End to end Speech Recognition With Transformers

-Master Thesis and internship report-

Master's Degree I.A.S.D.:
Artificial Intelligence, Systems, Data.

submitted by:
BOUSKILA Laurene
laurene.bouskila@gmail.com

Sept. 12th, 2021

Under the supervision of:
SUTTON Charles
charles@datascientest.com
Chief Technology Officer at DataScientest.com

Abstract

Datascientest.com is a french startup created in 2015 which aims at engaging learners on online education expanding around Data Science related fields. This startup develops its own content and its main target is the commitment of the users to learn and acquire valuable technological skills. The end goal can be very different if the user belongs to a B2B or a B2C *cohort*. Indeed, as DataScientest has a partnership with *Université Paris Sorbonne* and *Les Mines*, some users register to this formation to certify and develop skills, to specialize in a field in complement to their job, or even for a career change. DataScientest also possesses an R&D department called the *Studio* which conceptualizes *end-to-end* technological projects.

My internship at DataScientest.com gave me the opportunity to acquire and develop new technological as well as managing skills.

As a matter of fact, I had to bring my contribution to the users learning experience. This includes developing actual online content: notebooks, evaluations, explanatory videos, technological papers, teaching of recorded masterclasses and the creation of cheat sheets, to allow a partially asynchronous learning experience. I also had the chance to mentor several data science, data analyst and deep learning projects.

On the other half, I developed a project for the *Studio* on the following topic: *End-to-end Speech Recognition With Transformers*.
This topic was important to me to give coherence to my study track. As a matter of fact, I wanted to find a field that would combine my B.Sc. in electrical engineering and the knowledge acquired during the IASD M.Sc: *Signal Processing* thus seemed to be the appropriate option to go for.
To introduce this concept, I will follow the exact same steps I developed during my internship. That is, we will start by reviewing the core concepts of speech recognition. We will then analyze the speech to text concept and other applications of vocal recognition systems (especially through transformers). To conclude, we will talk about the *VoiceGlass* project which aims at providing a solution for deaf people thanks to connected glasses. This project, also developed in the *Studio* by alumnus (Malik ALY MARECAR & Lamia BOUGARA), uses speech recognition to transcribe voice into subtitles displayed on the glasses.

Acknowledgments

I would first like to start by thanking Charles Sutton not only for his devotion as an internship supervisor, but also for the time and precious advice he has given me. My professional experience would have been far from similar without his understanding and transpiring motivation to provide a great working environment.

Additionally, Thomas Boehler and Pierre Adeikalam have offered me support and assistance with regards to some demanding tasks, and for that I am truly grateful to them. In particular, Thomas has been greatly helpful as far as technical issues are concerned: he has been available on demand and always eager to help me find solutions in a timely manner.

It is of great importance for me to mention and thank Prof. Eli Gershon for his presence and continued assistance throughout my entire project. He has been a wise and precious source of advice and help for my postgraduate endeavours, and I am proud of having once been one of his students.

Finally, I want to thank Tristan Cazenave and Benjamin Negrevergne for their assistance during my studies, and I am happy to present and share with them the product of my work.

Table of Contents

1. Working at DataScientest...

In light of the fact that being an intern in a startup also means being committed to its evolution and understanding the basic needs of its clients, this section will emphasize my contribution to DataScientest. In this context, I had to develop a wide variety of materials and online contents as well as mentoring users' final projects at the end of their program.

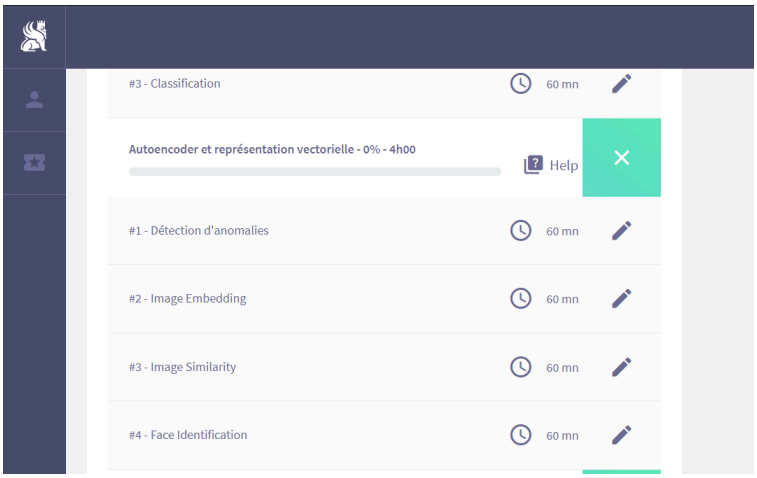
A. Content development

DataScientest offers various courses such as Data Scientist, Data Analyst, Data Engineer, Data Management amongst others... There are two intensities available for these courses: users can choose to follow them as a bootcamp or as a continuous formation, depending on the rhythm they want to attribute to the cursus.
Learning by doing: This mantra relies on the mainstays of the startup that focuses on providing practice-based instruction which melts theory into practice for a better understanding and ability to handle data science practical problematics.

Module development

When starting the program, users are given a dedicated educational trail which enumerates a list of modules to pass in order to graduate and is available on the DataScientest training platform. Each module is composed of exercises under the form of jupyter notebooks which provide theoretical explanation blocks followed by practical questions with hidden proposed solutions for each one of them. Once these exercises are validated, the user is given access to pass an evaluation which gathers and tests all the important skills the module comports.

Fig: Module and exercise screenshots for the Autoencoder and vectorial representation module (french version)



During my internship, I had to create some modules including exercises and evaluations in the context of the expansion of the proposed courses to Deep Learning.
Amongst them, we can count the development of three modules:

- **1. Autoencoder and vectorial representations:**

This module talks about the use of the autoencoder model through various topics.

- First the domain of *anomaly detection* is presented to the learner to give him an insight of practical use of autoencoders in nowadays society. As a matter of fact, the autoencoder is used in an unsupervised learning context in which the encoder is given non fraudulent inputs and the model should be able to predict whether an entry is fraudulent or not, in spite of the fact that it never met such an input during its training phase.

- Next, the concept of *image embedding* is observed through the computation of encoding vectors for each input image of the MNIST dataset. The notions of reconstruction by the decoder and denoising are also discussed.

- Another important topic to the next exercise is the *image similarity* concept which is reviewed through various clustering methods such as PCA, T-SNE, K-Means, LDA, euclidean distances, IsoMap...

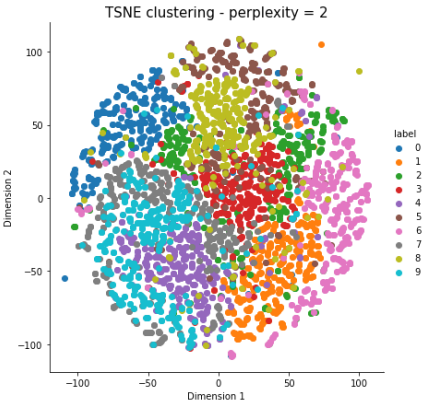


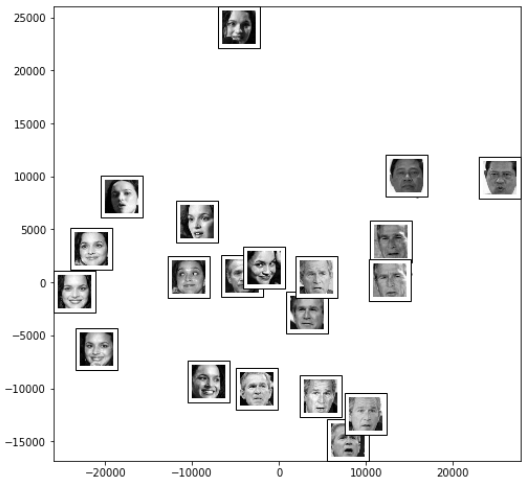
Fig: T-SNE representation (with a perplexity of 2) for the MNIST dataset

- The last exercise contained in this module is a *face identification* challenge which is made out of face detection and face recognition problems on the famous characters of the F.R.I.E.N.D.S TV series. The face identification problem is about determining the coordinates of a bounding box such that each image is cropped to this rectangle, so the model is not influenced by any bias that might exist in the image background (shapes, colors...).

Fig: Face recognition challenge on the characters of the series FRIENDS

Then the user has to encode each resulting image, and build a system that should be able to detect the character fed to the model. The observations are made out of three samples for each: two corresponding to the same person, and the third one to a different person. The autoencoder is pre-trained with these observations to minimize the distance calculated between the first two samples and maximize the distance to the third sample. These data being labeled, the model is trained in a supervised learning context. However, once the model is trained, the system is being fed with unlabeled data and it is its role to predict the face label of the image: this exercise is then an unsupervised learning problem.

- Finally, for the evaluation I decided to gather all these notions in a larger dataset available from Kaggle: lfw-funneled
  https://www.kaggle.com/atulanandjha/lfwpeople



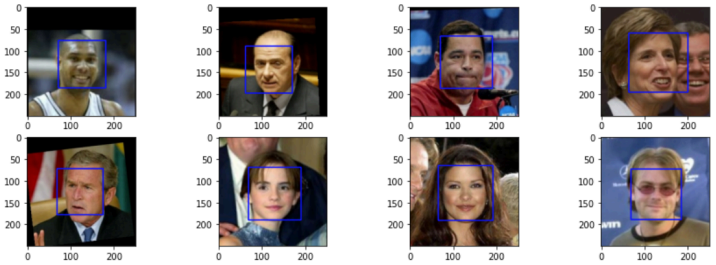


Fig: Autoencoder evaluation dataset sample and IsoMap for three different celebrities.

- **2. Speech recognition Core Concepts**

- **3. Speech Recognition advanced**

These last two modules review the basic concepts of signal processing and speech recognition systems necessary to understand how to implement these models in python. As they are part of my final project topic, we will go over their content later on in this report.

Research papers and articles

Research papers and articles are part of the learning process for users as well as for external people interested in a specific topic. During my internship I had to write two articles related to the Deep Learning field.

- DenseNet:

Fig: DenseNet article posted on the DataScientest *Blog* preview (FR)

This paper explains the configuration and the use of DenseNet architecture.
It starts by explaining the basic construction of a convolutional neural network and the different layers that compose it.

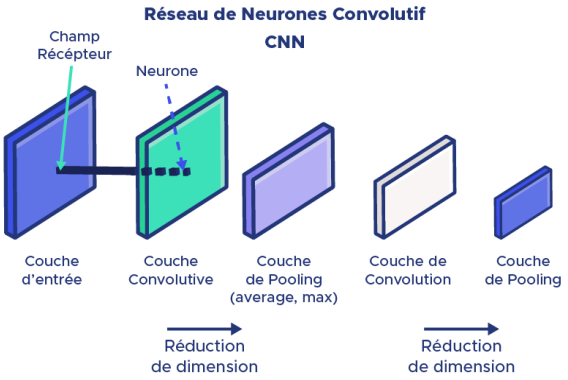


Fig: Architecture of a CNN (FR)

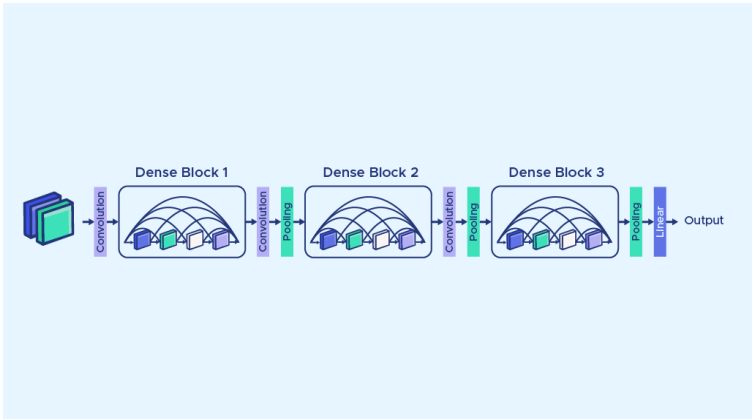


Fig: Architecture of the DenseNet

The DenseNet architecture belongs to the convolutional neural networks family and allows for the optimization of the performances especially in the images processing domain.
Its singular structure is made out of a stack of dense blocks, convolutional pooling layers successively. The pooling layers (average or max polling) ensure the data dimension restriction which justifies the efficiency of this kind of network.

From facial recognition to Natural Language Processing fields (NLP), the DenseNet applications to the field of AI are wide.

- Post treatment methods for Speech Recognition

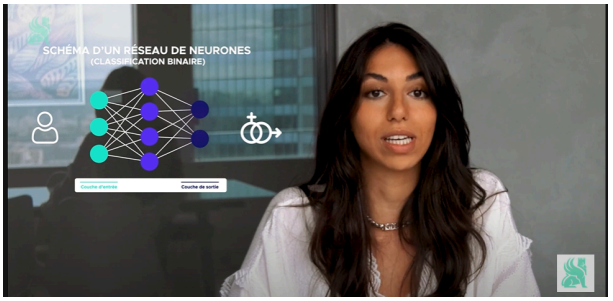
As this topic is part of the possible openings for the *End to End Speech recognizer,* it will be discussed in the second section of this paper.
The whole article can also be found in the appendix.

Videos

DataScientest is also active on social networks, and regularly publishes technological content such as videos on various topics of Data Science related fields. Not only does this enable the users to get visual content and graphic explanations on complex topics, but it also opens this access to any follower.
In this context, I was asked to record an explanatory video on several subjects such as:

- Autoencoder
- Word embedding and translation systems
- Feature Scaling

These videos can be accessed through the links available in the resource page.



Cheat sheets

Cheat sheets are useful tools to new learners that sum up most of the important tips and information of a module. They often contain portions of recurrent codes or explanations on a specific tool such as in the context of Power BI.
I created some of them on various topics:

- Keras (see appendix)
- Comparison of Machine Learning Algorithms (in process)
- Tableau (in process)

B. Project Mentorship

At the end of their formation, the users are gathered into groups according to the project topic they chose in the ”*catalogue*”. The project completion is mandatory to validate the program. This section ensures that the learners are able to put into practice all the theory learnt during the training phase and complete a typical data science project.
The mentor’s role is to guide the users and review their work to ensure that they are on the right track.
In this context, I had to mentor three projects listed below.

Online Portfolio Allocation

The financial and bank fields are some of the fields generating the biggest amount of data and these datas are often freely accessible and quite frequent. The portfolio allocation is particularly interesting since it is abundant and generalizable in a sense.

Fig: Backtesting strategy screenshot

The goal of this project is to create a portfolio allocation model which adapts its online strategy but can also adapt the machine learning approach to a similar domain.

This project was conceived around the NYSE, DJA and TSE data available from the following gitHub link:
https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

Some additional datasets were created from the financial api such as yahoo and google finance.
Here are some previews of the strategy and results.

Tennis betting

The main goal of this project is to try to beat the bookmakers' algorithms on the estimation of the probability of a winning team. This is a perfect problem to treat all the basic steps of a data science project.
A first step was to apply preprocessing methods to clean the dataset. Then, one should extract from the match history the appropriate features to evaluate the performance of a player or a team (ranking, winning streak...). Finally, from these features, the goal is to estimate the probability of a player/team A to beat a player/team B.

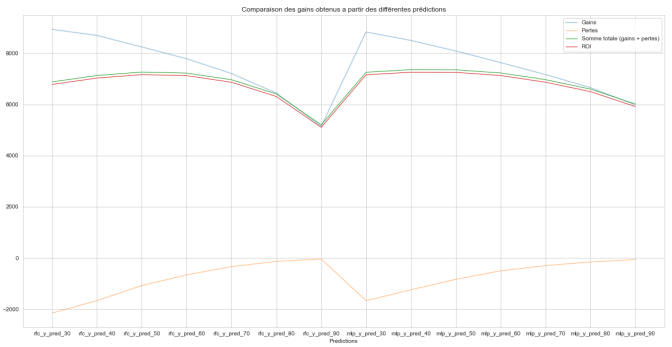


Fig: Wins comparison from different predictions preview

Once the training phase is over, the users should compare their model's performances to the bookmakers' model.
The dataset can be downloaded from the following Kaggle link:
https://www.kaggle.com/edouardthomas/atp-matches-dataset

C. Masterclasses

The masterclasses are synchronous sessions during which the data scientists intern and employees teach courses on specific topics to the cohorte of learners.
They are recorded so that the users can view their rediffusion. The masterclasses rely on both theoretical content and a typical practical application.
During my internship, I had to teach the following masterclasses:

- Introduction to python and errors handling
- Data Visualization in python (Matplotlib and Seaborn)
- Streamlit & Git
- Handling a data science project (end to end)
- Interpretability of machine learning models
- Deep learning
- Vocal Recognition systems
- Data profiling

2. Speech Recognition

A) Introduction

For a long time, speech recognition models have aroused a great interest to the computer science community who aimed at developing algorithms and methods to enable the conversion from audio signals to text intelligible by the computer.
This domain is also known under the name of Automatic Speech Recognition (ASR) or Speech to Text (STT).

The speech recognition problem relies on the audio clip data: that comes from the measurement of change in air pressure of the microphone against time.
The human ear has a physical structure that is able to measure the amount of intensity of different frequencies present in the audio. This step is encapsulated in what is called a *spectrogram*: this graph that displays the frequencies against time by taking into account the amount of energy/ intensity is commonly used as a preprocessing step of the audio.

The ASR (Acoustic Sound Recognition) domain has been a concern for a long time in engineering mainly due to the fact that the traditional calculation limits imposed by the data signal weights. With Deep Learning emergence aroused various ease in the time calculation and in the final cost of the operations.

Speech recognition field was significant to me since it closely relates to the Signal *Processing* domain which was one of the main specializations during my B.Sc. I wanted to find a final project topic that would use my previously acquired knowledge to the world of Data Science.

B) Core Concepts

In order to introduce the users to the module of speech recognition, I had to remodel the Core Concepts course content.

- **Sampling frequency**

When capturing the sound, the microphone transforms the acoustic pressure (physical measure of the sound) into an electric analog signal. It is a continuous signal across time. In order to deal with this signal, one should convert it into a discrete or numerical signal. We call the **sampling frequency** the number of samples (points) by second, at regular intervals of the analog signal.

$$d_t = \frac{1}{f_e}$$

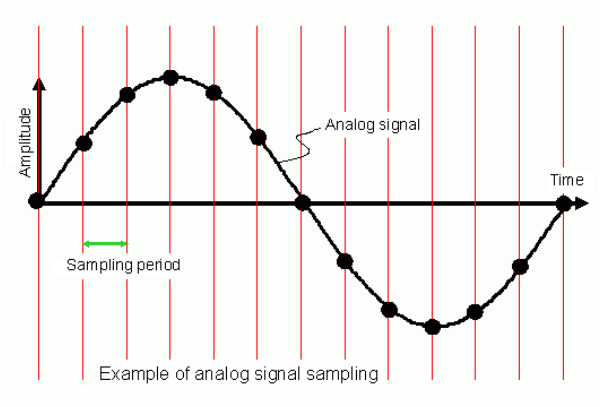


Fig: Sampling frequency scheme

We call the **sampling period** (in sec) the time between two successive measures.
The **Shanon theorem** states that a sampling frequency f_e can transmit information with no loss only frequencies lesser than 0.5*f_e. That is to say, the sampling frequency choice determines the highest frequency we will be able to represent in our signal. A high frequency means a quick oscillation or a short period. The distance between the two points in our discrete signal can be too big to encapsulate this quick oscillation.

- **Fourier Transform (FT) and Inverse Fourier Transform (IFT)**

A sound being the superposition of several frequencies, it is preferable to work in the frequency domain by focusing on the spectral representation. The spectral representation allows the decomposition of a sound into its frequential components (the different sinusoidal periods which compose it). Each sinusoidal has an intensity more or less important that we call amplitude in the spectral domain and which represents the associated energy of this frequency. This representation is calculated thanks to the Fourier transform.
The FT associates a time function f(t) definite on $\mathbb{R}$ and with real or complex values, to another function called the Fourier Transform. It can be interpreted as the frequency in physics. As the audio numerical signals are not continuous, we are going to use the

Discrete Fourier Transform (DFT).

$$\mathcal{F}(k) = \sum_{n=0}^{N-1} s(n) \cdot \exp(-2i\pi f_0 k)$$

$f_0 = \frac{n}{N}$ where f_0 is the fundamental frequency and the first harmonic of a sound. In other words, it corresponds to the lowest frequency.
The kth harmonic is the multiple by k of the fundamental frequency.
The amplitudes are represented by the module of the transform.

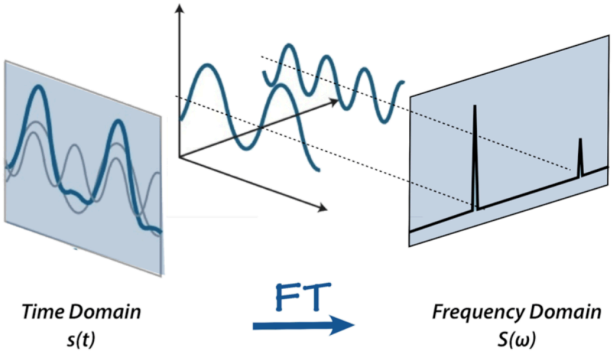


Fig: Fourier transform preview: time v/s frequency domain

- **Spectrogram**

A real sound is a sound which contains frequencies varying across time. To be able to analyze this, there exists a tridimensional representation called the spectrogram.
The spectrogram is divided in windows called *chunks* on which are applied the Fourier Transform individually. Thus, the spectrogram represents the evolution of the Fourier Transform of a signal across time.

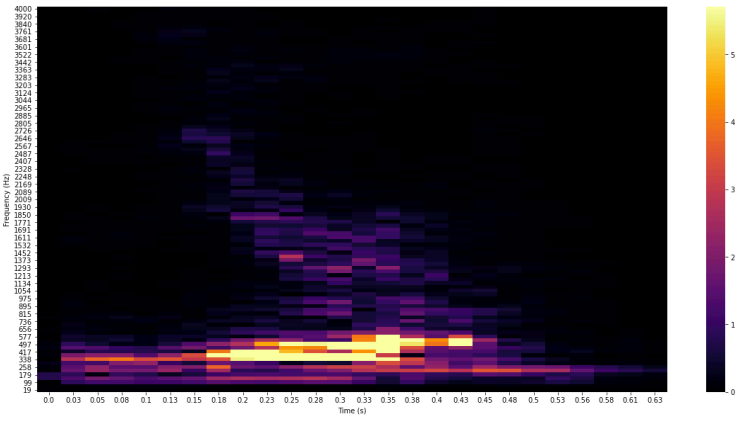


Fig: Spectrogram

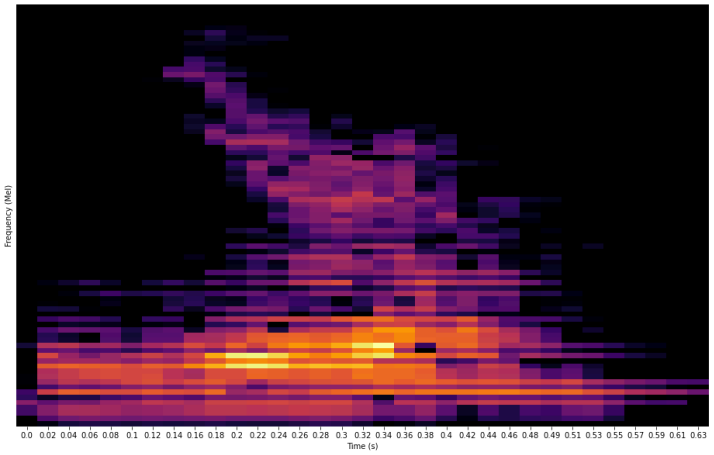
A spectrogram depends on three parameters:

- The number of points on which the FFT is calculated. It is preferable for this number to be a power of 2. Traditionally, we choose 2048 for music and 512 for speech. This parameter is related to the sampling rate.
- The window size that should be lesser or equal to the above criteria. A low value allows a good time resolution but a poor frequential resolution and vice versa.
- The hop length which generally equals to half of the window size. This parameter represents the number of points between two time windows.

Finally, the spectrogram has 3 dimensions: The time (x-axis), the frequencies (y-axis), the frequency amplitudes (coloration scale).

- **Mel Scale**

Fig: Mel Spectrogram

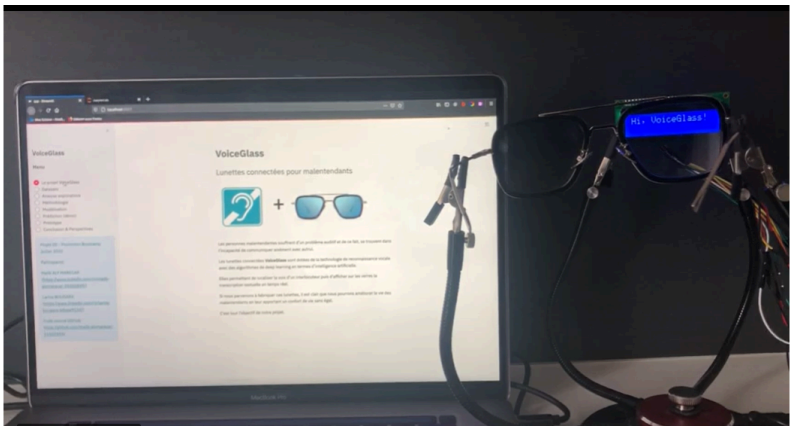


The Mel scale is a frequency scale which imitates the frequency of human perception. As a matter of fact, the human ear sensitivity is not the same in low and in high frequencies.
Moreover, this scale allows to reduce the data size which is beneficial for the training of models in deep learning.
Let's take the example of a low frequency of 1000 Hz. A sound at 2000 Hz will not be sensed as twice as great as the lowest frequency. However, if we do this change in the mel frequency domain, 1000 Hz corresponds to 1000 mels and a sound at 2000 mels which translates to 3428 Hz will be sensed as twice as high as the sound at 1000 Hz.

3. End to end speech recognizer through translation system integration

A) The VoiceGlass Project

The VoiceGlass project is a project integrated in the Studio (R&D department of DataScientest.com) initiated by Malik ALY MARECAR & Lamia BOUGARA which aims at providing a substantial solution for deaf people.



The Voiceglass connected glasses are endowed with the vocal recognition technology that uses deep learning algorithms and AI. They allow sensing of the voice of an interlocutor and the conversion of this audio signal into a text transcription displayed on the glasses in real time.
If you want to know more about this project, the streamlit demo can be found here: **https://studio.datascientest.com/project/voiceglass/**

B) Translation Systems implementation

The translation project is a project I conceived in order to adapt to an existing project implemented in the *Studio* called the *VoiceGlass* project presented above.
That is the reason why I had to conceive, with the precious help of one of my colleagues Thomas Boehler, the *step-by-step protocol* to follow for this project.

The end goal of the translation system project is to adapt the connected glasses to a translation system such that the glasses would sense an audio signal in a specific language and convert it to a text displayed directly on the glasses in the target language. In this way, the person wearing the glasses can read the translation of the spoken sentence pronounced by his interlocutor.

The datasets used can be downloaded from the following links:

- https://github.com/susanli2016/NLP-with-Python/tree/master/data
  small_vocab_fr, small_vocab_en
- https://github.com/facebookresearch/MUSE

In this context, four approaches have been considered.

1. A naive probabilistic approach:

We suppose that the word in language A keeps approximately the same position in the targeted sentence of the language B. We thus create a window array of variable size which encapsulates the words in the targeted sentence around the corresponding position and generates a probability of correspondence for the word (according to the number of occurrences in the whole dataset). As expected, even with some preprocessing steps, the performance is not sufficient (47.69% in the best case).

2. Word2Vec and word by word translation

The main idea behind this step is to encode the words into a representative vector so that the words can be plotted in space and their relations can be analyzed through the observation of the distances between them. Then we are able to learn a rotation matrix W from one monolingual corpus to its correspondent in the other language and associate the word pairs by applying a 1-NN algorithm.

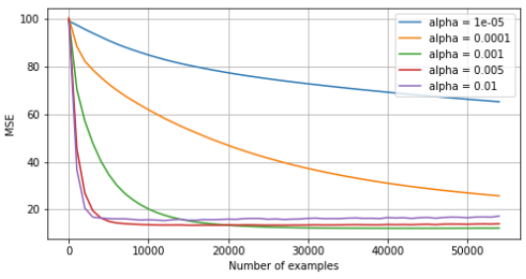


Fig: MSE evolution of the Learning process of the Rotation matrix W

Even though the accuracy is enhanced from the previous step (59.22%, meaning +11.53%), we reject the hypothesis that this method is satisfying for our final translation system version.

3. Seq2Seq translation

This step evaluates the ability of the system to translate sentences in sequences of word steps. The model is enhanced by integrating attention mechanisms to the initial model architecture.

The Seq2seq model is pretrained on a sequence of elements (words, letters, time series ...) and outputs another sequence of elements of the same kind. In the case of automatic translation, the input data are sequences of words in a specific language and the outputs are sequences of words in another language (target language).

In contrast to the previous algorithms that relied on translating sentences segment by segment and concatenating the translations, the Seq2Seq model prevents these brut and often incoherent translations. Indeed, when translating a document, we often need to read the sentence and understand it before translating it.
We talk about ***Automatic Neuronal Translation (NAT)*** which imitates this process and then avoids the lack of smoothness.

The NAT models generally rely on an autoencoder architecture made out of two milestones:

- The **encoder**: which creates the embedding (sequence of numbers) encoding the meaning of the sentence and that encapsulates the global significance of the sequence.
- The **decoder**: which receives this vector and *decodes*, this is the translation phase.

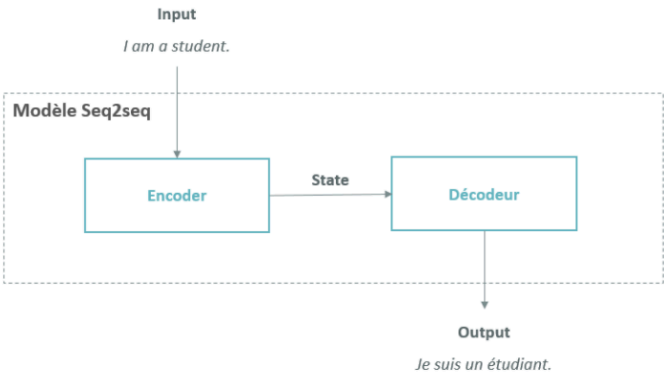


Fig: Autoencoder architecture of the NAT system

Thanks to this approach, the NAT models provide an efficient solution to the local translation problem: they are able to detect the intern dependencies of the sentences (syntaxical structure, conjugation etc...) and so to produce translation significantly smoother.

The NAT algorithms vary according to the encoder and decoder implementations. Accounting for the sequential origin of the data, the recurrent neural networks (RNN) are often used for both the encoder and the decoder. Nevertheless, several implementations are possible amongst the RNN configurations which differ according to the following criterias:

- Directionality: Unidirectional or bidirectional
- Depth: number of layers
- Type: simple RNN, Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU)...

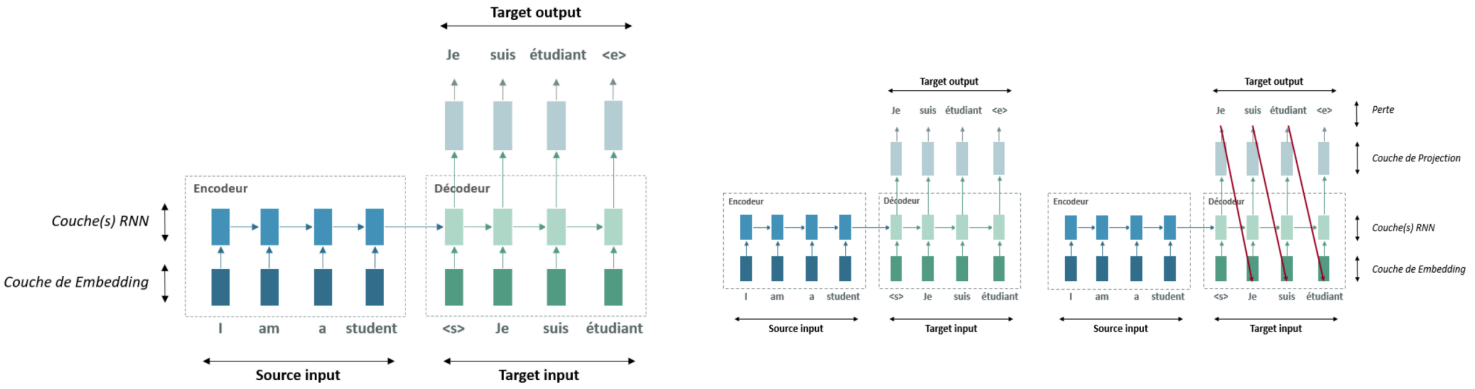


Fig: Encoder and decoder architectures schemes

A NAT model is also able to translate sentences which do not belong to its training set: this is what we call **inference.** Note that it is important to make the distinction between the training phase and the inference which belongs to the test phase. In this case the model does not have the origin sentence outputted by the encoder. There exist several approaches to perform the decoding necessary for the translation.

One of them is called the **Greedy approach.**

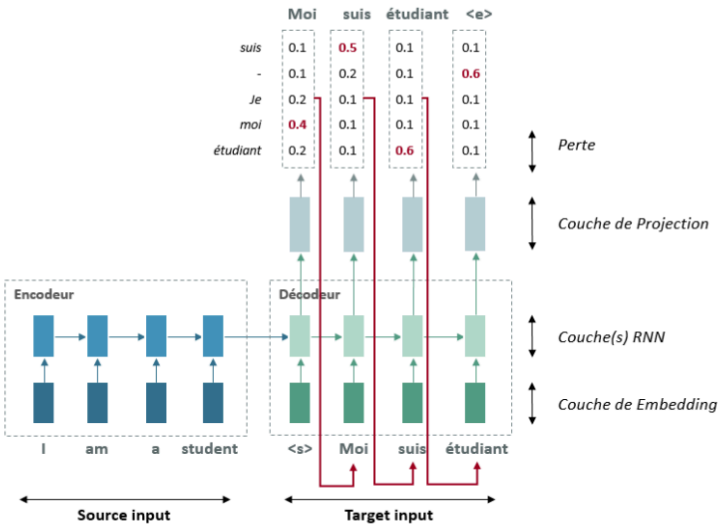


Fig: Inference, Greedy approach

The original sentence is at all times encoded in the same way by the encoder. Once the <start> point is received, the translation phase can start. At each time step, the output from the RNN decoder is treated as a logit

ensemble. The model chooses the most probable word from the associated id to the maximal logit value.
The greedy approach can lead to a reasonable enough translation.
Nevertheless it is possible to improve even more the performances through the use of approaches such as **Beam Search**. This algorithm relies on an exploration of the space of all the possible translations and the saving of a small number of candidates during the translation. The beam size is called the beam width and is generally around 10.

In the simple Seq2Seq model, we talked about the fact that during the decoding phase, the last sentence state goes from the encoder to the decoder. This mechanism is suitable for **short** to middle size sentences but becomes problematic as soon as the sentences become longer. Why is that?

- The encoder state is transmitted only to the first decoder node. For this reason, this information will be judged less and less relevant by the model while progressing in time.
- The model will struggle handling the syntaxical particularities of the languages.

Let's take an exemple: The sentence "He will soon go" corresponds to « Il partira bientôt ». We can see that the word "soon" does not occupy the same position in the sentence as its French equivalent « bientôt ».

The attention mechanism solves this problem, but **how** ?
It relies on the very basic idea to establish direct connections between the target and the original sentence by asking the model to "pay attention" to the most relevant words in the original sentence during the translation.
Thus, instead of removing all of the hidden states calculated in the RNN encoder, it allows the decoder to focus solely on the most important components given by the encoder.

The attention mechanism integration to the automatic translation model relies on the successive construction of two main vectors that we will describe: the alignment and the context vectors.

The alignment vector.
The alignment vector is a vector that has the same length as the original sequence and is calculated at each time step of the decoder. Each one of its values corresponds to the score (or the probability) that this word corresponds to the one of the sequence.

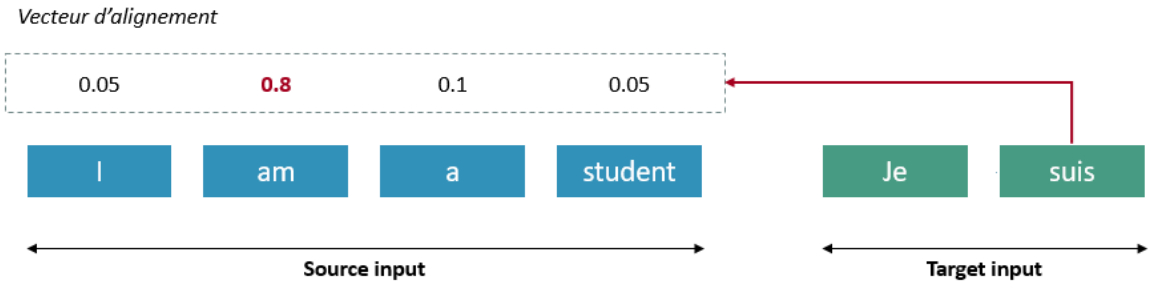


Fig: Alignment vector

Thus, the alignment vector associates each element of the original sequences to weights. In an intuitive manner, it indicates to the decoder what to focus on at each time step.

The context vector.

The context vector is the one used as input by the decoder. It is in reality the weighted average of the encoder output. It is obtained from the scalar product of the alignment vector with the encoder output.

There exist two versions of the attention mechanism depending on the scoring function used in the alignment vector calculation.

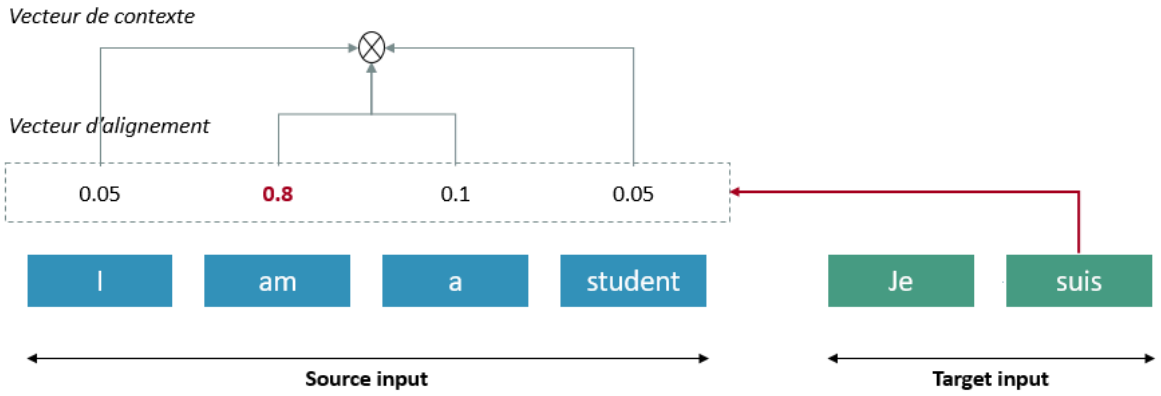


Fig: Context vector

The figure hereunder shows how the attention mechanism is incorporated to the Seq2Seq model.

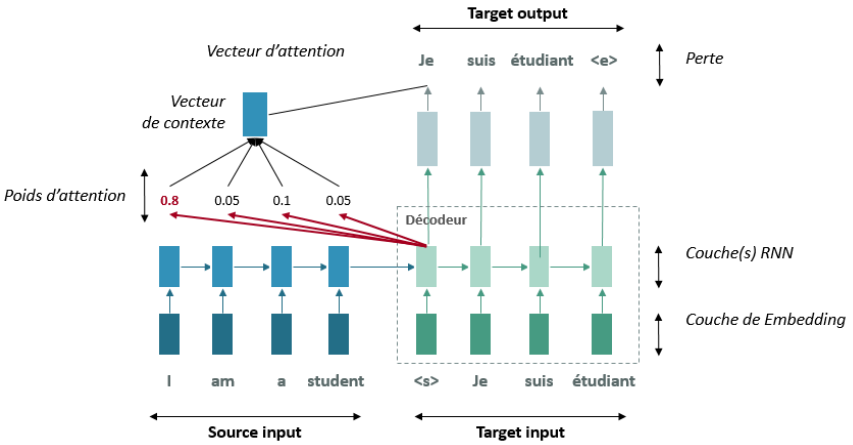


Fig: integration of the attention mechanism

As you understood, the key step resides in the attention weights calculation constituting the alignment vector. The following figure shows an example of this situation. The matrix represents the result of the alignment between an original sentence and its target from the attention mechanism.

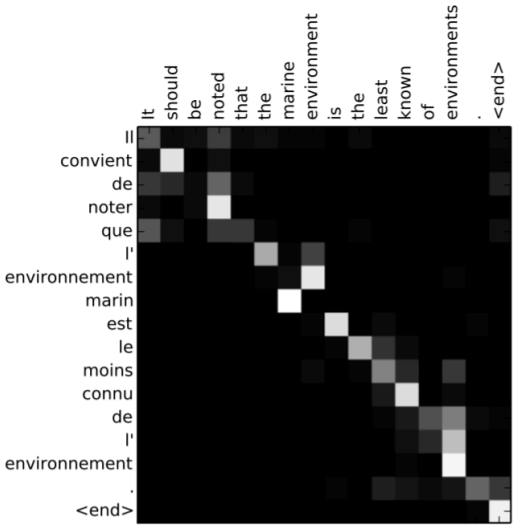
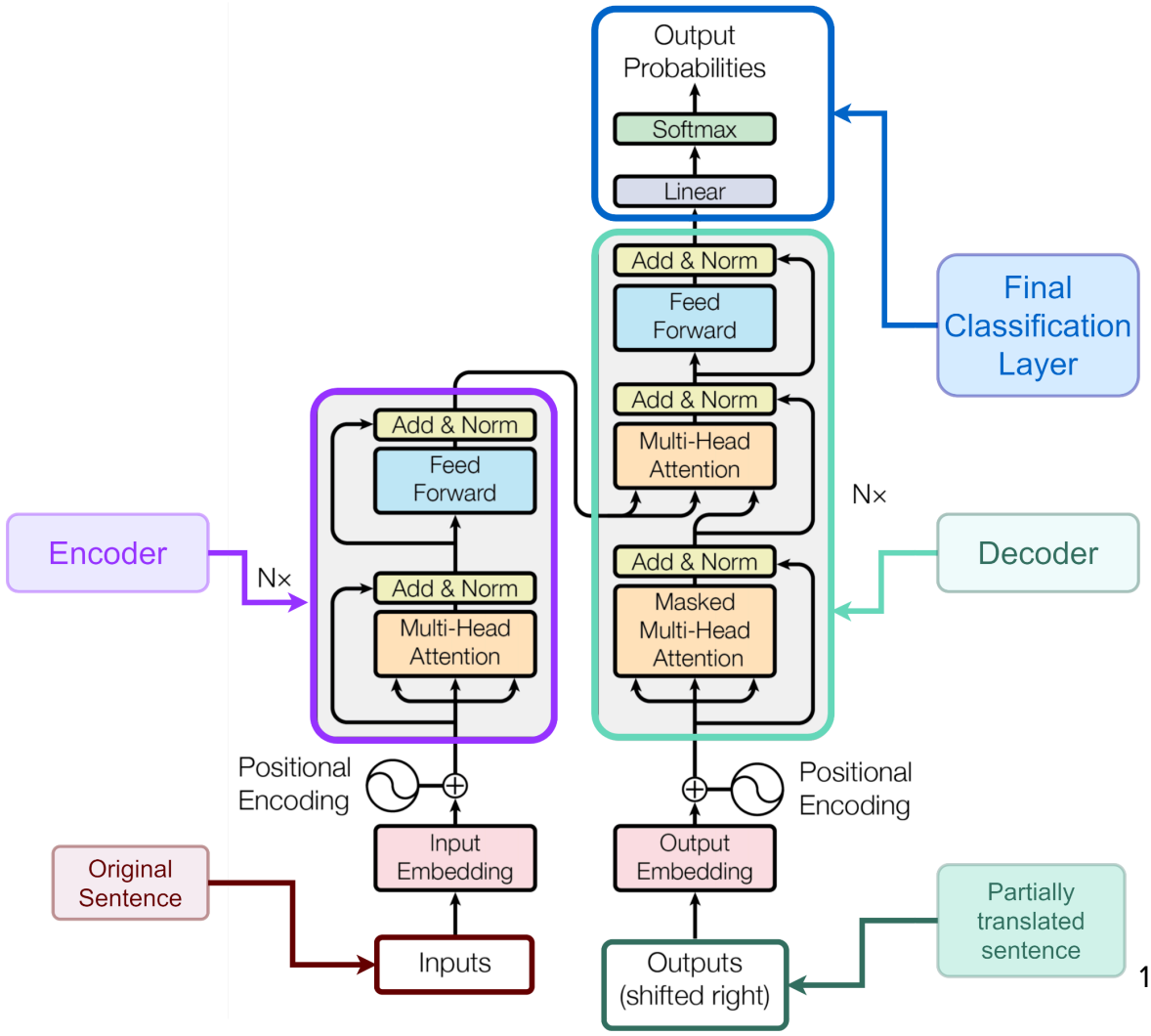


Fig: Alignment exemple

4. Transformers

Finally, we adapt the translation system by replacing the seq2se2 model with a transformer model.

Fig: Transformer basic architecture



As can be seen from the following scheme, the transformer is made out from the coupling of an encoder and a decoder. Keras enables an easy implementation of such a complex model by defining customizable layers.

The transformer uses three concepts: the attention modelisation, the attention by scalar product and the multi-head attention mechanism.

The encoder block fetches the information contained in the translated sentence. It is made out of:

- An encoding of the position to give the words order within a sentence
- A encoding layer made out of a multi-head attention mechanism and a Feed-Forward Network

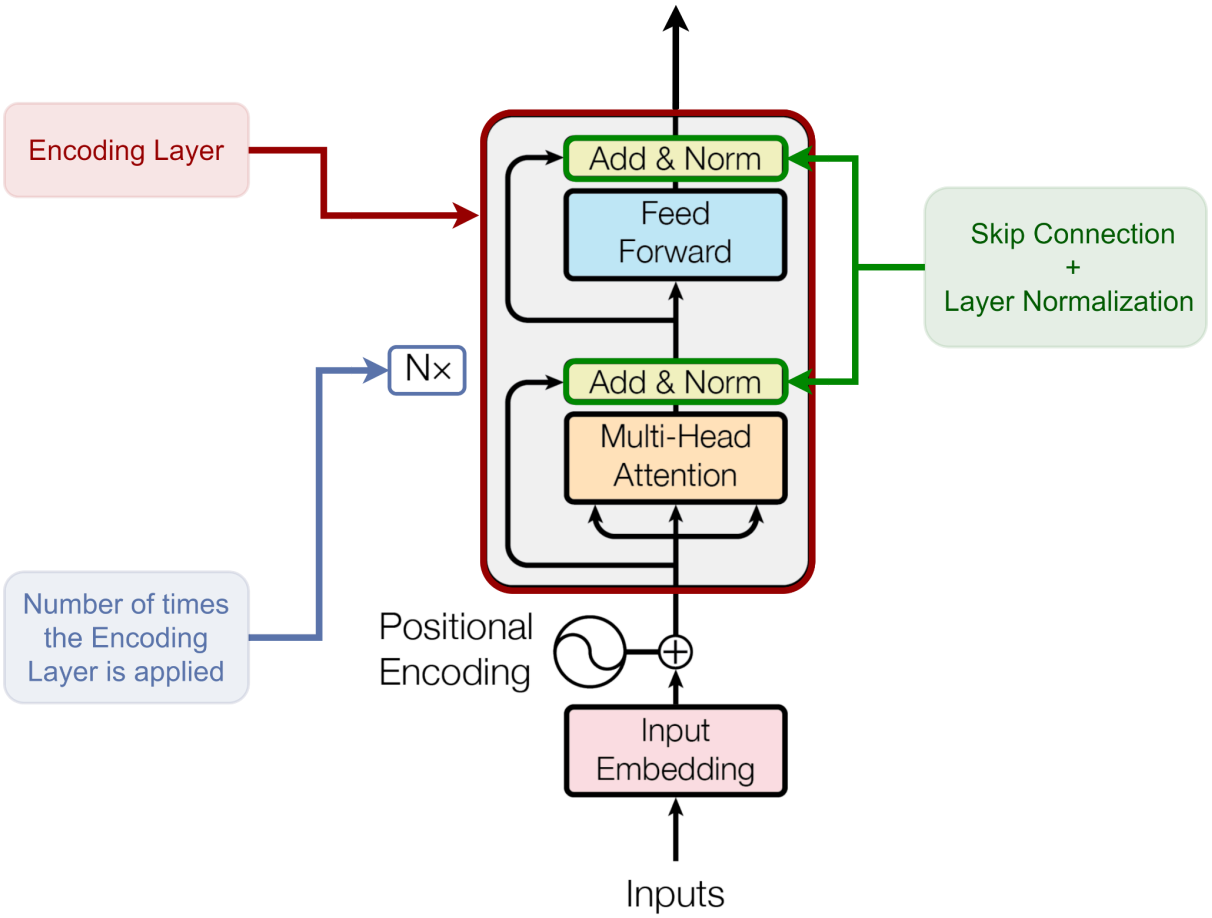


Fig: Encoder architecture

The decoder uses several mechanisms and layers previously defined to predict the next word in the translation:

- Embedding and encoding of the position
- Auto-attention and partial translation
- Attention with the encoder output
- Final classification

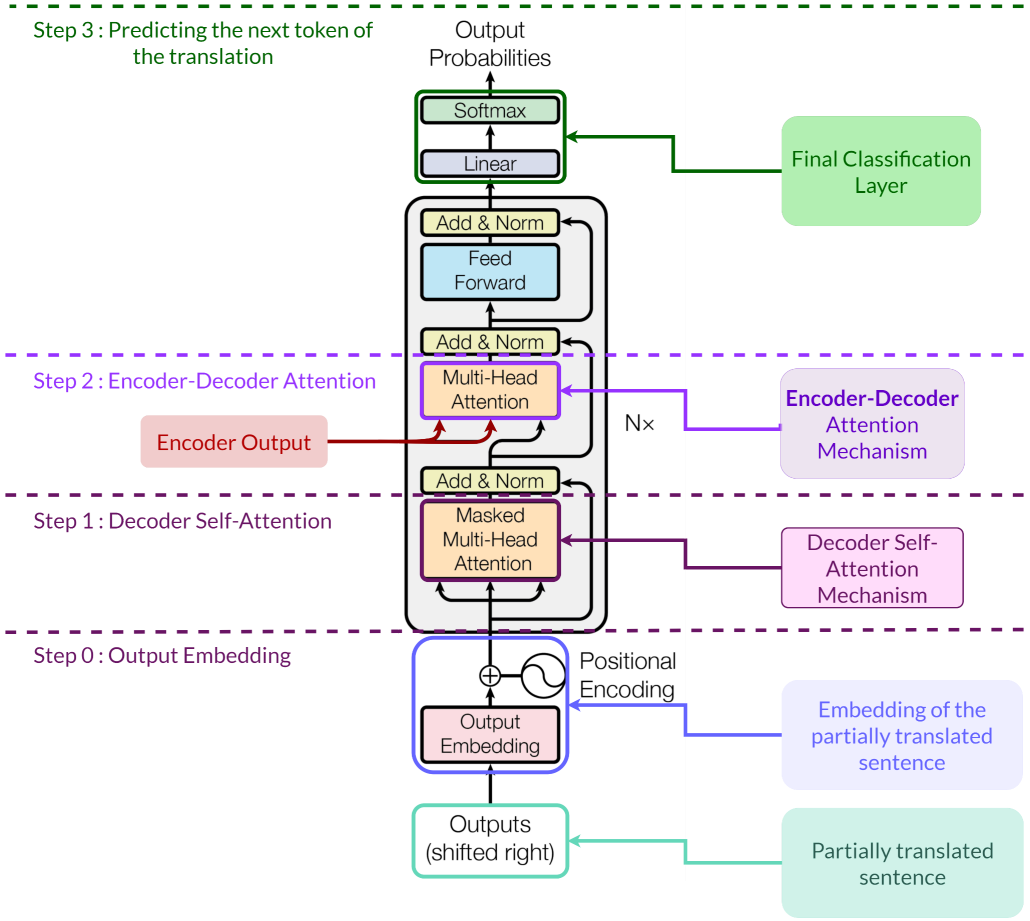


Fig: Decoder architecture

The transformer model has been trained on a french-english translation problem. To this end, we used:

- Padding and look-ahead masks
- An optimizer and a customized learning rate
- A customized training loop

For time efficiency reasons, this model is limited mainly because of:

- The limitation of the corpus to one fourth of its total length
- The size of the vocabulary used
- The corpus size

By adapting these parameters, one could improve the performances on a powerful machine between any two languages.

4. Applications and openings for further work

As you can imagine, there exist several promising openings for further work.
Aside from improving the performances of the translation system or of the speech recognizer, we can think in regards to two interesting suggestions.

A) Word Separation

This is about the idea of implementing a system that should learn the ability to separate voices in a recorded audio of a discussion in a party or in the street. This should be enabled by the frequency separation of the different voices and the subsequent application of a separate pre implemented speech recognizer model independently for each one of them.

B) Language Models

Speech recognition systems used to be represented by audio to phonemes conversion which are basically basic units of sounds. However the acoustic model on its own simply allows one to get a transcription (string of characters) from the audio.

The vocal recognition models deal with the CTC algorithm used to predict the letters instead of the words themselves (or the phonemes). To compensate for this drawback which alters the performances, language models, which are part of the post treatment techniques, were developed to capture texts or sentences with greater complexity.

This paper deals with two post treatment techniques, analyzes the specific example of text generation and treats the different fusion methods between the acoustic model and the post treatment models.
This article can be found in the appendix.

5. Conclusion

As presented in this thesis, the Speech recognition domain is a promising area of research for those who might be interested.
The ASR (Automatic Sound Recognition) field has been analyzed through the use of transformers and through the implementation and integration of a translation system to the connected glasses implemented by the VoiceGlass project.

There exist various areas of expertise that necessitate speech recognition performance improvements such as vocal assistants which are more and more present in nowadays society.

We mentioned some possibilities for opening this topic which are of course not an exhaustive list.

I personally plan on exploring these openings in further research and on trying to improve the performance of the so far implemented model.

Thanks a lot for your attention throughout this reading.

References

- End to end speech recognition
  - *Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin* [Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai...] http://proceedings.mlr.press/v48/amodei16.html
  - *Improved Noisy Student Training for Automatic Speech Recognition* [Daniel S. Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu and Quoc V. L] https://arxiv.org/pdf/2005.09629v2.pdf

- Word embedding and word translation articles
  - https://datascientest.com/nlp-word-translation
  - https://datascientest.com/le-word-embedding

- RNN and neural machine translation articles
  - https://towardsdatascience.com/using-rnns-for-machine-translation-11dded78ddf
  - https://datascientest.com/fonctionnement-des-reseaux-neurones
  - https://towardsdatascience.com/neural-machine-translation-15ecf6b0b

- Attention Mechanisms
  - *Effective Approaches to Attention-based Neural Machine Translation* [Minh-Thang Luong, Hieu Pham, Christopher D. Manning] https://arxiv.org/abs/1508.04025

  - *Neural Machine Translation by Jointly Learning to Align and Translate* [Szimitry Bahdanay, Jyunghyun Cho, Yoshua Bengio] https://arxiv.org/abs/1409.0473

- Transformers
  - *Attention Is All You Need* [Ashish Vaswani, Noam Shazeer...] https://arxiv.org/pdf/1706.03762.pdf

Developed content

Cheat Sheets:

- Keras (see appendix)
- Comparison of Machine Learning Algorithms (in process)
- Tableau (in process)

Research Papers and articles

- DenseNet: (see appendix)
  https://datascientest.com/reseaux-de-neurones-densenet
- Post treatment methods for speech Recognition (see appendix)

Videos

- Autoencoder: https://www.youtube.com/watch?v=XpiruMSG2FY
- Word Embedding and Translation systems:
  https://www.youtube.com/watch?v=jC4UnPf7oZo*
- Feature Scaling: (in process)

Appendix

Cheat Sheet - Keras

Modèles de bases et Preprocessing : Keras by DataScientest

Définition

Keras est une bibliothèque open source de Deep Learning permettant la génération de réseaux de neurones artificiels. Elle permet une expérimentation efficace de ces derniers de par sa simple utilisation et ses nombreuses possibilités de déploiement grâce à son intégration avec la librairie tensorflow.

Modèles

MLP

Binary Classification

```
from tensorflow.keras.layers import Dense
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Multi-Class Classification

```
from tensorflow.keras.layers import Dropout
model.add(Dense(512, activation='relu', input_dim=784))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))
```

Régression

```
model.add(Dense(64, activation='relu', input_dim=nb_features))
model.add(Dense(1))
```

CNN

```
from tensorflow.keras.layers import Activation,Conv2D,
MaxPooling2D,Flatten
model.add(Conv2D(32, (3,3), input_shape=(150,150,3)
activation='relu'))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
mocteladd(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
moctel.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Preprocessing

Une fois le jeu de données importé sous le format d'un array numpy, il peut être nécessaire de le préparer avant l'application du modèle.

Sequence Padding

```
from keras.preprocessing import sequence
X_train = sequence.pad_sequences(X_train, maxlen)
X_test = sequence.pad_sequences(X_test, maxlen)
```

Ensembles de test et d'entraînement

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size,
random_state)
```

Encodage des variables catégorielles
(One-hot-Encoding des variables cibles)

```
from keras.utils import to_categorical
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_train, num_classes)
```

Standardisation, Normalisation des variables

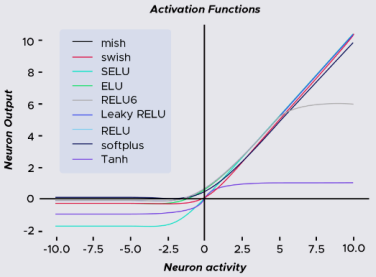
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
X_train_norm = scaler_transform(X_train)
X_test_norm = scaler_transform(X_test)
```

1

Fonctions d'activations

tf.keras.activations

- relu
- sigmoid
- softmax
- softplus
- softsign
- tanh
- seul
- elu
- exponantial



Exemples

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
data = np.random.random((1000,100))
labels = np.random.randint(2, size=(1000,1))
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics= ['accuracy'])
model.fit(data, labels, epochs = 10, batch_size=32)
predictions = model.predict(data)
```

Fonctions de perte

Pertes probabilistes

- binary_crossentropy
- categorical_crossentropy
- sparse_categorical_crossentropy
- poisson
- kl_divergence

Pertes de régression

- mean_squared_error
- mean_squared_absolute_error
- mean_squared_percentage_error
- mean_squared_logarithmic_error
- cosine_similarity
- huber
- log_cosh

Hinge Losses - Marge maximale

- hinge
- squared_hinge
- categorical_hinge

Métriques : évaluer les performances du modèle

Précision (fréquence des bonnes predictions)

- Accuracy
- BinaryAccuracy
- CategoricalAccuracy
- TopKCategoricalAccuracy
- SparseTopKCategoricalAccuracy

Métriques de régression

- MeanSquaredError
- RootMeanSquaredError
- MeanAbsoluteError
- MeanAbsolutePercentageError
- MeanSquaredlogarithmicError
- CosineSimilarity
- LogCoshError

Hinge Losses - Marge maximale

- AUC
- Precision
- Recall
- True Positives
- TrueNegatives
- FalsePositives
- FalseNegatives
- PrecisionAtRecall
- SensitivityAtSpecificity
- SpecificityAtSensitivity

Métriques probabilistes (calcul de la crossentropie)

- BinaryCrossentropy
- CategoricalCrossentropy
- SparseCategoricalCrossentropy
- KLDivregence
- Poisson

Métriques: hinge - 'maximum-margin'

- Hinge
- SquaredHinge
- CategoricalHinge

Métrique de segmentation d'image (IOU = true_positive J (true_positive + false_positive + false_negative)

- MeanIoU

Workflow and layers : Keras by DataScientest

Définir un modèle (define) → Compiler le modèle (compile) → Entraîner le modèle (fit) → Évaluer le modèle (evaluate) → Prédire (predict)

Entraînement

Entraîner le modèle sur jeu d'entrainement

```
model.fit(X_train, y_train, batch_size, epochs,
verbose, validation_data)
```

Évaluation

Déterminer la performance du modèle sur jeu de test

```
score = model.evaluate(X_test, y_test)
```

Prédiction

Prédire le jeu de test depuis le modèle entraîné

```
model.predict(X_test)
model.predict_classes(X_test)
```

Compilation

Une fois le modèle construit, préciser les paramètres d'entraînement

MLP : Classification binaire

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

MLP : Classification multiclasse

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=('accuracy'))
```

MLP : Régression

```
model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
```

Réseaux Récurrents

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Sauvegarder un modèle

```
model.save('model_file')
```

Charger un modèle

```
from keras.models
import load_model
model1 = load_model('model_file')
```

Les différentes couches

Les couches standards

Input()

Dense()

Reshape()

Flatten()

Les couches convolutionnelles

Conv1D/2D/3D()

UpSamping1D/2D/3D()

ZeroPadding1D/2D/3D()

Les couches récurrentes

SimpleRNN()

LSTM()

GRU()

Les couches de Pooling

MaxPooling1D/2D/3D()

AveragePooling1D/2D/3D()

Les couches de dropout

Dropout()

3

Les générateurs (la création du générateur de données avec Keras se fait en 2 étapes) :

Instancier un générateur

```
tf.keras.preprocessing.image.
ImageDataGenerator(featurewise_center=True,
featurewise_std_normalization=True,
rotation_range,
width_shift_range,
height_shift_range,
horizontal_flip=True,
validation_split)
```

Appliquer une des trois méthodes suivantes pour générer des lots de données augmentées

flow *: depuis des liste de données et de labels (array numpy) 1*
flow_from_directory *: depuis le chemin spécifie*
flow from dataframe *: depuis le dataframe*

Exemple

```
train_generator = train_datagen. flow_from_directory (
directory=r". /train/",
target-size= (224, 224),
color mode="rgb",
batch= size=32,
class_mode="categorical",
shuffle=True,
seed=42
)
```

Les callbacks

Sauvegarder les meilleurs poids du modèle au cours de l'entraînement

```
callbacks.ModelCheckpoint(filepath =
filepath, monitor = 'val_loss',
save_best_only = True,
save_weights_only = False,
mode ='min',
save freq = 'epoch')
```

Réduire automatiquement le learning rate

```
callbacks.ReduceIROnPlateau(monitor =
'val_loss', patience=5, factor=0.5,
verbose=2, mode='min')
```

Arrêter l'entrainement si le modèle n'évolue plus

```
callbacks.EarlyStopping(monitor =
'val_loss', patience = 8, mode = 'min',
restore_best_weights = True)
```

Transfer Learning

En quoi ça consiste ?

Le transfer learning consiste à passer des poids appris sur un problème vers un autre problème similaire. Il est généralement utile dans les tâches pour lesquelles le jeu de donnée d'entrainement est trop petit.

1. Instancier un modèle baseline et y charger les poids pré-entraînés

```
base_model = tf.keras.applications.Xception(
weights='imagenet',
input_shape=(150, 150, 3),
include_top=False)
```

2. Geler les couches de ce modèle

```
base_model.trainable = False
```

3. Créer un nouveau modèle par dessus la sortie des couches du modèle baseline

```
inputs = keras.Input(shape=(150, 150, 3))
x = base_model(inputs)
x = keras.layers.GlobalAveragePooling2D()(x)
outputs = keras.layers.Dense(1)(x)
model = keras.Model(inputs.outputs)
```

4. Entraîner le nouveau modèle sur le nouveau jeu de données. Cette phase consiste en la transformation des anciennes variables en prédiction du nouveau jeu de donnée.

```
model.compile(optimizer=keras.optimizers.Adam(),
loss = keras.losses.BinaryCrossentropy(from_logits = True),
metrics=[keras.metrics.BinaryAccuracy()])
model.fit(new_dataset, epochs= 20, callbacks = ... ,
validation_data = ... )
```

Research paper/ Article: DenseNet

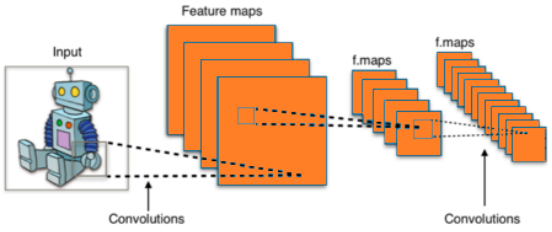
DenseNet

Vous avez déjà probablement entendu parler des réseaux convolutifs dans les contextes de reconnaissance d'image, de traitement vidéo ou encore dans le ciblage publicitaire et le traitement de langage naturel : NLP- *natural language processing* + lien article NLP . Cet article tente de vous apporter une première approche analytique et une compréhension de l'architecture de ces réseaux et plus particulièrement sur l'architecture des réseaux DenseNets.

I] Les réseaux de neurones convolutifs - CNN

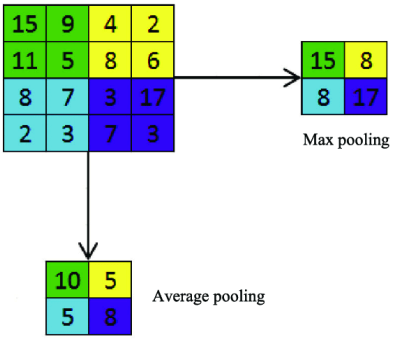
Prenons le cas de l'analyse d'image qui fait partie aujourd'hui d'une des grandes applications du machine learning. Chaque image est représentée par un très grand nombre de données et peut être visualisée comme un objet au sein d'un espace de très grande dimension.
Mais alors **comment exploiter les propriétés de ces images** de façon efficace compte tenu de la taille conséquentes des données qui les représentent?

Les réseaux de convolution apportent une solution efficace à ce problème: chaque couche intermédiaire du réseau est exposée à une portion de l'image appelée ***champ récepteur*** pour lequel plusieurs neurones effectuant différentes actions, calculent une **matrice de convolution** correspondante.



Mais alors chaque couche intermédiaire génère plusieurs matrices de convolution: tout cela laisse penser que nous étendons l'information plutôt que de la restreindre? Pas vraiment! Le partage de poids synaptique (ou *weight sharing*), définit le partage de poids entre les neurones d'une même couche. Ceci correspond à une **réduction considérable des paramètres** au sein de chaque couche. L'information à la sortie de ces couches est également restreinte grâce à une phase de **réduction de dimension** qui permet de rassembler les informations provenant de plusieurs neurones voisins appartenant au même canal, en une information commune au travers de l'ajout systématique d'une couche d'extraction en sortie des couches convolutives.
Il existe deux sortes de couche d'extraction:

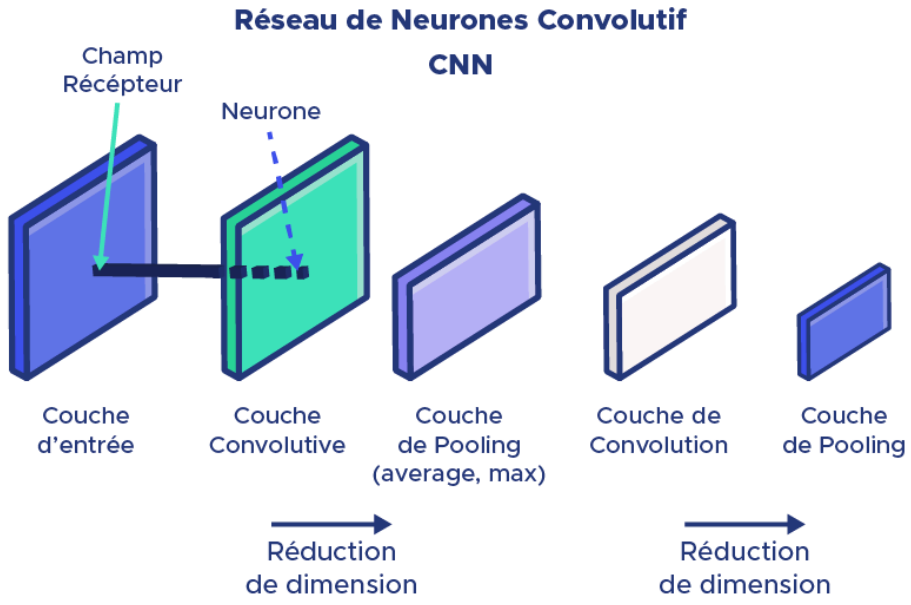
- Average pooling: qui calcule la valeur moyenne des informations



- Max pooling: qui ne retient que la valeur maximale des informations

et les utilise pour créer une *feature map* sous échantillonnée.

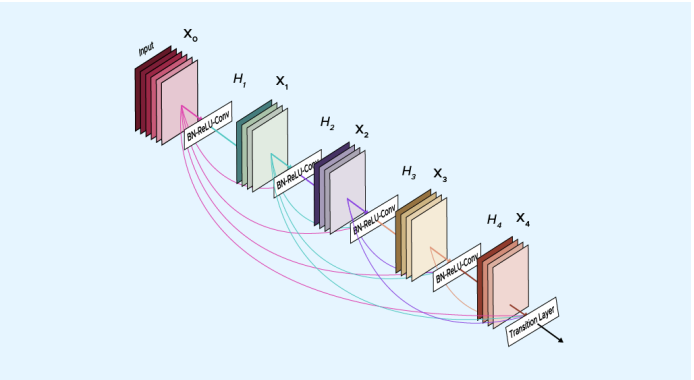
Nous obtenons donc en sortie de ces superpositions (couche concolutives et couche avg/max pooling), un vecteur de taille raisonnable et réduite représentant les informations extraites de l'image de façon certes, moins précise, mais beaucoup plus facilement exploitables.



II] DenseNet

Les réseaux DenseNet sont des réseaux de neurones convolutifs répondant à une certaine architecture communément appelée **réseaux de neurones densément connectés** (‘*densely connected convolutional network’* ou DenseNet).

Un challenge grandissant de ce type de réseau est l'**optimisation des performances**. Une solution naïve serait simplement d'empiler davantage de couches convolutives: un problème évident qui survient dès lors, est l'agrandissement de la profondeur du réseau. Lorsque l'on effectue la rétropropagation du gradient au travers des couches: on effectue en réalité une opération sur les dérivées partielles de chaque couche cachée du réseau et cela peut potentiellement compliquer les mises à jour des poids durant la phase d'entraînement.



La grande spécificité de cette architecture réside dans l'entrée des couches qui rassemble par concaténation toutes les entrées des couches précédentes (voir

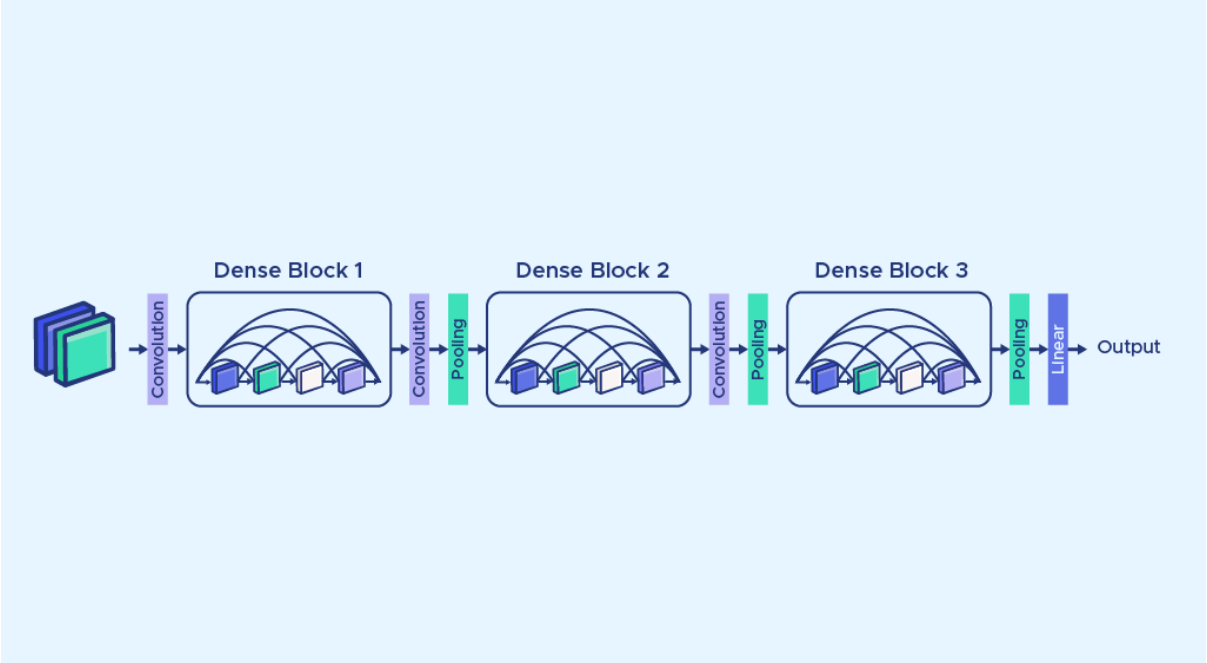
schéma): créant ainsi une *feature map,* tout en conservant une résolution spatiale identique: on parle de ***channel-wise concatenation***.

Ainsi, il devient inutile d'effectuer un processus de sélection d'information, contrairement à d'autres réseaux pour lesquels il est nécessaire de choisir quelles informations sont à transmettre aux couches suivantes.

https://www.google.com/search?q=densenet&rlz=1C1AVFC_enIL895IL895&sxsrf=ALeKk037se9dr4CEnysKxOngwPntOTG-ZQ:1624536987916&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjT0drwn7DxAhWG66QKHSu_B9cQ_AUoAXoECAEQAw&biw=1280&bih=577&dpr=1.5#imgrc=N5-2b2zADafhIM

Assemblage de plusieurs blocs Dense avec couches de transition

https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803



L'addition des *dense blocs* rend le réseau profond et peut parfois s'avérer problématique. Ceci peut être solutionné par la scission de ces blocs et par l'introduction de couches de convolution 1-by-1 qui conservent la résolution spatiale tout en réduisant la profondeur de la *feature map*.

Comme explicité dans la partie précédente, une couche de Pooling (MaxPool ou AvgPool) est souvent empilée à la sortie de ces couches dans le but de réduire la dimension de la *feature map* et de faciliter l'exploitation des données.

Il est important de noter que les feature map sont de même taille à l'intérieur de chaque bloc, ce qui rend la concatenation moins complexe.

Parmi les avantages du réseau DenseNet, nous pouvons mentionner la **rétropropagation facilitée du signal d'erreur** de manière directe. Nous pouvons également mettre en évidence l'**efficacité computationnelle de ce système** notamment grâce à la gestion du nombre de paramètres, DenseNet assure la diversité des features en comparaison avec un réseau convolutif traditionnel ainsi que leur faible complexité ce qui permet des performances optimales.

III] Conclusion

En conclusion, l'architecture de réseau DenseNet s'inscrit dans la famille des réseaux de neurones convolutifs et permet une optimisation des performances notamment dans les domaines de traitement d'images.
Sa structure singulière se compose d'un empilement de *dense block*, de couches de convolution et de pooling successivement. Les couches de Pooling (average ou max) permettent de garantir une restriction de la dimension des données ce qui justifie l'efficacité de traitement de ce type de réseau.
De la reconnaissance faciale jusqu'au traitement de langues naturelles, les applications du réseau convolutif DenseNet à l'AI en sont très larges.
Si vous voulez en savoir ces sujets d'applications, vous pouvez lire notre article sur le NLP et l'analyse de sentiments: https://datascientest.com/nlp-twitter-analyse-de-sentiment

Research paper/ Article: Post treatment methods for Speech Recognition

Post-treatment openings for speech recognition models.

I] Introduction: Speech recognition problem

For a long time, speech recognition models have aroused a great interest to the computer science community who aimed at developing algorithms and methods to enable the conversion from audio signals to text intelligible by the computer.

This domain is also known under the name of Automatic Speech Recognition (**ASR**) or Speech to Text (**STT**).

The speech recognition problem relies on the audio clip datas: that comes from the measurement of change in air pressure of the microphone against time.

The human ear has a physical structure that is able to measure the amount of intensity of different frequencies present in the audio. This step is encapsulated in what is called a spectrogram: this graph that displays the frequencies against time by taking into account the amount of energy/ intensity is commonly used as a preprocessing step of the audio.

Speech recognition systems used to be represented by audio to phonemes conversion which are basically basic units of sounds.

However the acoustic model on its own, simply allows one to get a transcription (string of characters) from the audio.

The vocal recognition models deal with the CTC algorithm used to predict the letters instead of the words themselves (or the phonemes).
To compensate for this drawback which alters the performances, language models, which are part of the post treatment techniques, were developed to capture texts or sentences with greater complexity.

We will first talk about two post treatment techniques and analyze the specific example of text generation.
We will then observe the different fusion methods between the acoustic model and the post treatment models.

II] Post treatment techniques

We call post treatment techniques, the methods that **train on raw text**, and that can be later on **fusioned** with the vocal recognition model to **enhance the performances**. In this context, we will present the language models (**LM**) and the spelling correction models (**SC**).

1) Language models **(LM)**

a) The role of the language models

Speech recognition models are in most cases fusioned with language models. These systems help the speech recognition model to establish the **probability** of a word sequence, without being influenced by the acoustic characteristics.

$$Y^* = \underset{Y}{\operatorname{argmax}} \; p(Y \mid X) \cdot p(Y)$$

$p(Y \mid X)$: The CTC conditional probability.

$p(Y)$: The language model probability.

In other words, when the speech recognizer faces a situation in which it is confronted with **similar** sentences (that sound similar): the language models provide a probability for each of them and allow the system to make the **right decision**.
For example, let's picture a situation in which the speech recognizer receives a sentence and hesitates between the two following interpretations:
1."There is a hole in my jeans."
2. "There is a whole in my jeans."

By just relying on the acoustics, these two sentences are very similar. However, by taking a second look at their **meanings**: it seems pretty clear that the first option is more likely to be the right one rather than the other one: and this can be guessed by the **context** of the sentence.

In such a case, the speech recognizer on its own will not be able to make the appropriate choice between the two. That is the reason why the language model evaluates the **likelihood** of each sentence and helps the speech recognizer to make the right guess.

b) Working process

Model definition

The language model relies on the corpus made out of audio recordings (speech samples) and is based on both basic and more complex algorithms that are supposed to handle the different language patterns complexity. It is not unusual to also use transfer learning techniques to enhance the system performances. The language model can rely on either supervised or unsupervised learning.

We denote two main types of language models:

- **HMMs (Hidden Markov Models):** these statistical models are used due to the pseudo stationarity of a speech signal approximation when windowing it to a short time scale. Moreover, HMMs have the undeniable benefit of being easy to use and to train.

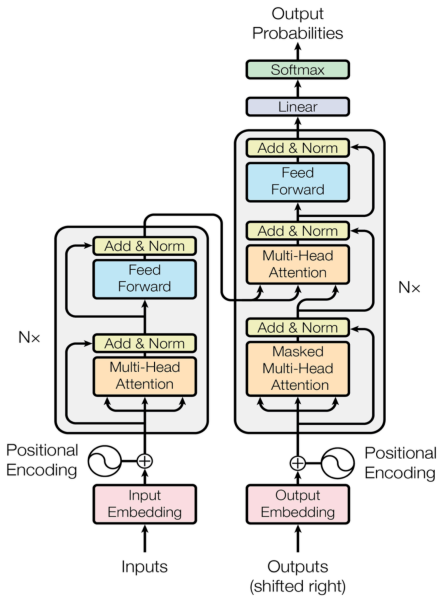


Figure 1: The Transformer - model architecture.

- **Artificial Neural Networks:** These deep learning models appeared in the 80's and compensate for the HMMs drawbacks in terms of **flexibility**.These layered systems are widely used also due to their training speeds. We can especially mention the transformer network which is shown in the following picture.

Training phase

Once the model is selected, the training phase can start: the model is fed with an audio sample and the system generates out of it a machine-encoded text that will be compared to ground truth transcripts to evaluate the error level of our model. This error provides indication on the likelihood of this word to be selected later on by our model. In the same logic, a greater error will penalize the word under analysis and decrease its chance to be guessed by the system.
The WER (Word Error Rate) enables the tuning of the next iteration of the training process.

Testing

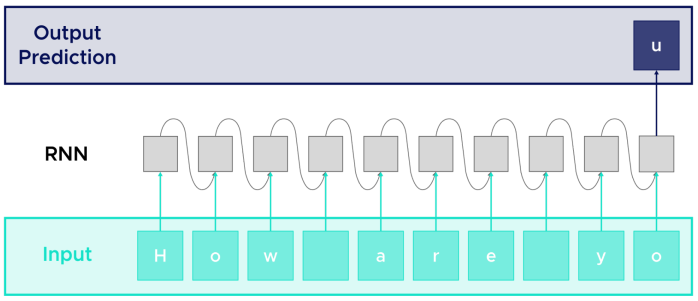
This language model can then be incorporated to several systems such as voice assistance or any other system that requires a speech recognizer. It allows to considerably enhance its accuracy by being used in inference problems. Inference refers to the conversion of input sequences (phonemes) into outputs (right words inferred from the model probabilities).

c) Text generation

The text generation problem is a practical example for the language model. It is about developing an algorithm that tries to **automatically complete a word** that a user is typing.

A first approach to this problem would be to predict the next character with respect to the ones the user already typed.

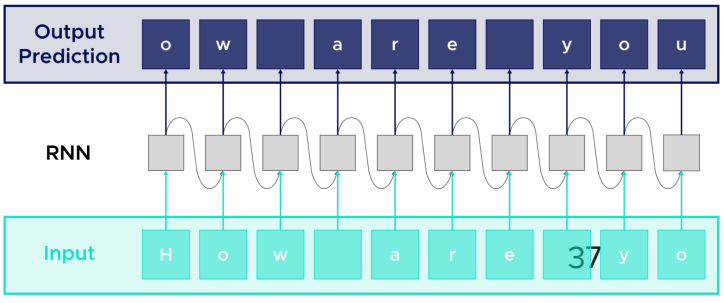
For example, if the user wanted to write the word "Hello" and that he only wrote "Hel", the algorithm should be able to predict an "l", which is the next character in the word, then the "o" and to notify that he actually finished the word prediction.



The hidden states roles are to store the information contained in the sequence history.

This problem can be solved through the implementation of a recurrent neural network that aims at predicting the **next character** by taking into account the previous ones.

A second and more complex approach tries to predict the next character **for every single first word** character as is described in the following picture:



2) Spelling correction models

The language model is trained only on audio- transcription pairs and for this reason, can lead to poor performance on rare words.
For this reason, the language model often **includes** a spelling correction model that tries to take into consideration the **characteristic error distribution made by the model.** The goal of the spelling correction to model is to correct these specific errors by training on **text-only data.**

The performance amelioration can be noticed by the decrease in the WER over a baseline model.
You can learn more about spelling correction models in the following article:
https://arxiv.org/pdf/1902.07178.pdf

III] Fusion methods

In conventional ASR models, the acoustic model, pronunciation model and language model are learned separately and folded together to form a single end-to-end system.
Fusion methods are integration techniques that aim at integrating a language model in a speech recognition model. Amongst them, we denote the *shallow, deep and cold* fusions that differ in the type and time of integrations.

1) Model integration

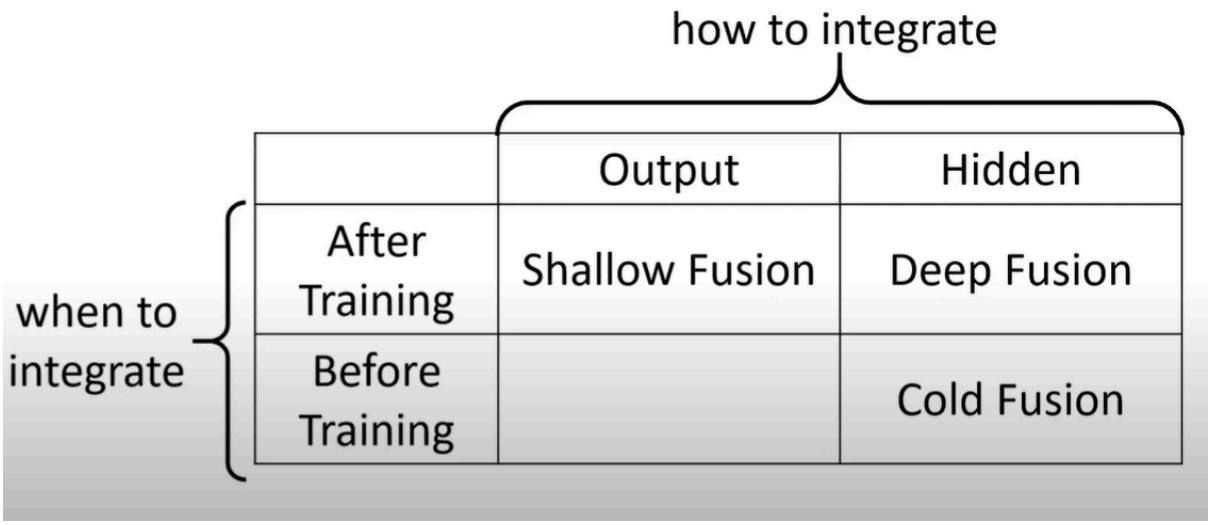
This criteria defines the point at which the Language model will be integrated in the whole ASR model computation.
For the cold and deep fusions, the hidden states of the language and the ASR model are combined and we can thus talk about a tight integration or integration on the hidden states. Whereas the shallow fusion, only the scores of the two models are combined and do not merge into a single model: thus, this type of integration happens on the outputs.

2) Training integration

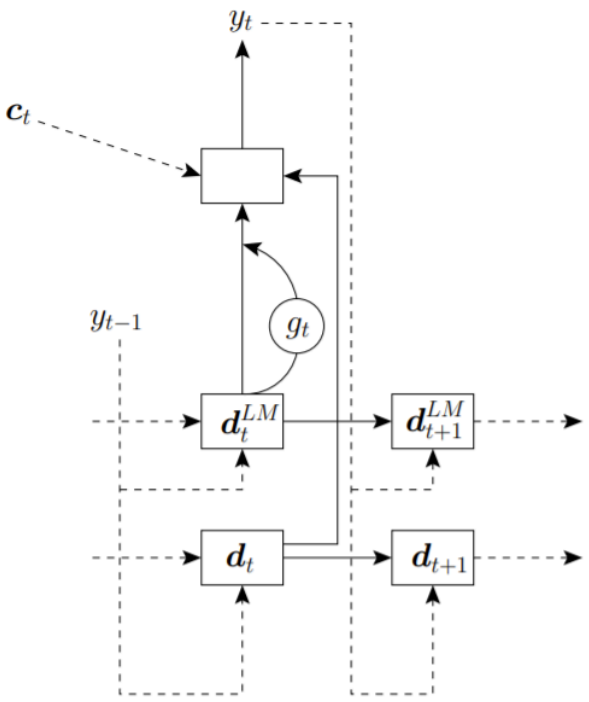
The training integration process focuses on the point at which the language model will be integrated in the ASR training phase.
The deep and shallow fusions, the ASR and language models are trained separately and then combined: this is called late training integration or integration after training.
In contrast, the cold fusion integrates the pretrained language model before the start of the ASR model training phase. Note that this option can be significantly costlier.

3) Global picture

The following table sums up the situation in which each fusion can be used.



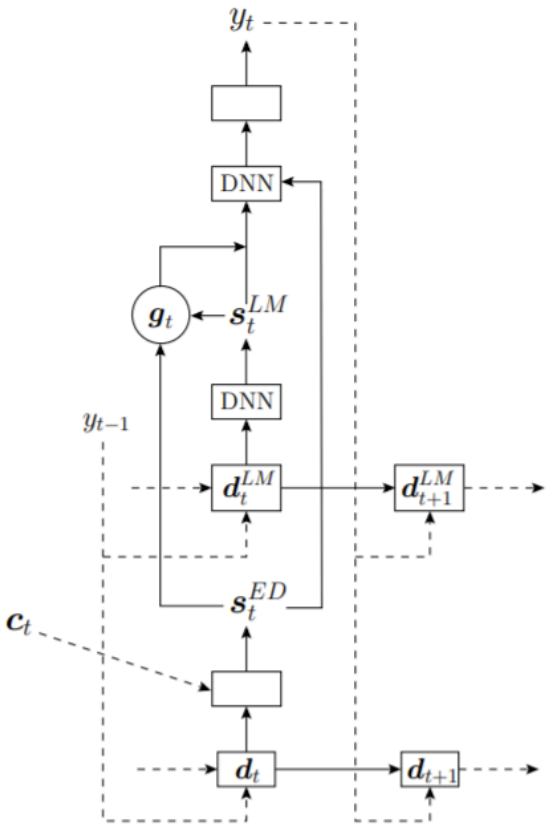
And in these schemes, you can visualize a simple single step of each fusion type.



This scheme represents a deep fusion single decoding step.

- The current hidden state of the language model d_t^{LM} is used to compute the next one d_{t+1}^{LM}
- The current LM hidden state and is fed to a gate function g_t that applies a sigmoid function
- The current hidden state of the decoder d_t is used to compute the next one d_{t+1}
- The current hidden states from the LM and the decoder hidden states are fused together with the context vector (from the attention mechanism) c_t to form y_t

Fig. 1: Illustration of a single decoding step of deep fusion.



This illustration depicts a cold fusion single decoding step.

You can notice that:

- It also integrates the LM model based on its hidden states
- However, in contrast to the deep fusion, the cold fusion requires an early integration: in other words, an integration before training.
- The LM is fused before the encoder decoder model ED starts its training
- Both LM and encoder decoder scores (s_t^{LM}, s_t^{ED}) are fed to a gate function

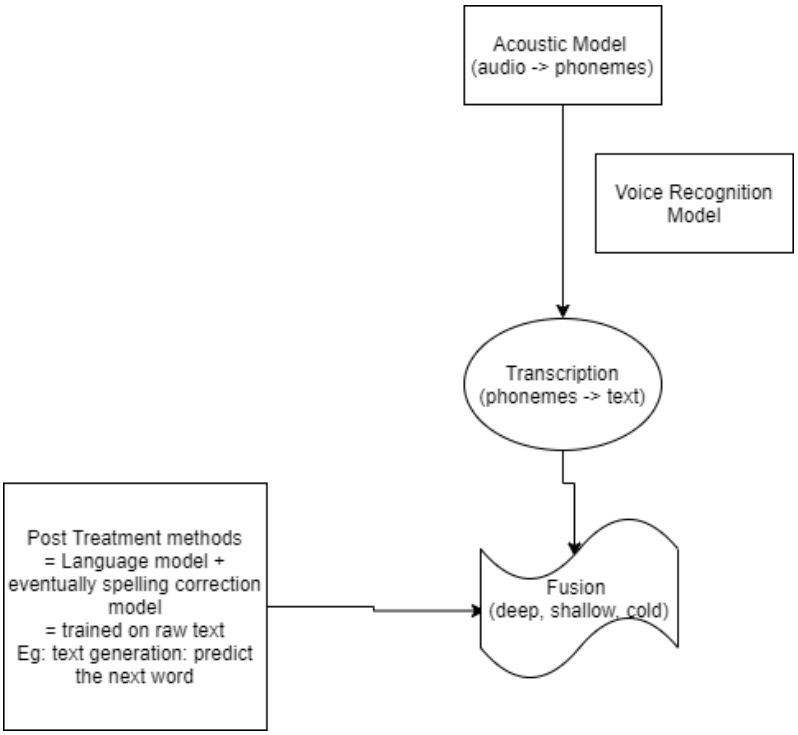
Fig. 2: Illustration of a single decoding step of cold fusion.

You can read more on their integrations to speech recognition systems in the following paper:
[1807.10857] A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition (arxiv.org)

IV] Conclusion

In conclusion, the global picture of the end-to-end speech recognition model includes the training of 3 models:

- an acoustic model: that converts the audio in transcription
- a language model: that is trained on transcribed audio-text pairs
- an alignment mechanism (single neural network)



The vocal recognition system converts the audio phoneme into a transcription (string of characters).

This transcription can be later on fusioned with systems built on a language model.
eg: a text generation model, to enhance the predictions.
That is an example in which the language models can be very important in the working process and in the intelligibility of speech recognition systems results.

Their implementation as well as their learning type provide a complexity to the whole system that allows for an estimation of the probability of a word to be correct in a specific sentence context.

Thus, they direct the decision of the speech recognizer towards a correct guess and enhance the system's overall performances.

It can be interesting to note that language modeling is also used in a variety of Natural Language Processing (NLP) applications such as document classifications.