Linking Stanford Typed Dependencies to Support Text Analytics

Fouad Zablith
Olayan School of Business
American University of Beirut
PO Box 11-0236, Riad El Solh, 1107 2020
Beirut, Lebanon
fouad.zablith@aub.edu.lb

Ibrahim H. Osman
Olayan School of Business
American University of Beirut
PO Box 11-0236, Riad El Solh, 1107 2020
Beirut, Lebanon
ibrahim.osman@aub.edu.lb

ABSTRACT

With the daily increase of the amount of published information, research in the area of text analytics is gaining more visibility. Text processing for improving analytics is being studied from different angles. In the literature, text dependencies have been employed to perform various tasks. This includes for example the identification of semantic relations and sentiment analysis. We observe that while text dependencies can boost text analytics, managing and preserving such dependencies in text documents that spread across various corpora and contexts is a challenging task. We present in this paper our work on linking text dependencies using the Resource Description Framework (RDF) specification, following the Stanford typed dependencies representation. We contribute to the field by providing analysts the means to query, extract, and reuse text dependencies for analytical purposes. We highlight how this additional layer can be used in the context of feedback analysis by applying a selection of queries passed to a triple-store containing the generated text dependencies graphs.

Categories and Subject Descriptors

E.1 [**Data**]: Data Structures—*Graphs and networks*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms

Performance

Keywords

Graph analysis, knowledge representation, linked data, performance management, semantic web, text analytics

1. INTRODUCTION

Text analytics is one of the core tasks in mining insights from the ever growing amount of text, especially online. For example in marketing campaigns, analysts aim to convert what consumers say about their products online into knowledgeable facts. One of the major differences between text-bases and other controlled data repositories (e.g. databases) is that text is unstructured and does not follow a predefined format for knowledge sharing and processing. There are many challenges involved in text analytics, mainly due to the unstructured nature of text that makes it hard to: (1) query the text source in a standardized way to derive insights, (2) trace the occurrence of textual terms and (3) study the cause and effect dependencies among terms.

Natural language processing (NLP) research has come a long way in performing various sophisticated tasks on text. Such tasks include for example named entity recognition [8], deriving structures for question answering [5], part-of-speech tagging [13] and word-sense disambiguation [10]. We present in this paper our work on linking dependency relations in text using the Resource Description Framework (RDF) specification [6], with the aim to improve text analytics. Dependencies in text (i.e. grammatical relations) naturally exist as a way to cognitively enable the reader to infer structural meanings from text. We highlight how capturing and linking such dependencies in RDF would provide a new analytical dimension of the involved text. Our aim is to create an additional analytical layer based on the linked dependencies among text elements. We implement a dependency-to-RDF translator that converts the Stanford typed dependency [4] to a linked data ready graph. We employ this translator to process comments aggregated from the evaluation of e-Government services [9], and push the generated RDF to a triple-store for analytical purposes and querying the aggregated data.

The aim of this work is to create on top of disparate pieces of text a layer that (1) connects the dots around the different textual elements, (2) can be queried and extended when new text emerges, and (3) can be used to derive insights around the flow of knowledge aggregated for example from user feedback. This semantic layer will enable us to go beyond syntactic analytics (e.g. the most frequently mentioned term), and to derive more in-depth information (e.g. which entities in the text are the most semantically modified). The contribution of our approach is that we are providing a linked layer of raw text dependencies on top of text corpora. We are preserving the provenance of all the text elements through Unique Resource Identifiers (URIs), which enables analysts to trace back exactly where the element came from in the text. Furthermore, thanks to the graph nature of RDF representations, additional analysis can be adopted by linking auxiliary information (e.g. polarity) to the specific elements in the graph of the text. In other words we are preserving all text dependencies in RDF, and leaving it to the analysts to process the text fragments as required by the application context.

We present in Section2 related work in the field. Then we discuss the linking process in Section 3, followed by a scenario where we apply some query examples on the aggregated text in Section 6. We finally conclude and discuss our future work in Section 7.

2. RELATED WORK

Text processing for analytics is gaining more and more traction in various fields. Businesses and organizations are understanding the value of this exercise with the increase of text and online discussions. Approaches are exploiting text to perform various objectives.

Acknowledging that the unstructured nature of text can be turned into a consumable body of knowledge, research around extracting entity relationships out of text documents has been ongoing for a while. For example existing work focused on creating ontologies from text documents to identify concepts and their corresponding relations [3, 12, 15]. The objective of such tools is mainly to build a representation of a specific domain (i.e. an ontology), out of a corpus. In other words, the aim is to identify the most accurate concepts and their corresponding relations from text sources. While this layer is a query compatible layer on top of text, some of the analytics operations such as filtering statements where opinions about an entity are expressed are not possible. This is due to the fact that most of such tools focus on concept-to-concept relations, and often do not capture descriptive statements that include for example adjectively modified entities.

Text dependencies have been used to perform many text processing tasks. For example it was used to identify polarity and sentiment analysis in text sentences [14]. It was also used to infer semantic relations among terms in text. For example in OntoLearn [12] it was used to derive *kind-of* relations between entities from term definitions. Others have relied on dependency relations to derive inference rules from text [7]. We observe that text analysts are mainly bound by the functionalities and algorithms implemented by the tools used. For example in tools where the focus is on sentiment analysis, the emphasis is on terms that affect polarity (e.g. adjectives), with little focus on conceptual relations. In other contexts, where the aim is to find conceptual relations around entities, adjectives that describe such entities in text are probably dropped. However in some cases, text analysts might require more flexibility in processing text, guided by their analysis objectives. Simple objectives could be for example to directly extract adjectives that are used by product or service reviewers, combined with their dependents mentioned in the text.

The aim of this work is to provide this flexibility by capturing and linking text dependencies as they occur in the text, and offer a query and inferencing enabled layer that merges the different pieces of text. Such layer can be directly used by analysts or tool developers with the aim to reuse text dependencies in other contexts or tool development to support text analytics.

3. LINKING PROCESS OVERVIEW

We highlight the main steps of the work in Figure 1. At a high level, a piece of text is passed to the text processing phase, where the part-of-speech (POS) and lexical parsers modules generate the text dependencies that we describe in further details next in the paper. The dependencies and POS information are passed to the linking module, where the links are created based on the dependency relations, and all the text elements are preserved in RDF triples with explicitly defined identifiers. The triples are then pushed to a triple-store to enable further processing and analytics applications.

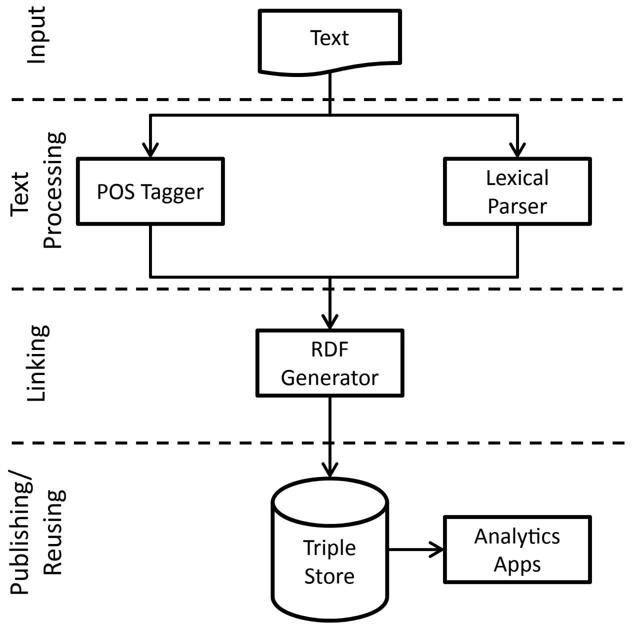

Figure 1: Overview of the Linking Process.

4. STANFORD TYPED DEPENDENCIES

We rely in our work on the Stanford NLP tools for processing text [2, 11]. Stanford typed dependencies identify grammatical relations among textual entities. The dependency relations follow a hierarchy, and are binary relations that include a *governor* (or head) and a *dependent* in the form of *depRelation(governor, dependent)*. Take for example the following statement:

It is an efficient service.

it includes the following Stanford dependency relations:

nsubj(service, It) - *nominal subject*
cop(service, is) - *copula*
det(service, an) - *determiner*
amod(service, efficient) - *adjectival modifier*

The full list of relations can be found in [4]. This binary representation is a good candidate for converting the relations to RDF triples as we show later in the paper.

In addition to dependency relations, Stanford NLP tools provide POS tagging features, through which the text entities are tagged based on their POS [11]. For the previous example, the entities are tagged as follows:

It:	PRP - *personal pronoun*
is:	VBZ - *verb, 3rd person singular present*
an:	DT - *determiner*
efficient:	JJ - *adjective*
service:	NN - *noun, singular*

5. GENERATION OF DEPENDENCY GRAPHS

In this part we present the methodology we followed to generate the RDF graphs of text dependencies. We first describe how the identifiers of text elements are generated; then we present the graph model we adopted to link the text dependencies; and later discuss the implementation of the dependency-to-RDF translator.

5.1 Unique Resource Identifiers of Text Elements

In our work, we represent each entity (i.e. terms, sentences, etc.) found in the text by a unique resource identifier (URI). This will enable us to create dependency relations that link unique terms in a text repository. The entities are linked to a context that also has a well defined URI. For example the term *service* has a URI and is linked through a relation to the sentence (with a URI) that forms the context of the term. In our design, we follow the below patterns for URI creation of text elements:

Sentence URI: NameSpace:`MD5`(sentence)

Term URI: NameSpace:term/`POS`(term)/`text`(term) /`MD5`(sentence)/_`positionIndex`(term)

Where `MD5`(sentence) is a function that returns the unique MD5 hash of the sentence; the `POS`(term) is a function that returns the part-of-speech tag of the term; the `text`(term) is a function that returns the string of the term; and the `positionIndex`(term) returns the position of the term relative to the sentence. Applying the URI patterns on the example we highlighted before will generate the list of URIs presented in Table 1.

5.2 Text Dependencies Graph Model

In order to translate the text entities into an RDF graph, we create a two-layered ontology schema. At a high level, we represent the schema of the Stanford dependencies (i.e. POS and Dependency relations) in RDF. The schema also includes the relations among the dependencies for improved inferencing. This will enable for example to extract all entities linked through a *modifier* relation, inferred from all its sub-properties such as adjectival modifier or adverbial modifier.

Table 1: URI Examples of a Sentence and its Terms

Entity	URI
It is an efficient service	NS:sentence/4c7aa81ba8fbcd3ad42996eb6bac18dc
It	NS:term/PRP/It/4c7aa81ba8fbcd3ad42996eb6bac18dc_1
is	NS:term/VBZ/is/4c7aa81ba8fbcd3ad42996eb6bac18dc_2
an	NS:term/DT/an/4c7aa81ba8fbcd3ad42996eb6bac18dc_3
efficient	NS:term/JJ/efficient/4c7aa81ba8fbcd3ad42996eb6bac18dc_4
service	NS:term/NN/service/4c7aa81ba8fbcd3ad42996eb6bac18dc_5

We present in Figure 2 the model we created to link the entities in text sentence. We start from a text, or at a more abstract level, a context. We have chosen this design to enable the linking of analysis across various contexts. For example product reviews of the same product can exist in various websites, which are considered contexts in our design. This representation will enable us to move from granular term levels, going back to higher level contexts and vice versa.

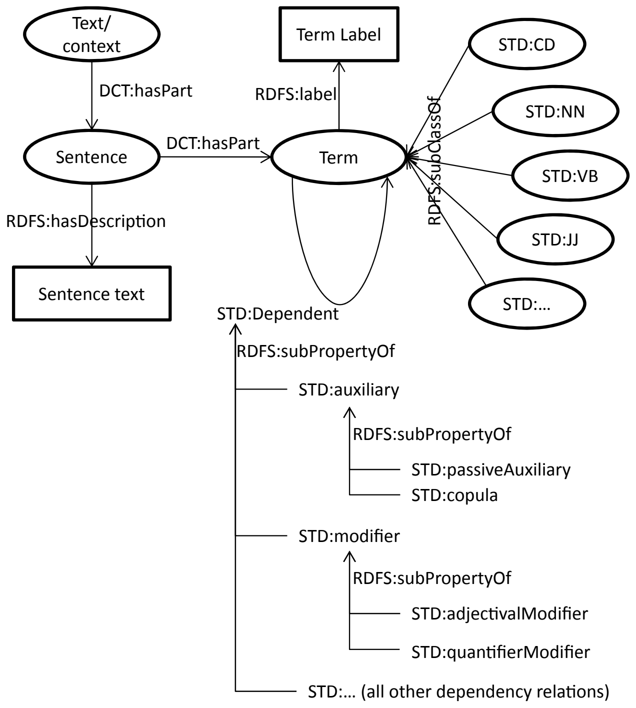

Figure 2: Text Dependencies RDF model.

From the text we move to the sentence level, which is linked to its terms through a *DCT:hasPart* relation taken from the Dublin Core terms vocabulary[1]. Links can be made down to the level of the part-of-speech term type. Terms are related through the generic Stanford dependency relation, which has all its sub-properties explicitly captured. We highlight in Figure 3 the dependency RDF graph generated of the ex-

[1]`http://purl.org/dc/terms/`

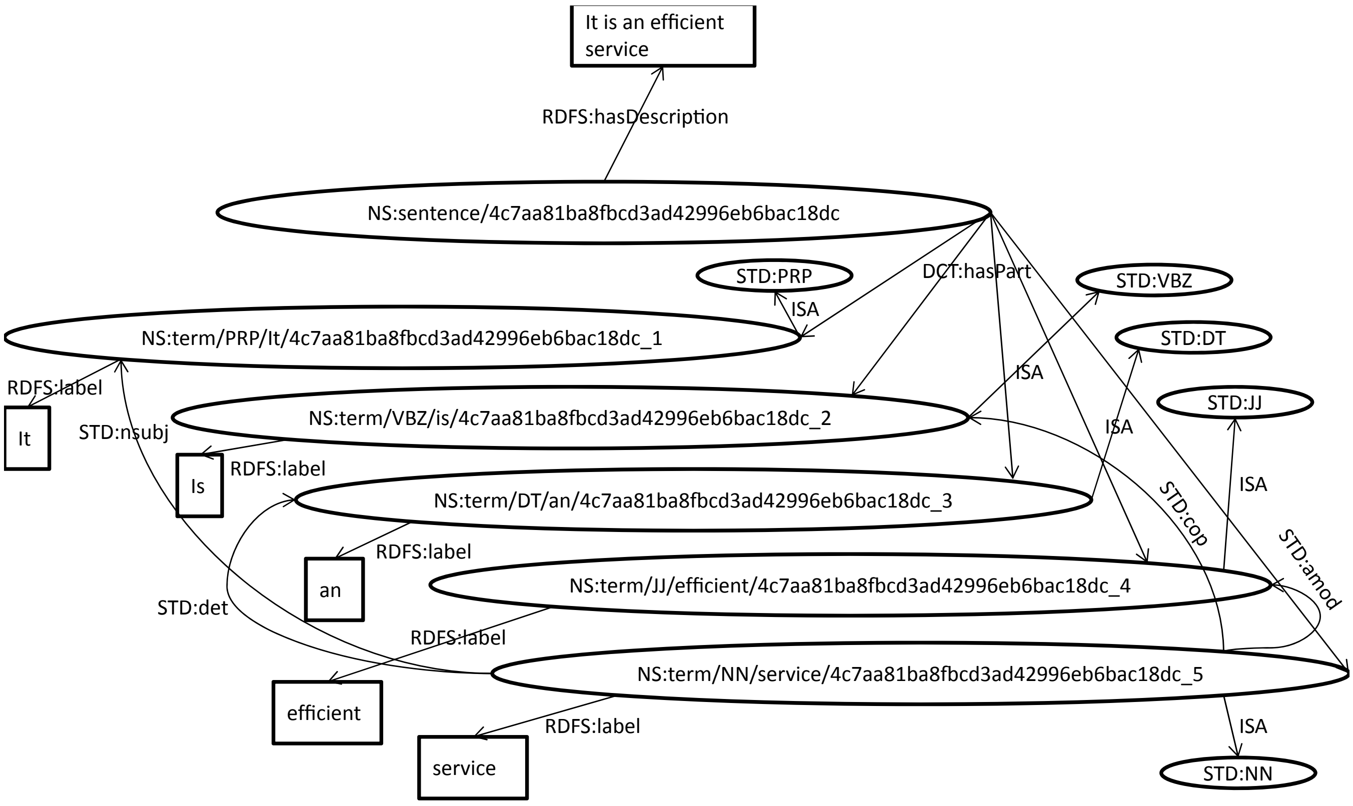
It is an efficient service
RDFS:hasDescription
NS:sentence/4c7aa81ba8fbcd3ad42996eb6bac18dc
DCT:hasPart
STD:PRP
ISA
STD:VBZ
NS:term/PRP/It/4c7aa81ba8fbcd3ad42996eb6bac18dc_1
STD:DT
RDFS:label
ISA
It
STD:nsubj
NS:term/VBZ/is/4c7aa81ba8fbcd3ad42996eb6bac18dc_2
STD:JJ
Is
RDFS:label
ISA
NS:term/DT/an/4c7aa81ba8fbcd3ad42996eb6bac18dc_3
STD:cop
ISA
RDFS:label
STD:amod
an
NS:term/JJ/efficient/4c7aa81ba8fbcd3ad42996eb6bac18dc_4
STD:det
RDFS:label
NS:term/NN/service/4c7aa81ba8fbcd3ad42996eb6bac18dc_5
efficient
RDFS:label
ISA
service
STD:NN

Figure 3: Sentence Dependencies RDF Example.

ample sentence "it is an efficient service".

5.3 Dependency-to-RDF Translator

We developed a dependency to RDF translator that takes a sentence as input, and generates an RDF graph. We rely on the Stanford NLP Lexical Parser tool [2] to parse the text, on which we apply our translation patterns. We used the combination of Jena for handling the creation of the model [1], and Fuseki-TDB triple-store[2] to handle serving the RDF code. We describe how we handle the translation process in Algorithm 1. Note that the `InferenceModel` function, which enables inferencing on the model, is not required if the RDF data will be pushed to a triple-store that includes an inferencing engine. However given that the Fuseki-TDB does not provide inferencing functionalities by default, the other option was to apply inferencing using Jena on the model before pushing it to the triple-store.

```
input  : A string $sentence$, LexicalizedParser $lp$
output: RDF model $r$
run the lexicalized parser lp on the sentence to get the
list of dependency relations "relationsList";
$sURI$ ← GenerateURI($sentence$);
for $i \leftarrow 0$ to $relationsList.size$ do
    loop through the dependency relations and add the
    triples to model $r$;
    $s$ ← GenerateURI($Relation[i].governor$);
    $p$ ← GenerateURI($Relation[i]$);
    $o$ ← GenerateURI($Relation[i].dependent$);
    $r$.AddTriple($s, p, o$);
    $r$.AddTriple($sURI, RDFS : hasPart, s$);
    $r$.AddTriple($sURI, RDFS : hasPart, o$);
    /* Add triples to $r$ related to the Stanford
    dependency relations hierarchy and the
    entity labels                         */;
    InferenceModel($r$)        /* This function will
    enable inferencing on the model */;
end
PushTotriple-store($r$)    /* The output model $r$ is
then pushed to a triple-store */;
```

Algorithm 1: RDF generator.

6. TEXT ANALYTICS SCENARIO

In order to preliminary show the value of this RDF layer on top of text, we apply our work in the context of the analysis of user feedback on e-government services. As part of the I-Meet research project [9], we collected the feedback from users around e-Government services. The questionnaire included closed-ended and two open-ended questions. While the analysis of the closed-ended questions was performed from different perspectives, the analysis of the open-ended questions was more challenging.

[2]`http://jena.apache.org/documentation/serving_data/`

We fed the 3,140 English written comments to our dependency-to-RDF parser, and generated 174,862 RDF triples that were pushed to a Fuseki-TDB triple-store. It is worth to note that without inferencing, the number of triples generated was 77,322. When the `InferenceModel` functionality is applied, Jena makes all inferencing statements as explicit triples. Hence the number of triples will be lower if the RDF is to be pushed to a triple-store with inferencing functionalities.

Sample Queries

With the text dependencies all represented in RDF and pushed to a triple-store, it is now possible to query the data to see how elements in the text interact with the aggregated view on user comments.

SPARQL Query 1: *What were the adjectives used by users to describe their experience from the most frequent, to the less frequent?*

```
SELECT ?adjLabel (COUNT(?adjLabel) AS ?count)
WHERE
{
    ?adj <http://www.w3.org/2000/01/rdf-schema#sub
        ClassOf> <http://linked.aub.edu.lb/term/JJ>.
    ?adj <http://www.w3.org/2000/01/rdf-schema
        #label> ?adjLabel
}
GROUP BY ?adjLabel
ORDER BY DESC(?count)
```

Based on the comments received by users, we got the first 10 results highlighted in Table 2.

Table 2: SPARQL Query 1 Sample Results

?adjLabel	?Count
easy	220
good	97
quick	72
other	67
available	32
able	31
simple	31
great	30
convenient	29
essential	29

SPARQL Query 2: *What were the "things" that users found "easy"?*

```
SELECT ?modifiedByEasy (COUNT(?modifiedByEasy) AS
?count)
WHERE
{
    ?adj <http://linked.aub.edu.lb/ontology/
        dependency/prep> ?modified .
    ?adj <http://www.w3.org/2000/01/rdf-schema
        #label> ?adjLabel.
    ?modified <http://www.w3.org/2000/01/rdf-schema
        #label> ?modifiedByEasy
    FILTER regex(?adjLabel, "^easy")
}
GROUP BY ?modifiedByEasy
ORDER BY DESC(?count)
```

Note that in this query we are querying the entities that are dependent on "easy" through the *prepositional modifier* (prep) dependency relation, which will indirectly include all the prepositional sub-property relations. For example the statement "easy to use" will have the following dependency relation from the Stanford parser: perp_to(easy, use). In our RDF translator, all sub-properties have been explicitly linked, which enabled us to run the query above. Part of the results are given in Table 3.

Table 3: SPARQL Query 2 Sample Results

?modifiedByEasy	?Count
use	11
access	2
acess (typos from users)	2
me	2
time	2
cost	1
excess	1
layout	1
problems	1
saving	1

SPARQL Query 3: *How is the term "Service" described by users?*

```
SELECT ?adjModifier (COUNT(?adjModifier) AS ?count)
WHERE
{
    ?x <http://www.w3.org/2000/01/rdf-schema#label>
        ?xLabel.
    ?z <http://www.w3.org/2000/01/rdf-schema#label>
        ?adjModifier.
    ?z <http://www.w3.org/2000/01/rdf-schema#sub
        ClassOf> <http://linked.aub.edu.lb/term/JJ>.
    {
        ?x <http://linked.aub.edu.lb/ontology/
            dependency/dep> ?z
    }
    UNION
    {
        ?z <http://linked.aub.edu.lb/ontology/
            dependency/dep> ?x
    }
FILTER regex(?xLabel, "^service")
}
GROUP BY ?adjModifier
ORDER BY DESC(?count)
```

In this query we looked for adjectives that impact the term service through the high level generic dependency relation in the aggregated user comments. Part of the results are shown in Table 4.

Table 4: SPARQL Query 3 Sample Results

?adjLabel	?Count
good	28
other	11
online	10
available	9
great	8
easy	5
useful	5
electronic	4
free	4
quick	4

7. CONCLUSION AND FUTURE WORK

We presented in this paper our work on linking text dependencies to improve text analytics. We discussed the linking process of text elements using RDF, following the Stanford typed dependencies representation. Our dependency-to-RDF translator was applied to process and link 3,140 aggregated user feedback around an e-Government services survey. This extracted RDF was pushed to a triple-store, and sample queries were run to highlight some of the added value and the type of analysis that can be applied on this new linked layer on top of text.

The contribution of the work is that we are making text dependencies more accessible for consumption by analysts and other applications. While keeping all the text dependencies available for processing, we are giving the freedom for the analyst to ask and extract the pieces of text that are relevant to the study in context. The benefit of having this linked layer extends also to the nature of RDF and the corresponding vocabularies used, through which we were able to apply inferencing on the Stanford dependencies representation. For example the links between properties through the sub-property relations, or between terms' POS types through sub-class relations, made it easier to traverse the graph around the text. Another benefit is that the access to text through SPARQL endpoints greatly improves the reuse and consumption of text snippets. For example now it is easier to apply pattern-based analysis on text by formulating rules directly through SPARQL queries. Consider the case where the analyst would like to enforce the rule of inverting the polarity of positive or negative terms that have been negatively modified in the text. To apply this, the SPARQL query 3 presented above can be easily modified to add this rule pattern in the SPARQL condition.

In our future work, we are planning to investigate how we can use this dependency graph layer on top of text to hook text documents to decision and performance management models. We believe that this can open up new channels to improve business analytics and derive insights from text.

8. REFERENCES

[1] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, 2004.

[2] D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

[3] P. Cimiano and J. Volker. Text2onto a framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 227–238, Alicante, Spain, 2005. Springer Berlin / Heidelberg.

[4] M.-C. De Marneffe and C. D. Manning. Stanford typed dependencies manual. Technical report, 2008.

[5] S. M. Harabagiu, M. A. Pasca, and S. J. Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 292–298. Association for Computational Linguistics, 2000.

[6] O. Lassila and R. R. Swick. Resource description framework (RDF) model and syntax specification. 1999.

[7] D. Lin and P. Pantel. DIRT- discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM, 2001.

[8] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[9] I. H. Osman, A. L. Anouze, N. M. Hindi, Z. Irani, H. Lee, and V. Weerakkody. I-meet framework for the evaluation eGovernment services from engaging stakeholders' perspectives. *European Scientific Journal*, 10(10), 2014.

[10] M. Sanderson. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 142–151. Springer-Verlag New York, Inc., 1994.

[11] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[12] P. Velardi, R. Navigli, A. Cucchiarelli, and F. Neri. Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies. *Ontology Learning and Population*, 2005.

[13] A. Voutilainen. Part-of-speech tagging. *The Oxford handbook of computational linguistics*, pages 219–232, 2003.

[14] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP*, pages 347–354. Association for Computational Linguistics, 2005.

[15] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, and M. Sabou. Ontology evolution: a process-centric survey. *The Knowledge Engineering Review*, 30(01):45–75, Jan. 2015.