Text Mining Techniques for Knowledge Extraction from Technical Documents

Filippo Chiarello

2018-09-06

Contents

Chapter 1

Introduction

1.1 Goal

Il problema non è sostituire domain knowledge. Idea vecchia ha fallito. E' insostibuibile perchè:

- Technology, interessa gli ingegneri
- Social Science, decision making

PErchè fallita: da una parte è andata avanti la knowledge rappresentation. E' impossibile rappresentare la conoscenza con regole, ma con altri strumenti si può rappresentare (bottom-up).

Inoltre ho text mining, capaità di processare testi. Parte di intelligenza artificiale. Questi fenomeni non sostituioscono l'esperto ma ne cambiano il modo di operare.

Oggi si integra. Vogliamo un esperto di dominio che faccia meglio il suo mestiere.

Abbiamo oggi più potenza e correzione errori.

Oltre ad efficienza e potenza nel correggere gli errori. Ora c'è anche la possibilità dio maggiore specificità. L'obiettivo è qiuindi poratre domain knowledge sia su technology sia ai decisori sociali.

1.2 Problem

Foresight

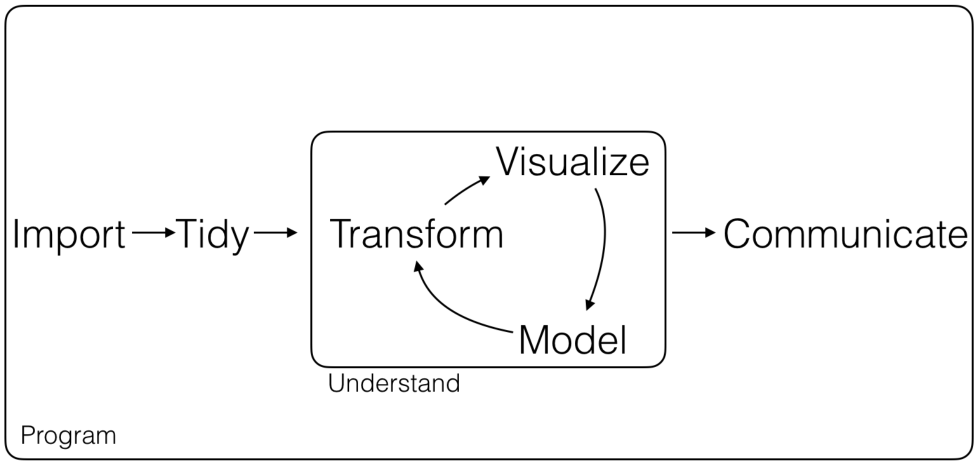

Figure 1.1: A general workflow for the process of data analysis. Readapted from Wickham (2016)

1.3 Solutions

1.4 Challenges: Understanding and Programming

1.4.1 Understanding

1.4.2 Programming

1.5 Research Questions

1.6 Stakeholders

Marketing

Research and Development

Design

Human Resources

Other Stakeholders

1.7 Knowledge Intensive Manegement Engineering

Tipicamente occupaimo di attività ad altà ripetitività. Ti porti dietro metodologie ingegneristiche applicate a sistemi inernti, andnano a operare in sistemi socio-tecnici. Hai fatto il tuo mestieri (ricerca operativa ecc..). Negli ultimi anni però le aziende le attività a maggior valore aggiunto sono non ripetitive. R&S, Design, marketing, HR ecc.. e quindi gestione della conoscena. Su situazione che sembrano uniche il gestionale rischia di perdere rispetto al creativo. Come disciplina voglio presidiare queste aree: non ci occupaimo di casi unici, ma costruire modelli in grado di incorporare conoscenza per essere usati in questi. La tesi ha l'obbiettivo di esploration and exploitation queste direzioni. Hon trovato nella data science è più nello specifico nel text mining gli strumenti adatti.

Chapter 2

State of the Art

The analysis of technical documents require the design of processes that rely both on programming and Natural Language Processing techniques and on the undestanding and knowldege of field experts. While the first techniques are codified and explicit, the second are sometimes implicit and always harder to systematize. In this section i treat these two groups of techniques in the same way to give to the reader a sistematic litterature review on these topics. For this reason the sections of this chapter has the sequent structure:

- At a first level we have two sections 2.1 and 2.2, reviewing respectivelly the processes of *programming and Natural Language Processing* and of *undestanding and knowldege of field experts application*;
- Section 2.1 has a subsection for each of the *phases* showed in figure 1.1. These subsections goes from 2.1.1 to 2.1.9;
- Each subsection from 2.1.1 to 2.1.9 contains the relative Natural Language Processing *task* that are relevant for the analysis of technical documents, for example Document Retrieval 2.1.2.1, Part-Of-Speech-Tagging; 2.1.6.3 or Named Entity Recognition 2.1.7.5.
- Each task subsection describes the relevant *techniques* to perform that task. I use the word techniques to include mainly algorythms and procedures but also more generic methods or frameworks;
- Since the second section 2.2 describes less systematics phases, task and techniques this section opens with a first subsection 2.2.1 that focuses on the studies of the problems of using expert knowledge in an analytical process and which are the techniques to convert this knowledge in a format that is usable in a Natural Language Processing workflow.
- Finally, always section 2.2 has a subsection for each of the technical *documents* I analyzed (aggiungi gancio con introduzione). These subsections goes from 2.2.2 to 2.2.7.

2.1 Phases, Tasks, and Techniques

In this section I make a review of the most important techniques for Natural Language Processing in the context of technical documents analysis. The techniques (mainly algorythms) are grouped in phases (Import, Tidy, Transform, Model, Visualize, Communicate) showed in figure 1.1 and each phases is dived in the NLP tasks that are the most important for the analysis of technical documents. The algorythms i reviewed in this section are summmarised in table tot, where the reader can see the relationship between tasks and techniques.

2.1.1 Program

- **Articoli Emily**

2.1.2 Import

- I tipi di codifica di testo
- Pachetti per import

2.1.2.1 Document Retrieval

- Letteratura query

2.1.3 Tidy

- Hadley

DTM

problems such as sparsity

2.1.4 Transform

Transforming in the context of Natural Language Processing is what in computational linguistic is called text normalization. Normalizing text means converting it to a more convenient, standard form. Most of the task of technical document analysis in fact relies on first separating out or tokenizing sentences and words, strip suffixes from the end of the word, determining the root of a word or transform the text using regular expressions.

2.1.5 Sentence Splitting

The analysis of technical documents require as first process, that the input text is segmented in sentences. Since documents do not encode this information in a non ambiguous manner (using dots) due to common abbreviations (e.g.: "Mr., Dr."), a sentence splitting process that does not rely only on a trivial *dot based* rule is required. This issue in the technical documents domain is even more problematic due to the presence of formulas, numbers, chemical entity names and bibliographic references. Furthermore, since sentece splitting is one of the first processes of an NLP pipeline, errors in this early stage are propagated in the following steps causing a strong decrease for what concerns their accuracy. One of the most advanced techniques are machine learning techniques: given a training corpus of properly segmented sentences and a learning algorithm, a statistical model is built. By reusing the statistical model, the sentence splitter is able to split sentences on texts not used in the training phase. ItalianNLP lab systems uses this approach (Dell'Orletta, 2009; Attardi and Dell'Orletta, 2009; Attardi et al., 2009). For this reason this algorythm is used for the most of the application presented in this Thesis.

2.1.6 Tokenization

Since documents are unstructured information, these has to be divided into linguistic units. The definition of linguistic units is non-trivial, and more advanced techniques can be used (such as n-gram extraction) but most of the times these are words, punctuation and numbers. English words are often separated from each other by whitespace, but whitespace is not always sufficient. Solving this problems and splitting words in well-defined tokensis defined as tokenization. In most of the application described in the present Thesis, the tokenizer developed by the ItalianNLP lab was integrated (Dell'Orletta, 2009; Attardi and Dell'Orletta, 2009; Attardi et al., 2009). This tokenizer is regular expression based: each token must match one of the regular expression defined in a configuration file. Among the others, rules are defined to tokenize words, acronyms, numbers, dates and equations.

2.1.6.1 Stemming

Stemming is a simpler but cruder methodology for chopping off of affixes. The goal of stemming is reducing inflected (or sometimes derived) words to their word stem, base or root form. The stem of a word and its morphological root do not need to be identical; it is sufficient that related words map to the same stem, even if this stem is not a valid root. One of the most widely used stemming is the simple and efficient Porter algorithm (Porter, 1980).

2.1.6.2 Lemmatisation

Lemmatization is the task of determining the root of a words. The output allow to find that two words have the same root, despite their surface differences. For example, the verbs *am*, *are*, and *is* have the shared lemma *be*; the nouns *cat* and *cats* both have the lemma *cat*. Representing a word by its lemma is important for many natural language processing tasks. Lemmatisation in fact diminish the problem of sparsity of document-word matrix. Futhermore lemmatisaion is important for document retrieval 2.1.2.1 web search, since we want to find documents mentioning motors if we search for motor. The most recent methods for lemmatization involve complete morphological parsing of the word (Hankamer, 1989).

2.1.6.3 Part-of-Speech Tagging

The part of speech plays an central role in technical document analysis since it provides very useful information concerning the morphological role of a word and its morphosyntactic context: for example, if a token is a determiner, the next token is a noun or an adjective with very high confidence. Part of speech tags are used for many information extraction tools such as named entity taggers (see section 2.1.7.5) in order to identify named entities. In typical named entity task these are people and locations since tokens representing named entities follow common morphological patterns (e.g. they start with a capital letter). For the application to technical documents, technical entities (like the possibile failures of a manufact) becomes more relevant. In this context a correct part-of-speech tagger becomes even more important since we can not rely on morphosyntactical rules. In addition part of speech tags can be used to mitigate problems related to polysemy since words often have different meaning with respect to their part of speech (e.g. “track”, “guide”). This information is extremelly valuable in patent analysis, and some patent tailored part-of-speech tagger has been designed (see section 2.2.2). The litterature on pos-tagger is huge, and goes behoind the scope of the present thesis to make a complete review. In most of the application presentend in this work, was employed the ILC postagger (Attardi, 2006). This postagger uses a supervised training algorithm: given a set of features and a training corpus, the classifier creates a statistical model using the feature statistics extracted from the training corpus.

2.1.6.4 Regular Expressions

Regular expression (regex) is a language for specifying text search strings, an algebraic notation for characterizing a set of strings. This language whidelly used in modern word processor and text processing tools.. They are particularly useful for searching in texts, when we have a pattern to search for.

A pattern could be at A regular expression search function will search through the corpus, returning all texts that match the pattern. The corpus can be a single document or a collection. For example, the Unix command-line tool grep takes a regular expression and returns every line of the input document that matches the expression. A search can be designed to return every match on a line, if there are more than one, or just the first match. In the following examples we generally underline the exact part of the pattern that matches the regular expression and show only the first match. We’ll show regular expressions delimited by slashes but note that slashes are not part of the regular expressions.

2.1.7 Model

Classi di modelli. Pedro Domingos

2.1.7.1 N-Grams

An n-gram is a sequence of N n-gram words: a 2-gram (or bigram) is a two-word sequence of words like “credit card”, “3d printing”, or ”printing machine”, and a 3-gram (or trigram) is a three-word sequence of words like “3d printing machine”. Statistical model can be used to extract the n-grams contained in a document. A first approach has the of predicting the next item in a sequence in the form of a $(n - 1)$–order Markov model(Lafferty and Zhai, 2001). The algorythm begin with the task of computing P(w|h), the probability of a word w given a word h.The way to extimate this probability is using relative frequency counts. To do that the algorythms count the number of times h is followed by the w. With a large enough corpus it is possibile to build valuable models, able to extract n-grams (Bellegarda, 2004). While this method of estimating probabilities directly from counts works for many natural language applications, in many cases a huge dimension of the corpus does make the model useful, and this is particularly true for technical documents (Brants et al., 2012). This is because technical language has a strong ratio of evolution; as new artifcat are invented, new chunks are created all the time, and has no sense to continuolly count every word co-occurrence to update our model(Gibson et al., 1994). A more usefull method for chunk extraction fro technical document uses part-of-speech-tagging and regular expression. Once a document is pos-taggerd each word is associated whit a particular part of speech: each sentence is rapresented as a sequence of part-of-spech. Once we have this rappresentation, it is possible to etract only certain sequences of part-of-speeches, the ones that whit an high level of confidence are n-grams.

[TROVA LAVORI SU QUESTO ARGOMENTO]

2.1.7.2 Document Classification

Classification is a general process that has the goal of taking an object, extract features, and assign to the observation one of a set of discrete classes. This process is largerly used for documents (Borko and Bernick, 1963) and there exist many methods for document classification (Aggarwal and Zhai, 2012).

Regardless of technological sector, most organizations today are facing the problem of overload of information. When it comes to classify huge ammount of documents or to separate the useful documents from the irrelevant, document classification techniques can reduce the proecess cost and time.

The simplest method for classifying text is to use expert defined rules, expert systems.Expert rule-based systems are programs that consist of rules in the IF form condition THEN action (if condition, then action). Given a series of facts, expert systems, thanks to the rules they are made of, manage to deduce new facts. The expert systems therefore differ from other similar programs, since, by referring to technologies developed according to artificial intelligence, they are always able to exhibit the logical steps that underlie their decisions: a purpose that, for example, is not feasible from the human mind or black box-systems. There are many type of documents for which expert based classifiers constitute a stateof-the-art system, or at least part of it. Anyway, rules can be useless in situations such as: - data change over time - the rules are too many and interrelated

Most systems of documents classification are instead done via supervised learning: we have a data set of input observations, each associated with some correct output (training set). The goal of the algorithm is to build a statical model able to learn how to map from a new observation (test set) to a correct output.

In the supervised document classification process, we have a training set of N documents that have each been tipically hand-labeled with a class: (d1, c1),....,(dN, cN). I say tipically, because other less expensive methods could be designed, as we will show for the task of Named Entity Recongition (another supervised learning task, that classifies words intstead of documents 2.1.7.5). The goal of the supervised document classification task is to learn a statistical model capable of assign a new document d to its correct class c

C. There exist a class of these classifier, probabilistic classifiers, that additionally will tell us the probability of the observation being in the class.

Many kinds of machine learning algorithms are used to build classifiers.

2.1.7.2.1 naive Bayes

2.1.7.2.2 Logistic regression

This chapter introduces naive Bayes; the following one introduces logistic regression. These exemplify two ways of doing classification. Generative classifiers like naive Bayes build a model of how a class could generate some input data. Given an observation, they return the class most likely to have generated the observation. Discriminative classifiers like logistic regression instead learn what features from the input are most useful to discriminate between the different possible classes. While discriminative systems are often more accurate and hence more commonly used, generative classifiers still have a role.

2.1.7.3 Sentiment Analsysis

2.1.7.4 Network Analysis

2.1.7.5 Named Entity Recognition

2.1.7.6 Vector Semantics

2.1.7.7 Topic Modelling

2.1.8 Visualize

2.1.9 Comunicate

2.2 Documents

2.2.1 Understand

Expertise (collins)

Sheela Jasanow

Taleb?

2.2.1.1 The problem of byases

2.2.1.2 The Importance of Lexicons for Technical Documents Analysis

2.2.1.3 Expert Systems

2.2.2 Patents

2.2.3 Papers

- Parte Barilari.Keyword base, defini i confini di area tecnologica. Hot-topics su paper (guaiè)
- Biblio

2.2.4 Projects

2.2.5 Wikipedia

2.2.6 Twitter

2.2.7 Job Profiles

Chapter 3

Methods

In this chapter I describe the methods applied for the analysis of different types of documents containing technical information. The methods are ensamble of Natural Language Processing (NLP) and Text Mining techniques described in @ref(sota_tools), re-designed depending on the analyzed document and the analysis goal.

Table tot summarise the relations between the documents under analysis (introduced in section @ref(sota_documents)) and the NLP techniques.

Table documents vs tools

Table algorithms vs tools

3.1 Patents

3.2 Papers

3.3 Projects

3.4 Wikipedia

3.5 Twitter

3.6 Job Profiles

Chapter 4

Applications and Results

Some *significant* applications are demonstrated in this chapter.

4.1 Patents

4.2 Papers

4.3 Projects

4.4 Wikipedia

4.5 Twitter

4.6 Job Profiles

Chapter 5

Future Developments

5.1 Marketing

5.2 Research and Development

5.3 Design

5.4 Human Resources

Chapter 6

Conclusions

We have finished a nice thesis

Chapter 7

Glossary

Morphology= the study of the way words are built up from smaller meaning-bearing units called morphemes.

Bibliography

Aggarwal, C. C. and Zhai, C. (2012). A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

Attardi, G. (2006). Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 166–170. Association for Computational Linguistics.

Attardi, G. and Dell'Orletta, F. (2009). Reverse revision and linear tree combination for dependency parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 261–264. Association for Computational Linguistics.

Attardi, G., Dell'Orletta, F., Simi, M., and Turian, J. (2009). Accurate dependency parsing with a stacked multilayer perceptron. *Proceedings of EVALITA*, 9:1–8.

Bellegarda, J. R. (2004). Statistical language model adaptation: review and perspectives. *Speech communication*, 42(1):93–108.

Borko, H. and Bernick, M. (1963). Automatic document classification. *Journal of the ACM (JACM)*, 10(2):151–162.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (2012). Large language models in machine translation. US Patent 8,332,207.

Dell'Orletta, F. (2009). Ensemble system for part-of-speech tagging. *Proceedings of EVALITA*, 9:1–8.

Gibson, K. R., Gibson, K. R., and Ingold, T. (1994). *Tools, language and cognition in human evolution*. Cambridge University Press.

Hankamer, J. (1989). Morphological parsing and the lexicon. In *Lexical representation and process*, pages 392–408. MIT Press.

Lafferty, J. and Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119. ACM.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.