


NON-PLAGIARISM UNDERTAKING

I, (full name of the undersigned) **Arthur Bartoli**

declare being fully aware that plagiarism by copying documents or a portion of a document published in all forms and media, including online publications, constitutes copyright infringement and related rights, as well as outright fraud.

Consequently, I hereby undertake to quote all sources and authors that I've used to write my internship report and its appendices.

Date: **11/09/2020**

Signature: 

Rapport de stage

Arthur Bartoli

Sommaire

1. Introduction (p. 3)
2. Alten (p. 4)
3. Étude (p. 4)
 - a. Contexte et objectif du stage (p. 4)
 - b. Nature R&D du stage (p. 5)
 - c. Analyse des contributions scientifiques (p. 10)
 - d. Démarche scientifique (p. 10)
 - i. Extraction du pdf et annotation du texte (p. 10)
 - ii. Identification des phrases contenant des exigences (p. 12)
 - iii. Extraction des triplets sujet/nombre cardinal/unité (p. 13)
 1. Transformers (p. 14)
 2. Du NER au MRC (p.15)
 3. Le terme d'interaction BiDAF (p.17)
 - iv. Tests et prototype (p. 18)
4. Conclusion (p. 19)
5. Bibliographie (p.20)

1. Introduction

Dans le cadre de ma formation à l'AMSE en tant que Data j'ai décidé de faire mon stage d'année dans l'entreprise Alten Sud-Ouest par le biais d'une annonce sur le site Indeed.com. Leur annonce proposait un stage autour du thème du Natural Language Processing.

Après avoir postulé j'ai eu un entretien avec la responsable commerciale Solène Cressant au cours duquel elle m'a expliqué les objectifs en cours de l'entreprise. Cet entretien a eu lieu avant la crise du Covid 19 et par conséquent l'entreprise était en pleine expansion, notamment au travers de son partenariat avec Airbus Toulouse.

Ce stage aurait dû servir d'introduction à l'entreprise au terme de laquelle j'aurais été embauché en CDI, sauf imprévu ou malentendu majeur.

À l'issue de cet entretien par visio-conférence, j'ai été contacté en Février pour venir dans les bureaux de l'entreprise passer un entretien d'embauche. C'est mon tuteur de stage, Sebastien Ricciardi, qui m'a accueilli et m'a fait passer cet entretien qui a tourné autour de questions de statistiques, de probabilités et des définitions de concepts tels que les réseaux de neurone, des procédures standards de NLP ou bien d'outils de développement.

Dans ce rapport de stage, nous allons dans un premier temps développer les activités de l'entreprise Alten puis expliciter le déroulement du stage et ses problématiques.

2. Alten

Alten est une entreprise multinationale française de conseil en technologies et de service numérique fondée en 1988 par Simou Azoulay, son actuel PDG et actionnaire majoritaire. Elle est implantée dans vingt-cinq pays et emploie 37200 salariés dont 32550 ingénieurs.

Ses services consistent à structurer la problématique d'entreprises et de leur louer les services d'ingénieurs durant une période de un jour à plusieurs années.

Ses employés techniques n'étant pas attribués à une mission sont en "intercontrat". Lors de cette période ils peuvent consacrer leur temps à d'autres projets en cours qui nécessite une aide, former un autre employé ou bien participer à un projet de R&D au terme duquel ils doivent rendre un bilan scientifique et technique (BST) rendant compte de leurs recherches.

L'entreprise est divisée en plusieurs pôles géographiques et spécifiques dont celui qui m'a employé, le pôle d'innovation Sud-Ouest situé à Toulouse. Ces pôles sont dirigés par des responsables administratifs, notamment les responsables commerciaux chargés des relations avec les clients et des embauches, comme dans mon cas.

De plus, l'activité de ses pôles est initiée et gérée par des responsables d'innovation et des chefs de projet.

3. Etude

a. Contexte et objectif du stage

L'industrie utilise régulièrement des fichiers pdf de taille très conséquente (de plus de 1000 pages) dans le but de définir des normes de production. Le but de ce stage était d'extraire ces normes et de les articuler dans une base graph.

Nous explorions dans un premier temps l'extraction des normes **et** leur articulation dans une base graph. Cependant ma tâche sur ce projet était de trouver des solutions à la première partie du problème et en même temps de former un autre employé en reconversion, Ludovic Caclin, aux technologies utilisées (réseau de neurone, python, concepts de nlp, etc...).

Ce stage a été effectué entièrement en télétravail pour des raisons sanitaires.

L'enjeu était de permettre une lecture automatique du fichier de normes d'une production et la détermination des exigences supplémentaires de production que ces normes impliquent à l'aide de la base graph.

C'est une amélioration significative du temps de développement d'un projet de production car il n'est pas nécessaire de faire expertiser les exigences de production (par exemple pour détecter deux exigences en réalité contradictoires).

L'objectif technique de l'opération d'extraction était de donner à l'algorithme un fichier pdf en entrée décrivant les exigences d'une production. La sortie de l'algorithme serait un fichier .csv contenant le sujet d'une exigence, la quantification de cette exigence et son unité.

Le fichier sur lequel nous avons fait notre étude est le fichier de norme CS-25 Large Aeroplanes.

Par exemple :

Les ailes de cet avion doivent faire 5 mètres chacune. —> ailes de cet avion / 5 / mètres

Nous avons utilisé deux méthodologies pour pouvoir accomplir cet objectif :

- Une méthode d'analyse de dépendance par le biais de Stanza permettant de déterminer les phrases contenant une exigence et d'isoler le sujet, la quantification de l'exigence et son unité issue de la thèse Natural Language Processing of Requirements for Model-Based Product Design with ENOVIA CATIA V6 (Pinquie et al., 2016)
- Une méthode d'analyse de dépendance par le biais de Stanza permettant de déterminer les phrases contenant une exigence ainsi qu'un modèle de reconnaissance d'entité par le biais d'un réseau de neurone en utilisant la technologie PyTorch.

b. Nature R&D du stage

Les principales difficultés concernant ce sujet sont liées à son domaine scientifique, la Natural Language Comprehension (NLC) qui est un problème de Data Science visant à extraire des données précises d'un texte écrit en langage naturel (humain).

Dans un premier temps il est nécessaire de nettoyer le texte et le découper en phrases. Ce problème est aisément résolu lorsqu'il s'agit d'articles, de courts textes ou bien de paragraphes concernant un sujet précis. Malheureusement, les documents de normes sont des fichiers structurés comportant de nombreux noms de section et allusions à des noms de normes tels que "**CS 36.8.3**". Ces termes sont interprétés par l'algorithme soit comme des phrases à part entière (ce qui les rend relativement

faciles à isoler et à écarter), soit comme faisant partie d'une phrase (ce qui complique la compréhension de la phrase par l'algorithme Stanza, qui l'interprète alors comme un nombre) ou soit comme de multiples phrases ("CS 36.6.3" —> "CS 36."/"8.3") ce qui les rend difficiles à identifier et à soustraire de l'analyse des dépendances.

Les algorithmes découpeurs de phrases essayés sont celui natif à Stanza et NLTK.

Dans un second temps, l'algorithme Stanza est parfois très inexact. En effet, les nombres, noms, verbes ne sont pas interprétés comme tels ou bien l'analyse de dépendance ne les lie pas entre eux. Le coeur de l'algorithme de l'analyse de dépendance repose sur l'étude des liens syntaxiques ou sémantiques entre les mots pour pouvoir en dériver la structure, par conséquent ces erreurs d'analyse bloquent l'algorithme. Ce dernier ne finit que par détecter la présence de normes chez un nombre infime de phrases (826 dans un texte de normes de 1000 pages) et ne parvient qu'en de rares cas à les extraire.

Exemples parmi les phrases détectées comme contenant des normes dans le document CS-25 Big Aeroplanes :

- "figure a 3 1 primary field view primary flight information information whose presentation required cs 25 1303 b cs 25 133 b arranged cs 25 1321 b" —> presentation | 25 | cs
- "however flight take thrust based maximum take thrust specified basic afm may used showing compliance landing approach climb requirements cs 25 119 25 121 provided availability take thrust upon demand confirmed using thrust verification checks specified paragraph 5 e" —> flight take thrust based maximum take thrust specified basic afm | 5 | without unit
- "functional elements used alerting information functions warning caution alerts must provide timely attention getting cues resulting immediate flight crew awareness least two different senses cs 25 1322 c 2" —> elements used | two | senses
- "conditions cs 25 519 require consideration aeroplane moore jacked condition wind speeds 120 km h 65 knots" —> conditions | 120 | km

La méthode que nous utilisons est trop sensible aux performances de Stanza et aux découpeurs de phrase alors que les documents que nous traitons sont trop imprévisibles dans leur formatage et difficiles à nettoyer.

Notre recherche s'est donc principalement portée sur une méthodologie à l'analyse plus souple, le nombre de cas de figure à prendre en compte pour chaque fichier de norme semblant beaucoup trop grand (noms de sections à éliminer, multiples sujets/exigences par phrase, mauvaise syntaxe,

mauvaise mise en page, noms d'unités inconnus...). De plus, la méthode exposée dans le papier Natural Language Processing of Requirements for Model-Based Product Design with ENOVIA CATIA V6 (Pinquie et al., 2016) avait été bien implémentée et son principal défaut résidait dans la technologie utilisée, nous n'avons donc pas cherché à l'améliorer.

Nous avons découpé le problème en deux :

- La détection de phrases contenant des normes
- L'extraction des normes de ces phrases sous la forme d'un triplet SUJET/NOMBRE/UNITE

Nous avons alors cherché à garder la méthode originale de détection de phrases contenant des normes pour nous concentrer sur l'extraction des normes dans des phrases déjà identifiées. Le but étant de concentrer la recherche mais aussi de pouvoir d'abord donner des résultats tels que nous les souhaitions à l'équipe chargée de la base de données en graph et de faciliter leur travail.

L'état de l'art nous a orienté vers des modèles de question/réponse propres au NLC. Le principe était de proposer une phrase à l'algorithme et de lui demander sous forme de question d'extraire chacun des éléments du triplet de la phrase.

Exemple :

- "Chaque aile doit faire 5 mètres de long."/"Quel est le sujet dans la phrase ?" → "aile"
- "Chaque aile doit faire 5 mètres de long."/"Quel est l'unité liée au sujet dans la phrase ?" → "mètres"
- "Chaque aile doit faire 5 mètres de long."/"Quel est le nombre lié à mètres dans la phrase ?" → "5"
- On obtiendrait le triplet aile/5/mètres que nous voulons.

Nous avons identifié 3 problèmes non-résolus par l'état de l'art :

- Capacité à identifier les informations recherchées dans le texte

Au final nous pouvons ramener cela à un problème de Named Entity Recognition (NER), qui consiste à "étiqueter" des mots dans une phrase.

- Capacité à identifier chaque élément du triplet recherché

Les questions posées doivent être précises ou doivent être assez bien comprises pour que chacune nous permette d'extraire un élément différent du triplet.

- Disponibilité des données d'entraînement

Les données d'entraînement destinées à l'identification de chaque élément du triplet

n'existent pas telles quelles. D'ailleurs, elles n'existent pas non plus pour l'identification des phrases contenant des normes.

	Capacité à identifier les informations dans le texte	Capacité à distinguer chaque élément du triplet	Données d'entraînement
Analyse de dépendance¹	Pour les raisons que nous avons exposées, l'algorithme original est très peu performant et ne parvient pas à repérer ni les phrases contenant une exigence, ni le triplet recherché.	L'algorithme ne parvient pas à détecter le sujet de l'exigence et les paramètres qui lui sont liés.	Cet algorithme ne nécessite pas de données d'entraînement car il se repose sur des réseaux de neurone pré-entraînés à cette tâche (Stanza, NLTK).
BiDAF² Modèle de question-réponse	Les tests nous montrent que le modèle est capable de repérer les éléments du triplet dans des phrases simples.	Cependant, il est rarement capable de distinguer le nombre cardinal des unités. Dans l'exemple donné ci-dessus, les réponses aux deux dernières questions seraient "5 mètres" à deux reprises.	Ce modèle s'entraîne sur SQUAD 1.0, un dataset classique des problèmes de NLC
Retroreader³ Modèle de question-réponse basé sur un affinement itératif de la réponse.	Les résultats sont très proches de ceux de BiDAF.	Les résultats sont très proches de ceux de BiDAF mais le modèle est capable de détecter l'absence de réponse (utile si paramètre sans unité)	Ce modèle s'entraîne sur SQUAD 2.0, un dataset classique des problèmes de NLC comportant des cas (~50%) où la réponse

			de ne se trouve pas dans le texte proposé.
Transformers⁴ Voir BERT ou ERNIE (entraînement sur la base d'un graph de connaissance, utile dans notre cas)	Cette architecture de réseau de neurone permet une représentation mathématique des relations complexes, contextuelles et grammaticales entre les mots par ré-interprétation successive de la matrice encodant une phrase.	-	-
Few-shots Learners⁵ Voir GPT2, ou tout récemment GPT3	-	-	Cette architecture de modèle Transformer permet du <i>few-shots learning</i> et donc d'entraîner le réseau de neurone à une tâche spécifique de NLP avec peu de données

c. Analyse des contributions scientifiques, techniques et technologiques

Nous avons mené nos recherches au travers de la technologie de l'apprentissage automatisé. Notre hypothèse était qu'un modèle plus robuste aux données était nécessaire et possible pour résoudre notre problème. De plus, l'algorithme original utilisant la technologie Stanza n'offrait pas les performances attendues.

Elles se sont effectuées par l'apprentissage de différentes technologies et la lecture de différents papiers et du code associé afin de trouver la meilleure solution à notre problème. Une fois une solution identifiée, elle était étudiée, comparée aux solutions que nous avons puis implémentées.

Il n'existe pas à ce jour et à notre connaissance d'algorithme permettant d'extraire plusieurs entités distinctes dans un très long document en langage naturel. Il s'agit d'un problème inédit du fait de la complexité du texte, de la précision requise des résultats et des progrès récents en data science qui permettent l'analyse de phrases complexes tant avec Stanza qu'avec des technologies Transformers. Ce domaine est nouveau, vaste et très dynamique. De plus, de nombreux papiers de recherche ne sont pas ou mal implémentés ce qui rend l'essai de certaines technologies impossibles.

L'architecture consiste en un programme qui prend en entrée un fichier au format pdf et donne en sortie un fichier .csv contenant les triplets sujet/nombre cardinal/unité du fichier.

La structure d'algorithme consiste en :

- Un algorithme qui extrait le texte du pdf, le nettoie, puis le divise en phrases.
- Un algorithme qui identifie parmi ces phrases celles qui contiennent une exigence.
- Un algorithme qui extrait les triplets sujet/nombre/unité de ces exigences.

d. Démarche scientifique et travaux réalisés

i. Extraction du pdf et annotation du texte

Nous avons très tôt dans le projet décidé d'abandonner le langage Java pour développer afin de passer à Python. Cette décision a été prise car la majorité des personnes travaillant sur le projet sont plus portées en Python qu'en Java et car la majorité des outils de NLP et de Data Science sont avant tout disponibles sur Python avant d'être disponible en Java.

La décision originale de développer sous Java venait du fait que la librairie Stanza est elle-même développée en Java. Cependant, nous avons préféré nous tourner rapidement vers Python afin de ne pas avoir à réécrire l'ensemble du projet si aucune autre option ne se présente à nous lors du développement du projet. De plus, la librairie Stanza inclut un environnement python (par-dessus son implémentation Java).

Nous avons utilisé le module PyPDF2 afin d'extraire le texte du fichier pdf dans la fonction *read_pdf* dans le fichier *FileFunctionsAndParser.py*.

Si le paramètre *load* est vrai, alors la fonction charge le fichier .txt dont le nom est précisé dans le paramètre *save_file*.

Si non, le module crée un objet pdf à partir du chemin d'accès du fichier et concatène son contenu par page. Ce résultat est stocké dans un fichier .txt sous le nom *save_file* si il est précisé en paramètre.

Dans tous les cas le texte complet est renvoyé en sortie.

Le texte devait ensuite être séparé (parsée) en phrases puis découpé en tokens afin de pouvoir être analysé par Stanza.

Ce découpage en phrase est assuré par le module nltk car son algorithme était rapide et considéré comme performant. De plus, séparer les phrases par un double retour à la ligne dans une fichier .txt permettait à l'algorithme de paralléliser l'analyse des dépendances. Nous avons également considéré une simple séparation par point (qui a été peu performante, notamment par rapport au chapitrage du document que nous avons mentionné auparavant) et celle native à Stanza qui donnait un résultat similaire à nltk sans permettre de paralléliser l'algorithme par phrase.

Avant de transmettre le texte à l'algorithme Stanza, nous avons tenu à nettoyer le texte en développant les contractions, éliminant les caractères spéciaux inutiles (', ", #, etc...) et en appliquant des expressions régulières communément utilisées lors du pre-processing de textes. L'option d'éliminer les stopwords est ouverte mais n'a pas montré de variations significatives de résultat.

À ce niveau-ci de l'algorithme, le résultat est sauvegardé dans *data/parsed_sent.txt*. Si le paramètre *load* est vrai et le chemin de *data/parsed_sent.txt* précisé, la fonction chargera le résultat précédent.

La chaîne de caractère parsée, formatée et nettoyée est transmise à Stanza qui en crée un objet *doc*. Cet objet contient les phrases brutes, tokenisée et lemmatisées, le Part Of Speech Tag (POS-tag) de chaque token ainsi que l'annotation nécessaire à l'analyse des dépendances de chaque phrases. Cet objet est ensuite sauvegardé à l'aide de la librairie pickle sous le protocole 4 et sous le nom de *documentNLP.pkl*. Il est à noter que pour un pdf de 1000 pages, ce pickle peut faire jusqu'à 250mo et peut prendre de 3 à 5h à générer sur une machine standard (bi-core 2.7Ghz SSD et 8go de RAM) **avec parallélisation**. C'est donc un calcul long dont le résultat est lourd. Il est à noter que cet algorithme se base sur une technologie de réseau de neurones et donc une carte graphique pourra grandement accélérer le processus. Ainsi il nous était difficile d'élaborer les tests sur le document original et nous n'avons pu faire fonctionner cet algorithme que quelques fois, des tests limités sur quelques pages pouvant nous laisser sans résultat.

Nous avons demandé à utiliser un cluster haute performance afin d'avoir une puissance de calcul suffisante. Malheureusement, ce projet était déjà en cours de réalisation depuis un temps et lors de la rédaction de ce rapport, son élaboration est toujours en cours.

À ce stade du projet nous avons donc un objet "document Stanza" contenant le texte annoté et lemmatisé et le code d'analyse des dépendances venait d'être traduit en python.

ii. Identification des phrases contenant des exigences

La prochaine étape est de classer les phrases selon qu'elles contiennent des exigences à analyser ou non. Seules phrases qui sont classifiées comme telles sont transmises à l'analyse de dépendance.

Nous n'avons pas fait évoluer la méthode de Pinquie qui consiste à vérifier pour chaque phrase :

- * la présence d'un **terme prescriptif** (*shall, must, should, have to, require, need, want, expect, wish* ou *desire*).

- * D'un **domaine de définition**

- * D'une **condition**

Par exemple : The Control_Subsystem **shall** open the Inlet_Valve in **less than 3 seconds**, **when the temperature of water in the Boiler is less than 85 °C**.

Le domaine et la condition étant très complexes à détecter, et dans un but de test, nous n'avons retenu comme critère que la présence d'un terme prescriptif. En effet, la détection du domaine et de la condition était a priori aussi complexe que l'extraction du triplet. Par conséquent, proposer une solution à l'extraction du triplet pouvait offrir une voie vers la détection de ces critères.

Malgré tout, la simple détection du terme prescriptif ne permettait que la détection de 800 phrases sur un document de 1000 pages.

Manquant d'alternatives si ce n'est l'apprentissage supervisé, nous avons cherché à améliorer directement l'extraction des paramètres de chaque exigence plutôt que leur détection.

À ce moment-là du projet, la problématique du manque de données d'exemple ou d'entraînement s'est posée plus concrètement. À moins d'improviser de nouvelles règles de détection des différentes phrases, pour autant toujours conditionnelles aux performances de Stanza, toutes les autres solutions auxquelles nous pensions nécessitaient un grand nombre de données d'entraînement pour pouvoir classer correctement ces phrases.

iii. Extraction des triplets sujet/nombre cardinal/unité

La plus grande partie de notre étude était centrée autour de l'extraction de ce triplet sujet/nombre cardinal/unité.

Tout d'abord, nous avons analysé les résultats de l'analyse de dépendance afin de repérer des problèmes inhérents à l'analyse.

- Nombres cardinaux : Les noms de sections rattachés par erreur à une phrase seront la majorité du temps pris pour des nombres cardinaux.
De plus, lorsque la phrase en contient deux, l'algorithme choisira le premier rencontré, ce qui peut être amélioré en choisissant celui ayant une relation de dépendance avec une unité. Cependant, ce problème est minoritaire et un changement de l'algorithme n'a pas eu d'impact significatif sur le résultat.
- Unités : Les unités sont extraites en remontant la relation de dépendance "nummod" du nombre à l'unité. Malheureusement, sa détection dépend donc de la détection correcte du nombre cardinal.
- Sujet : Le sujet est extrait en remontant les relations de dépendance du terme prescriptif et en repérant ceux dont le pos-tag est sujet, un sujet passif ou la dépendance nominale d'un autre nom.

Nous avons alors cherché une solution qui nous permettait d'extraire un maximum d'information textuelles et précises avec un minimum d'entraînement. Nous nous sommes alors tournés vers des modèles de NLP dédiés à la compréhension de texte.

En effet, il était nécessaire de développer un modèle qui était capable d'intégrer une notion de nuance assez important pour faire la différence entre un paramètre et son unité (5/mètres par exemple) mais également de faire preuve de contextualisation afin d'avoir l'intuition qu'il s'agit d'un document technique voire aéronautique sans entraînement préalable.

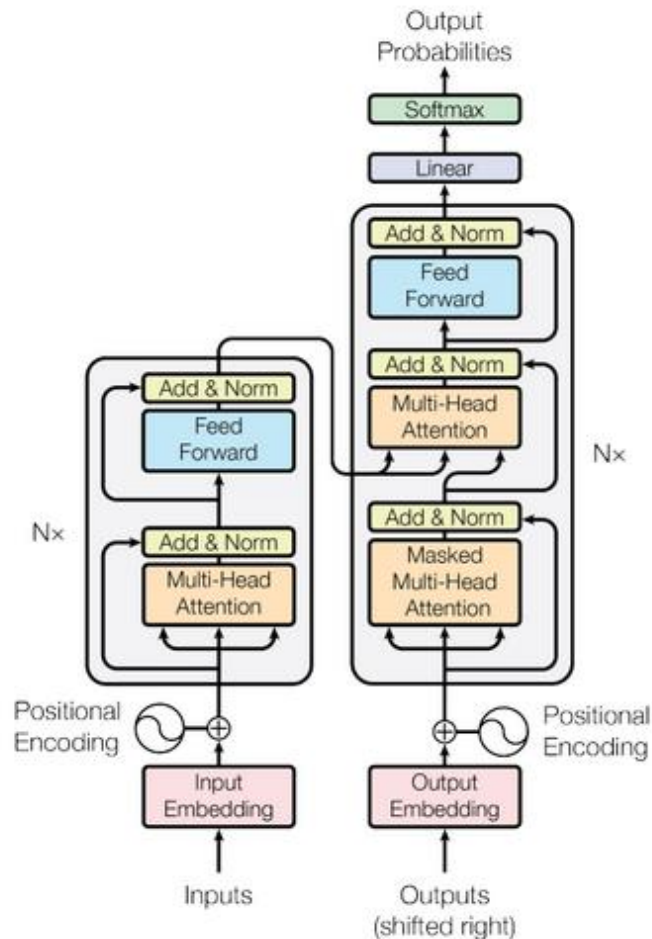
1. Transformers

Nous nous sommes alors tournés vers le champ d'étude NLP du *Named Entity Recognition* (NER) ou reconnaissance d'entité nommée qui consiste à extraire d'une phrase des mots correspondant à une entité précise.

Exemple : Extraire le sujet de la phrase "Le ciel est nuageux aujourd'hui" → "Le ciel"

La plupart de ces modèles se basent sur la technologie *Transformers*. Un transformer est un modèle entraîné à prédire un mot masqué dans une phrase à l'aide des autres mots, c'est donc un modèle

de langage. Ces modèles peuvent être intégrés pré-entraînés dans un modèle puis ré-entraîné afin d'affiner ses paramètres à une tâche précise (traduction, question-réponse, summarisation..). C'est l'étape de *fine-tuning* qui nécessite en général peu d'exemples.



Son architecture se base sur une ré-interprétation successive de la matrice représentant la phrase (**input embedding** sur ce schéma).

Un *embedding* ou “plongement” est une représentation mathématique dans un espace à grande dimension d’un objet ou structure.

Soit X une matrice de vocabulaire de dimension (T, J) telle que $X_{t,j} = 1$ si le mot j d’une phrase correspond au mot t .

Soit $X = U \Sigma V^T$ la SVD de la matrice X et E l’embedding de dimension d :

$$E = U_{1:d} (\Sigma^{\alpha}_{1:d, 1:d}) \text{ où } \alpha \in [0 ; 1]$$

À chaque re-interprétation (qu’on appelle couche d’attention, ici **Multi-Head Attention** et **Add & Norm**) le modèle découvre de nouvelles caractéristiques au texte en combinant la représentation mathématique de plusieurs mots pour ré-encoder chacun d’entre eux. Ce schéma ne comporte

qu'une seule couche de ré-interprétation.

Nous recommandons ce [lien](#) afin d'avoir une bonne compréhension des différents composants du modèle.

Une fois les couches d'encodage passées, cette représentation est habituellement décodée (partie de droite sur le schéma) afin de donner en sortie les uns après les autres le mot correspondant (le mot masqué lors de l'entraînement ou bien une traduction lors du fine-tuning).

La sortie de cette couche d'encodage est une matrice dont chaque colonne est la représentation d'un mot, une phrase est représentée par une matrice.

Ce fonctionnement est assez similaire dans le principe à un réseau convolutionnel re-combinant différentes parties d'une image pour la décrypter.

L'immense avantage d'un transformer est d'avoir accès à une représentation mathématique très fidèle d'une phrase qui prend compte du contexte, de la grammaire, du champ lexical et de la position de chaque mot.

De plus, la partie encodage est parallélisable. Or, comme nous le verrons par la suite, seule cette partie nous intéresse lors de notre implémentation.

2. Du NER au MRC

Cependant, les modèles les plus élaborés de NER se reposent sur des bases de données (payantes comme ACE 2004, ACE 2005, GENIA et KBP2017) dédiées à la reconnaissance d'entités particulières comme des verbes ou des sujets. Nous cherchions une solution nous permettant de retrouver une entité que nous aurions défini, nous avons alors cherché des solutions plus flexibles.

Nous nous sommes tournés vers une architecture de MRC (Machine-Reading Comprehension, à base de questions-réponses) pour résoudre un problème de NER⁶.

Le principe est d'utiliser la représentation mathématique de la phrase et de la question en sortie du transformer pour créer un **terme d'interaction**.

Ce terme d'interaction est une matrice dont chaque colonne est la représentation mathématique de chaque mot du texte conditionnellement à la question. Chaque colonne est utilisée pour classer chaque mot de la phrase en une réponse à la question posée ou non. Si la question est sous la forme "Dans cette phrase, quel est [nom de l'entité] ?" et comporte un certain nombre de synonyme du nom de l'entité recherchée, la réponse sera l'entité recherchée. Une autre méthode serait de nommer l'entité selon l'intitulé de sa page Wikipédia, celle-ci ayant certainement été utilisée dans l'entraînement.

Pour des questions de simplicité, la partie longueur (span) de la réponse n'a pas été implémentée.

La dernière couche du modèle est un triple classifieur binaire :

- Si chaque mot est un début de réponse, en prenant en entrée le terme d'interaction correspondant à ce mot.
- Si chaque mot est une fin de réponse, en prenant en entrée le terme d'interaction correspondant à ce mot.
- Enfin, quelle fin de réponse correspond à quel début de réponse, en prenant en entrée le terme d'interaction correspondant au mot de début et au mot de fin de réponse.

Si plusieurs fins de réponse sont associées à plusieurs débuts de réponse, la plus pertinente (plus haute probabilité ou score) est choisie.

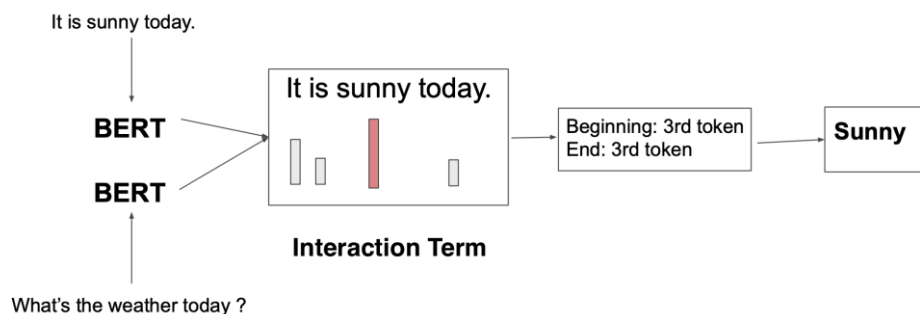
Les fins sans début, les fins placées avant les débuts et inversement ne sont pas comptabilisés.

Une absence de réponse peut être correcte, tout particulièrement dans le dataset SQUAD 2.0.

D'ailleurs, et comme mentionné plus haut, plusieurs datasets consacrés au NER (et utilisés dans le papier) sont payants et difficiles d'accès. Nous avons préféré utiliser un jeu de donnée très généraliste et produit par Stanford, le SQUAD 2.0 (Stanford QUESion Answering Dataset).

Ce dataset comporte 36 thèmes d'une vingtaine de textes. Chaque texte est lié à une douzaine de questions, une moitié avec plusieurs réponses possibles dans le texte et l'autre sans réponses.

En convertissant la réponse en langage naturel du dataset en indice de début et de fin de la réponse, on le transforme en un problème de classification.



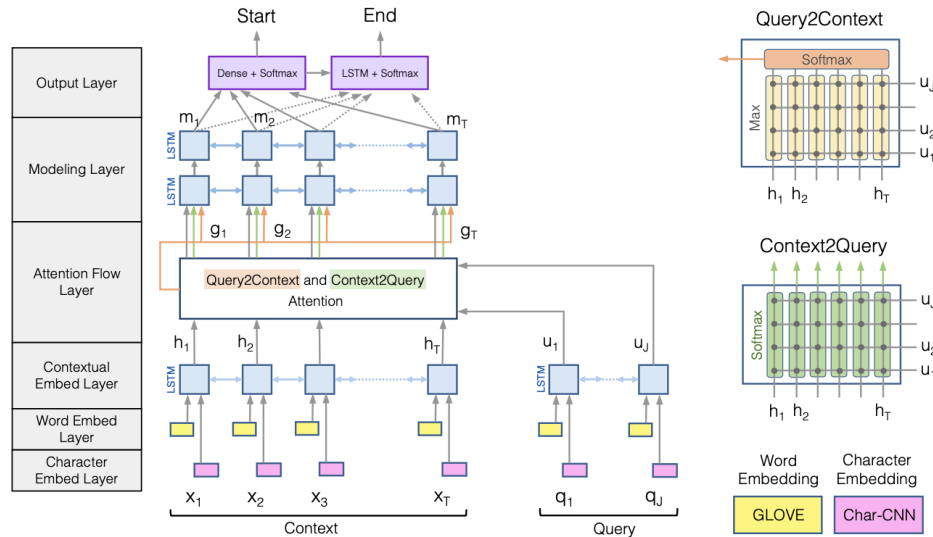
La publication utilise le modèle Transformer BERT, le modèle original de Transformer tel que conçu par Google Mind et en libre accès sur internet. Il s'agit d'un modèle de langage, et par conséquent un *few-shots learners*⁵. Malheureusement, c'est le premier de son type et son entraînement est trop réduit pour qu'un *fine-tuning* sur quelques exemples suffise.

Beaucoup d'autres modèles plus évolués/entraînés de ce type existent, tels que GPT2 qui est développé par OpenAI (auteurs de [Language Models Are Few-Shots Learners](#)) et que nous

recommandons. En effet, il est en libre accès et il est désormais établi que c'est un des modèles de langage les plus versatiles et les plus efficaces pour des tâches de NLP.

3. Le terme d'interaction BiDAF

Nous avons décidé d'utiliser le terme d'interaction BiDAF² pour sa simplicité (pratiquement uniquement des calculs d'algèbre linéaire) et ses bonnes performances.



Voici l'architecture du modèle MRC proposée dans la publication présentant BiDAF.

Malheureusement, plusieurs de ces composants sont obsolètes et ont été amélioré dans le modèle que nous proposons :

- Les trois couches inférieures d'embedding (Contextual Embed Layer, Word Embed Layer et Character Embed Layer) sont remplacées par un Transformer.
En effet tout type de Transformer est plus rapide et plus performant que deux embeddings différents traités par un réseau LSTM (lourd en calcul et peu performant)
- Les deux couches supérieures de classification (Modeling Layer et Output Layer) ont été remplacées par les trois classifications binaires mentionnées auparavant.

Deux composants du terme d'interaction sont calculés à partir de la matrice S tel que $S_{tj} = \alpha(H_{:t}, U_{:j})$ où α est une fonction à apprendre calculant un score de similarité entre les deux vecteurs représentant un mot t du texte et un mot j de la question resp.

Dans le papier et dans notre implémentation, $\alpha(h, u) = w_{(s)}^T [h; u; h \circ u]$, où $w_{(s)}$ est un vecteur à apprendre, \circ est une multiplication d'élément à élément et $[\cdot]$ est une concaténation de vecteurs.

On calcule alors :

- Query2Context, est une représentation des mots du texte les plus similaires à au moins un mot de la question.

Pour un mot t du texte,

$$Q2C_t = \text{sum}(b_t H_{:,t}) \text{ où } b = \text{softmax}(\max_{col}(S))$$

- Context2Query, est une représentation des mots de la question partageant un contexte avec chacun des mots de la question.

Pour un mot t du texte et un mot j de la question,

$$C2Q_{:t} = \text{sum}_j(a_{tj} U_{:j}) \text{ où } a_t = \text{softmax}(S_{t,:})$$

Finalement, on a $G = [h, C2Q, h \circ C2Q, h \circ Q2C]$

iv. Tests et prototype

Notre prototype a été codé à l'aide du Framework *pytorch*. Ce framework est développé par Facebook AI Research Lab et comporte deux avantages au projet :

- Un réseau de neurone est construit en héritant d'une classe abstraite "nn.Module" et son fonctionnement est défini par du code Python et non pas par un graph comme avec le framework Tensorflow.
Le graph servant à la propagation du gradient est généré en même temps que les tenseurs Pytorch sont manipulés dans le code. Cette implémentation est faite en C, un langage plus bas niveau que Python, donc plus rapide, et ne présente pas de ralentissement de l'algorithme.
Cela rend la conception du modèle plus lisible et facile.
- Une meilleure flexibilité par rapport au framework Keras (qui est lui-même une interface à Tensorflow) qui nécessite de définir les unes après les autres les couches du réseau.
De plus, la création de couches personnalisées est peu développée ou revient à utiliser Tensorflow

Malheureusement, l'implémentation a souffert de bugs à l'apprentissage qui poussaient le réseau à attribuer un poids identique à chaque mot. Sans apprentissage cependant, le modèle donne en moyenne un poids plus élevé au(x) mot(s) correspondant à une réponse.

Nous estimons donc que le prototype est en bonne voie et que son problème d'apprentissage pourra être résolu.

Il est à noter que le texte et la vraie réponse lors de l'apprentissage sont passés dans le tokenizer utilisé par le réseau de neurone pour s'assurer qu'il s'agisse des mêmes tokens.

Deux autres modèles à avoir été testés sont RetroReader et le modèle original de BiDAF. Tous deux ont donné des résultats similaires et parviennent à isoler chaque entité dans la phrase mais ne parvient pas à séparer le paramètre des unités.

Par exemple : Dans la phrase *“Les ailes doivent faire 5 mètres”* il trouverait le triplet *“Ailes/5 mètres/5 mètres”*.

Ce résultat pourrait être affiné à l’avenir.

4. Conclusion

En conclusion, ce projet a rencontré plusieurs obstacles lors de sa conception qui ont dû être contournés par la recherche de l’état de l’art.

Tout d’abord, le format du document crée de la confusion lors de l’étape d’annotation de l’algorithme Stanza; or il ne s’agit pas d’un format bien établi qu’il est possible de s’approprier. Il varie très certainement d’un document technique à l’autre, est inhérent à ce type de document (noms de section, plan, vocabulaire technique...) et crée de nombreux problèmes lors de l’analyse du texte.

Nous nous sommes alors tournés vers une solution plus robuste aux données : les technologies de réseau de neurone dans le cadre du NER.

Cependant, par manque de données d’entraînement autour des entités à rechercher (le triplet sujet/paramètre/unité de mesure) nous avons cherché à ramener le problème à un problème de question réponse pour lesquels les jeux de données généralistes existent (le dataset SQUAD 2.0).

De plus, l’utilisation de modèles Transformers plus évolués que BERT pourrait réduire la charge d’entraînement, voire nous permettre de nous passer de datasets généralistes et créer le nôtre.

Malgré son contexte unique ce stage m’a apporté une profonde compréhension des technologies de pointe de traitement du langage naturel. En effet, malgré ma formation en data science, ce champ est si vaste et dynamique que se pencher attentivement sur le simple traitement du langage a largement rempli la durée de mon stage.

Aussi, la tâche de formation qui m’a été confiée a été très formatrice par l’effort d’explication de mes connaissances (qui les a consolidées), le partage de sources d’apprentissage et de pistes de réflexion avec Ludovic, la confrontation de nos solutions et les nombreuses discussions sur le milieu professionnel des nouvelles technologies.

5. Bibliographie

- ¹ – Natural Language Processing of Requirements for Model-Based Product Design with ENOVIA CATIA V6, Pinquie, Veron, Segonds, Croue, 2015
- ² – Bi-Directional Attention Flow, Seo, Kembhavi, Farhad, Hajishirzi, 2018
- ³ – Retrospective Reader for Machine Reading Comprehension, Zhang, Yang, Zhao, 2020
- ⁴ – Attention is all you need, Vaswani et al., 2017
- ⁵ – Language models are few-shots learners, Brown et al., 2020
- ⁶ – A Unified MRC Framework for Entity Recognition, Li et al., 2020