A First Step Towards Automatic Consolidation of Legal Acts: Reliable Classification of Textual Modifications

Samuel Fabrizi, Maria Iacono, Andrea Tesei and Lorenzo De Mattei
Aptus.AI / Pisa, Italy
{samuel,maria,andrea,lorenzo}@aptus.ai

Abstract

The automatic consolidation of legal texts with the integration of its successive amendments and corrigenda might have an important practical impact on public institutions, citizens and organizations. This process involves two steps: a) the classification of the textual modifications in amendment acts and b) the integration within a single document of such modifications. In this work we propose a methodology to solve step a) by exploiting Machine Learning and Natural Language Process techniques on the Italian versions of European Regulations: our results suggest that the methodology we propose is a reliable first milestone toward the automatic consolidation of legal texts.

1 Introduction

Consolidation consists of the integration in a legal act of its successive amendments and corrigenda.[1] Consolidated texts are very important for legal practitioners. However, their maintenance is a tedious task. Some regulatory publishers such as Normattiva[2] provide continuously updated consolidated texts, others such as Eur-Lex[3] do times to times, some other do not. The automation of this process could let institutions to save resources and practitioners to access continuously updated consolidated documents. This achievement would let organizations stay compliant with the normative more easily. The consolidation process involves two main steps: a) the identification and classification of the textual modifications in amendment acts; b) the integration within a single document of the textual modifications identified in the previous step. The first step can be expressed as the automatic classification of textual modifications inside a legal document. In this work, we focus on step a).

Several authors tried to solve this task using standard Natural Language Processing (NLP) techniques. Ogawa et al. (2008) showed that amendment clauses described in the Japanese statutes can be formalized in terms of sixteen regular expressions. Lesmo et al. (2009) tried to identify and classify integrations, substitutions and deletions using a three-step approach: 1) prune text fragments that do not convey relevant information, 2) perform the syntactic analysis of the retrieved sentences, 3) semantically annotate the provision using a rule-based approach based on tree. In this last step, they also used a knowledge base that describes the provisions taxonomy (Arnold-Moore, 1997).[4] Brighi et al. (2008) and Spinosa et al. (2009) followed a similar approach. In both cases, semantic analysis is carried out on the syntactically pre-processed text using a rule-based approach. The difference is related to the starting point of the semantic analysis. The former's system relied on a deep semantic analysis of the textual modifications. The latter started from the shallow syntactically parsed text. Garofalakis et al. (2016) presented a semi-automatic system for the consolidation of Greek legislative texts based on regular expressions. Francesconi and Passerini (2007) defined a module that automatically classifies paragraphs into provision types. Each paragraph is represented using Bag of words either with TF-IDF weighting (Salton and Buck-

[1] Eur-Lex, About consolidation, https://bit.ly/2VFyGhv

[2] Normattiva, https://www.normattiva.it/

[3] Eur-Lex, https://eur-lex.europa.eu/

[4] A legislative provision represents the meaning of a law part from a legal point of view. Obligations, definitions and modifications are specific types of provision.

ley, 1988) or binary weight. The authors showed an experimental comparison of the different representation methods using the Naive Bayes and Multiclass Support Vector Machine (MSVM) models. This paper describes our approach in the classification of textual modifications, namely substitution, addition, repeal and abolition. The proposed approach is based on standard statistical NLP techniques (Manning and Schutze, 1999). Our method involves i) the use of XML-based standards for the annotation of legislative documents, ii) the construction of the dataset assigning a label to each word according to the tagging format used, and iii) the implementation of NLP models to identify and classify textual modifications. We carried out a systematic comparison among several feature extraction techniques and models. The main contribution of this paper is the application of machine learning models to classify textual modifications. In contrast to rule-based or regular expression techniques, our models do not need expert knowledge about the application domain's properties. They try to extract formulas used to introduce a textual modification without the need for an explicit definition of all the formulas. Our approach leads to lower maintenance costs and hopefully increased robustness of the system.

2 Data

We extracted the data from Daitomic[5], a product that contains all the regulations from a set of legal sources encoded automatically in Akoma Ntoso standard format (Palmirani and Vitali, 2011). We collected from this product all the Italian versions of the amendment documents originally extracted from Eur-Lex and we randomly sampled 260 legal documents for manual labelling.
Accordingly to the Eur-Lex web service specifications[6], we identified seven different types of textual modifications:

- *replacement* annotates a substitution which may concern a part of a sentence (expression, word, date, amount) or a whole subdivision of the document (article, paragraph, indent). Usually, this type of textual modification includes also the following subcategories:
  - *from* annotates the replaced words ("novellando").
  - *to* annotates the words that replace the previous ones ("novella").
- *replacement_ref* is a type of replacement. We use it to handle textual modifications that include attachments.
- *addition* annotates textual modifications that add or complete a part of a legal document.
- *repeal* indicates the removal or reversal of a law. It is used to invalidate its provisions altogether.
- *abolition* indicates the removal of a law part. It is used to replace the law with an updated, amended or related law. This textual modification could just involve single words or whole subdivision as in the replacements.

Category	Total
replacement	308
from	95
to	95
replacement_ref	34
addition	96
repeal	93
abolition	92

Table 1: Total number of textual modifications for each category

Table 2 reports an example for each of the mentioned categories. Table 1 shows the total number of textual modifications per category. The number of *replacements* examples is greater than that the others types of modifications because substitutions can be introduced by different formulas that determine their specific meaning. Indeed, from a preliminary experiment, we understood that there is a relationship of proportionality between the number of formulas used to introduce textual modifications and the number of examples needed to train the models. For this reason, we needed a different number of examples for each category to train our models.
Given the differences among the nature of each modification type, we preferred to split the original problem into five subtasks, namely:

1. *replacement* classification that also contains the *replacement_ref* category;
2. *addition* classification;

[5]Daitomic, https://www.daitomic.com/

[6]Eur-Lex, How to use the webservice?, https://bit.ly/393qt9Z

3. *repeal* classification;

4. *abolition* classification;

5. *from_to* classification.

The manual annotation consisted in assigning one label at each token of the selected document for each subtask that indicates if it represents or not a textual modification. We defined three different tagging formats: Inside-Outside-Beginning (IOB), Inside-Outside (IO), Limit-Limit(LL). The first two tagging formats are standard.[7] The last one, instead, uses the prefix "L-" to indicate that the token is either the beginning or end of a textual modification. We adopted a specific tagging format for each model based on our preliminary results. The tagging format was one of the most critical choices to improve model performance.
The dataset used for the last subtask is different. Indeed, the *from* and *to* tags are always enclosed within the *replacement* tags. We could not use any of our tagging formats because their syntax does not permit any nesting (Dai, 2018). Therefore, we decided to change the dataset itself to train the models. We considered only the tokens inside the sentences representing a replacement and tagged them using the aforementioned tagging formats. In this way, we avoided the nesting issue.

2.1 Preprocessing

Each model needs a different preprocessing method to process the raw text legal documents, depending on the feature extractor used. There are only a few preprocessing operations common to all models:

1. substitution of the special characters $\ll$ and $\gg$ with the quote marks;

2. substitution of words between quote marks with the special token `QUOTES_TEXT`. This step has allowed us to limit the number of tokens in each paragraph. The words between quote marks often represent a whole article (for example to substitute or to add). We decided to substitute these words with a special token because they are redundant for our task. This consideration permits us to improve the performances of all models. In the *from* and *to* subtask, we avoided substituting the text between quotes because it has led to a performance improvement.

3 Experiments

For each task, we gathered the documents that contain one or more occurrences of that specific modification. Then, we split the dataset into a training and a test set. More precisely, we used the 80/20 ratio adopting a stratified technique (Trost, 1986). We used the training set to validate the hyperparameters of each model. Once computed the final models, we made use of the test set to measure their generalization ability. It is important to emphasise that we never used the internal test set before the definition of the final models.
The general pipeline is composed of the following steps:

1. The annotated documents are tokenized.

2. Each token is associated with one label for each category following the tagging formats previously defined.

3. From each token, we extract its representation using either hand-crafted features or character level N-grams or word embeddings. Depending on the model used, both tagging format and features extraction change.

4. We execute the model selection phase exploiting K-fold cross-validation. In our experiments, we set the K parameter to 3 so that validation sets size is reasonable. The purpose of this step is to find the best hyperparameters of each model.

5. For each subtask, we chose the model with the best performance in the previous step.

6. After choosing the best configuration of each model, we computed and compared their performances over the test set.

3.1 Feature Extraction

We applied several feature extraction techniques to figure out which one was the most effective. In this section, we will explain these techniques with an in-depth description. Considering the nature of the task, all the features are extracted at the word level. We define different sets of features according to the models' needs. We logically divided our features into hand-crafted features, n-gram features and word embeddings.

[7] Breckbaldwin, Coding Chunkers as Taggers: IO, BIO, BMEWO, and BMEWO+, `https://bit.ly/3DzuqBc`

replacement	All'articolo 7 della decisione 2005/692/CE, la data del <replacement> « <from> 31 dicembre 2010 </from> » è sostituita da « <to> 30 giugno 2012 </to> » </replacement>.
replacement_ref	L'allegato II al regolamento (CE) n. 998/2003 è sostituito dal testo dell' < replacement_ref > allegato </replacement_ref> al presente regolamento.
addition	È aggiunto il seguente allegato: <addition> "ALLEGATO III [...]" </addition>
repeal	Il regolamento (CEE) n. 160/88 è abrogato. <repeal></repeal>
abolition	nel titolo i termini <abolition>"raccolti nel 1980" </abolition>sono soppressi

Table 2: Annotations examples

In the following we list the **hand-crafted features** extracted and their meaning:

- is_upper: boolean value indicating whether the token is in uppercase
- is_lower: boolean value indicating whether the token is in lowercase
- is_title: boolean value indicating whether the token is in titlecase
- is_alpha: boolean value indicating whether the token consists of alphabetic characters
- is_digit: boolean value indicating whether the token consists of digits
- is_punct: boolean value indicating whether the token is a punctuation mark
- pos_val_cg: coarse-grained part-of-speech from the Universal POS tag set (Kumawat and Jain, 2015): the text has been POS tagged with SpaCy Italian model[8]
- is_alnum: boolean value indicating whether all characters in the token are alphanumeric (either alphabets or numbers)
- word_lower: token in lowercase
- word[-3:]: last three characters of the token
- word[-2:]: last two characters of the token

Then, we decided to use a more complex representation. We used a **Count Vectorizer** (Sarlis and Maglogiannis, 2020) computed over all the Italian legal documents contained in EUR-Lex at the date we created it. It converts a collection of text documents to a matrix of n-gram counts. From each set of words, it produces a sparse vector representation that captures a large number (376037) of character n-grams features.
Finally, we decided to use a **word embedding** lexicon as it has been shown that provides good performances in other Italian tasks (De Mattei et al., 2018; Cimino et al., 2018). We tested a few different in-domain and general purpose embeddings lexicons trained using both fastText (Bojanowski et al., 2017) and word2vec (Mikolov et al., 2013), we obtained the best results with fastText pretrained Italian model (Grave et al., 2018).
The features extracted from each token do not contain enough information to discriminate the true amendment class. For this reason, we decided to introduce the *sliding window* concept (Dietterich, 2002). It represents a set of tokens that precede and/or follow each token, like a "window" with a fixed size that moves forward through the text. For each feature extraction technique, we introduced two parameters, `window_size` and `is_bilateral_window`. The former indicates the dimension of the window. The latter is a boolean value indicating whether the window considers only the preceding tokens (False) or both preceding and following tokens (True). For example, the sentence "È aggiunto il seguente allegato" with a bilateral sliding window of size 1, becomes ⟨(PAD, È, aggiunto), (È, aggiunto, il), (aggiunto, il, seguente), (il, seguente, allegato), (seguente, allegato, PAD)⟩ where PAD indicates the padding value. The introduction of the sliding window has made it possible to improve the evaluation metric of all models.

3.2 Models

We want to find a fully automatic approach based on the extraction of interesting features. For this reason, we developed a systematic comparison

[8]Spacy, Models, `https://spacy.io/models/it`

among three models: **Support Vector Machine (SVM)** with n-gram features, **Conditional Random Field (CRF)** with hand-crafted features and a **Neural Network (NN)** that uses word embeddings. This latter model is a rather general convolutional network architecture. The inputs of our NLP tasks are the words that compose the sliding window represented as a matrix. Each row of the matrix corresponds to the word embedding representation of one token. We decided to use a convolutional layer given its efficiency in terms of both representation and speed; it permits us to capture local and position-invariant features (Yin et al., 2017) useful for our purpose. Then, we added a Batch Normalization layer. It significantly reduces the training time in feedforward neural networks (Ba et al., 2016). During the experiment phase, we observed that layer normalization offers a speedup over the baseline model without normalization and it stabilizes the training of the model. We have also tried to use a Bidirectional Long Short-Term Memory based model with an additional CRF layer (Bi-LSTM-CRF) to solve our task (Huang et al., 2015). Its application leads to poor performance in terms of scores and speed. The results obtained show the need to solve our task using simple models that are able to discover local patterns.

4 Results

The objective of the evaluation was to define a systematic comparison among the models' performance with respect to F1 macro, precision and recall. In the model selection step, we used the F1 macro score as the evaluation metric since the frequency distribution of the labels turned out to be strongly unbalanced in all the subtasks.

After some preliminary experiments, we fixed the sliding window size and the tagging format for each model. We found that both the CRF and NN models are more inclined to use a bigger sliding window size (5) than the SVM models (1) from a performance-based perspective. We think this difference comes from the *Curse of Dimensionality* problem that could be encountered in the SVM models (Bengio et al., 2005). Concerning the tagging format, we adopted the `LL` tagging for all the models. Our experiments show that it increases the f1 score of about 20 percentage points.

Table 3 reports the mean results among the 3-fold obtained by the best configuration of each model. The CRF outperforms other models in almost all the subtasks. We think that it is due to the nature of this model. Indeed, CRFs naturally consider state-to-state dependencies and feature-to-state dependencies (Lafferty et al., 2001). Once completed the model selection phase, we chose the best model and its configuration for each subtask. We considered both the mean and standard deviation of the f1 metric among the folds. Then, we re-trained the best model on the whole training set. Table 4 reports the results and the average score of the precision, recall and F1 metrics over the internal test set. The precision score is higher than recall in all except one subtask which may be good for an application perspective.

Subtask	SVM	CRF	NN
Replacement	0.868	**0.881**	0.841
Addition	0.825	**0.852**	0.796
Repeal	0.915	**0.938**	0.924
Abolition	0.823	0.878	**0.939**
From_To	0.748	**0.873**	0.800

Table 3: Average results in terms of F1 macro score obtained in the validation phase

	Model	Prec.	Rec.	F1
Replacement	CRF	0.949	0.864	0.902
Addition	CRF	0.790	0.865	0.823
Repeal	CRF	0.937	0.912	0.924
Abolition	NN	0.951	0.912	0.931
From_To	CRF	0.977	0.841	0.899

Table 4: Precision, recall and F1 scores of the best model for each subtask

The models' performances are improved compared to the results achieved in the model selection phase, probably thanks to the larger training set provided.

5 Conclusion

We presented and analysed a machine-learning approach to the problem of the classification of textual modifications. We compared different tagging formats, feature extractor techniques and machine learning models. Our experiments show that the sliding window approach, combined with char count vectorizer or word embeddings, allows the models to capture most of the formulas that introduce textual modifications. Following Occam's

razor principle, we defined simple models that obtained good performances in all the subtasks. Our approach does not need any expertise in the law field since it tries to formalized rules to identify textual modifications. We use different NLP techniques to extract hidden features from the words inside a window.

Results validate our approach in terms of both correctness and stability. They represent the first step to build a fully automatic model capable to identify and integrates textual modifications.

References

Timothy Arnold-Moore. 1997. Automatic generation of amendment legislation. In *ICAIL '97*, pages 56–62, 01.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2005. The curse of dimensionality for local kernel machines. *Techn. Rep*, 1258:12.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Raffaella Brighi, Leonardo Lesmo, Alessandro Mazzei, Monica Palmirani, and Daniele Radicioni. 2008. Towards semantic interpretation of legal modifications through deep syntactic analysis. volume 189, pages 202–206, 01.

Andrea Cimino, Lorenzo De Mattei, and Felice Dell'Orletta. 2018. Multi-task learning in deep neural networks at evalita 2018. *Proceedings of the Wvaluation Campaign of Natural Language Processing and Speech tools for Italian*, pages 86–95.

Xiang Dai. 2018. Recognizing complex entity mentions: A review and future directions. In *Proceedings of ACL 2018, Student Research Workshop*, pages 37–44, Melbourne, Australia, July. Association for Computational Linguistics.

Lorenzo De Mattei, Andrea Cimino, and Felice Dell'Orletta. 2018. Multi-task learning in deep neural network for sentiment polarity and irony classification. In *NL4AI@ AI* IA*, pages 76–82.

Thomas G. Dietterich. 2002. Machine learning for sequential data: A review. In Terry Caelli, Adnan Amin, Robert P. W. Duin, Dick de Ridder, and Mohamed Kamel, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30, Berlin, Heidelberg. Springer Berlin Heidelberg.

Enrico Francesconi and A. Passerini. 2007. Automatic classification of provisions in legislative texts. *Artificial Intelligence and Law*, 15:1–17, 01.

John Garofalakis, Konstantinos Plessas, and Athanasios Plessas. 2016. A semi-automatic system for the consolidation of greek legislative texts. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, PCI '16, New York, NY, USA. Association for Computing Machinery.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

Deepika Kumawat and Vinesh Jain. 2015. Pos tagging approaches: a comparison. *International Journal of Computer Applications*, 118(6).

J. Lafferty, A. McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Leonardo Lesmo, Alessandro Mazzei, and Daniele Radicioni. 2009. Extracting semantic annotations from legal texts. In *HT '09*, pages 167–172, 01.

Christopher Manning and Hinrich Schutze. 1999. *Foundations of statistical natural language processing*. MIT press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Yasuhiro Ogawa, Shintaro Inagaki, and Katsuhiko Toyama. 2008. Automatic consolidation of japanese statutes based on formalization of amendment sentences. In Ken Satoh, Akihiro Inokuchi, Katashi Nagao, and Takahiro Kawamura, editors, *New Frontiers in Artificial Intelligence*, pages 363–376, Berlin, Heidelberg. Springer Berlin Heidelberg.

Monica Palmirani and Fabio Vitali, 2011. *Akoma-Ntoso for Legal Documents*, pages 75–100. Springer Netherlands, Dordrecht.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

S. Sarlis and I. Maglogiannis. 2020. On the reusability of sentiment analysis datasets in applications with dissimilar contexts. In Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 409–418, Cham. Springer International Publishing.

Pierluigi Spinosa, Gerardo Giardiello, Manola Cherubini, Simone Marchi, Giulia Venturi, and Simonetta Montemagni. 2009. Nlp-based metadata extraction for legal text consolidation. In *ICAIL*, pages 40–49, 01.

Jan E Trost. 1986. Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies. *Qualitative sociology*, 9(1):54–57.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.