Appunti di
Information Retrieval

A cura di:
Francesco Refolli
Matricola 865955

Anno Accademico 2024-2025

Chapter 1

Introduction to Information Retrieval

1.1 Introduction

Predictive Text Analysis a computer program that automatically recognize or detect **a particular concept** within a text.

Explorative Text Analysis a computer program that discovers **patterns or useful trends** in a collection of texts.

Opinion Mining a program that detects whether an opinionated text expresses a positive or negative opinion over an argument.

Sentiment Analysis a program that detects emotional state of the author of a text.

Bias Detection a program that detects whether the author of a text favors a specific viewpoint.

Information Extraction detects that some words refer to a specific entity within a text.

Relation Learning detects that some relation instates between two entities in a text.

Text-driven Forecasting monitors incoming text (as stream) making predictions about events or properties.

Temporal Summarization summaries a stream of text.

Structured Text Formal languages often have a structure which has some semantic relevance and meaningful data can be extracted within this structure (the abstract field of a paper).

1.2 Basic Text Processing

Document The unit of textual resource is the **word** which may have a structure (logical or physical, or whatever), some medias, and some metadata. A term is meaningful data; a word is lexical term which could appear in a real world language; an index is used to retrieve faster or link documents.

Descriptive Metadata Is related to the document as unit (TODO: example).

Semantic Metadata Is related to the content of the document (TODO: example).

1.2.1 Text Pre-Processing

How to represent a text? tokenization (and lemmization) of text. The units of informations should be words, since sentences or whole documents don't offer sufficient granularity in order to extract meaningful entities and relations. Words are the basic "features" of the text (leads to Word Embedding).

Tokenization splitting a stream of characters into unit of information (lexing rules strictly depends on the language, pictograms != wordlings != soundcarriers). Each token is a candidate for being a meaningful terms (index, key ...).

Normalization establishing rules in order to have the same notation for the same entity (ex: U.s.A -> USA).

Stemming reducing variations of words into the same form (ex: authorization -> authorize), but is optional.

Noise removal erasing useless or noisy words (ex: conjunctions, articles, prepositions, punctuation), optional. It may be an advantage when dealing with the implementation of a search engines (reduces the size of the language of keys).

Types and Tokens just as in lexers, the type is an abstract kind of token, while the latter is just an instance of a type.

The *vocabulary* is constructed on the corpora, it's useless to use a huge vocabulary from the entire language X.

Heaps/Herdan's law There's a general empiric law for size of vocabulary $|V| = kN^{\beta}$, where $0.67 < \beta < 0.75$, and N is the number of words in the text.

Corpora words are part of a context which is the corpus (further the corpora), and has an author, a time/space location, a target language/audience, a function. A document may be written in more than one language/slang. with or without junctions.

Segmentation of Sentences identification of boundaries of sentences and text with punctuation.

Clitic a word that is represented with punctuation and other symbols (ex: *we're* contains implicit are).

Language issues in complex languages like German, as well as when using pictograms, dividing words with a compound splitter is useful. Some languages also don't use space or punctuation as separation of words or sentences. Inaccurate splitting may end changing the meaning of the sentences.

1.2.2 Word Tokenization

Alternative tokenization methods

- Byte Pair Encoding
- Unigram language modeling tokenization
- WordPiece

All of these are statistical methods and have a **token learner** which learns the vocabulary from the data, and the **token segmenter** which applies this vocabulary to a random stream of text. Also regular expressions could be used to lex tokens, as done with compilers.

Chapter 2

Text Representation

2.1 Basics

Incidence Matrices The most simple representation is the **bag of words**. Putting on more formal representations, we see the **Incidence Matrix** in which the *columns* are the documents, and the *rows* are the words in the vocabulary. Each cell represents if a word appears in the document, or the number of occurrences of such word in the document (**Count Matrix**). Of course this matrix should be a sparse matrix.

Bag of Words with N-grams An **N-gram** is a continuous sequence of N tokens from a document (ex: "Sistema di Controllo Marcia Treno", 2-grams are: "Sistema_di", "di_Controllo", "Controllo_Marcia", "Marcia_Treno"). This concept is somehow used also in computational genetics to represent alignments, because this way is possible to retain information about relative position of words. Of course it comes with the cost of space and parametricity.

Statistics The *Zipf's law* says that the product of frequency $f(W)$ and rank order $r(w)$ of words is somewhat constant. So one could write $f(w) = \frac{1}{r(w)} = \frac{K}{f(w)}$ for some arbitrary K constant value. Words with most occurrences usually are stop words or meaningless interjections. Also is possible that some words appear too often and don't offer advantages when used as indices, while also extremely rare words usage could be very limited (depending from the context and frequency they may be removed). However erasing non-informative and useless words is optional.

Luhn's Analysis Each word is associated to a weight, where very rare/frequent words have very low weight (and could be eliminated). Word occurrences may be normalized by dividing the counter by the total number of words in a document or by the maximum counter for that word in the collection. Another weight could be driven by $log(\frac{N}{df_t})$, the logarithm of N (the number of documents) and df_t the number of documents in which a word appears, so that when a word is extremely common, the value reaches 0.

Chapter 3

POS, NER, NLP

3.1 Part of Speech Tagging / Named Entity Recognition

POS Marking words in a text with the role in that sentence / text (ex: "chasing" -> "Verb", is" -> "auxiliary"). «Word classes may be classified as open or closed: open classes (typically including nouns, verbs and adjectives) acquire new members constantly, while closed classes (such as pronouns and conjunctions) acquire new members infrequently, if at all» [1].

We need a reference for tagging: Penn Treebank has a set of tags which annotate a corpus. The may be some ambiguity within a tense due to context which influences the meaning and role of a term. Well, many words can only be assigned to only one TAG, others have a most-probable TAG which is by far the best. There may be also correlations between a term being a TAG and another TAG/word appearing nearby. Ambiguity is thus solved with probability and statistics.

3.2 Statistical Language Model

A Language Model is a kind of representation for a text (based on probability distributions) by modelling a sequence of words as an event and assigning a probability to it. If such model is used to generate new data from a collection, then it's called also Generative Model. The probability for a sentence is $P(Ws) = P(w_1, w_2...w_n)$ and the probability of an upcoming word (Prediction) is $P(w_{n+1}|w_1, w_2...w_n)$. A model which computes such probabilities is thus called *Language Model*. An *N-gram* is a tuple $(w_n, (w_1, w_2...w_{n-1}))$ where i compute probabilities as $P(w_n|w_1, w_2...w_{n-1})$. A Language Model is *well-formed* over an alphabet *Sigma* iff $\Sigma P(S) = 1$ with $S \in \Sigma^*$. With the *chain rule* we apply $P(w_n|w_1, w_2...w_{n-1}) = P(w_n|w_{n-1}, w_{n-2}) * P(w_{n-1}|w_{n-2}, w_{w-3}))...etc.$

When using a Language Model, a different model is used for each document.

Smoothing Is a technique in which all the probabilities of words seen in the document are lowered by some amount in order to create a difference from the total probability of the words and 1. This difference is reassigned (equally divided) to the words which didn't appeared. In such a way there aren't $P(W) = 0$ for any word W (preventing zero denominator in the maximum likelihood estimation formula $P(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$).

Perplexity is the multiplicative inverse of the probability given by a language model on a test set which has not been seen before. It's useful to evaluate multiple language models. A language model which is less surprised (perplexed) is the one which is more accurate. In the formula there is a normalization factor (which applies the root-by-N) where N is the number of words. For obvious reasons, n-grams with more N are the ones with less perplexity.

Chapter 4

Named Entity Recognition

definition is the task of identifying the class or relationships between names in a text.

Example: «Professor Pajeet doesn't speak very well»; here, "Pajeet" is a **Person**. Both POS and NER are tagging a sequence of tokens, but if POS works on the lexical level (recognizes grammatical syntax, like a Parser), the NER is working on the semantics level (recognizes roles, meanings, relationships, like a Type Checker).

Chapter 5

IR Models

The role of an IR system is to assess **relevance** (topical). There isn't a single model used in practice, but a mixture of more models.

5.1 Boolean Model

It is based on set theory. A document is formally represented as a set of index terms and binary weights are associated with those index terms: $R(d_j) = t_j | w_{ij} == 1 | \forall wij \in 0, 1$.

A query is formally represented as a Boolean expression on terms.

- AND is Set Intersection
- OR is Set Union
- NOT is Set Difference
- ADJ/NEAR is a Context search

The matching mechanism applies set operations. Relevance is modeled as a binary property of documents (Retrieval Status Value either 0 or 1).

When performing exact matching, the representation is inverted ($R(d_1) = t_1, t_2, t_3 \rightarrow R(t_1) = d_1, d_2$) so that i can apply set operations on $R(t_j)$ values.

Queries are performed with Lazy Evaluation (TODO: recursive and dynamic bottom-up matching algorithm).

5.2 Vector Space Model

It's based on n-dimensional vector space (where n is the size of the vocabulary). I assume linear independence of terms to simplify computation. A document is a linear combination of the terms which appear: $d_j = \Sigma_{i=1}^{N} w_{ij} * t_i$. Without the assumption of term independence, the cosine similarity of terms should be computed. Queries that want to elaborate OR, AND and so on, must use distinct queries and then join the results.

5.3 Binary Independence Model

The Odds of an event A is the ratio $\frac{P(A)}{1-P(A)}$. Search engines can rely on *relevance feedback* (the user marks the found documents with whether they were relevant or not). Some

assumptions: documents and queries are represented as binary vectors, and query terms / documents are assumed to be Bayes independent.

The probability that the document x is relevant with this query q: $P(R = 1|x) = \frac{P(x|R=1,q)P(R=1,q)}{P(x,q)}$. Then using the concept of odds we compute the Retrieval Status Value: $RSV(x,q) = \frac{P(R=1|x)}{P(R=0|x)}$. The initial rank uses $P(x|R = 1) = 0.5$ and $P(x = R = 0) = \frac{n}{N}$ (the rate of appearance for that term in documents of the collection). Then the probabilities are improved with $P(x|R = 1) = \frac{V_t}{V}$ (the number of truly relevant documents over the number of documents) and $P(x = R = 0) = \frac{n-V_t}{N-V}$.

Bibliography

[1] Wikipedia contributors. *Part of speech — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-October-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Part_of_speech&oldid=1251632448.