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. À partir de celles-ci, nous peuplons des ontologies et visualisons les résultats à l'aide d'un outil de navigation dans des graphes RDF. Nous décrivons ensuite le 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 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 sur des éoliennes. Ce corpus contient environ 8000 fiches de maintenance. Si la taille de ce corpus semble restreinte, nous avons néanmoins obtenu des résultats concluants (fig), les textes étant très répétitifs ; la taille de notre dictionnaire est d'environ 5000 mots après prétraitements. Nous avons 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 :

1. Si voulu, récupération de données via Elastic Search.
2. Prétraitements : suppressions des caractères non désirés.
3. Découpage en phrases et en mots.
4. Appel de TreeTagger (Schmid, 1994) pour un étiquetage morphosyntaxique : nous sommes partis de l'hypothèse que l'apprentissage de Word2Vec serait plus fin avec l'ajout de méta-données de type morphosyntaxique sur les mots de notre corpus. Word2vec apprend sur la forme, le lemme et la catégorie morphosyntaxique du terme. Nous considérons qu'il s'agit d'une première étape de désambiguïsation.
5. Nettoyage des mots : suppression des mots vides.

6. Appel du package gensim en python pour le calcul de la similarité avec Word2Vec.
7. Création d'une sortie contenant les résultats de Word2Vec au format RDF.

Cette dernière étape va permettre de créer une base de connaissance sur les résultatss du traitement des documents du corpus contenant les phrases et mots qu'ils contiennnent et pour chaque mot du corpus :

— Son label
— Sa fréquence
— Sa catégorie morphosyntaxique
— Son lemme
— Les 15 premières phrases dans lesquelles il apparaît
— Ses 15 candidats termes

4 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'humain. Nous avons ainsi pu détecter automatiquement les types d'éléments suivants :

— des pluriels (ex : « moteur », « moteurs »)
— des abréviations (ex : pour « transfo », « transformateur », qui n'était pas présent dans le lexique constitué manuellement)
— des fautes/mots collés (« bloc » et « dublock »)
— des hyponymes (pour « équipements » on retrouve « modem », « serveur » etc.)
— des mots de la même famille (ex : « ventilateur » et « ventilation »)
— des synonymes (ex : pour « remplacement » on retrouve avec W2V « changement », « échange » etc.
— des phénomènes de multilinguisme : pour le terme « contrôle » : « controle » sans accent, termes anglais associés plus ou moins bien orthographiés « cheik » et « check ».
— pour le terme « arrêt » on retrouve « stop »

Il nous reste à présent à effectuer une analyse quantitative poussée pour établir le taux de rappel et de précision en injectant ces lexiques à notre chaîne de traitement. Cette étape pourra être facilitée par notre outil d'exploration de termes et par CuriosiText.

5 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 est un premier palier en vue du peuplement d'ontologies métiers.

Nous souhaitions une visualisation sous forme de graphe où pour chaque mot du corpus, considéré comme étant un Représentant, nous affichons ses quinze Candidats Termes les plus similaires.

Pour visualiser ces résultats, nous avons utilisé l'outil SemVue développé en 2014 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

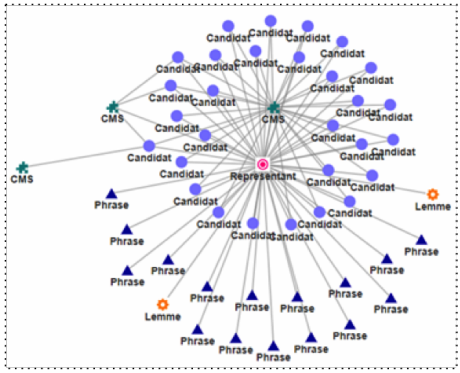

FIGURE 1 – Graphe Type de l'ontologie pour Word2Vec

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.

Cette première visualisation nous a permis d'optimiser le paramétrage de Word2Vec et de valider qualitativement nos résultats. Ainsi nous observons pour le terme changement 2

Afin de visualiser les résultats de Word2Vec formatés en RDF, nous avons créé une première ontologie modélisée sur Word2Vec. Elle est un premier palier en vue du peuplement d'ontologies métiers.

6 Peuplement semi-automatique d'ontologies métiers

Problématique du bruit

Nous avons 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 ce qui permet d'utiliser CuriosiText pour des cas d'usage très divers, sans dépendre d'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'Elastic Search 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 (fig 3) développée en amont à peupler. Pour ce faire, nous avons mis en place un scénario de peuplement de l'ontologie finale illustré ici par la requête du terme "changement". Pour aider au peuplement de l'ontologie, une ontologie linguistique (fig **??** a été intégrée dans l'application. L'utilisateur doit renseigner pour le terme requêté sa classe lexicale (Terme_Canonique) et sa classe métier (Action).

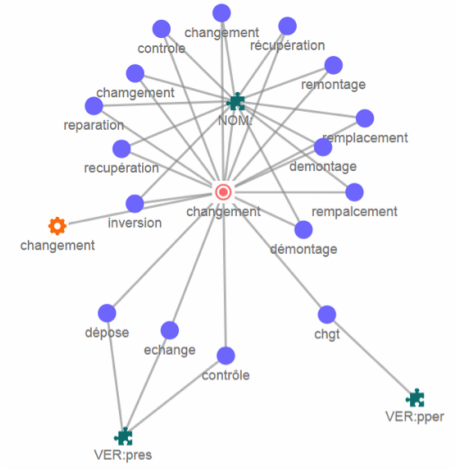

FIGURE 2 – Graphe sur le mot changement

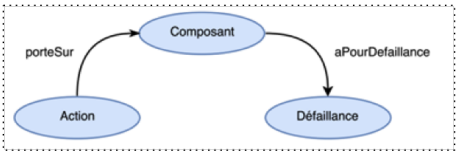

FIGURE 3 – Ontologie métier

L'intérêt d'utiliser directement un entrepôt RDF comme backend de l'application nous permet de bénéficier, en plus des capacités de requêteage et de stockage, les possibilités de raisonnement définies lors de la création des ontologies. En effet, les déductions effectuées par l'entrepôt permmettent à l'interface de ne fournir que les informations minimales à celui-ci : il se charge de compléter celles-ci 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 va permmettre éventuellement de déduire l'appartenance d'un individu à une classe métier (ici Action) grâce à l'axiome ci-dessous [1] :

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

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 ne nous permettant pas d'écrire des axiomes sur des ensembles de classes. L'interface ne fait que qu'ajouter ou retirer des triplés 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.

1. Notons que nous sommes obligé de dupliquer ce type d'axiome pour chaque classe concernée.

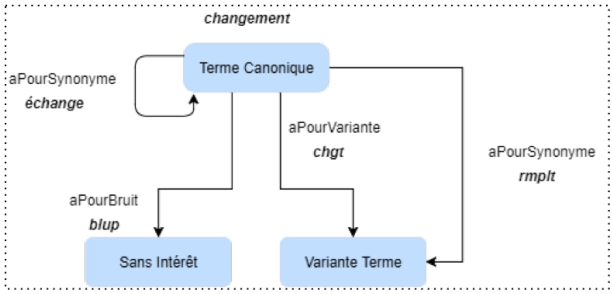

FIGURE 4 – Ontologie linguistique

7 Conclusion et perspectives

- Évaluer quantitativement - Évaluer l'apprentissage Word2Vec avec et sans étiquetage morphosyntaxique pour montrer l'apport. - Utiliser l'ontologie peuplée semi-automatiquement comme corpus de référence pour de l'apprentissage supervisé en vue de la détection automatique de termes pertinent. Cela permettrait de réduire la tâche d'annotation de classes et de relations. - Tester le paramétrage de Word2Vec sur les expressions multi-mots.

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.