# Génération automatique de billets journalistiques : singularité et normalité d'une sélection

Jérémy Vizzini, Cyril Labbé, François Portet

Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
CNRS, LIG, F-38000 Grenoble, France
nom.prenom@imag.fr

**Résumé.** La génération automatique de billets journalistiques est utilisée pour transcrire en texte des informations répondant à une requête utilisateur : prévisions météo, résultats d'élections, sportifs ou boursiers... En général, le texte généré se contente de décrire les données pertinentes sans souligner leurs singularités ou normalité par rapport à un sur-ensemble. Or, la plupart du temps, les connaissances disponibles permettent de souligner ce type d'informations. En effet, les données décrivent une même information à différents instants ou pour différents groupes. Il est donc envisageable de détecter automatiquement le caractère exceptionnel d'un prix ou d'une défaite électorale. Cet article présente une démarche et un outil permettant de spécifier un générateur de texte comparant une sélection de données à l'ensemble des données disponibles. Les singularités et les régularités exprimables à partir d'une sélection de données sont illustrées à travers un exemple (résultats d'élections). L'explicitation des connaissances (modèles, ressources langagières,...) nécessaires à la construction du générateur de textes a pour objectif de rendre l'approche générique et applicable à différents domaines (météo, élections, sports,...). Le prototype `Summy` a été réalisé pour valider la faisabilité de l'approche en utilisant les données des élections régionales. Ce prototype permet notamment de montrer comment automatiquement exprimer en langage naturel la singularité ou la normalité d'un sous-ensemble de données par rapport à l'ensemble.

## 1 Introduction

Si on dit qu'un graphique vaut mieux qu'un long discours il peut être aussi avancé qu'un petit texte synthétique, ciblé et nuancé vaut mieux que de nombreux graphiques, ou du moins qu'il vient les compléter avantageusement. En effet, la richesse des langues permet de mieux résumer des informations complexes et nombreuses en les rendant accessibles à tous : un texte peut être écouté par des malvoyants et le discours peut être adapté à l'expertise du destinataire. Le domaine de la génération automatique de textes (GAT) offre donc des perspectives pertinentes et intéressantes pour transmettre des connaissances riches, complexes et personnalisées.

Cependant, le développement d'un système de GAT est fortement dépendant du domaine et de l'application ciblée. De plus, si pour la génération à partir de données (data to text), malgré

la qualité indéniable des textes générés, on peut observer que ceux-ci restent dans une visée purement descriptive des données sélectionnées. Or, lors de la communication d'une information, il est souvent intéressant de contextualiser le contenu avec des informations exprimant des similarités et différences observables. On peut ainsi mentionner des évolutions ou des corrélations en rapport avec la sélection mais n'en faisant pas partie explicitement. Par exemple, les prévisions météo ou les résultats d'élection concernant une localité peuvent être comparés aux données concernant des localités ayant des propriétés identiques, p.ex., de la même région etc. Comparaisons qui peuvent être ensuite être insérées dans le texte.

Cet article présente une approche pour résumer des données sous forme textuelle tout en mettant en exergue les singularités et/ou la normalité d'un sous-ensemble peuvant être transcrites dans un texte. La démarche consiste à identifier et expliciter les ressources et connaissances (modèles, ressources langagières etc.) nécessaires à la construction du générateur de texte. L'objectif est de rendre l'approche générique et applicable à moindre coût dans différents domaines (météo, élections, sports...). Un prototype `Summy` a été réalisé et testé avec des données d'élection régionales.

La section 2 présente un exemple concret d'application : la génération de résultats d'élection qui sera repris tout le long de l'article ; qui permettra de mettre en lumière l'intérêt de l'approche. La section 3 décrit le processus de génération, l'architecture fonctionnelle et les différents traitements. La section 4 présente plus en détail chacune des phases de traitement en formalisant la notion de sélection de données et en décrivant le processus de macro et micro planification de textes. La section 4 présente le prototype et un bref positionnement par rapport à la littérature du domaine (section 5) avant de conclure en section 6.

## 2 Enrichir les textes décrivant les résultats des élections

La Figure 1 présente une modélisation possible des données représentant les élections régionales françaises de 2015. Les résultats sont disponibles sur `data.gouv.fr` pour les 18 régions françaises ainsi que pour chaque département et commune data.gouv.fr, Plateforme ouverte des données publiques françaises (2016). Pour les sites d'information, le défi réside en la génération en un temps minimal d'un grand nombre de textes présentant les résultats individualisés de chaque zone de vote. Étant donné que pour des élections de grande ampleur il serait bien trop coûteux de faire réaliser le travail de rédaction par l'homme, certains sites d'information ont mis en place des systèmes de génération automatique de textes tels que *The Associated Press* avec la solution d'`Automated Insights`, ou encore le site `lemonde.fr` qui a utilisé la solution `Data2Content` développée par la société Syllabs (2016).

L'exemple 1 est un exemple de texte généré automatiquement et publié sur le site `lemonde.fr`. Le texte est composé en premier lieu d'un préambule où les candidats de la région sont présentés. Ensuite, les résultats de la commune sont donnés. Ce texte est fluide et conforme aux attentes de l'utilisateur. Cependant, il contient exactement et uniquement les informations de la ville de Grenoble en 2015 au deuxième tour [1].

Or la base de données contient des informations d'autres temporalités (les élections précédentes) ainsi que les résultats des autres villes, des autres départements et régions. Techniquement, il est donc possible, à l'aide de ces connaissances, de générer un texte contenant d'autres

1. L'absence d'autres informations peut provenir de différentes causes : difficultés techniques, risque d'erreurs plus important ou simplement problèmes d'acceptabilité d'un point de vue éthique, sociale ou déontologique.

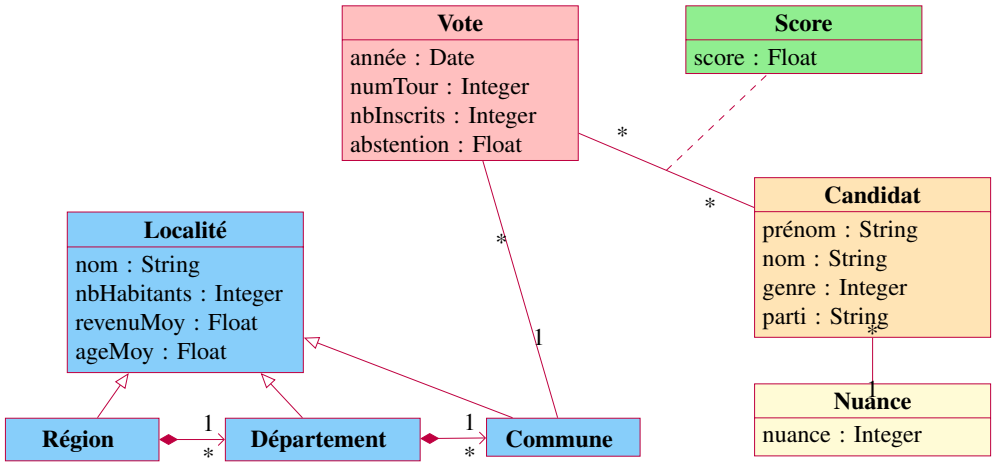



FIG. 1 – *Modélisation des données des élections régionales*

**Exemple 1** (Extrait du texte des résultats de la ville de Grenoble (`Data2content`).)**.** *Au second tour des élections régionales 2015, la région Auvergne-Rhône-Alpes voyait trois candidats s'affronter : M. Jean-Jack Queyranne (liste Union de la Gauche), M. Laurent Wauquiez (liste Union de la Droite) et M. Christophe Boudot (liste Front National).*
*Voici les résultats du second tour pour la ville de Grenoble : la liste Union de la Gauche de M. Jean-Jack Queyranne est arrivée en première position avec 57,19 % des voix. Elle a devancé la liste Union de la Droite de M. Laurent Wauquiez qui a obtenu 29,78 % et la liste Front National de M. Christophe Boudot , qui a recueilli 13,03 % des voix. Dans cette localité, 46,51 % des inscrits se sont abstenus.*

informations contextuelles. Pour cela, il faut estimer le niveau de similitude entre les données sélectionnées par l'utilisateur (résultats dans la commune) et d'autres ayant une caractéristique similaire (dates différentes, autres communes de la région, communes de taille similaire...). Lors de la génération du texte, il est intéressant de souligner le caractère *normal* ou *exceptionnel* de ces élections. Par exemple, souligner que le parti du gagnant dans la commune est commun aux autres villes de la région ou au contraire exceptionnel, signaler que le score du parti vainqueur est habituel ou non par rapport à l'historique de la ville. Dans les deux cas, l'ensemble sélectionné (résultats dans la commune) est comparé à un sur-ensemble composé d'éléments ayant une caractéristique commune (villes différentes, même région ou même ville, dates différentes).

L'exemple 2 concerne le même sujet que l'exemple 1 mais présente plus d'information contextuelle. Le lecteur obtient à la fois les informations sélectionnées mais aussi leurs mises en perspective qui permet une interprétation plus large.

Cet article présente une démarche et un outil permettant de construire un générateur de texte personnalisable et offrant la possibilité d'inclure les enrichissements présentés ci-dessus. Dans la suite de l'article, seule la singularité ou la régularité du taux d'absentions servira d'exemple. La démarche se veut générique et adaptable à de nombreux domaines.

**Exemple 2** (texte des résultats des élections avec enrichissements)**.** *Au second tour des élections régionales 2015, la région Auvergne-Rhône-Alpes voyait trois candidats s'affronter : M. Jean-Jack Queyranne (liste Union de la Gauche), M. Laurent Wauquiez (liste Union de la Droite) et M. Christophe Boudot (liste Front National).*
*Les résultats de la ville de Grenoble sont* très similaires *à ceux des autres communes de la région. La liste Union de la Gauche est arrivée en première position avec 57,19 % des voix. Elle a devancé la liste Union de la Droite qui a obtenu 29,78 % et la liste Front National, qui a recueilli 13,03 % des voix.*
*Cette tendance à voter à Gauche est* habituelle *pour une ville ayant une population jeune aux revenus moyens . Entre le premier et le second tour, le taux d'abstention a* diminué *de près de 10 %, passant de 55,50 % à 46,51 %. Un tel score et une telle diminution ne sont pas* surprenants *au regard des précédentes élections régionales.*

# 3 Processus de génération

Un concepteur de générateur de textes, spécialiste de son domaine (élection, météo, sport) doit pouvoir définir chacune des étapes de la génération et créer ou compléter les ressources et connaissances du domaine nécessaires à l'interprétation des données. La section 3.1 présente le processus global de génération de textes enrichis ainsi que les ressources nécessaires. Les sections suivantes détaillent chacune de ces étapes.

## 3.1 Vue d'ensemble

L'approche que nous avons mise en œuvre s'inspire de l'architecture en pipeline du domaine de la GAT telle que présentée par Reiter et Dale (2000). La génération commence par la détermination du contenu qui ici est soumise à la requête de l'utilisateur (Figure 2 *Sélection du contenu*). Dans notre domaine applicatif, cela correspond à sélectionner une "entité d'intérêt" à décrire parmi l'ensemble des données disponibles. Par exemple, pour le cas des élections, le lecteur sélectionne la ou les villes dont il veut connaître les résultats. Par ailleurs, des fonctions d'évaluation et de comparaison permettent d'obtenir des informations provenant du contenu sélectionné (nombre de tuples, max, min,...) et du reste de l'ensemble. Ces fonctions de comparaison sont définies par le concepteur : elles mesurent la similarité ou dissimilarité de la sélection par rapport à un autre ensemble.

Puis la structure du document est déterminée (Figure 2 structuration du document). Cette structure est spécifiée par le concepteur à l'aide d'un langage impératif de haut niveau décrivant les sections, les paragraphes et les phrases du document.

Le processus se termine par le micro-planning (Figure 2, finalisation des phrases et réalisation de surface). Cette phase s'appuie sur les structures de phrases et le lexique nécessaire à la génération en langage naturel des données de la base. Ces connaissances se composent d'un ensemble de dictionnaires associés à la base. Ces dictionnaires peuvent être préexistants ou enrichis par le concepteur.

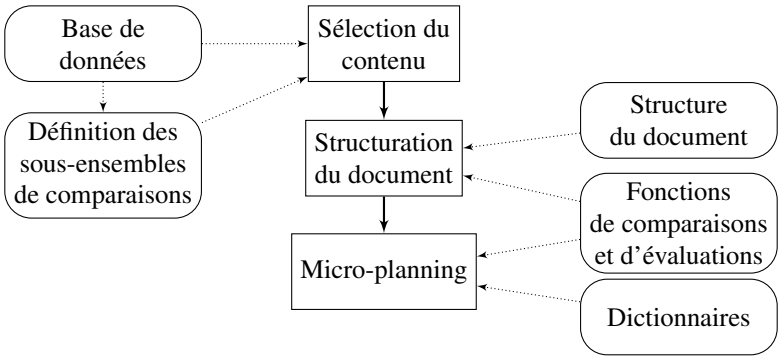



FIG. 2 – *Schéma du processus de génération*

## 3.2 Sélection du contenu

Quand l'entité sur laquelle le résumé doit porter est sélectionnée (dans l'exemple, la ou les villes), `Summy` utilise deux ensembles de données pour générer le texte décrivant l'entité et les comparaisons.

Le premier ensemble de données est qualifié de "primaires" ($TP$ dans la suite), il regroupe toutes les connaissances disponibles sur l'entité sélectionnée par l'utilisateur. $TP$ doit être défini par le concepteur du générateur de textes.

**Exemple 3.** *Le lecteur choisit un ensemble de villes. L'ensemble de données primaires peut donc être une table regroupant l'ensemble des informations reliées à la ville (cf. figure 1) :* $TP_VILLES(Annee, Num_Tour, Nom_Departement, Nom_Region, Nom_Commune, Habitants, Revenu_Moy, Age_Moy, Inscrits, Abstention)$

Le deuxième ensemble de données est dit "secondaire" ($TS$), il regroupe l'ensemble des informations qui seront résumées textuellement.

**Exemple 4.** *Dans l'exemple des élections,* $TS$ *est constitué des scores de chaque candidat dans chaque localité :* $TS_CANDIDATS(ID_Localite, Annee, Num_Tour, Annee, Prenom, Nom, Genre, Parti, Nuance, Score)$

De ces deux ensembles on en déduit $S$ qui est l'ensemble des données qui serviront à la génération du texte final. $S$ est une restriction (au sens de Codd (1970)) de l'ensemble $TS$ par l'ensemble des données choisi par l'utilisateur dans $TP$. La restriction est définie par un ensemble de propriétés communes entre les deux ensembles $TP$ et $TS$. Dans la suite, cet ensemble de propriétés communes sera nommé "filtre".

**Exemple 5.** *Pour l'exemple des élections, la sélection* $S$ *des données à transcrire est un sous-ensemble de* `TS_CANDIDATS` *où les tuples concernent la même commune, la même année et le même tour donnant ainsi la liste des candidats pour une ville sélectionnée dans l'ensemble primaire* `TP_VILLES`.

Les ensembles de données ainsi que les filtres doivent être définis par le concepteur. Les filtres sont aussi utilisés par le concepteur pour définir les ensembles auxquels $S$ doit être

comparé. Par exemple, l'ensemble des résultats pour les villes de même région que la ville de $S$.

**Exemple 6.** *La comparaison de l'abstention dans une ville peut se faire par une restriction de la table $TP$ et le filtre $\{Nom_Region\}$.*

## 3.3 Structuration du document : macro planning

`Summy` propose un langage de haut-niveau s'inspirant des schémas McKeown (1985) permettant de placer les éléments structurants du texte : les sections, les paragraphes et les listes.

Le contexte courant du générateur est défini par un ensemble de données (i.e. une table), un tuple courant et un ensemble de variables globales contenant des métadonnées (nombre de tuples de la table courante, résultat de comparaison,..). Une liste de directives et de variables globales permettant de définir la structure d'un texte sont données dans l'exemple 7.

**Exemple 7.**

- `Iteration(table_name)` *itération des tuples de la table* `table_name`*, fixe la table courante.*
- `Sentence(sent_id)` *ajoute au texte la phrase identifiée par* `sent_id`*. La phrase est construite en utilisant le contexte courant (table, tuple , variables globales).*
- `Link(link_name)` *ajoute au document un mot de liaison* `link_name`*.*
- `Similarity(evaluator, filter_id)` *mesure de la* normalité *du tuple courant à l'aide de* `evaluator`*. Le filtre d'identifiant* `filter_id`*) définie la restriction de la table courante par le tuple courant.*
- `Paragraph()` *ajoute un nouveau paragraphe.*
- `COUNT` *variable globale contenant le nombre de tuples de la table courante.*
- `SIMILARITY` *variable globale contenant le score de 0 à 1 de la dernière fonction de similarité calculée.*
- `FILTER` *variable globale contenant l'identifiant du dernier filtre utilisé.*

La figure 3 donne un exemple de structuration de document. Le document commence par une phrase préambule identifiée par `phrase_preambule`. Puis on énumère (`Iteration()`) les villes sélectionnés par le lecteur. Pour chaque ville, les candidats sont générés avec leur score électoral. Enfin, une recherche de similarité est effectuée entre le taux d'abstention du tuple courant (`TU_GET(ABSTENTION)`) par rapport à un ensemble défini par le filtre (`filtre_meme_tour_region`). Le comparateur $Similarity(evaluator, filter)$ est à définir par le concepteur en fonction de la sémantique associée aux données. Différents algorithmes de fouille de données et/ou de classification peuvent être utilisés. Pour l'abstention, une valeur est considérée exceptionnelle ou normale en fonction de son écart à la moyenne et de l'écart-type observé dans l'ensemble de comparaison.

## 3.4 Lexicalisation des variables

Deux types différents d'éléments textuels peuvent être placés dans le document : les mots de liaisons et les modèles de phrases.

```
Sentence(phrase_preambule)

Iteration() {
    Paragraph()
    Sentence(phrase_candidats_region)

    Sentence(phrase_candidats_score)
    Iteration(PROJ_CANDIDATS) {
        Sentence(phrase_score_parti)
    }

    Similarity(TU_GET(ABSTENTION), filtre_meme_tour_region)
    Sentence(phrase_abstention_vile)
}
```

FIG. 3 – *Exemple de macro-planning*

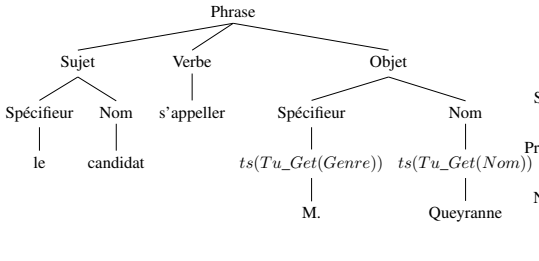



FIG. 4 – *Modèle de phrase contenant deux fonctions de lexicalisation.*

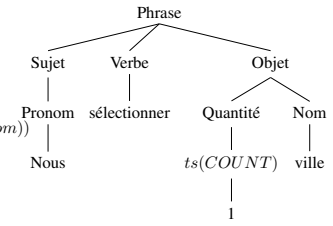



FIG. 5 – *Modèle de phrase contenant une fonction de lexicalisation.*

Les mots de liaison ont été séparés des modèles de phrases car ceux-ci sont dépendants de la structure du document : par exemple les mots de liaison indiquant des énumérations. Lors d'une énumération des mots tel que 'premièrement', 'deuxièmement', etc peuvent être ajoutés. Au contraire, le système doit supprimer cette liaison si une seule ville est transcrite.

Les phrases sont définies sous une forme permettant la réalisation de surface. La structure de la phrase est donnée sous forme d'arbre dont certaines feuilles doivent être calculées au moment de la lexicalisation qui définit les termes à associer à une donnée particulière (cf. Figure 4 et 5).

La fonction de lexicalisation $ts$ associe un mot à une valeur d'attribut, à un attribut ou à un objet du modèle. `Summy` implémente ces fonctions sous forme de dictionnaires.

La fonction de lexicalisation agit sur une donnée résultant de l'appel à un évaluateur. Un évaluateur est une fonction qui associe une valeur à une ensemble de données. Par exemple, $Tu_Get(Tuple, Attr)$ renvoie la valeur du tuple pour l'attribut donné ou encore $Ta_MostFreq(Table, Attr)$ calcule la valeur la plus fréquente pour l'attribut donnée.

Des modèles génériques sont mis à disposition du concepteur qui doit uniquement complé-

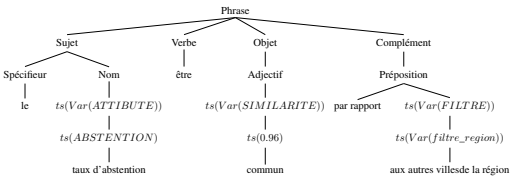



FIG. 6 – *Modèle de phrase générique attaché à la similarité*

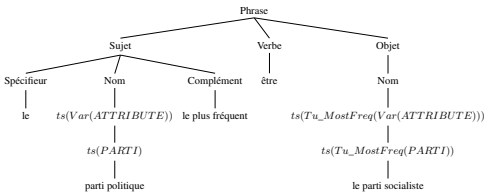



FIG. 7 – *Modèle de phrase générique attaché à $Tu_MostFreq$*

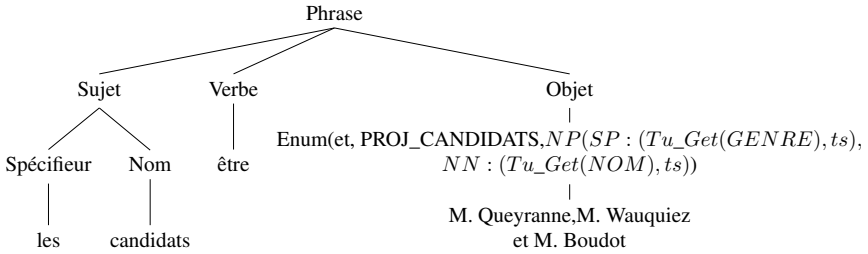



FIG. 8 – *Modèle de phrase contenant une énumération*

ter la génération des noms, des attributs ou le nom des filtres utilisés. Les figures 6 et 7 sont des exemples de modèles de phrases génériques. Ils sont fournis par le système au concepteur qui utilise leur identifiant lors de la structuration du document 3.3.

Un opérateur particulier permet d'exprimer une énumération. La figure 3.4 montre un exemple d'itération sur les différents tuples de $TS_CANDIDAT$. Cet opérateur permet à `Summy` de ne pas mettre en œuvre l'étape d'agrégation consistant à regrouper les phrases décrivant chacun des tuples.

# 4 Implémentation de Summy

L'implémentation du prototype a été effectuée en `Java` et utilise la version bilingue (franco-anglaise) de la bibliothèque `SimpleNLG` pour la réalisation de surface Vaudry et Lapalme (2013). Les tests ont été effectués sur les résultats des élections régionales françaises de 2015 stockés dans un SGBD (`MySQL`) selon le modèle décrit dans la section 3.2. La table `TP_VILLES` comporte 8092 tuples, la table `TS_CANDIDATS` 456699.

Le prototype est composé d'une interface graphique qui permet au concepteur de définir un fichier de configuration (format `JSON`) comportant les ressources linguistiques et informationnelles décrites Figure 2. Le concepteur définit les ensembles de données ($TS$,$TP$,$S$), les filtres et fonctions pour les comparaisons. La structure des documents à générer est définie à l'aide du langage présenté dans la section 3.3.

Lors de la phase de génération, l'application se connecte à la base de données et effectue la génération de texte. Le fichier de configuration ainsi que l'archive `jar` du prototype peuvent être intégrés à des applications afin de mettre en place facilement un système de génération de textes directement au sein d'un logiciel métier. Les dictionnaires ont été créés à partir des textes disponibles sur `lemonde.fr` afin d'y ajouter les enrichissements de manière à générer des textes tels que celui de la figure 2.

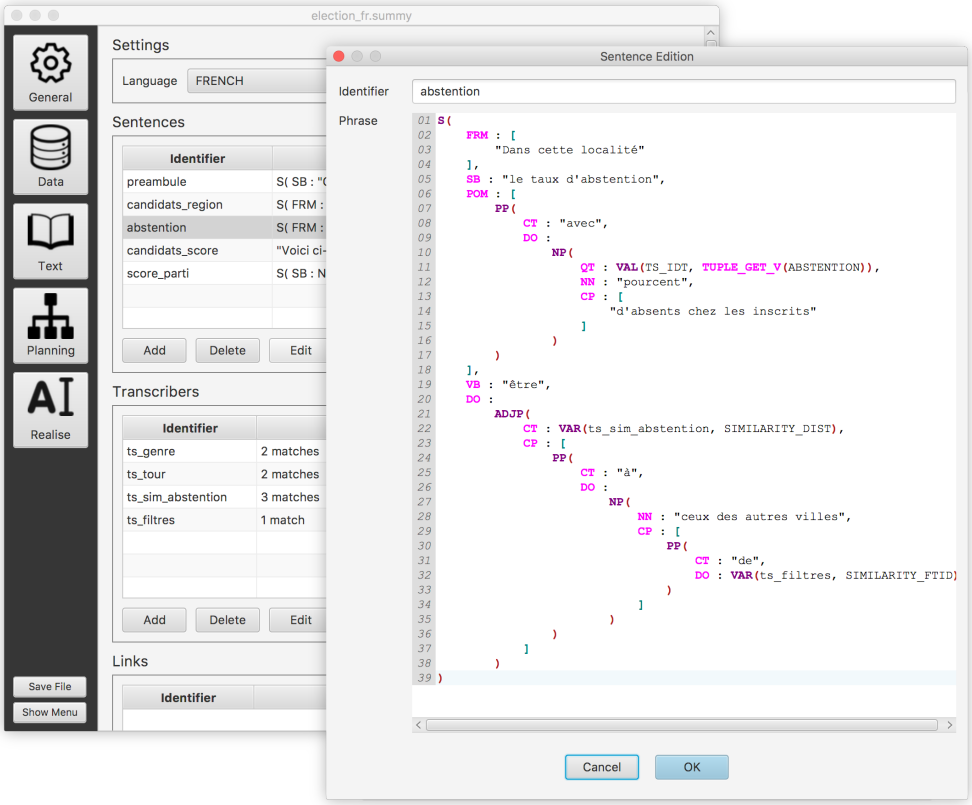



FIG. 9 – *Interface d'édition d'un modèle de phrase*

## 5 Travaux relatifs

Le résumé textuel d'informations structurées a déjà fait l'objet de nombreux travaux notamment dans le domaine de la Génération Automatique de Textes (GAT), des Bases de Données et du web sémantique. On peut citer ONTOSUM Bontcheva (2005) et NaturalOWL Androutsopoulos et al. (2013) dont l'objectif est de rendre accessible des informations contenues dans une ontologie de type RDF/OWL par sélection d'un concept et génération d'une description

textuelle des propriétés et concepts liés. L'architecture adoptée étant le pipeline classique de la GAT. Si ces approches contribuent de manière significative à un web sémantique plus accessible (cf. le KBGen challenge Banik et al. (2013)) elles reposent cependant sur une sémantique très riche bien loin des bases de données classiques.

Un autre domaine d'application est celui du résumé textuel de graphiques à destination des personnes ayant des déficiences visuelles (synthèse vocale de descriptions). En effet les systèmes tels de SIGHT Demir et al. (2012) ou iGraph Dumontier et al. (2010) permettent à l'aide de primitives de graphiques d'extraire les segments pertinents d'un graphique et de les décrire textuellement afin d'augmenter l'accessibilité des pages web. Si certains éléments de discours sont communs avec notre approche (comparaison) ces approches restent spécifiques au domaine des graphiques.

Le domaine des bases de données est assez riche en solutions de génération de textes à partir de données, que cela soit dans le domaine industriel (Syllabs dont le slogan est "Faites parler vos données", Arria, Yseop, etc.) qui surfe sur le *data analytics* ou dans le domaine académique Labbé et al. (2015); Portet et al. (2009); Labbé et Portet (2012). Si dans l'approche développée dans Babytalk Portet et al. (2009) un ensemble important et hétérogène de données médicales sont analysées, liées par des relations rhétoriques et résumées en texte, cette approche a requis un grand nombre de connaissances du domaine et n'est donc pas facilement portable dans un autre domaine. L'approche développée par Labbé et al. (2015), présente quant à elle un système capable de générer du texte à partir du résultat de requêtes de base de données. Le langage de ces requêtes permet de restreindre la génération à des primitives génériques pour lesquelles le passage d'un domaine à un autre est trivial. Ce système n'embarque cependant pas de comparaison automatique et reste donc uniquement guidé par les primitives disponibles pour la requête. L'approche présentée dans cet article permet donc d'apporter un complément utile d'un point de vue analyse des données à Labbé et al. (2015) tout en conservant l'aspect générique du système.

# 6 Conclusion

Dans cet article, nous avons proposé une approche pour résumer dans un texte des données issues d'une requête utilisateur. Cette approche a été implantée dans un logiciel et testée sur des données de résultats d'élections. Elle permet de concevoir une application de génération de textes avec une paramétrisation réduite. La prochaine étape consistera à mettre en place une évaluation sur un ensemble plus diversifiés de données (sport, bourse, etc.) en mesurant d'une part, les performances techniques et qualitatives (temps d'exécution, qualité linguistique et cohérence du texte généré). Il conviendra en particulier de mesurer le temps de développement nécessaire pour les utilisateurs-développeur et d'évaluer la préférence des lecteurs lors d'une étude comparative. Une réflexion doit aussi être menée sur la pertinence et la validité des faits automatiquement exposés ainsi que sur la portée éthique et sociale de ce type de solutions techniques.

# Références

Androutsopoulos, I., G. Lampouras, et D. Galanis (2013). Generating natural language descriptions from OWL ontologies : the naturalowl system. *J. Artif. Intell. Res. (JAIR) 48*, 671–715.

Banik, E., C. Gardent, et E. Kow (2013). The KBGen Challenge. In *the 14th European Workshop on Natural Language Generation (ENLG)*, Sofia, Bulgaria, pp. 94–97.

Bontcheva, K. (2005). Generating tailored textual summaries from ontologies. In *The Semantic Web : Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, pp. 531–545.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM 13*(6), 377–387.

data.gouv.fr, Plateforme ouverte des données publiques françaises (2016). Sélection thématique : élections régionales 2015. `https://www.data.gouv.fr/fr/datasets/selection-thematique-elections-regionales-2015/`. Accessed : 2016-10-14.

Demir, S., S. Carberry, et K. F. McCoy (2012). Summarizing information graphics textually. *Computational Linguistics 38*(3), 527–574.

Dumontier, M., L. Ferres, et N. Villanueva-Rosales (2010). Modeling and querying graphical representations of statistical data. *J. Web Sem. 8*(2-3), 241–254.

Labbé, C. et F. Portet (2012). Towards an abstractive opinion summarisation of multiple reviews in the tourism domain. In *The First International Workshop on Sentiment Discovery from Affective Data (SDAD 2012)*, pp. 87–94.

Labbé, C., C. Roncancio, et D. Bras (2015). A Personal Storytelling about Your Favorite Data. In *15th European Workshop on Natural Language Generation (ENLG 2015)*, Brighton, United Kingdom.

McKeown, K. R. (1985). *Text Generation : Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press.

Portet, F., E. Reiter, A. Gatt, J. Hunter, S. Sripada, Y. Freer, et C. Sykes (2009). Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence 173*(7-8), 789–816.

Reiter, E. et R. Dale (2000). *Building Natural Language Generation Systems*. New York, NY, USA : Cambridge University Press.

Syllabs (2016). Nos robots rédacteurs collaborent avec le monde. http ://blog.syllabs.com/le-monde-elections-departementales-syllabs-robotjournalisme/. Accessed : 2016-10-14.

Vaudry, P.-L. et G. Lapalme (2013). Adapting simplenlg for bilingual english-french realisation. In *Proceedings of the 14th European Workshop on Natural Language Generation*, Sofia, Bulgaria, pp. 183–187. Association for Computational Linguistics.

Génération automatique de textes : singularité et normalité


## Summary

Natural language generation is used to describe, in natural language, the data answering a user query : weather forecast, elections or sport results... In most cases, the generated text do not show the singularity nor the normality of the selected data compared to the whole. Most of the time, the knowledge needed to express these particular kind of information is available. It is thus possible to detect particular exceptional or banal features of a selection and to generate automatically a text that exposes these interresting facts. This paper presents a tool aiming at specifying generators of texts containing comparisons of the selected data to the whole set. Singularity and banality exprimable using the tool are showed thanks to an example (election results). Reification of the needed knowledge (models, language ressources,... ) aims for genericity of the approche and easy reuse for other domains (weather forecast, sports,...). The prototype `Summy` was built to validate the approach and to demonstrate how particularity or normality of a subset of data compared to the whole can be automatically expressed.