Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 31e édition),
Traitement Automatique des Langues Naturelles (TALN, 27e édition),
Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition)
Nancy, France, 08-19 juin 2020

Quel type de systèmes utiliser pour la transcription automatique du français ? Les HMM font de la résistance.

Paul Deléglise[1] Carole Lailler[2]
(1) LIUM, Université du Maine, avenue Olivier Messiaen, 72085 Le Mans Cedex 09, France
(2) Scribe-conseil, 9 avenue Adrien Daurelle, 05100 Briançon, France
paul.deleglise at gmail.com,c.lailler chez scribe-conseil.com

RÉSUMÉ

Forts d'une utilisation couronnée de succès en traduction automatique, les systèmes *end-to-end* dont la sortie réside en une suite de caractères, ont vu leur utilisation étendue à la transcription automatique de la parole. De nombreuses comparaisons ont alors été effectuées sur des corpus anglais libres de droits, de parole lue. Nous proposons ici de réaliser une comparaison entre deux systèmes état de l'art, non pas sur de la parole lue mais bel et bien sur un corpus d'émissions audiovisuelles françaises présentant différents degrés de spontanéité. Le premier est un *end-to-end* et le second est un système hybride (HMM/DNN). L'obtention de résultats satisfaisants pour le *end-to-end* nécessitant un lexique et modèle de langage dédiés, il est intéressant de constater qu'une meilleure intégration dans les systèmes hybrides (HMM/DNN) est source de performances supérieures, notamment en Français où le contexte est primordial pour capturer un énoncé.

ABSTRACT

What system for the automatic transcription of French in audiovisual broadcasts ?

With a successful use in machine translation, Character-based Neural Machine were naturally used in automatic speech transcription. Many comparisons were then made on English free corpora of speech read. We deal here with a comparison between two state-of-the-art systems, not on read speech but rather on a corpus of French audiovisual broadcasts with different degrees of spontaneity. The first is a *end-to-end* and the second is an hybrid system (HMM / DNN). As obtaining good results for the *end-to-end* requires a dedicated vocabulary and a language model, the goal is to see if better integration in hybrid systems (HMM / DNN) is really a source of superior performances, especially in French language where context is essential for syntax.

MOTS-CLÉS : Transcription Automatique de la parole, Évaluation, end-to-end, HMM/DNN.

KEYWORDS: ASR, Evaluation, end-to-end, HMM/DNN.

1 Introduction

Depuis leur apparition dans le domaine de la transcription automatique de la parole en proposant en 2013 une transcription en phonèmes suivie en 2014 par une transcription en mots sans connaissances phonétiques (Graves & Jaitly, 2014), les systèmes *end-to-end* s'imposent comme une alternative sérieuse aux systèmes à base de HMM. (Amodei *et al.*, 2016) rapportent une évaluation sur le test du corpus Librispeech (Panayotov *et al.*, 2015), obtenue en utilisant un corpus d'apprentissage d'une dizaine de milliers d'heures. Les résultats sont notamment comparés avec la performance d'un

locuteur humain. La taille du corpus employé pose cependant un net problème de reproductibilité des expériences. (Zeghidour *et al.*, 2018) font état de meilleurs résultats avec une architecture de réseaux différente sur Librispeech (3,26% partie *clean* , 12,76% partie *other*) en utilisant "seulement" 960 heures qui correspondent à l'apprentissage de ce même corpus. Dans ce même article, les résultats sont comparés avec ceux d'un système hybride (HMM/DNN) (Han *et al.*, 2017) (3,51% partie *clean*, 8,58% partie *other*). Enfin, (Lüscher *et al.*, 2019) présentent deux architectures : l'une hybride (HMM/DNN), l'autre de type *end-to-end* dont les résultats sont respectivement de 5.0% et 9.3% sur la partie *other*. Toutes les modélisations *end-to-end* utilisent un modèle de langage construit sur des données textuelles en appui du DNN appris, lui, sur les corpus oraux. Il ressort que les différences avec un modèle hybride portent sur l'utilisation de connaissances phonétiques préalables comme le dictionnaire de phonétisation et le processus d'alignement signal/symbole : HMM dans un cas, neuronal (convolution, encodeur-décodeur) dans l'autre.

Comme l'ensemble des évaluations de ces systèmes *end-to-end* est présenté en langue anglaise sur des corpus de parole lue, nous proposons d'étudier ces différences sur du français, avec un corpus d'émissions audiovisuelles comprenant une part non négligeable de parole spontanée. Nous pourrons ainsi considérer l'influence de la langue, y compris et surtout dans sa dimension spontanée, sur les résultats. L'article se compose d'une description des systèmes et corpus utilisés. Nous présentons ensuite les processus d'apprentissage, les expériences de transcription et leur évaluation. Enfin, nous analysons les résultats obtenus.

2 Description des systèmes

Avant de décrire les 2 types de systèmes utilisés, nous présentons les 2 composants communs des systèmes construits, à savoir une étape de segmentation automatique préalable au décodage et une modélisation du langage.

La segmentation automatique est obligatoire puisqu'aucun des deux systèmes ne permet un décodage en flux. En outre, il n'est pas question d'utiliser une segmentation manuelle pour l'évaluation car la segmentation automatique influence les résultats. Cette segmentation est réalisée par le système Lium Spkdiarization (Barras *et al.*, 2006; Meignier & Merlin, 2010) sans utiliser la tâche de détection parole/silence/musique, car le décodeur réalise cette tâche de meilleure manière. En outre, le regroupement de locuteurs à l'aide du critère de vraisemblance croisée pour obtenir une homogénéité des classes n'est pas utile dans le décodage. Il est donc omis.

Pour réaliser les modèles de langage, nous avons opté pour une modélisation avec repli, la différence avec les modèles continus étant limitée jusqu'à l'apparition des modèles de langage à *Transformer* introduit dans (Lüscher *et al.*, 2019). Nos modèles de langage sont construits avec l'outil POCOLM [1] qui optimise le poids des différents corpus en fonction de la cible de manière plus pertinente que les autres boîtes à outils.

2.1 Système hybride

Nous avons choisi comme représentant d'un système hybride (HMM/DNN), un système utilisant la boîte à outils KALDI (Povey *et al.*, 2011) dont les performances ont fortement augmenté suite

1. https://github.com/danpovey/pocolm

à l'utilisation de DNN pour le calcul de la probabilité d'une trame (Povey *et al.*, 2016), avec un apprentissage discriminant de type LF-MMI. Un modèle de Markov caché utilisant des mélanges de gaussiennes est construit par étape. L'avant-dernière étape utilise une représentation par un vecteur de 40 paramètres issu d'une LDA/MLLT calculée sur la concaténation des paramètres de 9 trames consécutives où chaque trame est paramétrée par 13 coefficients MFCC auxquels sont ajoutées les dérivées premières et secondes. Dans la dernière étape, une adaptation au locuteur est réalisée par un processus de fMLLR (Gales, 1998). Ce modèle ne servant qu'à fournir, pour l'apprentissage du DNN, un alignement des états des HMM/signal de parole, l'apprentissage discriminant pour le modèle gaussien présent dans la recette KALDI n'est pas effectué.

Pour le DNN, nous avons choisi l'architecture tdnn7n présente dans la recette *Switchboard* de Kaldi. Elle comprend schématiquement 11 couches de TDNN intercalées avec des couches de régularisation de type ReLu pour un total de 20 millions de paramètres et une couche de sortie de taille 11500, qui correspond au nombre d'états partagés de l'étape fMLLR. Les paramètres d'entrée sont des MFFC calculés sur 40 bandes spectrales en gardant tous les coefficients cepstraux d'une part, et un i-vector de dimension 100 calculés sur une fenêtre glissante caractérisant le locuteur d'autre part.

Le décodage après le calcul des i-vectors s'effectue en 3 passes :

1. une première utilise le HMM hybride avec un modèle de langage d'ordre 2 pour calculer un treillis pour chaque segment,
2. puis, une réévaluation de ces derniers par un modèle de langage d'ordre 4 est effectuée,
3. enfin, le calcul de la meilleure séquence de mots en utilisant différentes valeurs pour le poids du LM et la pénalité d'insertion des mots à partir des treillis est opéré.

2.2 Système *End-to-end*

Pour ce système, nous avons opté pour le *wav2letter++* (Pratap *et al.*, 2019) dans sa version convolutionnelle (Zeghidour *et al.*, 2018) avec un critère d'apprentissage de type *AutoSegCriterion*. Les paramètres d'entrée du réseau sont 40 coefficients log-Mels calculés à la volée, solution qui s'est montrée plus efficace que l'utilisation de MFCC sur Librispeech. La sortie du réseau de neurones est une suite de symboles : ici des lettres. Ainsi, une liste de mots est obtenue directement en utilisant un symbole spécial indiquant la fin d'un mot. Pour obtenir de meilleures performances, un élargissement à l'utilisation d'un lexique et d'un modèle de langage est proposé. Ce dernier peut être à base de mots ou de lettres. Cela correspond à la version *conv_glu* de la recette dédiée à Librispeech. Ce réseau comprend 19 couches convolutives intercalées de couches de normalisation, dropout, et Relu. Il a 200 millions de paramètres. Par ailleurs, il faut noter l'absence de toute indication temporelle associée aux mots dans la sortie.

3 Données d'apprentissage

3.1 L'acoustique

Les données acoustiques pour l'apprentissage sont les audios associés à leur transcription fine provenant des différentes campagnes d'évaluation du français. Elles comportent des émissions de radio pour les 2 campagnes ESTER (Gravier *et al.*, 2004) dont la parole est, le plus souvent, préparée, auxquelles sont ajoutées les émissions de télévision de la campagne ETAPE (Gravier *et al.*, 2012)

qui contient davantage de parole spontanée. Des transcriptions larges de podcasts complètent ce corpus d'apprentissage. Les corpus de développement et de test sont ceux de l'évaluation finale du défi REPERE (Giraudel *et al.*, 2012). Le tableau 1 donne le nombre d'heures transcrites des différents corpus. Le corpus d'apprentissage est utilisé tel quel pour l'apprentissage des GMM. Pour l'apprentissage des DNN en revanche, une augmentation de corpus est effectuée. Pour cela, on pratique une modification de la vitesse de 0,9 et de 1,1, conjuguée à une variation aléatoire de l'amplitude par émission : cela multiplie par 3 le nombre d'heures disponibles pour l'apprentissage.

Corpus	Taille en heures	Nb émissions
Apprentissage campagne évaluation	291h	759
Apprentissage podcast	225h	1229
Total Apprentissage	617h	1988
Développement	5h30	28
Test	9h25	62

TABLE 1 – Les corpus audio

3.2 Les données textuelles et les modèles de langage

Nous regroupons dans ce paragraphe les informations sur les données textuelles et les modèles de langage utilisés par les 2 types de systèmes.

3.2.1 Les textes

Les textes proviennent de 6 sources différentes auxquelles il faut ajouter la transcription du corpus de développement qui sert de cible pour l'optimisation des LM. Le vocabulaire a une taille de 160 000 mots. Il est construit sur la réunion des mots présents dans les transcriptions des corpus des campagnes d'évaluation ESTER et des mots les plus fréquents sur les autres corpus. La table 2 donne le nombre d'occurrences de mot par corpus. Les extraits de corpus sont obtenus par une méthode d'entropie croisée entre la source originale et le corpus de développement pour ne garder que les phrases les plus pertinentes.

Corpus	Nb occurrences
Audio transcrites	8M de mots
Sites web de télévision	5M
Extraits (40%) de Google News ($\leq$ 2012)	80M
Extraits (75%) de French Gigaword ($\leq$ 2012)	753M
Extraits (84 %) du journal le monde de 1988 à 2003	316M
Sous titre de journaux télévisés (OCR et télétexte)	11M
Transcription du corpus de développement	5,2k

TABLE 2 – Les textes

3.2.2 Les modèles de langage

Deux modèles sont construits : un modèle 2-gram et un modèle 4-gram. Le modèle 2-G n'est cependant utilisé que par la première passe du modèle hybride. Les tailles de ces modèles sont présentées dans

le tableau 3. Trois modèles de langage sur les séquences de lettres sont également calculés pour le modèle *end-to-end*. Ils sont respectivement d'ordre 2, 7 et 10 et contiennent respectivement 2192, 7M et 26 M n-grams pour des perplexités respectivement de 7.9, 3.2 et 2.87.

modèle	1-gram	2-gram	3-gram	4-gram	perplexité
2G	160 k	1,70 M	-	-	152,39
4G	160 k	24,00 M	140 M	338 M	84,35

TABLE 3 – Modèles de langage

4 Apprentissage de modèles et optimisation des paramètres

4.1 Système hybride

Pour ce dernier, seul un dictionnaire de phonétisation lui est fourni. Il contient l'ensemble des mots issus des transcriptions du corpus d'apprentissage (phase apprentissage) et le vocabulaire des modèles langage (phase décodage). Ce dictionnaire est construit à l'aide de BDLEX, puis de LIAPHON contrôlé par un alignement forcé sur le corpus d'apprentissage. Ensuite, il ne reste plus qu'à lancer le script fourni dans la recette. L'apprentissage du modèle acoustique prend 70 heures sur une machine ayant 24 cœurs et de 2 cartes GPU de modèle GTX Titan X dotées chacune de 12 Go de ram.

L' optimisation des paramètres est effectuée sur le corpus de développement. Elle consiste à choisir le meilleur poids pour le modèle de langage et la pénalité d'insertion. Comme ces poids ne servent que dans le calcul des CTM à partir des treillis, un algorithme purement combinatoire est utilisé sur les 39 combinaisons les plus susceptibles de contenir l'optimal sur cette passe très rapide.

4.2 Système *end-to-end*

L'apprentissage doit commencer par une phase de préparation des données avec la création d'un fichier par segment de parole. En outre, la représentation des mots doit être spécifiée : nous avons tout d'abord choisi de les représenter, classiquement, par la suite des lettres en testant divers regroupements que nous détaillerons dans la partie 5.1.1. Nous avons rajouté 2 mots et 2 symboles spécifiques pour représenter d'une part les silences et d'autre part, les inspirations et autres bruits. Pour lancer l'algorithme sur la machine dont on dispose, nous devons limiter la taille des minibatchs à 4. Par ailleurs, l'algorithme d'apprentissage ne progresse pas si le corpus comprend des segments trop longs. Le plus usuel est de commencer en restreignant le corpus à des segments de petite taille comme dans (Lüscher *et al.*, 2019). Toutefois, cela peut conduire à l'introduction d'un biais dans l'initialisation du modèle. Nous avons donc préféré utiliser les alignements effectués lors de l'apprentissage du système hybride pour redécouper le corpus en segment de moins de 8 secondes, grâce aux silences présents dans le signal. Lors de cet apprentissage, le corpus de développement sert à contrôler le nombre d'itérations. La convergence prend une vingtaine de jours sur la machine décrite dans 4.1. Cette durée importante ne nous a pas permis de tester les métaparamètres de l'apprentissage du modèle.

Une phase d'optimisation est aussi réalisée pour les paramètres du décodage : poids du modèle de langage, probabilité du silence et des autres bruits, probabilité de production d'un mot. Cette étape est faite avec l'algorithme du simplex en quelques jours de calculs.

5 Expériences et évaluation

5.1 Expériences

En plus de la comparaison entre les 2 types de systèmes, nous avons aussi testé différentes variantes dans la représentation des mots et dans l'algorithme de décodage pour le système *end-to-end*.

5.1.1 Représentation de mots

Nous avons testé 3 solutions pour cette représentation :

1. Représentation par la suite de lettres suivies d'un symbole de fin de mot : cette solution produit 44 lettres correspondant aux 26 lettres de l'alphabet et à leurs versions accentuées ainsi que 4 symboles spécifiques « `|` `'` `S` `B` ». Cette représentation sera notée *Base*.
2. Dans le but de diminuer la couche de sortie et de tenir compte des faibles différences de prononciation de certaines lettres accentuées, nous avons regroupé l'ensemble des ces lettres sur leur version non-accentuée à l'exception des lettres *é* et *è* réunies ensemble. Ceci réduit l'ensemble à un total de 35 symboles. Cette représentation sera notée *Regr*.
3. Comme une des spécificités du français est une plus grande coarticulation entre les mots allant jusqu'à la liaison, nous avons voulu tester une représentation sans symbole de fin de mots. Elle sera notée *NoSep*.

5.1.2 Algorithme de décodage

Plusieurs possibilités existent pour guider la recherche en faisceau du système *end-to-end*. En plus d'une transcription contrainte par un lexique, nous avons testé pour la solution *Base*, la comparaison entre un modèle de mots 4-grammes et des modèles sur les lettres d'ordre 2, 7 et 10 notés *Lettre2*, *Lettre7* et *Lettre10*. Nous n'avons pas été jusqu'à un ordre de 15, comme dans (Likhomanenko *et al.*, 2019), car le peu de variabilité de nos corpus de textes entrainait des problèmes de convergence numérique dans les algorithmes d'élagage du modèle de langage. Pour les solutions *Regr* et *NoSep*, nous avons utilisé seulement le modèle en mots.

5.2 Évaluation

Pour cette dernière phase de travail, nous avons utilisé les outils du défi REPERE (Giraudel *et al.*, 2012) qui demandent un étiquetage temporel des mots. Pour obtenir ce dernier pour les sorties du système *end-to-end*, nous avons dû procéder en plusieurs temps :

— une équi-répartition temporelle des mots dans le segment de la transcription automatique,
— suivie d'une première évaluation qui fournit 2 suites *sentences* parallèles ; l'une avec le texte de la référence, l'autre avec celui de l'hypothèse.
— l'utilisation de *mwerSegmenter*[2] a permis de répartir les mots de l'hypothèse sur chaque *sentence* de la référence (et par là-même l'intervalle de temps desdites *sentences*). Cette réaffectation s'est faite à la marge, après la première évaluation.
— enfin, une nouvelle équi-répartition des mots a été réalisée en utilisant les temps de l'étape précédente.

Nous présentons ci-après, dans le tableau 4, les résultats de l'ensemble des systèmes sur les corpus de développement et de test.

2. https://www- i6.informatik.rwth- aachen.de/web/ Software/mwerSegmenter.tar.gz

Corpus	Système	Cor.	Sub.	Sup.	Ins.	WER
Dev.	Kaldi	88,18	7,83	3,63	3,14	**14,60**
Dev.	Base	76,84	9,72	12,95	4,14	26,81
Dev.	Regr	74,69	7,27	17,54	0,83	**25,63**
Dev.	NoSep	72,38	17,20	9,93	3,17	30,31
Dev.	Lettre2	60,86	23,91	14,74	5,54	44,19
Dev.	Lettre7	73,59	18,17	7,76	6,61	32,53
Dev.	Lettre10	75,52	16,82	7,17	6,55	30,54
Test	Kaldi	88,63	7,17	3,58	2,80	**13,55**
Test	Base	80,19	10,21	8,95	4,41	**23,58**
Test	Regr	74,55	6,69	18,07	0,63	25,40
Test	NoSep	73,65	16,68	9,02	2,96	28,65
Test	Lettre2	59,49	24,63	15,22	5,31	45,16
Test	Lettre7	73,74	17,71	7,90	6,36	31,97
Test	Lettre10	75,72	16,25	7,37	6,48	30,11

TABLE 4 – Résultats de l'évaluation

6 Discussions

6.1 Les *end-to-end*

Avant même de comparer avec le système KALDI, un des premiers enseignements à tirer réside dans le fait que le système *end-to-end* a besoin d'un séparateur de mots bien que celui-ci n'ait aucune réalité en français (*Base* vs *Regr*). Pour le regroupement des lettres, la réduction de nombre de paramètres (11830 sur 200 millions) ne semble pas jouer. La différence entre les résultats du corpus de test par rapport au corpus de développement nécessite une analyse des résultats par type d'émissions. Concernant les décodages avec LM lettres, ils s'améliorent certes avec l'ordre mais les temps de calcul deviennent aussi importants que pour le LM classique. On s'aperçoit rapidement que le modèle d'ordre 2 est effectivement intéressant dans sa vitesse d'exécution, notamment pour une recherche en mots clés avec tolérance à l'orthographe, ce qui peut présenter un véritable intérêt dans un couplage avec des *embedings*.

6.2 KALDI vs Wav2letter

Le système KALDI présente une amélioration d'environ 40% en relatif par rapport au meilleur *end-to-end* : *Base*. Ce taux est similaire à celui présenté dans (Lüscher *et al.*, 2019). Cet écart important a deux explications principales. La première réside dans l'utilisation de connaissances phonétiques à travers le dictionnaire. Celles-ci ont l'avantage de mentionner les variations de prononciation des homographes hétérophones : « *Les poules du couvent couvent.* ». Le deuxième point concerne un problème algorithmique : avec les seuils utilisés dans la recherche en faisceau, le temps de décodage pour KALDI est de 7 minutes pour 5h30 de parole *vs* 1h30 pour le *end-to-end* en utilisant tous les coeurs de la machine. Il semble donc difficile d'élargir le champ de recherche pour le *end-to-end*. On pourrait sans doute augmenter son efficacité en faisant une première recherche avec un ML sur les lettres, puis en utilisant un ML sur les mots. Toutefois, l'efficacité de KALDI réside principalement dans la réunion, en un seul réseau de type FST (Mohri *et al.*, 2002), de l'ensemble

des connaissances acoustiques, phonétiques, lexicales, langagières. Après compilation de ce réseau, les poids des arcs sont poussés au maximum vers le début du réseau, ainsi les procédures d'élagage peuvent fonctionner au niveau de la trame avec le maximum d'informations. Il n'y a pas de "coupure" entre l'acoustique, le lexique, et le modèle de langage. Or, dans le cas du *end-to-end*, ce sont bel et bien trois entités distinctes lors de la recherche en faisceau. Pour une étude plus fine, une comparaison entre les 2 systèmes est présentée dans le tableau 5 sur les différents types d'émissions du corpus de test. L'émission LCP_topQuestions est une émission très facile à décoder, seulement ce type de langage relativement contraint dialogiquement est faiblement présent dans le corpus d'apprentissage acoustique. L'ajout du ML en fin de processus avec la limitation de la recherche en faisceau ne permet pas de faire survivre la bonne solution. Ce phénomène est présent, dans une moindre mesure, pour LCPActu. L'émission Culture et Vous, qui se caractérise par un français très spontané, décousu et bruité, présente, tout en ayant des écarts de scores en relatif stables, des variations en absolu allant jusqu'à rendre la transcription presque inutilisable, dans le cas du *end-to-end*.

Émissions	Kaldi	Base
BFMTV_BFMStory	14,24	23,03
BFMTV_CultureEtVous	21,61	41,14
BFMTV_RuthElkrief	16,29	24,95
LCP_CaVousRegarde	12,26	22,34
LCP_EntreLesLignes	11,57	22,15
LCP_LCPActu	9,06	17,61
LCP_LCPInfo	13,78	22,43
LCP_PileEtFace	10,75	21,59
LCP_TopQuestions	7,83	18,06

TABLE 5 – WER sur le corpus de test par type de systèmes

7 Conclusion

Dans cet article, nous avons présenté une comparaison entre un système hybride et un système *end-to-end*. Le peu de disponibilité de corpus textuels par rapport aux enregistrements audio implique la nécessaire utilisation, pour chacun, d'un modèle de langage. Or, l'intégration de ce dernier dans le processus de décodage du système hybride permet un élagage performant au niveau de la trame, ce qui rend ce dernier réellement plus efficace. Il faut noter aussi que, compte tenu du découplage entre langage et réseaux de neurones, une meilleure adaptation est réalisable en ne travaillant que sur un ML dédié (pour les émissions TOP_QUESTION par exemple). Ce phénomène confère au système hybride un avantage des plus sérieux, notamment si l'on souhaite l'intégrer en première brique à une chaîne de traitements complète comme celles des *chatbots*. Par ailleurs, ce système hybride présente l'intérêt d'un apprentissage nettement plus écologique : on constate ainsi un rapport de 1 à 10.

Références

AMODEI D., ANUBHAI R., BATTENBERG E. & ALL (2016). Deep speech 2 : End-to-end speech recognition in english and mandarin. In *The 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 173–182 : JMLR.org.

BARRAS C., ZHU X., MEIGNIER S. & GAUVAIN J.-L. (2006). Multi-stage speaker diarization of broadcast news. *IEEE Transactions on Audio, Speech and Language Processing*, **14**(5). DOI : 10.1109/TASL.2006.878261, HAL : hal-01434241.

GALES M. (1998). Maximum likelihood linear transformations for hmm-based speech recognition. *Comp. Speech and Lang.*, **12**, 75–98.

GIRAUDEL A., CARRÉ M., MAPELLI V., KAHN J., GALIBERT O. & QUINTARD L. (2012). The REPERE corpus : a multimodal corpus for person recognition. In *The Eighth International Conference on Language Resources and Evaluation (LREC'12)*, p. 1102–1107, Istanbul, Turkey : European Language Resources Association (ELRA).

GRAVES A. & JAITLY N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *The 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, p. II–1764–II–1772 : JMLR.org.

GRAVIER G., ADDA G., PAULSSON N., CARRÉ M., GIRAUDEL A. & GALIBERT O. (2012). The ETAPE corpus for the evaluation of speech-based TV content processing in the French language. In *The Eighth International Conference on Language Resources and Evaluation (LREC'12)*, p. 114–118, Istanbul, Turkey : European Language Resources Association (ELRA).

GRAVIER G., BONASTRE J.-F., GEOFFROIS E. & ALL (2004). The ESTER evaluation campaign for the rich transcription of French broadcast news. In *The Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

HAN K. J., CHANDRASHEKARAN A., KIM J. & LANE I. (2017). The capio 2017 conversational speech recognition system. *Preprint :* arXiv:1801.00059 [cs.CL].

LIKHOMANENKO T., SYNNAEVE G. & COLLOBERT R. (2019). Who needs words ? lexicon-free speech recognition. In *Interspeech 2019* : ISCA. DOI : 10.21437/interspeech.2019-3107.

LÜSCHER C., BECK E., IRIE K., KITZA M., MICHEL W., ZEYER A., SCHLÜTER R. & NEY H. (2019). Rwth asr systems for librispeech : Hybrid vs attention. *Interspeech 2019*. DOI : 10.21437/interspeech.2019-1780.

MEIGNIER S. & MERLIN T. (2010). LIUM SPKDIARIZATION : AN OPEN SOURCE TOOLKIT FOR DIARIZATION. In *CMU SPUD Workshop*, Dallas, United States. HAL : hal-01433518.

MOHRI M., PEREIRA F. & RILEY M. (2002). Weighted finite-state transducers in speech recognition. *Comput. Speech Lang.*, **16**(1), 69–88. DOI : 10.1006/csla.2001.0184.

PANAYOTOV V., CHEN G., POVEY D. & KHUDANPUR S. (2015). Librispeech : An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 5206–5210. DOI : 10.1109/ICASSP.2015.7178964.

POVEY D., GHOSHAL & ALL (2011). The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding* : IEEE Signal Processing Society. IEEE Catalog No. : CFP11SRW-USB.

POVEY D., PEDDINTI V., GALVEZ D. & ALL (2016). Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech 2016*, p. 2751–2755. DOI : 10.21437/Interspeech.2016-595.

PRATAP V., HANNUN A., XU Q., CAI J., KAHN J., SYNNAEVE G., LIPTCHINSKY V. & COLLOBERT R. (2019). Wav2letter++ : A fast open-source speech recognition system. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 6460–6464.

ZEGHIDOUR N., XU Q., LIPTCHINSKY V., USUNIER N., SYNNAEVE G. & COLLOBERT R. (2018). Fully convolutional speech recognition. *Preprint :* arXiv:1812.06864 [cs.CL].