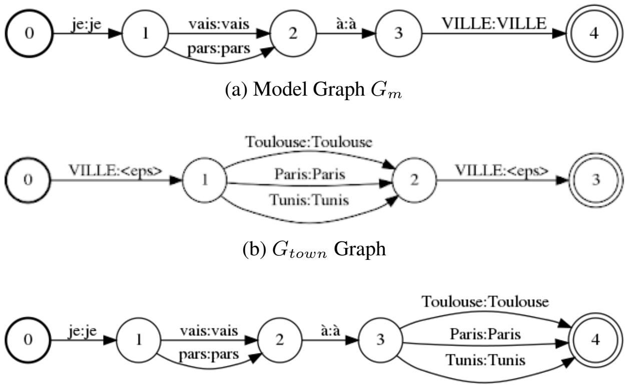
4. The Language Model

The language model is the second component of the ASR engine and of the SLU pipeline. Like the acoustic model, it is designed to optimize a trade-off between transcription, precision and computational cost.

The LM converts the sequence of probabilities over acoustic units predicted by the AM into a probable word sequence, while taking into account the probability of word co-occurrence (Chelba et al., 2012b). Once the text is obtained, the NLU module extracts the intent of the user from the decoded query. The LM and NLU have to be mutually consistent in order to optimize the accuracy of the SLU engine, and together they make up the language modeling component.

A large vocabulary LM consisting of several million n-grams (Chelba et al., 2012a) can adequately capture the linguistic knowledge of a target language (e.g. syntax and semantics) in order to more accurately transcribe a sequence of acoustic units. But this results in a large search space that greatly increases decoding time for the LM, and effects ASR performance as a whole. To avoid such limitations to

[10] https://github.com/kaldi-asr/kaldi



(a) Model Graph G_m

(b) G_{town} Graph

(c) $G = ReplaceFST(G_m, G_{town})$

Figure 6: Graph G creation process with a simple query of the form “je vais à #VILLE” (in English “I am going to #TOWN”

the performance of our SLU, we restrict the vocabulary of the LM to a particular use case domain.

This domain is defined by the set of skills the assistant is designed to handle (e.g. SmartRoomEquipment, MeetingReport, MeetingControl for a smart business assistant, or SmartLights, SmartMusic skills for smart home assistant). Each skill contains one or multiple intents (i.e. user-intentions[11]), that furnish a concentrated vocabulary which is sufficient to correctly model the sentences encountered by the assistant. While the AM is trained once per language, this specialized LM can be trained quickly on the reduced vocabulary, which is sufficient to with improvements to the overall performance of the SLU.

4.1. LM training data

The data used to train the specialized LM (as well as the NLU) consists of sentences expressing the desired intents. These are the queries listed manually by the user to which one or more entities can be bound. For example, the skill MeetingReport contains the intent ShowMeetingReport, to which the entities *date* and *report-type* are bound, giving “show the meeting *summary* of the date *21 April 2019*” to the dialogue manager which then executes the correct action.

Entities are predefined and provided by the platform – e.g. numbers, colors, date and times, etc, and can be bound to different intents. Using the same example, the values associated with the date and report-type entities in the query can be specified as follows: “show the meeting (summary)[report-type] of the date (21 April 2019)[date]”. The entity report-type is a custom list of values (e.g. full transcription, summary, notes).

4.2. Model generation

The map from the output of the acoustic model to the most likely sequence of words is carried out using a Viterbi search in a decoding graph which is a weighted finite state transducer (wFST) (Mohri et al., 2002). This graph is the composition of four wFST graphs: H contains the HMM definitions, C represents the context-dependency, L is the lexicon, and G is a bigram or a trigram language model. For more details refer to (Mohri et al., 2002) and references therein. In this paper, we focus on the construction of the L graph and G graph since they are the most important parts of the decoding graph that must be adapted to the domain-specific data.

The lexicon (i.e. dictionary) consists of a set of words with their phonetic pronunciations. This pair (word/phonemes) allows us to associate a sequence of words with the predicted phonetic sequence provided by the acoustic model. In this work we use an open-source lexicon (French Prosodylab dictionary[12]), composed of more than 100,000 words with their SAMPA phonetization (McAuliffe et al., 2017). Since this lexicon may not include all possible words, the new words in the data used to train the G graph are first automatically phoneticized and then added to the lexicon in order to allow the decoding graph to correctly predict these words. The word’s phonetization is performed using a Grapheme-to-Phoneme (G2P) model trained using phonetisaurus toolkit[13]. The obtained lexicon is then converted into L graph using a Kaldi function.

After preparing the L graph, the first step in building G is the preparation of the set of queries that the user may make to the assistant. Given the data described in section 4.1., the values of each entities in the queries are replaced by an identifier of the entity. For example, the query “show the meeting (summary)[report-type] of the date (21 April 2019)[date]” is mapped to “show the meeting *#report-type* of the date *#date*”. Next, an n-gram model is trained on the resulting queries and then converted to a wFST, which we called the main graph G_m.

In order to bind the entities to the main graph, each entity e is converted into an acceptor graph G_e that encodes the list of values of that entity. Next, these graphs G_e are merged with the main graph G_m to obtain the final graph G. This process is illustrated by a sample query in Figure 6.

[11] Overall, a user intent spots what a user is looking for when conducting a search query

[12] https://montreal-forced-aligner.readthedocs.io/en/latest/pre-trained_models.html

[13] https://github.com/AdolfVonKleist/Phonetisaurus

5. Performance Evaluation

The aim of this work is to develop an SLU system that achieves high performance in terms of response time and accuracy. This system must correctly predict the intent and the entities of a spoken utterance in order to execute the correct action. In this paper, our focus is to improve the performance of the ASR engine which has a strong impact on the SLU performance.

In this section, we present the different experiments carried out under different conditions. The objective is to evaluate the ASR engine and in particular the LM of a smart business assistant. While the response time is used to analyse the speed of the decoder, the accuracy of the transcription is usually measured using the word error rate (WER). The experiments are conducted on French language. Thirteen queries related to the context of the meeting are chosen for evaluation. Examples are:

- "Baisse la lumière de (10)[number] pourcent" (Turn down the light 10 percent)
- "Active le chronomètre pour (4 heures)[time] de réunion" (Activate the chronometer for 4 hours)
- "Affiche le compte-rendu de la réunion du (12 septembre 2018)[date] concernant le (résumé automatique)[subject]" (Post the meeting minutes from the September 12 meeting about automatic summarization.
- "Invite (Sonia)[person] à la visioconférence de (demain)[date]" (Invite Sonia to the video-conference tomorrow)

These queries are recorded in order to build an evaluation corpus. For this purpose, we record the queries with 16 speakers (14 native and 2 non-native, including 10 men and 6 women between 21 and 30 years old). The recording are carried out with the YETI – Blue Microphone in a meeting room, with a sampling frequency of 16 Khz and a speaker/microphone distance of 80 cm.

We varied the number of the queries (intents) on which the LM is trained to evaluate the quality of recognition based on the number of skills. Four models are built, defined as follows:

- First Model: using only the thirteen evaluation queries
- Second Model: using 76 queries which represents the following 7 skills:
  - control-equipment: manage the meeting room equipment (contains 8 queries);
  - time: gives the time (contains 8 queries);
  - meeting_time: gives the meeting time (contains 11 queries);
  - chrono: adjust the stopwatch of the meeting (9 queries);
  - meeting-report: which gives the meeting report (5 queries);
  - meeting-participant: manage the meeting participants (5 queries);
  - videoconf: manage the video conference (17 queries).
- Third Model: We use 171 queries in this model which represent 14 skills: weather, news, mail, note, meeting-control (recording, meeting mode), traffic, control-Linto, control-equipment, time, meeting-time, chrono, meeting-report, meeting-participant, videoconf.
- Fourth Model: In this model, we use 237 queries which represents 22 skills.

In order to evaluate the impact of the adapted language model over the large vocabulary model on the response time, we use a large vocabulary model trained on the text of the speech corpus.

Results

The results of the evaluations in terms of WER, SER (Sentence/Command Error Rate) as well as the response time are presented in the following table.

Table 1: Performance evaluation in terms of WER for the different language models.

	WER[%]	SER[%]	Time[s]
Large vocabulary model	25.75	70.67	237
Adapted language model (1)	4.28	20.67	195
Adapted language model (2)	4.62	22.12	199
Adapted language model (3)	5.23	25.00	204
Adapted language model (4)	5.37	25.48	216

The first evaluation consists of comparing the results obtained using a conversational recognition mode (using the large vocabulary model) on the one hand, and an adapted language model on the other. The objective of this evaluation is to highlight the performance and response time of the control systems.

As shown in Table 1, the WER results of the control systems are better than those obtained by the large vocabulary system with a gain of 20%. It can be concluded that the models adapted to the assistant are more advantageous. Additionally, the response time of these models is shorter than that of the conversational system thanks to the relatively small size of the language models in terms of vocabulary (words).

The second series of evaluations allows us to analyze the consequence of the complexity of the language model in terms of the number of commands trained on the recognition performance.

Table 1 shows a slight loss in WER (1%) obtained by switching from a 13-queries model (Model 1) to a 237-queries model (Model 4). On the other hand, in terms of response time, we see a loss of 21s / 208 evaluated queries from the M4 model compared to the M1 model.

6. Conclusion

In this paper, we presented LinTO, an open-source voice platform for business environments. It is a spoken language understanding system that achieves high performance in terms of accuracy and response time, is portable and scalable, and can be tailored to meet the needs of specific business contexts in the form of skills. We described the techniques and adaptations applied to the system's acoustic model and language model: for the acoustic model, we took advantage of recent advances in machine learning while optimizing computational cost; for the language model we trained the adapted automatic speech recognition component to correctly model only the sentences that are found in the domain supported by the SLU, which is implemented in a particular context. Overall, these techniques optimize a trade-off between the accuracy and computational cost by biasing and thus reducing the size of the language model.

7. Acknowledgements

This work was carried out as part of the research project assessed by Bpifrance (Public Investment Bank): LinTO - Open-source voice assistant that respects personal data for the company. The latter is financed by the French government as part of the "Programme des Investissements d'Avenir - Grands Défis du Numérique".

8. Bibliographical References

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE.

Chelba, C., Bikel, D., Shugrina, M., Nguyen, P., and Kumar, S. (2012a). Large scale language modeling in automatic speech recognition. *arXiv preprint arXiv:1210.8440*.

Chelba, C., Schalkwyk, J., Harb, B., Parada, C., Allauzen, C., Johnson, L., Riley, M., Xu, P., Jyothi, P., Brants, T., Ha, V., and Neveitt, W. (2012b). Language modeling for automatic speech recognition meets the web: Google search by voice.

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., et al. (2018). Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Garimella, S., Mandal, A., Strom, N., Hoffmeister, B., Matsoukas, S., and Parthasarathi, S. H. K. (2015). Robust i-vector based adaptation of dnn acoustic model for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Lu, L. (2019). A transformer with interleaved self-attention and convolution for hybrid acoustic models. *arXiv preprint arXiv:1910.10352*.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. (2017). Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, pages 498–502.

Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Peddinti, V., Povey, D., and Khudanpur, S. (2015). A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Pfleging, B., Schneegass, S., and Schmidt, A. (2012). Multimodal interaction in the car: combining speech and gestures on the steering wheel. In *Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 155–162.

Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. (2016). Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755.

Serdyuk, D., Wang, Y., Fuegen, C., Kumar, A., Liu, B., and Bengio, Y. (2018). Towards end-to-end spoken language understanding. *CoRR*, abs/1802.08395.

Tur, G. and De Mori, R. (2011). *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

TURING, I. B. A. (1950). Computing machinery and intelligence-am turing. *Mind*, 59(236):433.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., and Zweig, G. (2016). Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*.