Extraction automatique et visualisation de termes de maintenance pour le peuplement d'ontologies métiers

Meryl Bothua Laurent Pierre
EDF Lab Paris-Saclay,7 bd Gaspard Monge, 91120 PALAISEAU, France
meryl.bothua@edf.fr, laurent.pierre@edf.fr

RÉSUMÉ

EDF a développé une chaîne logicielle destinée à extraire et analyser l'information contenue dans les rapports de maintenance. Les expériences menées avec Word2Vec nous ont permis de créer des ressources lexicales à partir de ces documents. Grâce à celles-ci, nous peuplons des ontologies et visualisons les résultats à l'aide d'un outil de navigation dans des graphes RDF. Nous avons mis en place un processus semi-automatique de correction et de validation des termes lexicaux destinés à peupler l'ontologie métier finale.

ABSTRACT

Automated Extraction and Visualization of Terms Related to Maintenance in Order to Populate Business Ontologies

EDF is developing a process dedicated to extract and analyze information from maintenance reports. Experiments with Word2Vec were conducted for creating lexical resources from these sets of documents. We populated ontologies from the Word2Vec outputs in order to visualize and check the accuracy of the results, thanks to our ontology driven RDF graph UI. We describe the semiautomation of the processing line dedicated to correction and validation of lexical terms intended to populate the final business ontology.

MOTS-CLÉS : similarité entre mots, plongement de mots, visualisation de données, recherche d'informations, peuplement d'ontologie.

KEYWORDS: word similarity, word embedding, data visualization, information retrieval, ontology population.

1 Introduction

EDF est aujourd'hui au coeur de la transition numérique et mène de nombreuses études sur l'exploitation de données diverses. Dans cette optique, des méthodes de fouille de texte sont testées afin d'exploiter les données non structurées. Une chaîne de traitement a été mise en place pour extraire et analyser des informations à partir de rapports de maintenance. Des tests ont permis de mettre en évidence l'apport de Word2Vec (Mikolov *et al.*, 2013) pour l'aide à la constitution de ressources lexicales. L'automatisation du processus met en exergue des éléments auparavant noyés dans la masse des données. Le gain est double : une économie de temps est réalisée grâce à la proposition de termes candidats au peuplement de lexiques ; nous produisons actuellement une sortie RDF avec une ontologie dédiée et proposons une visualisation sous forme de graphe avec l'outil SemVue. Pour chaque terme du corpus, des candidats sont donnés après prétraitements. Le gain est également

qualitatif : des synonymes, des abréviations, des possibles fautes d'orthographe et des phénomènes de multilinguisme sont retournés par le système. Nous développons actuellement une application web, CuriosiText, pour l'aide au peuplement d'ontologie par des utilisateurs non experts.

2 Contexte de l'étude

Une chaîne de traitement de fouille de texte a été mise en place en 2017 à l'aide de la plateforme GATE (*General Architecture for Text Engineering*). Cette chaîne a pour objectif d'extraire automatiquement des actions de maintenance réalisées sur des composants (exemple : `remplacement de l'anémomètre`). Dans un premier temps et afin de démontrer la faisabilité de notre cas d'étude, les gazetteers (ou dictionnaires) de cette chaîne ont été développés manuellement. Des experts les ont constitués en pointant les termes pertinents d'une partie du corpus des comptes rendus de maintenance. En vue du passage à l'échelle, nous avons cherché à automatiser cette étape coûteuse en temps ; nous avons ainsi adopté la méthode Word2Vec pour extraire automatiquement les termes similaires sur l'ensemble du corpus et augmenter le rappel de la chaîne de traitement.

3 Extraction de termes avec Word2Vec

3.1 Présentation des données

Nous avons entraîné un modèle Word2Vec sur un corpus de comptes rendus de maintenance. Ce corpus contient environ 8000 fiches de maintenance. Si la taille de ce corpus semble restreinte, nous avons néanmoins obtenu des résultats concluants (fig 2), les textes étant très répétitifs. En effet, la taille de notre dictionnaire est d'environ 5000 mots après prétraitements. Nous avons également calculé l'indice de richesse lexicale (Mckee *et al.*, 2000), quotient entre le nombre d'occurrences d'un token et le nombre total de tokens du texte. Un token est défini comme formé exclusivement de caractères alphabétiques, sans distinguer majuscules et minuscules. Notre corpus comprend 97 065 tokens dont 5887 tokens uniques donnant une richesse lexicale de 0.0607 avant prétraitements.

3.2 Algorithme

Nous avons mis en place un script python qui effectue plusieurs opérations. Il est ainsi possible de charger un corpus local ou de récupérer des données externes avec Elasticsearch [1]. Un ensemble de prétraitements est ensuite appliqué sur les données à savoir la suppression des caractères non désirés, le découpage en phrases et en mots. Concernant ces deux dernières étapes, nous avons utilisé la fonction dédiée du package NLTK. Nous sommes également partis de l'hypothèse que l'apprentissage de Word2Vec serait plus fin avec l'ajout de métadonnées grammaticales sur les mots de notre corpus. Nous avons donc intégré TreeTagger (Schmid, 1994) à nos traitements pour un étiquetage morphosyntaxique : Word2Vec apprend ainsi sur la forme, le lemme et la catégorie grammaticale du terme. Nous considérons qu'il s'agit d'une première étape de désambiguïsation. À la suite de TreeTagger, les mots vides sont supprimés et nous appelons Word2Vec pour le calcul de la

1. www.elastic.co

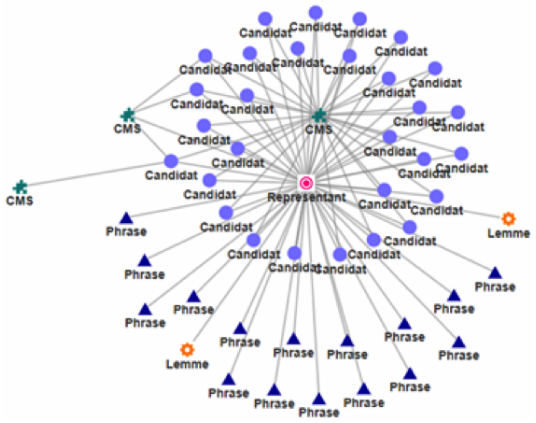

FIGURE 1 – Modélisation des résultats Word2Vec

similarité. Une sortie contenant les résultats de Word2Vec au format RDF est alors générée. Cette dernière étape permet de créer une base de connaissance sur les résultats du traitement des documents du corpus contenant les phrases et mots qu'elles contiennent et pour chaque mot du corpus : son label, sa fréquence, sa catégorie morphosyntaxique, son lemme, les 15 premières phrases dans lesquelles il apparaît et les 15 candidats termes les plus similaires (fig **??**).

4 Exploration de termes sous forme de graphe RDF

Afin de visualiser les résultats de Word2Vec formatés en RDF, nous avons créé une première ontologie modélisée sur Word2Vec. Elle sert de premier palier en vue du peuplement d'ontologies métiers. Nous produisons une visualisation sous forme de graphe où pour chaque mot du corpus, considéré comme étant de type `Representant`, nous affichons ses quinze termes candidats (classe `Candidat`) les plus similaires. Pour visualiser ces résultats, nous avons utilisé l'outil SemVue développé à EDF. Cet outil propose une interface de navigation dans un graphe RDF extrêmement facile à mettre en œuvre à chaque étape du développement d'un entrepôt de données. Le cœur de l'interface est défini à l'aide d'une ontologie associée à la modélisation métier proprement dite dans un entrepôt de données (Motik *et al.*, 2014) et permet ainsi à l'interface web de lancer des requêtes SPARQL. La stratégie d'affichage de SemVue est définie à l'aide d'axiomes OWL, ceux-ci configurant dynamiquement le sous-graphe à exposer à l'utilisateur. Grâce à cette visualisation nous avons d'optimisé de manière itérative le paramétrage de Word2Vec et validé qualitativement nos résultats (fig 2).

5 Evaluation

Nous avons réalisé une analyse qualitative en considérant les termes présents dans le lexique métier constitué manuellement. L'apport de Word2Vec a ainsi pu être démontré, certains termes ayant été omis par l'annotateur. Nous avons ainsi détecté automatiquement plusieurs éléments

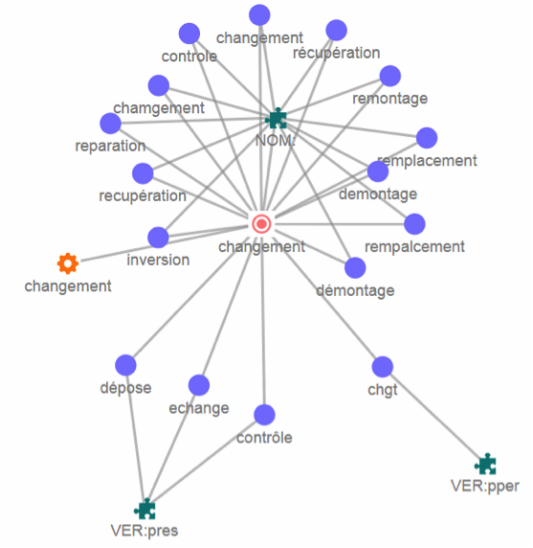

FIGURE 2 – Graphe sur le mot "changement"

pertinents à savoir : des pluriels (ex : `moteur` / `moteurs`), des abréviations (ex : `transfo` / `transformateur`), des mots collés (`bloc` / `dublock`), des fautes de frappe (ex : `changement` / `hamgement`), des hyponymes (pour `équipements` on retrouve `modem`, `serveur`), des mots du même radical (ex : `ventilateur` / `ventilation`), des synonymes (ex : pour `remplacement` on retrouve `changement`, `échange`) et des phénomènes de multilinguisme (ex `contrôle` / `cheik` / `check`). Il nous reste à présent à effectuer une analyse quantitative pour établir les taux de rappel et de précision en injectant ces lexiques à notre chaîne de traitement. Cette étape pourra être facilitée par nos outils d'exploration de termes et de peuplement, respectivement SemVue et CuriosiText.

6 Peuplement semi-automatique d'ontologies métiers

Bien que Word2Vec fournisse des termes similaires pertinents, du bruit est toujours présent. Aussi il n'est pas possible pour l'heure d'automatiser complètement le processus. Nous avons donc développé une application web, CuriosiText, permettant à un utilisateur non expert de visualiser les résultats de Word2Vec et de peupler semi-automatiquement une ontologie métier. Celle-ci est indépendante de l'architecture de l'application qui s'adapte au cas d'usage étudié, sans dépendance à un domaine spécifique. Dans un premier temps, l'utilisateur est invité à charger un corpus texte local ou peut effectuer une requête à l'aide d'Elasticsearch pour récupérer des données distantes. Il peut ensuite appliquer Word2Vec sur ses données. Les paramètres de base de la méthode sont paramétrables à savoir : la dimension des vecteurs, la taille de la fenêtre, la fréquence minimale d'apparition d'un terme pour l'apprentissage, le modèle (CBOW ou Skip-Gram) et le nombre d'itérations. L'utilisateur accède ensuite à la troisième partie de l'application à savoir le peuplement d'une ontologie. Il doit charger l'ontologie métier qu'il souhaite peupler et qu'il a modélisée en amont (fig 3). Pour ce

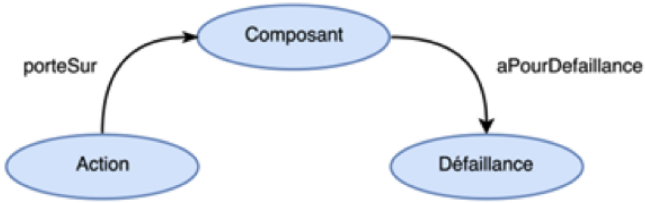

FIGURE 3 – Ontologie métier

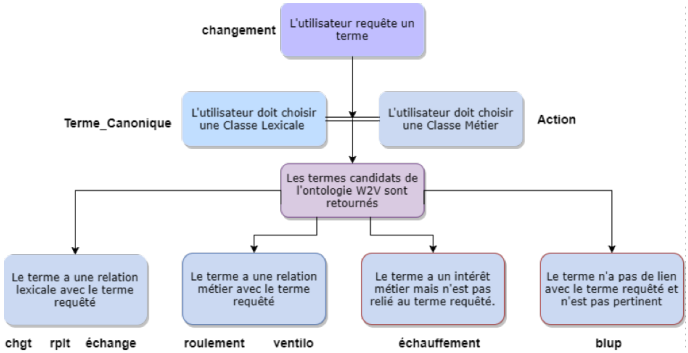

FIGURE 4 – Scénario de peuplement d'ontologie semi-automatique

faire, nous avons mis en place un scénario de peuplement de l'ontologie finale illustré ici par la requête du terme `changement` (fig 4). Pour aider au peuplement de l'ontologie, une ontologie linguistique (fig 5) a été intégrée dans l'application. L'utilisateur doit renseigner pour le terme requêté sa classe lexicale (`Terme_Canonique`) et sa classe métier (ici `Action`). Grâce à l'utilisation d'un entrepôt RDF comme backend de l'application, nous bénéficions, outre les capacités de requêtage et de stockage, des possibilités de raisonnement définies lors de la création des ontologies. Ainsi, l'interface ne fournit que les informations essentielles issues des déductions effectuées par l'entrepôt ; il se charge de les compléter à l'aide de la formalisation OWL (RL)(Motik *et al.*, 2009) définie dans les ontologies. Par exemple, le fait d'ajouter un lien de synonymie ou de variance dans les données permet le cas échéant de déduire l'appartenance d'un individu à une classe métier (ici `Action`) grâce à l'axiome ci-dessous :

$$\exists\, \texttt{aPourSynonyme.Action} \sqcup \exists\, \texttt{aPourVariante.Action} \sqsubseteq \texttt{Action}$$

En logique de description, cela signifie que tout individu qui a un synonyme ou une variante dans la classe `Action` appartient à la classe `Action` lui-même. Nous faisons la même chose pour les classes `Composant` et `Defaillance`, la logique du premier ordre n'autorisant pas l'écriture d'axiomes sur des ensembles de classes. L'interface ne fait qu'ajouter ou retirer des triplets RDF élémentaires, l'intelligence de l'entrepôt se charge de faire les mises à jour déclenchées par le raisonnement et éventuellement de détecter des incohérences dans les données.

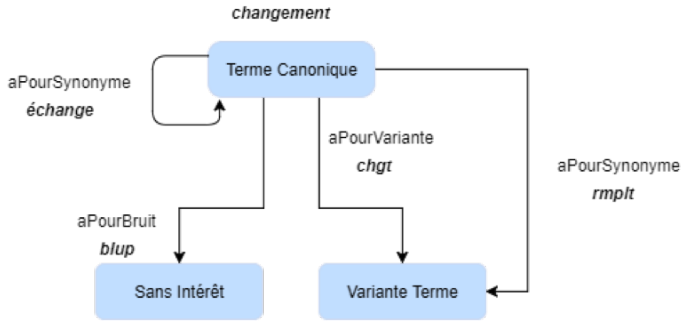

FIGURE 5 – Ontologie linguistique

7 Conclusion et perspectives

Les premiers résultats de cette étude se sont révélés très prometteurs relativement à nos besoins métiers. Nous avons pu utiliser des méthodes de plongement de mots, de visualisation par graphes et de peuplement d'ontologie pour un cas industriel concret. Il nous reste à évaluer quantitivement ces résultats sur l'ensemble de nos données. Nous souhaiterions également évaluer l'apprentissage de Word2Vec avec et sans étiquetage morphosyntaxique pour en démontrer l'apport. Par ailleurs, il serait intéressant d'utiliser l'ontologie peuplée semi-automatiquement comme corpus de référence pour de l'apprentissage supervisé en vue de la détection automatique de termes pertinents. Cette approche réduirait la tâche d'annotation en classes et en relations. Le paramétrage de Word2Vec sur les expressions multi-mots fait également partie de nos perspectives afin de gagner en pertinence.

Références

MCKEE G. T., MALVERN D. & RICHARDS B. J. (2000). Measuring vocabulary diversity using dedicated software.

MIKOLOV T., CHEN K., CORRADO G. & DEAN J. (2013). Efficient estimation of word representations in vector space. *CoRR*.

MOTIK B., GRAU B. C., HORROCKS I., WU Z., FOKOUE A. & LUTZ C. (2009). *OWL 2 Web Ontology Language Profiles (Second Edition)*. `http://www.w3.org/TR/owl2-profiles`.

MOTIK B., NENOV Y., PIRO R., HORROCKS I. & OLTEANU D. (2014). Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems. In C. E. BRODLEY & P. STONE, Eds., *Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI 2014)*, p. 129–137, Québec City, Québec, Canada : AAAI Press.

SCHMID H. (1994). Probabilistic part-of-speech tagging using decision trees.