Spotle.ai Study Material
Spotle.ai/Learn

Medical Reports Standardization

Document Clustering

Part 1

Objectives

It takes, on an average, 3 months to write a single Clinical Study Report (CSR) document. Over 40% of the text in the CSR document is taken as it is from various input sources. Multiple resources (Internal + CROs) are involved in writing CSR. One of the biggest challenges in medical writing is inconsistency of the text and language across the CSRs from the same Pharma company.

Clinical research companies such as Johnson and Johnson, Novartis, Pfizer and many many other clinical research companies need to submit thousands of reports in different phases of drug development process to regulation authorities such as Food and Drug Administration (FBA) in USA or European Medicines Agency (EMA). It is obvious that drug development process is a highly regulated process. The regulation authorities demand and accept reports only in specific formats. And it's a mammoth work to put thousands and thousands pages of documents into specific formats. Based on their thousands old reports clinical research companies are aiming to create standard templates that will categorise words, sentences, intents etc for such reports. This will reduce 60% of reporting time and hence the cost.

The objective of this study is to create a state of the art AI framework that helps find similar clinical content from various input source documents and allows you to create/manage standard clinical content database.

Introduction

With the rapid increase in internet resources, Natural Language Processing (NLP), including document clustering, has become a widely researched topic for many decades. Many models have been built to handle natural language processing, such as N-gram and Hidden Markov Model. All of these are used to represent documents via extracting the semantic features. But these methods suffer from high dimension problem. For example, if we convert strings with only letters from the English alphabet, which has 26 letters, into single character 3-grams, we get a 26^3 - dimensional space. The first dimension measures the number of occurrences of "aaa", the second dimension measures the number of occurrences of "aab", dimension measures the number of occurrences of “aac” and so on for all possible combinations of three letters).

Word2vec is considerably a new method which embed each word into a low dimensional continuous vector space in which words can be easily compared for similarity. It is capable of capturing context of a word in a document, finding semantic and syntactic similarity and relation with other words etc.

This case study consists of three steps. First, create a new vector space model which is suitable for medical reporting. Since Word2vec model gives vector representation of a word, an additional functionality is required to compute similarities of sentences. And finally use cluster analysis for grouping of sentences.

Background

The following chapter briefly covers the foundations needed to understand the work. We will explain the basic classical notions of natural language processing (NLP) in a bottom-up approach, starting at character level and building up to understanding the meaning at a more abstract level.

Preprocessing

Texts in the real world are highly subjective to deal with.

As an example, the sentence "Tomorrow you are going to come to my house" may have many interpretations. The following sentences will mean many things to our interpretation depending upon where the focus is when we say this sentence.
"**Tomorrow** you are going to come to my house. *This means when a meeting event is happening*."
"Tomorrow **you** are going to come to my house." *This means who is responsible to move for the meeting event.*
"Tomorrow you are going to come to **my house**." *This means where the meeting event is happening.*
"Tomorrow you are going to come to my house**?"** *This raises a question about the meeting event.*

Moreover, “Tomorrow, my house, you are going to come” - means the same thing as “Tomorrow you are going to come to my house”. “My house, tomorrow” - would also mean the same thing where ‘you are going to come is meant automatically without even mentioning it.

It so difficult to categorize sentences just by reading the text.

It is understood that language is highly inconsistent, even in the most formal settings. In this section we will cover the standard NLP tool-set that is used to transform text from a raw sequence of characters to a form that is cleaner and easier to deal with computationally.

Tokenization

Tokenization is the task of splitting the text into more meaningful character sequence groups, usually words or sentences. It is almost always the first step in any NLP pipeline. Characters are hard to interpret by a computer on their own, but words are mostly self-contained semantic units. It is a comfortable level of abstraction to work at, a lot of NLP operations act directly on words. A naive approach to tokenization would be simply to split by spaces and remove any punctuation. It is a good baseline, but not ideal. Even for common English there is a number of tricky cases. For example, the use of apostrophe for contractions. Should "aren’t" be a single token or two ("are", "n’t"), meaning "are not". What about the different cases of hyphenation like "co-occurrence" or composite borrowed names like "New Delhi", or "Los Angeles". Such corner cases are language and domain specific. NLP libraries like NLTK offer a variety of language specific tokenizers that cover most of the corner cases. Each implementation has different priorities and one can choose the most appropriate one for each use-case.

Parsing

After isolating and cleaning words, it is common to add yet another layer of processing to be able to work at higher abstraction levels. Parsing encompasses a set of annotation tasks to identify the role of each word with respect to its context in some text, usually a sentence. Parsing is usually the core of NLP tasks that requires deep understanding. POS tagging or Part-Of-Speech tagging is the computational equivalent of morphological linguistic analysis. *In linguistics, morphology is the study of words, how they are formed, and their relationship to other words in the same language. It analyzes the structure of words and parts of words, such as stems, root words, prefixes, and suffixes. Morphology also looks at parts of speech, intonation and stress, and the ways context can change a word's pronunciation and meaning. - Wiki*

POS tagging annotates each token with a morphological class, say substantive, adjective or verb. Some taggers also go deeper and specify substantive properties, such as plurality, or verb tenses.

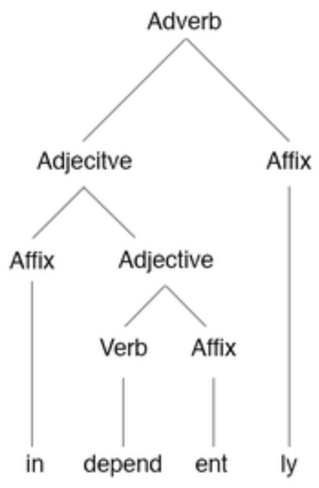

Figure 1 - Morpheme-based morphology tree of the word "independently" (Wiki)

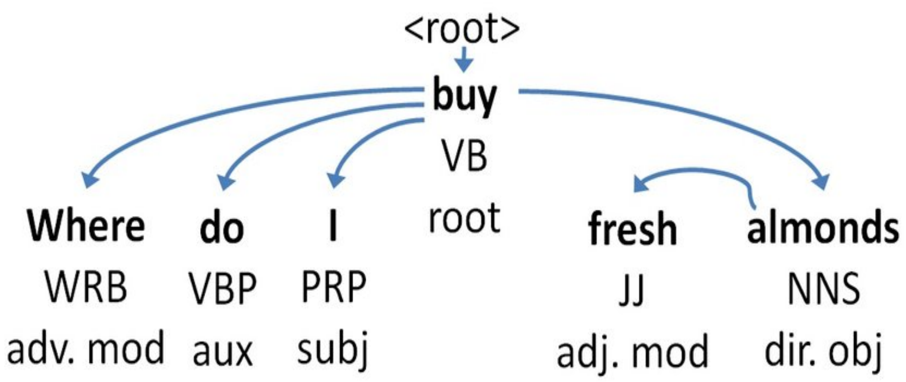

Figure 2 - POS tagging and dependency parse tree for the question "Where do I buy fresh almonds?" (researchgate.net)

Syntactical analysis is the second step of parsing. Much like in linguistic analysis, syntax follows after morphological analysis. The task here is to determine the role of each word in the sentence and capture dependencies between each component. This is an even harder challenge than POS tagging. However, complete syntactical analysis is not required for many use-cases. For example, probabilistic models that extract subject verb-object triplets have been effective for a while. Such triplets are already extremely useful for most information extraction tasks.

Embeddings
Embeddings are numerical representations of semantic units. The most common form of embeddings are word embeddings. Word embeddings assume that there is no ambiguity. So they end up capturing multiple senses in the same vector representation, which is not ideal. Sense embeddings are much harder to train where word embeddings can be efficiently learned just by scanning big corpora. There are sense embedding algorithms that utilize some form of clustering to distinguish senses, either during or after training.

Vector Space Model
The Vector Space Model (VSM) is a model for representing documents in algebraic form. It is a widely accepted standard that is used whenever a numerical representation of text is needed. The basic idea of the VSM is to represent text as a Bag of Words (BoW). In order to have a compact representation, the ordering of the words in a document is ignored and the document is represented by a vector of word frequencies.

Word Embeddings

Word embeddings are vector representation of the meaning of words. In practice, this usually means that word embeddings are placed in a high dimensional space where the embeddings of similar or related words are close to each other and different word embeddings are placed far from each other. Word embeddings also acquire more complex geometric structures as a side effect of some algorithms. A typical example for this is real world analogies that can be discovered using simple vector arithmetic:

king - man + woman = queen

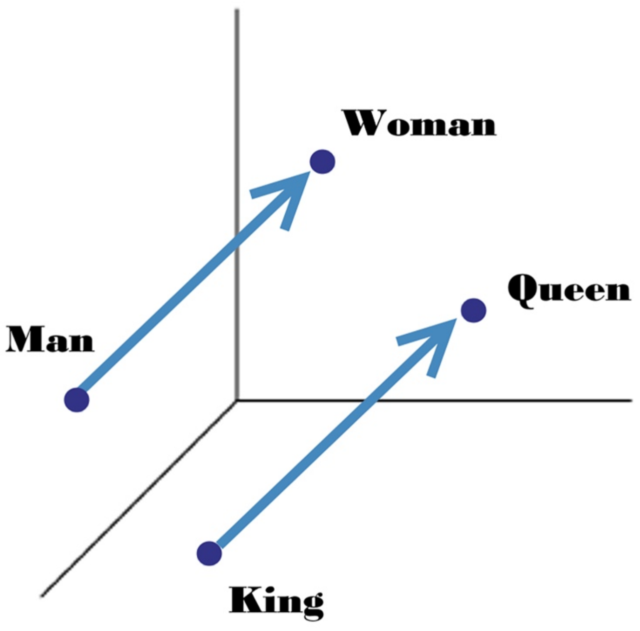

Figure 3 - king - man + woman = queen

Word or sense embeddings can be trained on knowledge graphs. But the most popular algorithms learn these vector representations just by scanning big corpora. All of these algorithms rely on a single assumption: **words that appear in similar contexts have similar meanings**. Most state-of-the-art corpus word embedding algorithms are variations of the original Word2Vec. Word2Vec and most other embedding algorithms do the factorization implicitly. They do this by scanning the corpus with a fixed sized window. The window has a central word, called the target and a few neighboring words that are called the context. The goal is to minimize the distance, usually dot product, between words and their contexts. Each time a target word is found with a context, their vectors are pushed together slightly, depending on the learning rate. These models are very simple and, therefore, very efficient. They can run through very big corpora and they only need a few scans to achieve convergence. This is precisely the key to the success of this new wave of embedding algorithms. Shallow models like these win over more complex deep neural models by being way faster and by consuming far more training data. Although technically senses are the purest form of semantic unit, it is often assumed that there is a one-to-one correlation between words and senses and word embeddings are used as the basic input for many tasks. This is a reasonable assumption, as polysemy and synonymy are relatively rare. It is also far easier to train word embeddings using the framework described above. Word embeddings can then be composed into more abstract structures, such as phrases, sentences, paragraphs or documents. Compositionality is still an open problem, the state-of-the-art is fairly narrow in this task. A lot of unsupervised sentence and document embedding algorithms still use a very similar framework to word embedding algorithms inspired by the original Word2Vec. In fact, some document embedding algorithms are specialized word embedding algorithms that optimize word embeddings such that just averaging them or performing a similar simple composition provides meaningful document embeddings.

Word2Vec

Word embeddings have a long history in academic. The Neural Network Language Model was a very influential work. It is a neural architecture that simultaneously learns word embeddings and a statistical language model. Over the last two decades, many iterations on this seminal model have been proposed, such as the replacement of a simple feed-forward network with an RNN. However, word embeddings did not take off until 2013. In 2013 Mikolov, K. Chen and others proposed Word2Vec based on two very simple log-linear models that outperformed all previous complex architectures. Most importantly the proposed models drastically reduced the time complexity. The newly increased scalability made it possible to use much bigger corpora, which opened the door to more accurate embeddings, embeddings that could reliably be used as the basis for all kinds of NLP models. Since the publication of the article in 2013, word embedding has become the foundations of modern deep-NLP. Ever since there have been many more proposals to improve on Word2Vec, but the models proposed by Mikolov, K. Chen and others have been proven to be a solid baseline and is still used regularly as the default source of embeddings.