Chapter 6
Text Mining with Unstructured Text

Abstract This chapter introduces computational techniques that enable us to extract concepts, relations, and other patterns from text documents, and from scientific publications in particular. After targets of interest have been extracted and annotated, text mining techniques can be applied to identify higher-order patterns and trends that may not be obvious from individual documents. The basic concepts and the general procedure for applying these tools to the study of scientific publications are explained with illustrative applications.

Natural Language Processing

The ultimate goal of Natural Language Processing (NLP) is to capture meaning from an input of words (sentences, paragraphs, pages, etc.) in the form of a structured output (which varies greatly depending on the application) so that further analytics can be applied to the output of NLP.

There are a variety of approaches for NLP, which can be classified into three approaches: symbolic, statistical, and hybrid. The symbolic approach to NLP is based on human-developed rules and lexicons, which is based on a set of accepted rules of speech within a given language that are materialized and recorded by linguistic experts for computer systems to follow. The statistical approach is based on observable and recurring examples of linguistic phenomena. Models based on statistics recognize recurring themes by mathematical analysis of large text corpora. The system can develop its own linguistic rules that it will use to analyze future input and/or the generation of language output by identifying trends in large samples of text. The hybrid approach is a combination of the symbolic and statistical approaches. This approach starts with generally accepted rules of language and tailors them to specific applications from input derived from statistical inference.

C. Chen and M. Song, *Representing Scientific Knowledge*,
https://doi.org/10.1007/978-3-319-62543-0_6

Modeling and Analytic Tools

There are many open source as well as commercial NLP tools. In Table 6.1, we only listed some of well-known, open source tools that are pertinent to the objectives of this book.

Table 6.1 A list of well-known, open source NLP tools

Tool	Description	Platform
Stanford's CoreNLP	A pipeline framework of tools for processing English, Chinese, and Spanish. Includes tools for tokenization (splitting of text into words), part of speech tagging, grammar parsing (identifying things like noun and verb phrases), named entity recognition, sentiment analysis, and more. There are several spin-off projects based on Stanford's CoreNLP http://nlp.stanford.edu/software/corenlp.shtml	Java
GATE and Apache UIMA	GATE combined with UIMA provides a placeholder for building complex NLP workflows which need to integrate several different processing steps. In these cases, a framework like GATE or UIMA is a good option for standardizing and abstracting much of the repetitive work that goes into building a complex NLP application https://gate.ac.uk/	Java
Natural language toolkit	Similar to the Stanford CoreNLP, it includes capabilities for tokenizing, parsing, and identifying named entities as well as many more features http://www.nltk.org/	Python
Apache Lucene and Solr	While originally targeted at solving Information Retrieval problems, Lucene and Solr contain a number of powerful tools for working with text ranging from advanced string manipulation utilities to powerful and flexible tokenization libraries to blazing fast libraries for working with finite state automatons http://lucene.apache.org/	Java
Apache OpenNLP	Using a different underlying approach than Stanford's CoreNLP, the OpenNLP project is an Apache-licensed suite of tools to do tasks like tokenization, part of speech tagging, parsing, and named entity recognition. While not necessarily state of the art anymore in its approach, it remains a solid choice that is easy to get up and running http://opennlp.apache.org/	Java
ScalaNLP	ScalaNLP is the umbrella project for several libraries, including Breeze and Epic. Breeze is a set of libraries for machine learning and numerical computing. Epic is a high-performance statistical parser and structured prediction library http://www.scalanlp.org/	Scala

(continued)

Table 6.1 (continued)

Tool	Description	Platform
Snowball	Snowball is a string processing language designed for creating stemming algorithms for use in Information Retrieval in many different languages including English, French, Spanish, Portuguese, Italian, Romanian, German, Dutch, Swedish, Norwegian, and Danish http://snowball.tartarus.org	Java C
Deeplearning4j	Deeplearning4j is designed to be used in a big scale setting in business environments, rather than as a research tool. It is a Java-based, industry-focused, commercially supported, distributed deep-learning framework https://deeplearning4j.org/	Java
Torch	Torch is written in Lua, and used at NYU, Facebook AI lab and Google DeepMind. It claims to provide a MATLAB-like environment for machine learning algorithms. Lua is easily to be integrated with C so within a few hours' work, any C or C ++ library can become a Lua library." With Lua written in pure ANSI C, it can be easily compiled for arbitrary targets http://torch.ch/	Lua
TensorFlow	TensorFlow is an open source library for numerical computation using data flow graphs (which is all that a Neural Network really is). Originally developed by the researchers on the Google Brain Team within Google's Machine Intelligence research organization, the library has since been open sourced and made available to the general public https://www.tensorflow.org/	Python

Information Extraction

Information extraction (IE) is a research topic to automatically extract target information from unstructured text. IE is involved in two major tasks: entity extraction and relation extraction. Extracting entities such as people, organizations, locations, times, dates, prices, etc. from unstructured text. Entities are objects that often the major nouns in texts. Extracting relations is associated with identifying the relation between two entities. Example relation types are located in, employed by, part of, and is associated with.

Extracting Entities from Text

Extracting entities is mainly studied in the field of Named Entity Extraction (NEE) or Named Entity Recognition (NER). The NER problem is a tagging task, similar to part-of speech (POS) tagging. Thus, if entity extraction is carried out by a supervised learning approach, the task is typically uses sequence classifiers like

Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs). In that case, features used to train classifier usually include words, POS tags, word shapes, orthographic features, gazetteers, etc.

With the huge amount of accessible biomedical literature nowadays, extracting entities from the literature has been receiving more and attention. Entity extraction from biomedical literature can be used to automatically extract useful biomedical information, particularly those key concepts dealing with genes, proteins, diseases and associations among them. The information extracted from biomedical literature has notable potential to automate database construction in biomedicine, with minimal human effort. Entity extraction can also be useful in other areas. Query suggestion, for instance, is another important application area, where concepts can be output as correction suggestions for misspelled queries. Entities are also widely used for text categorization tasks. Most of text categorization techniques are based on word and/or phrase analysis of the text. It has been shown that concept-based text categorization can help improve the precision of clustering documents by topic. Also entity extraction is important and useful in areas like automatic text summarization, information retrieval, question answering, and so forth.

There are various approaches to handle the entity extraction problem. Most of them are based on statistical features such as word counting, inverse document frequency (IDF) as well as semantic features. Attempting to automatically extract useful biomedical information from web accessible biomedical literature, particularly the key concepts dealing with genes, proteins, drugs and diseases and associations among these concepts, Fu et al. (2002) developed a system called VCGS (Vocabulary Cluster Generating Systems) that automatically extracts and determines associations among tokens from biomedical literature. They used three local databases to validate tokens extracted that are gene names or protein fragments. Both statistical and semantic features were used for token extraction. They proposed a clustering algorithm to identify specific groups of tokens, collectively represented as centroids, which are different from each other in terms of their separation as individual clusters. Similarly, Shehata et al. (2007) exploited the semantic structure at both sentence and document level. Their model combined the selected statistical features and the conceptual ontological graph representation that they built. Majoros et al. (2003) proposed a method of improving the quality of automatically extracted noun phrases by employing prior knowledge during the hidden Markov model (HMM) training procedure for the part-of-speech tagger. They modified the basic Markov model tagger with states corresponding to part-of-speech tags and an alphabet of symbols corresponding to individual words.

External ontologies and thesauri are also widely used for concept extraction tasks in biomedical domain. Rindflesch et al. (2000) developed a system that extracts information about drugs and genes relevant to cancer from the biomedical literature. Two external ontologies were used to build their system: the MEDLINE database of biomedical citations and abstractions and the Unified Medical Language System (UMLS), which provides syntactic and semantic information about the terms identified in the biomedical abstracts. Zhou et al. (2006) introduced an approximate dictionary lookup technique to capture significant words rather than

```
The [European Commission ORG] said on Thursday it
disagreed with [German MISC] advice.
Only [France LOC] and [Britain LOC] backed
[Fischler PER] 's proposal .

"What we have to be extremely careful of is how
other countries are going to take [Germany LOC]
's lead", [Welsh National Farmers ' Union ORG]
( [NFU ORG] ) chairman [John Lloyd Jones PER]
said on [BBC ORG] radio .
```

Fig. 6.1 An example of entity extraction results for location, organization, and person

all words in a concept name. They also used UMLS as the dictionary to train the significance score of each word to biological concepts containing that word. A set of simple rules were applied to identify the boundary of a concept candidate and their experimental results show that their approach can dramatically improve the extraction recall while maintaining the precision.

Given unstructured text, the goal of a NER tool is to tag the sequence of words denoting a target entity type. For instance, the below example show the results of NER tagging. For the task of extracting four types of entities such as ORG, LOC, PER, MISC, the below examples shows that a NER classifier tags the words or phrases predicted to be an entity (Fig. 6.1).

Extracting Entities from Biomedical Literature

To demonstrate the process of entity extraction, we use our concept extraction system, which is publicly available at http://informatics.yonsei.ac.kr/tsmm/uncertainty_book/ConceptExtraction.zip.

First, we start with a parsing procedure using biomedical Named Entity Recognition (NER) software to extract key entities from the input text. In particular, we utilize Lingpipe's NER API and a statistical model trained on the Genia corpus. In addition, we use a biomedical domain ontology to map the extracted entities into concepts. We choose the Unified Medical Language System (UMLS) installed on our server as a MySQL database of biomedical concepts and relationships between them. UMLS offers a semantic network that allows retrieving higher level semantic types of a 'is-a' link nature. Since such semantic types are usually general enough for humans to interpret, we will use them in the final stages of the algorithm to extract meaningful concept descriptions from text documents.

The mapping is based on matching entities to corresponding concept strings used as concept labels in the database. We only use exact string matching because most named entity strings are defined exactly the same way as concept labels in the database. Although this leads to a lower recall, the precision remains very high. The number of concepts matched is usually sufficient for building a good graph representation of a document.

After mapping the extracted entities, the next step is to build concept graphs. Nodes in a graph represent concepts in a document. Edges represent their relationships. For each concept node, we search for additional related nodes so as to enhance the concept extraction process later on. We query the UMLS database, resultant concepts are added as new nodes to the graph unless they already exist, in which case we add the relation only.

During this process, some concepts might occur repeatedly as commonly related concepts. We keep track of the occurrence count and use it later on in a weighting scheme. At this point, graphs include many concept nodes that may or may not be related to one another. Within the same graph, the domain similarity of concepts varies since a single graph may include more than one general idea from the text. We then group similar concepts using the k-Medoids method to compact a group of concepts into the most representative one for the final extraction phase.

k-Medoids is a variant of the popular k-Means clustering algorithm. The main difference is that in k-Medoids, the center chosen for each cluster is one of the existing data elements in the set as opposed to finding a mean value for each cluster. We divide each graph into k clusters, where k is chosen so that it matches to either the number of author-provided keyword labels or the number of the top labels generated by the KEA package (http://www.nzdl.org/Kea).

Initially, concepts are added to the clusters randomly. In the following steps, the algorithm tries to find a better medoid candidate for each cluster based on concept weights. The weights are calculated for each concept based on (1) the average distance to all other nodes in the cluster and (2) the concept occurrence count mentioned earlier. The distance between two concepts is a compound value based on their text similarity and the relationship between them. Next, the distance between each node and the medoid of each cluster is calculated. If a node is closer to another cluster other than the one it currently belongs to, it will be placed in the new cluster. This process is repeated until all medoids in the graph are fixed. The process is summarized as follows:

1. Apply NER to extract biomedical named entities
2. Map entities to UMLS concepts using string matching and add concept weights
3. Add related nodes (parents and synonyms)
4. Use k-Medoids to group the top k concepts and extract the medoids

- Node Distance is calculated based on text similarity and on relationships
- Concept occurrence frequency is also used in the medoid calculation score

Supposed the sample text from a text file called input.txt contains the following sentences:

> The occurrence of subsequent neoplasms has direct impact on the quantity and quality of life in cancer survivors. We have expanded our analysis of these events in the Childhood Cancer Survivor Study (CCSS) to better understand the occurrence of these events as the survivor population ages.

Use the following command to extract named entities from the text. In addition, the graph object is generated in the concept extraction project directory for visualization.

```
java -Xms64m -Xmx12550m -cp .:./bin:./lib/* ce.Main4.
```

The extracted named entities are listed as follows:

```
processing input.txt
named entitiy: subsequent neoplasms(subsequent neoplasms)
named entitiy: cancer survivors(cancer survivors)
named entitiy: Childhood Cancer Survivor Study(Childhood Cancer Survivor Study)
adding related…
Concept: childhood cancer survivor study
named entitiy: CCSS(CCSS)
adding related…
Concept: ccss
named entitiy: survivor population(survivor population)
named entitiy: neoplasms(neoplasms).
```

Extracting Relations from Text

An important step to understand human natural language automatically is relation extraction. If we can turn unstructured text into structured by annotating semantic information in a programmatic way, knowledge buried in the sheer volume and heterogeneity of data can be available to create new values for humanity. The reliable, accurate relation extraction is not a trivial task.

Examples of relations are person-affiliation and organization-location. Existing named entities recognizers (NER) (e.g., Bikel et al. 1999; Finkel et al. 2005) can automatically label data with high accuracy. However, the computer needs to know how to recognize a piece of text having a semantic property of interest in order to make a correct annotation. Thus, extracting semantic relations between entities in natural language text is an important step towards natural language understanding applications.

A relation is defined in the form of a tuple t = (e1, e2, …, en) where the ei are entities in a pre-defined relation r within document D. Most relation extraction systems focus on extracting binary relations. Examples of binary relations include located-in (CMU, Pittsburgh), father-of (Manuel Blum, Avrim Blum). It is also possible to go to higher-order relations as well. For example, in the sentence "At codons 12, the occurrence of point mutations from G to T were observed" exists a 4-ary biomedical relation. The biomedical relationship between a type of variation,

Table 6.2 The sample relation types in the news domain

Relations		Examples	Types
Affiliations	Personal organizational artifactual	Married to, mother of, spokesman for, president of, owns, invented, produces	PER → PER PER → ORG (PER|ORG) → ART
Geospatial	Proximity directional	Near, on outskirts, southeast of	LOC → LOC LOC → LOC
Part-of	Organizational political	A unit of, parent of annexed, acquired	ORG → ORG GPE → GPE

its location, and the corresponding state change from an initial-state to an altered-state can be extracted as point mutation(codon, 12, G, T).

Depending on the domain that relation extraction is applied to, the list of relation types will be determined. For example, in relation extraction for the news articles, the following would be an example of relation types (Table 6.2).

Another example of relation types is from the Automatic Content Extraction (ACE) program held in 2003. The goal of the program is to develop technology to automatically infer from human language data the entities being mentioned, the relations among these entities that are directly expressed, and the events in which these entities are involved. Data sources include audio and image data in addition to pure text, and Arabic and Chinese in addition to English. One of the tasks offered by ACE 2003 was relation extraction called the relation detection and characterization task (RDC). This task requires detection and characterization of relations between (pairs of) entities. There are four general types of relations, some of which are further sub-divided, yielding a total of 24 types/subtypes of relations:

ROLE: relates a person to an organization or a geopolitical entity
subtypes: member, owner, affiliate, client, citizen
PART: generalized containment
subtypes: subsidiary, physical part-of, set membership
AT: permanent and transient locations
subtypes: located, based-in, residence
SOCIAL: social relations among persons
subtypes: parent, sibling, spouse, grandparent, associate.

To discover the hidden knowledge from the unstructured text, NLP techniques were adopted to reveal the relation extraction patterns, which splits the sentences into word or presents syntactic structures (Zhou and He 2008; Bui et al. 2010). Bui et al. (2010) extract the drug-mutation relation from PubMed abstract by applying a rule-based approach. To extract the relation, they justify the two rules. The first rule is <keyword, relation keyword> pattern which is mostly common in sentences. The second rule is <relation, keyword1, keyword2> which calculates the distance and number of occurrences in the phrase. Also, Koike et al. (2005) present an extraction method from the biomedical text by using a shallow parser. NLP helps to assign Gene Ontology ID to PubMed abstracts and then they use shallow parsing

approaches to break down and analyze the sentences. After parsing the sentences, they extract ACTOR-OBJECT relation from the sentence structure. Huang et al. (2006) propose a new approach, a hybrid method using shallow parsing and pattern matching, to extract relation between two proteins from biomedical literature. They use rule-based shallow parsing that defines heads of each chunk and processes appositive and coordinative structures. The result indicates that pattern matching is remarkably improved with shallow parsing.

Several researches adopted feature based approaches to extract the relation on the biomedical text. To extract the relation on Protein-Protein Interaction (PPI), Song et al. (2011) propose the relation extraction technique called PPISpotter which is a combination of active learning and semi-supervised SVM techniques. They extract features from MEDLINE records by using NLP techniques. Chowdhury et al. (2011) extract the relation on drug—drug interaction (DDI). They employ the feature-based method which uses different the feature selection technique compared to Song et al.'s study (2011). Their features are word features, morpho-syntactic features, trigger words, and negation. Using SVM classifier with selected features, they evaluate their performance of DDI extraction.

Many researches employ different feature based approaches to Protein-Protein interaction extraction including Song et al.'s study (2011). Lin et al. (2011) extract the PPI relation by using a multiple kernels learning based approach which ensembles the feature-based kernel, tree kernel and graph kernel. Furthermore, they propose a lexical feature-based technique which considers not only bag of word features but also n-gram features. Yang et al. (2010) propose a BioPPISVMExtractor to extract PPI from the biomedical literature which is based on SVM classifier. They select various features including word features, keyword features, protein names distance feature and link path features. Also, Chen et al. (2011) propose PPIEor to extract PPI pairs from the biological literature. They use SVM classifier to extract features based on clause parsing output. Features they use include word feature, distance feature and location feature.

There are a set of common steps that are involved in relation extraction in biomedical literatures. Figure 6.2 illustrates how the task of relation extraction can be carried out in a common scenario.

For binary classification of relation where only true and false labels exist, the feature set for relation extraction are generated by the following three techniques: Named Entity Recognition, Shallow Parsing, and Negation. Of course, other types of features may be used, but for the simplicity reason, we used those three representative features for the tutorials.

Named Entity Recognition

The Named Entity Recognition (NER) technique automatically extracts pre-defined Named Entities (NEs) like gene, protein, and cell in text. It tags each word whether it is located in the starting or ending position, or outside the target entity. Most

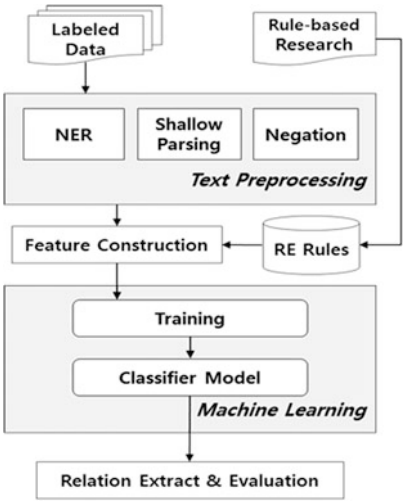

Fig. 6.2 An architecture of a typical supervised relation extraction system

corpora for relation extraction provide NE annotations that have information about target entities in a given text. We extract NEs by using a LingPipe tool introduced earlier in this chapter.

Shallow Parsing

Shallow parsing, also known as text chunking, splits sentence into phrases, such as Noun Phrase (NP), Verb Phrase (VP), Prepositional Phrase (PP), and Adverb Phrase (ADVP). This shallow parsing result gives us an important clue to extract relation in that relation of between entities is usually expressed in [entity1…verb… entity2] structure in a sentence. We apply shallow parsing to all sentences by the Apache OpenNLP toolkit (https://opennlp.apache.org/).

Negation

The negation technique examines whether a sentence is negated or not by finding negation terms (‘neither’, ‘not’, etc.) and a negation scope. In relation extraction, negation terms change the relation judgment in an opposite direction. We use NegEx (Chen et al. 2011) toolkit for negation detection.

We combine a rule-based approach with a machine learning (ML) based approach in order to efficient relation extraction. In particular, the hybrid framework consists of the rule-based feature selection and the ML-based classification algorithm.

Feature Construction from Defined Rules

In a rule-based approach, rules are constructed by combination of complex factors such as sentence structure, relation keyword, distance of between entities, grammatical relation and so on. Since those factors appear differently along with relationship type, it can be treated as variables for statistical algorithm. Those factors are the clue that predicted the difference to differ between relationships involving sentence and the others. Our approach is to choose features that represent the key clue for extracting relation in a rule-based approach.

For relation extraction, we used seven features using relation keyword, negation, distance of between two entities, location, order of entities and relation keyword etc. The seven features are as follows:

1. Predicate: a main verb that is located inside or nearest two entities. It must be found in the BioVerb list
2. Predicate POS: part-of-speech of predicate
3. Number of left words: the number of words in the left side of the first appearance of a named entity in a sentence
4. Number of right words: the number of words in the right side of the last appearance of a named entity in a sentence
5. Number of words in between entities: the number of words in the first left named entity and the extreme right named entity in a sentence.
6. Negation: sentence is negated or not
7. LinkPath: link path between two named entities exists or not.

ML-Based Classification

The pattern matching method finds a sentence in accordance with the predefined patterns. It could give more accurate results if patterns are precisely articulated. However, at the same time, it is very difficult to detect matches due to wide variations. Hence, we apply a machine learning algorithm for relation extraction with the rule-based feature set. We treated relation extraction as binary classification task. A sentence is classified depending on whether relation between entities exists or not. We use the WEKA toolkit for classification algorithms.

Given the following input,

> 11218788 Larsen J, Arnberg A, Brosen K: [Tramadol and oxazepam. Ugeskr Laeger. 2001 Jan 22; 163(4):458-60. Effect on pulmonary function in elderly patients with chronic obstructive lung disease]. Many patients with chronic obstructive pulmonary disease (COPD) suffer from osteoporotic pain as a result of glucocorticoid treatment and nervous symptoms partly related to their lung disease. There seems to be some reluctance to treat these patients with an opioid or benzodiazepine. Upon request, the Drug Information Centre in Odense made an extensive literature search on the subject. No documentation was found that tramadol additionally depresses the respiration in patients with COPD, nor has

> oxazepam in clinically relevant doses been found to exacerbate their lung disease. The clinical effect is subject to large interindividual variability, and the use of these drugs should, to a greater extent, rest on experience with the individual patient. There seems to be no reason to maintain a priori this rigoristic reluctance to use tramadol and/or oxazepam in patients with COPD.

The following results are produced:

ID 11218788 ANSWER N LEFT ENTITY tramadol RIGHT ENTITY COPD

The above report means that no relation between tramadol and COPD is predicted. For readers who are interested in reproducing the procedure, download the tool at http://informatics.yonsei.ac.kr/tsmm/uncertainty_book/RE.zip. Once you download and un-compress it, change to RE and run the following command:

```
java -Xms64m -Xmx15550m -cp .:./bin:./lib/* evaluation. PolyDrugGeneEvaluation
```

Well-Known Relation Extraction Tools

There are several tools that do relation extractions including PKDE4J, OpenIE, Stanford CoreNLP OpenIE, GATE, etc. PKDE4J will be explained in the next chapter. Here a well-accepted relation extraction tool, OpenIE, will be described.

Open IE

Open IE, standing for the Open Information Extraction (Open IE) system, was developed by the Etzioni's group at University of Washington. Open IE takes natural language sentences as an input and extracts relations in text. For example, consider the following sentence:

> The U.S. president George W. Bush gave his speech on Friday to hundred thousands of people.

There are many binary relations in this sentence that can be expressed as a triple (A, B, C) where A and B are arguments, and C is the relation between those arguments. Since Open IE is not aligned with an ontology like WordNet, the extracted relation is a phrase of text. The following list shows binary relations extracted from the sentence above:

1. (George W. Bush, is the president of, the U.S.)
2. (George W. Bush, gave, his speech)
3. (George W. Bush, gave his speech, on Friday)
4. (George W. Bush gave his speech, to hundred thousands of people).

The first result above is a “noun-mediated extraction”, because the extraction has a relation phrase is described by the noun “president”. The above results show that an n-ary extraction represents them in an informative way. Here is a possible list of the n-ary relations in the sentence:

(George W. Bush, is the president of, the U.S.)
(George W. Bush, gave, [his speech, on Friday, to hundred thousands of people])

To use the Open IE system, first install the system by typing the following command in the UNIX like prompt: sbt compile. Open IE uses Java 7 SDK and the sbt build system. The sbt command makes downloading dependencies and compiling very simple. The sbt command results in the jar file called “openie-assembly.jar” that contains all required dependent libraries. Once the jar file for Open IE is ready, execute the following command:

```
java -jar openie-assembly.jar.
```

The Open IE system takes one sentence per line unless the argument “–split” is specified. If the argument “–split” is presented, the input text will be split into sentences. Input can be fed into Open IE either as a file (an option first argument) or in an interactive mode where you type sentences interactively. Results will be written to the console unless a second option argument is specified for an output file.

Open IE takes a number of command line arguments. All of available arguments are displayed if you run java -jar openie-assembly.jar–usage. There are several interesting arguments. The first argument is “–binary” that generates the triple output. The second argument is “–split” that partitions the input text into sentences. The third argument is “–ignore-errors” that allows for Open IE to continue to execute even if an exception is encountered. Regarding the output format, There are two formats: simple and column. The argument “–format simple” enables to make ease of reading whereas a columnated format is used for machine processing.

Extracting Semantic Predications with SemRep

SemRep, standing for Semantic Knowledge Representation, is an automatic program that extracts semantic predications (subject-predicate-object triples) from biomedical free text (Rindflesch and Fiszman 2003). SemRep was developed at developed at National Library of Medicine. SemRep uses MetaMap to map noun phrases to UMLS concepts. Through its rule-based summarization system, it maps the syntactic elements to semantic network predicates. About 36.7 millions of sentences extracted from titles and abstracts of PubMed generate the predication analysis. SemRep detects about 12.7 millions of unique predicate instances and 58 unique predicate types (Aronson 2001).

Semantic predications are extracted based on the UMLS knowledge sources where subject and object are UMLS Metathesaurus concepts and the predicate to a relation type in the UMLS Semantic Network. SemRep extracts a wide range of predicates including (1) clinical medicine such as TREATS, DIAGNOSES, and PROCESS_OF, (2) substance interactions such as INTERACTS_WITH, INHIBITS, and STIMULATES, (3) genetic etiology of disease such as ASSOCIATED_WITH and CAUSES, and pharmacogenomics such as AFFECTS, AUGMENTS, and DISRUPTS. SemRep can be run interactively or in batch mode using the SKR Scheduler. SemRep program is also available as a stand-alone program on Linux platform.

For example, given the input text:

> dexamethasone is a potent inducer of multidrug resistance associated protein expression in rat hepatocytes

SemRep generates three semantic predications as follows:

- Dexamethasone STIMULATES Multidrug Resistence—Associated Proteins
- Multidrug Resistance—Associated Proteins PART_OF Rats
- Hepatocytes PART_OF Rat

SemRep is part of the SKR project that maintains a database of 84.6 million SemRep predications extracted from all MEDLINE citations (Hristovski et al. 2006). SKR stands for Semantic Knowledge Representation which is available at https://skr.nlm.nih.gov/. The SemRep database supports the Semantic MEDLINE Web application, which integrates PubMed search, SemRep predications, automatic summarization, and data visualization. The goal of the application is to assist users to manage the results of PubMed searches. Output is visualized as an informative graph with links to the original MEDLINE citations. Convenient access is also provided to additional relevant knowledge resources, such as Entrez Gene, the Genetics Home Reference, and UMLS Metathesaurus.

As a tool designed for automatic identification of semantic predication from biomedical literature, SemRep operates by applying a set of linguistic rules to sentences found in MEDLINE abstracts. Semantic relations identified by SemRep (Ahlers et al. 2007; Hristovski et al. 2006) have been used in literature-based discovery (LBD) (Wilkowski et al. 2011), among many other approaches to mining information from biomedical literature. Biomedical articles on which SemRep is designed to operate contain explicit and implicit mentions of relationships between various medical concepts. For example a TREATS relation between a medication and a disorder may be found in a single sentence in a MEDLINE citation containing the following text: “Metamorphosia associated with topiramate for migraine prevention.”

One example of using SemRep for semantic predications is creating RDF triples. It would be interesting to observe whether semantic predications can be leveraged to create network representations (will leverage the OWL-NETS abstraction to create networks containing only the mechanisms of interest). Supposed that we take

Table 6.3 Concept table

CONCEPT_ID	CUI	TYPE	PREFERRED_NAME	GHR	OMIM
1844	C0003873	META	Rheumatoid Arthritis	NULL	180300:604302
1276072	215	ENTREZ	ABCD1	NULL	NULL

Table 6.4 CONCEPT_SEMTYPE table

CONCEPT_SEMTYPE_ID	CONCEPT_ID	SEMTYPE	NOVEL
2628	1844	dsyn	Y
1481123	1276072	gngm	Y

Table 6.5 PREDICATION table

PREDICATION_ID	PREDICATE	TYPE
87120	PROCESS_OF	semrep

drug repurposing such as Rapamycin, Tamoxifen as a use scenario. There was a previous effort that SemRep was converted into RDF.[1]

The following words and phrases can be used to search for Tamoxifen for Repurposing in PubMed: Tamoxifen (C0039286), Bipolar Disorder (C0005586), Manic (C0338831), Protein Kinase C (C0033634), Protein Kinase C Inhibitor (C1514555). RDF "schema" for the SemRep predications consists of the following three:

1. UMLS CUI, relationship, UMLS CUI—annotation triple
2. UMLS CUI, rdfs:label, <preferred term>—label triple
3. UMLS CUI, umls:semtype, <semantic type>—semantic type triple

An example for the SemRep annotation is as follows:
Protein Kinase C Inhibitor TREATS Bipolar Disorder

umls:C1514555 umls: TREATS umls:C0005586
umls:C1514555 rdfs: label "Protein Kinase C Inhibitor"
umls:C0005586 rdfs:label "Bipolar Disorder"
umls:C1514555 umls: semtype "mobd"
umls:C0005586 umls: semtype "phsu"
(mobd = Mental or Behavioral Dysfunction)
(phsu = "Pharmacologic Substance")

The transformation of the SemRep tables into triples results in the following tables (Tables 6.3, 6.4, 6.5 snd 6.6).

[1] https://github.com/OHDSI/KnowledgeBase/tree/master/LAERTES/SemMED.

Table 6.6 PREDICATION_ARGUMENT table

PREDICATION_ARGUMENT_ID	PREDICATION_ID	CONCEPT_SEMTYPE_ID	TYPE
176604	87120	2628	S
176605	87120	21437	O

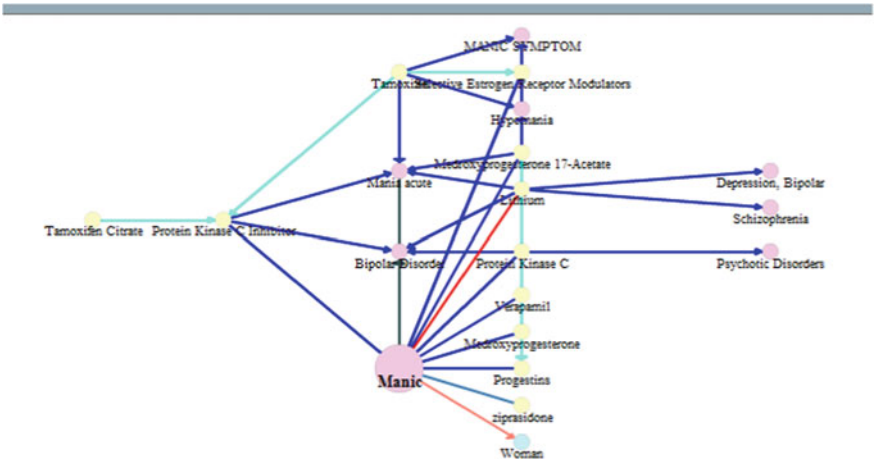

Fig. 6.3 A resulting RDF graph for drug reposition

The example query for the concept table would look like "column = CUI

With prefix umls: if TYPE = META
With prefix e.g if TYPE = ENTREZ
(ignore other types)

For label: object column = PREFERRED_NAME (literal)"
The example query for the concept_semtype table would look like "Subject column = CONCEPT_ID

With prefix umls: or e.g.: depending on what was assigned in the CONCEPT table
For semantic type: column = SEMTYPE (literal)"
For a given PREDICATION_ID
Subject = PREDICATION_ARGUMENT.CONCEPT_SEMTYPE_ID (where TYPE = S)
Predicate = PREDICATION.PREDICATE
Object = PREDICATION_ARGUMENT.CONCEPT_SEMTYPE_ID (where TYPE = O)

Given these transformed semantic predications, we are now able to visualize a RDF graph like Fig. 6.3.

Topic Modeling

Topic modeling methods are known to be useful for analyzing and summarizing large scale textual data in an unsupervised manner. Topic modeling have been applied in various different data sources including biomedical data, images, videos, and social media (Blei 2012). The goal of topic modeling is to group sets of words which are co-occurred within texts as topics by giving high probability for the words under same topics. The prominent feature of topic modeling lies in its ability not to require any training datasets which normally demand tremendous manual efforts of annotating or labeling and make quality of output heavily depend on training datasets.

Among the topic modeling algorithms, Latent Dirichlet Allocation (LDA) is the simplest and most well-accepted algorithm. LDA is a generative model (e.g., Naïve Bayes) which is a full probabilistic model of all the variables. In generative modeling, data is derived from a generative process which defines a joint probability distribution of observed and hidden variables. It contrasts to discriminative model (e.g., linear regression) which only models the conditional probability of unobserved variables on the observed variables. In LDA, the observed variables are words in the documents, and the hidden variables are topics. It follows the assumption that authors first decide a number of topics for an article, then pick up words related to these topics to write the article. In LDA, all documents in the corpus have the same set of topics, but each document has different portions of those topics (Blei 2012).

The basic assumption of LDA is that one document contains multiple topics and each of those requires specific words to describe them. For example, a paper, entitled "Artificial Intelligence in Biomedical Literatures", discussed the application of artificial intelligence algorithms to discover hidden associations between biological entities. Words such as neural network, autoencoder, neuron are from the topic of artificial intelligence; disease, gene, and protein are used to describe the biomedical topic; bioinformatics, genomics, cheminformatics are used for the topic of computer applications in biology.

Topic modeling algorithms aim to capture topics from a corpus automatically by using words observed in documents to infer the hidden topic structure (e.g., document topic distribution, and word topic distribution). The number of topics is usually decided by perplexity and can be heuristically set between 20 and 300 (Blei 2012). Topics are represented by distributions of words in the entire collection. Each document is generated by selecting a distribution over the topics. For each word, a topic assignment is chosen. In addition, the word from the corresponding topic is chosen.

The perplexity is often used to measure how a probability distribution fits a set of data. The perplexity is the inverse of the geometric mean per-word likelihood and is used to evaluate the models. A lower perplexity means a better model (Blei et al. 2003). The inference mechanics in topic models are independent of languages and contents. They capture the statistical structure of using language to represent

thematic content. LDA approximates its posterior distribution by using inference (e.g., Gibbs sampling) or optimization (e.g., variational methods). The detailed explanations of how to run LDA with Mallet are provided in the later section.

Latent Semantic Indexing

Latent Semantic Indexing (LSI) tries to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval (Deerwester et al. 1990). LSI assumes that there is some underlying or latent structure in word usage that is partially obscured by variability in word choice. A truncated singular value decomposition (SVD) is used to estimate the structure in word usage across documents. Retrieval is then performed using the database of singular values and vectors obtained from the truncated SVD. Performance data shows that these statistically derived vectors are more robust indicators of meaning than individual terms.

The SVD projection is computed by decomposing the document-by-term matrix $A_{t\times d}$ into the product of three matrices, $T_{t\times n}$, $S_{n\times n}$, $D_{d\times n}$:

$$A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^T$$

where t is the number of terms, d is the number of documents, n = min(t, d), T and D have orthonormal columns, i.e.,

$$TT^T = D^T D = I,$$

$$\text{rank(A)} = \text{r},$$

$$\text{S} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n),\ \sigma_i > 0 for\ 1 \le i \le r,\ \sigma_j = 0 for\ j \ge r+1.$$

We can view SVD as a method for rotating the axes of the n-dimensional space such that the first axis runs along the direction of largest variation among the documents, the second dimension runs along the direction with the second largest variation and so forth. The matrices T and D represent terms and documents in this new space. The diagonal matrix S contains the singular values of A in descending order. The ith singular value indicates the amount of variation along the ith axis.

By restricting the matrixes T, S and D to their first $\text{k} < n$ rows, one obtains three truncated matrices $T_{t\times k}$, $S_{k\times k}$, $(D_{d\times k})^T$. Their product $\hat{A}$ is the best square approximation of A by a matrix of rank k in the sense defined in the equation $||\Delta = A - \hat{A}_2||$

$$\hat{A}_{t\times k} = T_{t\times k} S_{k\times k} (D_{d\times k})^T$$

Choosing the number of dimensions (k) for $\hat{A}$ is an interesting problem. While a reduction in k can remove much of the noise, keeping too few dimensions or factors may lose important information. As discussed in (Deerwester et al. 1990) using a test database of medical abstracts, LSI's performance improved considerably after 10 or 20 dimensions, peaked between 70 and 100 dimensions, and then began to diminish slowly. This pattern of performance—initial large increases and slowly decreases—is observed with other datasets as well. Eventually the performance becomes the same as standard vector methods because, with k = n factors, $\hat{A}$ is the same as the original term by document matrix A. The fact that LSI works well with a relatively small (compared to the number of unique terms) number of dimensions or factors k shows that these dimensions are, in fact, capturing a major portion of the meaningful structure (Berry et al. 1995).
There are several open source packages available for LSI, including

1. Text Mining Library for Latent Semantic Analysis at http://tml-java.sourceforge.net/
2. Weka at http://weka.sourceforge.net
3. airhead-research (a.k.a. s-space) at https://code.google.com/archive/p/airhead-research/

Out of these three packages, airhead-research is used for describing how LSI can be used due to its simplicity and robustness compared to the other two packages. airhead-research was developed in Java and provides an API. Thus, the provided LSI functionalities in airhead-research can be used in two ways: (1) API and (2) command line.

To use airhead-research, download and uncompress its Java package. airhead-research supports a variety of options. For instance, the argument "-n" or "–dimensions <int>" sets how many dimensions to use for the LSA vectors. Another basic option is "-p" or "–preprocess <class name>", which specifies an instance of a transform class to use in preprocessing the word-document matrix compiled by LSA prior to computing the SVD. More advanced options include "-S" or "–svdAlgorithm", which specifies manually a particular SVD algorithm should be used internally. Valid options are SVDLIBC, MATLAB, OCTAVE, JAMA and COLT. Since LSA will select the fastest algorithm available, use this option only when it is necessary.

Depending on the number of options to be used, several combinations of options can be used. For example, in order to remove stop words from the corpus while processing, the following command can be used:

```
java -Xmx8g -jar lsa.jar -d corpus.txt -F exclude=stopwords.txt my-lsa-output-no-stopwords.sspace.
```

To generates a 500-dimension LSA space, use the following command:

```
java -Xmx8g -jar lsa.jar -d corpus.txt -n 500 my-lsa-output-500dim.sspace.
```

Table 6.7 The list of related terms to "farm"

Term	Relevance score
Hay	0.64
Farmer	0.87
Farming	0.78
Farmland	0.72
Landowner	0.67
Cattle	0.66
Homestead	0.65
Agricultural	0.65

To generates an LSA space with known compound words, use the following command:

```
java -Xmx8g -jar lsa.jar -d corpus.txt -C my-list-of-ngrams.txt my-lsa-output-with-ngrams.sspace
```

Once the LSI model is built, the user can query the model with a term of interest. For instance, for the query "farm," the LSI model returns a list of related terms (Table 6.7).

Latent Dirichlet Allocation (LDA)

As described in the topic modeling section, LDA is a type of generative, probabilistic model for the latent topic layer (Blei et al. 2003). For a document d, a multinomial distribution θ_d over topics is sampled from a Dirichlet distribution with parameter α. For each word w_{di}, a topic z_{di} is chosen from the topic distribution. A word w_{di} is generated from a topic-specific multinomial distribution $\phi_{z_{di}}$. The probability of generating a word w from a document d is:

$$P(w|d,\theta,\phi) = \sum_{z \in T} P(w|z,\phi_z)P(z|d,\theta_d)$$

Therefore, the likelihood of a document collection D is defined as:

$$P(Z,W|\Theta,\Phi) = \prod_{d \in D}\prod_{z \in T}\theta_{dz}^{n_{dz}} \times \prod_{z \in T}\prod_{v \in V}\phi_{zv}^{n_{zv}}$$

where n_{dz} is the number of times that a topic z has been associated with a document d, and n_{zv} is the number of times that a word w_v has been generated by a topic z. The model can be explained as: to write a paper, an author first decides what topics and

then uses words that have a high probability of being associated with these topics to write the article.

For the tutorial of how to do topic modeling, we introduce Mallet that was developed by McCullum and his team at University of Massachusetts Amherst. Mallet is a Java-based tool that provides various techniques including statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text.

To use Mallet for topic modeling, download it at http://mallet.cs.umass.edu/download.php. The latest version is 2.0.8. Once MALLET has been downloaded and installed, the next step is to import text files into MALLET's internal format. The following instructions assume that the documents to be used as input to the topic model are in separate files, in a directory that contains no other files. For detailed information on how to import data in MALLET, we refer the reader to instructions available at http://mallet.cs.umass.edu/import.php.

Once the MALLET package is successfully installed, it is ready to use. Simply change to the MALLET directory and run the following command:

```
bin/mallet import-dir --input data/topic-input --output topic-input.mallet  --keep-sequence --remove-stopwords
```

The input data is assumed to be under the MALLET packgage's data sub directory called "topic-input." To learn more about options available in MALLET, use the argument "–help". To build a topic model, use the train-topics command, assuming that documents are formatted properly for MALLET. For example, the following command will create 100 topics and save the trained topic model, again, assuming that the MALLET instance object has already been created with the input data:

```
bin/mallet train-topics --input topic-input.mallet --num-topics 100 --output-state topic-state.gz
```

If you want to know more about available options in MALLET, use the option-help to get a complete list of options for the train-topics command. There are several options that are frequently used when you run Mallet for topic modeling (See Table 6.8).

You may download the sample input file from http://informatics.yonsei.ac.kr/tsmm/uncertainty_book/ISI_Abstract_original.txt and generate the same results as shown in Table 6.9. The following command includes several options such as the number of topics (10), the number of iterations (1000), applying the stopword list to create the MALLET instance object. The "keep-sequence" option in the command denotes that the input text is converted into a sequence of features, and it is normal that topic modeling in MALLET assumes that the input is converted to a feature sequence.

```
bin/mallet import-dir --input ISI_Abstract_original.txt --output topic-input.mallet
--num-topics 10 --num-iterations 1000 --keep-sequence --remove-stopwords
```

Table 6.8 The list of core options available in MALLET

Option	Description
–input [FILE]	Use this option to specify the MALLET collection file you created in the previous step
–num-topics [NUMBER]	The number of topics to use. The best number depends on what you are looking for in the model. The default (10) will provide a broad overview of the contents of the corpus. The number of topics should depend to some degree on the size of the collection, but 200 to 400 will produce reasonably fine-grained results
–num-iterations [NUMBER]	The number of sampling iterations should be a tradeoff between the time taken to complete sampling and the quality of the topic model
–optimize-interval [NUMBER]	This option turns on hyperparameter optimization, which allows the model to better fit the data by allowing some topics to be more prominent than others. Optimization every 10 iterations is reasonable
–optimize-burn-in [NUMBER]	The number of iterations before hyperparameter optimization begins. Default is twice the optimize interval
–output-model [FILENAME]	This option specifies a file to write a serialized MALLET topic trainer object. This type of output is appropriate for pausing and restarting training, but does not produce data that can easily be analyzed
–output-state [FILENAME]	Similar to output-model, this option outputs a compressed text file containing the words in the corpus with their topic assignments. This file format can easily be parsed and used by non-Java-based software. Note that the state file will be GZipped, so it is helpful to provide a filename that ends in.gz
–output-doc-topics [FILENAME]	This option specifies a file to write the topic composition of documents. See the–help options for parameters related to this file
–output-topic-keys [FILENAME]	This file contains a "key" consisting of the top k words for each topic (where k is defined by the–num-top-words option). This output can be useful for checking that the model is working as well as displaying results of the model. In addition, this file reports the Dirichlet parameter of each topic. If hyperparamter optimization is turned on, this number will be roughly proportional to the overall portion of the collection assigned to a given topic
–inferencer-filename [FILENAME]	Create a topic inference tool based on the current, trained model. Use the MALLET command bin/mallet infer-topics–help to get information on using topic inference

Table 6.9 The number of topics and top terms generated by LDA

Topic no.	Top terms
0	Cell cells tissue study engineering differentiation bone potential nanofiber regeneration factors culture scaffolds critical scaffold mechanical control stem increase
1	Structure based technology polymer design fabricated carbon band high performance size circuit interconnect materials capacity electrical interconnects significantly improved
2	Graphene surface growth electronics epitaxial electronic material electron use layer magnetic layers high scattering chemical multilayer demonstrated surfaces landau
3	New formation effects vascular important known cells elsevier shown number factor development reserved rights sod network reduced vegf role
4	Energy power zno applications piezoelectric potential voltage output approach mechanical low cmos flexible area current density thin reduce crystal
5	Used results function different data show large small including webs elements delivery indicate application aligned electrospun activity direct resistance
6	Model using paper process well proposed mems structures parameters present simple fabrication response range presented linear stochastic mode nonlinear
7	Method time two order zoning system efficiency found optimization algorithm significant provides higher optical films experimental compared study air
8	Properties effect solar using devices temperature carrier charge morphology doping silicon transport film interface than transfer provide bulk device
9	Expression complex gene novel rna changes essential hsp genes stress specific functional cmr rnai species cofactors redox predicted patterns

Semantic Networks and Ontology

A semantic network is a propositional knowledge structure consisting of a set of nodes that are selectively connected to each other by links labeled by the relationship between each pair of connected nodes (Stillings et al. 1987). Semantic networks as a representation of knowledge have been in use in artificial intelligence (AI) research in a number of different areas. Some of the first uses of the nodes-and-links formulation were in the work of Collins and Quillian (1969), where the networks acted as models of associative memory. Their work centers on how natural language is understood and how the meanings of words can be captured in a machine.

Building a semantic network was previously done manually, which requires experts to put a significant amount of time and effort. Therefore, automatic construction of a semantic network was a recent, focal point of the semantic web community (Harrington 2009; Harrington and Wojtinnek 2011). One of the recent efforts for automatic construction of a semantic network is a hybrid set of classification systems based on weakly, distant or semi-supervised learning systems. These systems require a smaller set of training material that focuses on either two independent categories or utilizes two different classification methods. After the

intermediate classifiers are run on non-annotated documents, the results are analyzed and the documents that best represent of the categories are added to the training data to improve the classifier. This process is repeated until some predefined condition is met (Aggarwal and Zhai 2012). While words contain a lot of information about the document under inspection, they also create a high-level of dimensionality and ambiguity. Different words can be used to describe the same meaning (synonyms), for example *earth* and *dirt.* Using both words as separate terms in a VSM creates a high-level of dimensionality. We can use natural language processing (NLP) techniques to recognize and consolidate synonyms to reduce dimensionality but a second problem arises. Some words, like *earth,* have multiple meanings (polysemy) (Aggarwal and Zhai 2012) and these meanings can be domain dependent. It is in these cases that information extraction techniques such as concept hierarchies can be used to determine appropriate meaning (Feldman and Dagan 1995).

Concept hierarchies are created through analyzing the relationships of tokens found in a document. Relationships can be defined manually, based on token distributions, or specified through background knowledge. Zheng et al. (2009) defined a concept as a set of words, usually noun phrases, which have semantic relationships. Feldman and Sanger (2007) emphasize that a concept hierarchy can be used to describe a document which contains one or more concept nodes going from a more generalized meaning to more specific meaning. Representing a document as a set of concepts, or concept signature, provides a richer representation which, when used with clustering techniques, makes the resulting index scheme more useful (Zheng et al. 2009). The explosive growth in digital content emphasizes the need to develop automated management (organizational) and access (discovery) tools to support the processing of digital content for information access systems. Organization of this generally unstructured content requires one to identify the scope, concepts, and purpose of the resource and then analyze the relationships of the concepts to provide an overall understanding of the document (Tseng et al. 2007).

Early text classification schemes were built on labor intensive training sets that were used to model the predefined categories to be identified and required sufficient text in the document being classified to ensure good accuracy (Zelikovitz and Hirsh 2000). Due to these challenges research started to explore the use of background knowledge, that is, domain-specific heuristics that can be used as constraints to reduce the ambiguity of natural languages and help in the feature selection process. Taxonomies, controlled vocabularies, and ontologies are various types of formalized specification that provide a conceptualization of a domain of interest (Gruber 1995). It is generally agreed that a controlled vocabulary is the most basic form of background knowledge. It can be used for keyword or concept identification. Taxonomies take controlled vocabularies and identify relationships between concepts, such as an "is-a" relation that is used to identify synonyms of terms. Ontologies are the most complex of the three specifications and add on to taxonomies additional domain specific rules. An ontology contains a shared, controlled

vocabulary which models a specific domain with the definition of concepts and their properties and relations.

WordNet

WordNet is considered by most an implementation of the general English language ontology (Miller et al. 1990). It identifies words and word phrases, includes morphological and semantic relationships, and identifies a hierarchy of relationships (hypernym and hyponym). It has been used in query expansion (Hsu et al. 2008), text classification (Elberrichi et al. 2008), and text clustering (Hotho et al. 2003). Using WordNet's background knowledge, text documents are analyzed for concepts based on relationships between terms. Common linguistic relationships are antonyms (opposite meaning), synonyms (similar meaning), hypernyms ("IS-A" generalization of a term), hyponyms (more specific meaning of a term), holonyms ("PART-OF" relationship), and meronyms ("HAS-A" relationship). These relationships are shown in Fig. 6.4.

Hypernym relationships form a directional "IS-A" connection between two terms that moves from a specific meaning to a more generalized one ("Earth IS-A planet"). Many studies have been performed to automatically extract these relationships from unstructured text, such as in (Snow et al. 2004). Unlike hypernyms, terms which are synonyms can replace each other and still hold a similar meaning. For example, "sunshine" and "sunlight" terms may be used interchangeably in a sentence without significant loss of meaning. Meronyms are a bit more complex. Girju et al. (2006) defined six types of meronyms which WordNet consolidates three categories; *member-of* (faculty HAS-A professor), *stuff-of* (tree HAS-A wood), and *part-of* (solar system HAS-A sun). Additionally, Girju et al. identifies the *part-of* category as the most prominently used while Miller et al. (1990) indicate meronym transitivity may be optional as one moves away from the original relationship. For example, "Earth HAS-A moon" but the "plant HAS-A moon" relationship is optional (not all planets have moons).

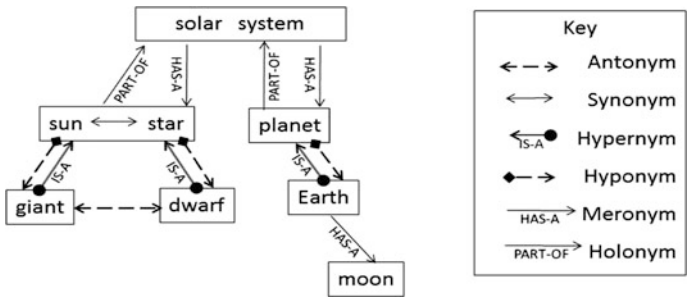

Fig. 6.4 Concept map using natural language relationships

WordNet has been used in numerous document-clustering experiments. Some of the earliest uses of WordNet in text categorization supported techniques to address effectively the classification of low frequency categories (Rodriguez et al. 1997). Green (1999) used WordNet's hypernym and hyponym links to build lexical chains to analyze the similarity between information in different paragraphs. Hotho et al. (2003) showed utilizing background knowledge (i.e., relationships) between terms improved document-clustering. Hung and Wermter (2004) present three text vector representations, two of which used hypernym as concepts to improve classification accuracy. Zheng et al. (2009) used WordNet relationships with noun phrases to analyze clustering improvements. Wang and Taylor (2007) used WordNet to capture hypernym relations in short text documents creating clusters of concepts called concept forests to represent a document. Elberrichi et al. (2008) used WordNet to create a concept vector format they compared to traditional bag-of-word vector representation. Except for Zheng et al. (2009), all these methods use single term analysis (using synonyms) and calculate term frequency from hypernyms. In fact, many of the papers listed suggest using more than one relationship as a future area research.

Accurately identifying concepts for categorization purposes is fraught with time-consuming manual analysis by content experts and librarians. A digital library catalog/index must represent the digital content and reflect the expectations of its users. Automating this process requires new techniques in concept extraction to analyze any size document and capture main concepts based on the appropriate domain. In this paper, we describe an extension to existing natural language and machine-learning techniques to improve the accuracy of extracting concepts from small text based resources and grouping them appropriately.

The selection of terms is a critical first step in concept generation. Terms with multiple meanings (polysemy) create ambiguity, while a term that is similar (synonyms) to others or have a degree of generalization (hypernym) can strengthen the importance of a concept. For these reasons, term frequency calculations often use hypernym and synonym information once ambiguity is resolved. We also use this approach in our algorithm but the novelty of our approach is the inclusion of meronyms. The choice of meronyms comes from the idea of finding mechanisms to improve frequency measures for significant terms in short text documents without over constraining larger documents. Some meronyms studies have been conducted as outlined by Yang and Callan (2009). Basu et al. (2001) developed a set of measures for different lexical relationships, including meronyms to identify the average semantic difference (i.e., the weight of an edge between two terms). Meronyms were given the same weight as hypernyms in this study. Girju et al. (2006) suggest techniques for identifying meronyms for the specific use of incorporating them into taxonomies so they may be used in concept extraction. Zheng et al. (2009) used meronyms as the relationship to support clustering and found it to be not as good as hypernyms and holonyms. The novelty of our study examines the effects of weighing meronyms differently than synonyms or hypernyms when incorporating them into a frequency count for text characterization.

In addition to a general English ontology, domain specific ones exist. In the realm of education, there are many used to define guidelines for knowledge goals. Strand Map Benchmarks is a representation of the AAAS' Project 2061, a "statement of what all students should know and be able to do in science, mathematics, and technology by the end of grades 2, 5, 8, and 12." … "It provides educators with sequences of specific learning goals they can use to design a core curriculum".

The basic statistics of WordNet 3.0 are provided as follows (Table 6.10–6.11):

By and large, WordNet can be used in two ways. First approach is use WordNet online. WordNet is accessible online at http://wordnetweb.princeton.edu/perl/webwn. Once you type in a query and choose options for displaying results, WordNet returns the matched results (Fig. 6.5).

The second option is to download and install WordNet to a local machine. Depending on the operating system, you need to download different version. The most recent Windows version of WordNet is 2.1, released in March 2005. Yes, it has been a long time. For the Unix or Linux OS, version 3.0 is available for download, which was released in December, 2006. However, database files are updated to the version 3.1 and can substitute for the 3.0 files on the Unix or Linux OS.

Regarding database files, the following standoff files provide further semantic information to supplement the WordNet 3.0:

- Semantically annotated gloss corpus
- Evocation database
- Morphosemantic Links (Semantic relations between morphologically related nouns and verbs)
- Teleological Links (an encoding of typical activity for which artifact was intended)

Table 6.10 Number of POS, words, Synsets, and sense pairs

POS	Unique strings	Synsets	Total word-sense pairs
Noun	117798	82115	146312
Verb	11529	13767	25047
Adjective	21479	18156	30002
Adverb	4481	3621	5580
Totals	155287	117659	206941

Table 6.11 Polysemy information

POS	Monosemous words and senses	Polysemous words	Polysemous senses
Noun	101863	15935	44449
Verb	6277	5252	18770
Adjective	16503	4976	14399
Adverb	3748	733	1832
Totals	128391	26896	79450

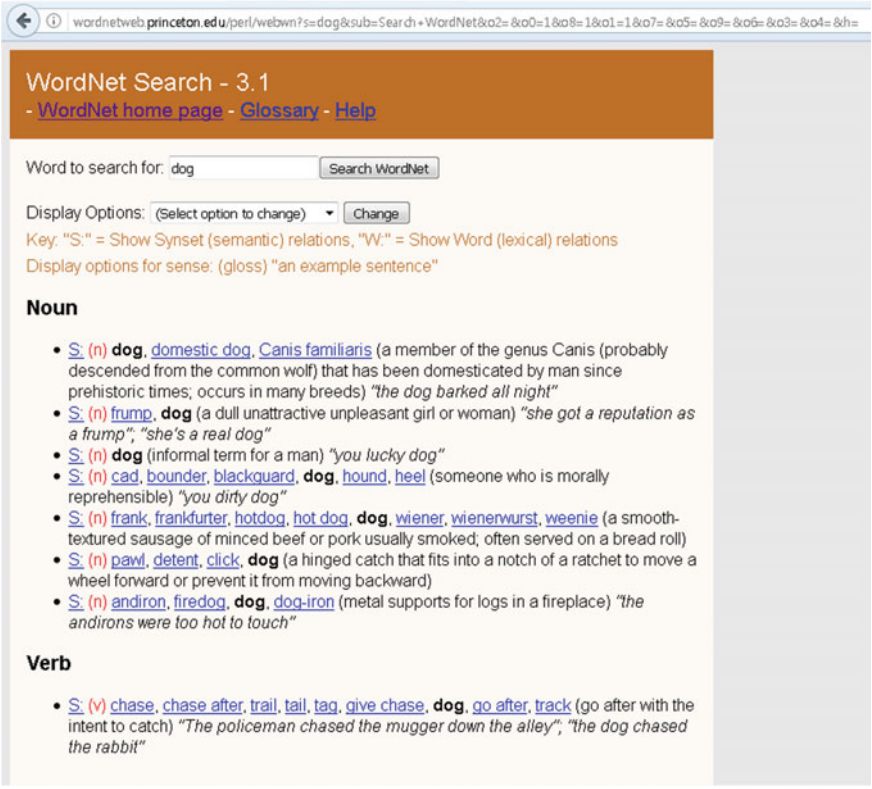

Fig. 6.5 The homepage of WordNet search

- "Core" WordNet (5000 more frequently used word senses)
- Logical Forms (logical forms for glosses)

WordNet can be utilized with NLTK, the Python based text mining tool. WordNet is a NLTK corpus reader, and it is imported with the following import statement:

```
>>> from nltk.corpus import wordnet
```

To examine a word with the NLTK WordNet module, we can use the NLTK function called synsets(). This function has an optional pos argument which lets you constrain the part of speech of the word:

```
>>> wn.synsets('dog') # doctest: +ELLIPSIS +NORMALIZE_WHITESPACE
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'),
Synset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01'), Synset('chase.v.01')]
>>> wn.synsets('dog', pos=wn.VERB)
[Synset('chase.v.01')]
```

Table 6.12 Statistics of BabelNet 3.7

1971744856	Total number of RDF triples
745859932	Total number of Babel senses
380239084	Total number of lexico-semantic relations
40709194	Total number of glosses (textual definitions)
13801844	Total number of Babel synsets
10767833	Total number of images
7735448	Total number of named entities
6393568	Total number of other forms
6066396	Total number of concepts
2948668	Total number of Babel synsets with at least one picture
2675385	Total number of Babel synsets with at least one domain
743296	Total number of compounds
271	The number of languages

BabelNet

BabelNet is a very large multilingual encyclopedic dictionary and semantic network (Navigli and Ponzetto 2012). It integrates the largest multilingual Web encyclopedia with the most popular computational lexicon of English such as WordNet, other lexical resources such as Wiktionary, OmegaWiki, Wikidata, and the Open Multilingual WordNet. The integration is performed by an automatic linking algorithm and by filling in lexical gaps with the aid of machine translation algorithms. The result is an encyclopedic dictionary that provides Babel synsets including concepts and named entities lexicalized in many languages and connected with large amounts of semantic relations.

BabelNet's current version is 3.7, which includes many feature such as FrameNet (lexical units), more than 2500 Babel synsets identified as key concepts, mappings with several versions of WordNet integrated, more than 2.6 million Babel synsets labeled with domains, more than 625 million new senses, 6.4 million surface forms for Babel synsets, and 3.5 million YAGO external links (Table 6.12). BabelNet also provides both Java and HTTP RESTful APIs.[2]

BabelNet can be utilized in two ways. The first method is to use the web interface of BabelNet. The second method is to use Java API or REST API. For the REST API, one can query BabelNet through an HTTP interface that returns JSON. The user can append the *key* parameter to the HTTP requests as shown in the examples below. To obtain an API key please read the page. All requests must be executed using the GET method and they should include Accept-Encoding: gzip as the header in order to obtain compressed content. The example of a REST API is as follows:

[2] http://babelnet.org/download.

https://babelnet.io/v4/getVersion?key={key}.

where the {key} denotes the API key obtained after signing up to BabelNet. Another example is to retrieve the IDs of the Babel synsets (concepts) denoted by a given word:

https://babelnet.io/v4/getSynsetIds?word={word}&langs={lang}&key={key}.

In this example, there are seven options that can be added to the REST API (Table 6.13).

The results of the ID retrieval REST API are shown in Table 6.14.

Another example of to retrieve the senses of a given word from BabelNet using the REST API. Table 6.15 shows a list of options.

Table 6.13 Options available for ID retrieval in BabelNet API

Name	Description
word	The word you want to search for
langs	The language of the word. Accepts multiple values
filterLangs	The languages in which the data are to be retrieved. Default value is the search language and accepts not more than 3 languages except the search language
pos	Returns only the synsets containing this part of speech (NOUN, VERB, etc.). Accepts only a single value
source	Returns only the synsets containing these sources (WIKT, WIKIDATA, etc.). Accepts multiple values
normalizer	Enables normalized search
key	API key obtained after signing up to BabelNet

Table 6.14 The results of the ID retrieval REST API

[
{“id”:“bn:15409009n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:00046063n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:03345344n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:00055685n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:01204395n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:02131227n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:02799103n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:15586454n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:00088150v”,“pos”:“VERB”,“source”:“BABELNET”},
{“id”:“bn:00355636n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:00090750v”,“pos”:“VERB”,“source”:“BABELNET”},
{“id”:“bn:00071669n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:02363694n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:01610649n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:03783607n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:01683382n”,“pos”:“NOUN”,“source”:“BABELNET”},
{“id”:“bn:15010220n”,“pos”:“NOUN”,“source”:“BABELNET”}]

Table 6.15 Options available for word sense retrieval of BabelNet API

Name	Description
word	The word you want to search for
lang	The language of the word. Required
filterLangs	The languages in which the data are to be retrieved. Default value is the search language and accepts not more than 3 languages except the search language. Example
pos	Returns only the synsets containing this part of speech (NOUN, VERB, etc.). Accepts only a single value
source	Returns only the synsets containing these sources (WIKT, WIKIDATA, etc.). Accepts multiple values
normalizer	Enables normalized search
key	API key obtained after signing up to BabelNet

https://babelnet.io/v4/getSenses?word={word}&lang={lang}&key={key}.

The results of the word sense retrieval REST API are shown as follows.


```
[
      {
            "lemma":"Simians in Chinese poetry",
            "simpleLemma":"Simians in Chinese poetry",
            "source":"WIKIRED",
            "sensekey":"",
            "sensenumber":0,
            "frequency":1,
            "position":1,
            "language":"EN",
            "pos":"NOUN",
            "synsetID":{"id":"bn:15409009n","pos":"NOUN","source":"BABELNET"},
            "translationInfo":"",
            "pronunciations":{"audios":[],"transcriptions":[]},
            "bKeyConcept":false
      },
      {
            "lemma":"Simians_(Chinese_poetry)",
            "simpleLemma":"Simians",
            "source":"WIKI",
            "sensekey":"",
            "sensenumber":0,
            "frequency":10,
            "position":1,
            "language":"EN",
            "pos":"NOUN",
            "synsetID":{"id":"bn:15409009n","pos":"NOUN","source":"BABELNET"},
            "translationInfo":"",
            "pronunciations":{"audios":[],"transcriptions":[]},
            "freebaseId":"0 frc72",
            "YAGOURL":"Simians (Chinese poetry)",
            "bKeyConcept":false
      },
       …
]
```

Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. An artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in Artificial Intelligence (AI) that has networks which are capable of learning unsupervised from data that is unstructured or unlabeled.

One of the most common AI techniques used for processing Big Data is Machine Learning. Machine learning is a self-adaptive algorithm that gets better and better analysis and patterns with experience or with new added data. If a digital payments company wanted to detect the occurrence of or potential for fraud in its system, it could employ machine learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set, and point out any anomaly detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a non-linear approach. A traditional approach to detecting fraud or money laundering might rely on the amount of transaction that ensues, while a deep learning non-linear technique to weeding out a fraudulent transaction would include time, geographic location, IP address, type of retailer, and any other feature that is likely to make up a fraudulent activity. The first layer of the neural network processes a raw data input like the amount of the transaction and passes it on to the next layer as output. The second layer processes the previous layer's information by including additional information like the user's IP address and passes on its result. The next layer takes the second layer's information and includes raw data like geographic location and makes the machine's pattern even better. This continues across all levels of the neuron network until the best and output is determined.

Recently, deep learning approaches have obtained very high performance across many different NLP tasks. These models can often be trained with a single end-to-end model and do not require traditional, task-specific feature engineering. There several good reasons for using deep learning for NLP problems. First is that it is quite suitable for learning representation. Hand crafting features is time-consuming. The features are often both over-specified and incomplete. The work has to be done again for each task/domain, etc. We must move beyond handcrafted features and simple ML. Humans develop representations for learning and reasoning. Our computers should do the same. Deep learning provides a way of doing this. Second, Current NLP systems are incredibly fragile because of their atomic symbol representations. Distributed representation enabled by deep learning based NLP can relax this problem. Learned word representations help enormously

in NLP. They provide a powerful similarity model for words. Distributional similarity based word clusters greatly help most applications.

Distributed representations can do even better by representing more dimensions of similarity. Distributed representations deal with the curse of dimensionality. Generalizing locally (e.g., nearest neighbors) requires representative examples for all relevant variations. Classic solutions: Manual feature design, assuming a smooth target function (e.g., linear models), Kernel methods (linear in terms of kernel based on data points). Neural networks parameterize and learn a "similarity" kernel. Third, deep learning is suitable for unsupervised feature and weight learning. Today, most practical, good NLP& ML methods require labeled training data (i.e., supervised learning). But almost all data is unlabeled. Most information must be acquired unsupervised. Fortunately, a good model of observed data can really help you learn classification decisions. Despite prior investigation and understanding of many of the algorithmic techniques before 2006, training deep architectures was unsuccessful. But since then, faster machines and more data help DL more than other algorithms. New methods for unsupervised pre-training have been developed (Restricted Boltzmann Machines = RBMs, autoencoders, contrastive estimation, etc.). More efficient parameter estimation methods. Better understanding of model regularization.

Word Embeddings

Word embeddings are one of the most well accepted deep learning algorithms that has been applied to NLP, which the original concept was introduced by Bengio et al. (2003). Word embedding algorithms are one of the best options to gain intuition about why deep learning is effective. Let's discuss the basic notion of word embeddings.

A word embedding W:words $\rightarrow$ Rn is a parameterized function mapping words in some language to high-dimensional vectors (perhaps 200 to 500 dimensions). For example, we might find:

$$W(''cat'') = (0.2, -0.4, 0.7, \ldots)$$
$$W(''mat'') = (0.0, 0.6, -0.1, \ldots)$$

Typically, the function is a lookup table, parameterized by a matrix, θ, with a row for each word: $W\theta(wn) = \theta n$. W is initialized to have random vectors for each word. It learns to have meaningful vectors, which can be used for advanced NLP tasks such as sentiment analysis or information retrieval. For example, one task we might train a network for is predicting whether a 5 g (sequence of five words) is valid. We can generate a number of 5 g from Wikipedia (e.g., "cat sat on the mat") and then make half of them invalid by switching a word with a random word (e.g., "cat sat **song** the mat"), since that will almost certainly make our 5 g nonsensical.

The model we train will run each word in the 5 g through W to get a vector representing it and feed those into another 'module' called R which tries to predict if the 5 g is valid or invalid, and it will result in the following:

$$R(W(''cat''), W(''sat''), W(''on''), W(''the''), W(''mat'')) = 1$$
$$R(W(''cat''), W(''sat''), W(''song''), W(''the''), W(''mat'')) = 0$$

In order to predict these values accurately, the network needs to learn good parameters for both W and R. Although it could be helpful in detecting grammatical errors in text, but what is interesting is to learn W. One way to understand the word embedding space is to visualize them with t-SNE, a sophisticated technique for visualizing high-dimensional data. Figure 6.6 shows the results of t-SNE with the word2vec model built on the news articles related to companies producing platform software and hardware.

This visualization of words helps us make sense of word associations. Similar words are close together. Another way to get at this is to look at which words are closest in the embedding to a given word. Again, the words tend to be quite similar. Figure 6.7 below shows how the similar words to the word "IBM" changed over time.

It may be adequate for a network to make words with similar meanings have similar vectors. If a word is replaced with a synonym (e.g., "a few people sing well" → "a *couple* people sing well"), the meaning of the sentence still remains the same. Thus, we may say that even if the input sentence has changed a lot, if W maps synonyms (like "few" and "couple") close together, from R's perspective not much changes are made. This implies many important points. There is the enormous number of possible 5 g whereas we have a comparatively small number of data

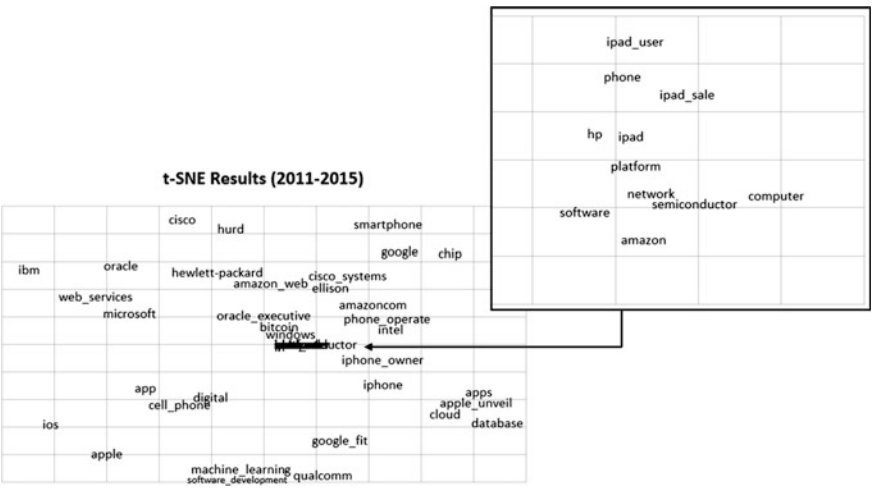

Fig. 6.6 t-SNE visualizations of word embeddings

2000-2005		2006-2010		2011-2015		2016-2017	
dell	0.959924936	oracle	0.79832989	buy_open-source	0.688270867	ibm_watson	0.820717812
unix	0.954558194	mainframe	0.77750206	idataplex	0.676672041	watson	0.808973908
solaris	0.952200234	sun	0.762942851	nirvanix_software	0.662971854	cognitive	0.803402245
server	0.950758517	orcl	0.756299734	reinvigorate_dell	0.660516798	ibm_cloud	0.757027686
datum_center	0.950342536	sap	0.742992461	ibm_provide	0.653238475	watson_internet	0.742083311
source	0.950308442	novl	0.738868535	bleed_market	0.652676821	cognitive_computing	0.741557717
move	0.948748887	patent_battle	0.735488415	teradataexadata	0.651375115	watson_unit	0.734454691
operate_system	0.947559893	seven-year_patent	0.734089375	provide_vertical	0.64939189	ibm_collaborations	0.727057636
run	0.946791053	sun_microsystems	0.72926867	require_pace	0.648596585	ibm_research	0.726451933
proprietary	0.944843233	open-source_linux	0.726028204	s&o_professional	0.647573769	ibm_continue	0.723082423
support	0.94289434	seven-year	0.725172222	dell_bring	0.646723807	watson_cognitive	0.720367968
hewlett-packard	0.942260742	larry_ellison	0.724186003	boost_ibm	0.646304667	ibmcomoutthink	0.717018962
linux	0.940122664	ibm_corp	0.723489523	systems_hope	0.64626497	watson_developer	0.716962516
develop	0.939396739	software_giant	0.721525371	teradataexadata_hardware	0.645682216	watson_health	0.716135025
hardware	0.935241699	java_patent	0.718840897	hp_struggle	0.645025253	cognitive_solution	0.716092885
pc	0.933961868	oracle_acquire	0.715612531	ibm_build	0.644733489	watson_education	0.711184084
corporate	0.933870077	outlooksoft	0.715118825	software_ip	0.644324541	ibmi_suh	0.705735028
websphere	0.932072341	business_object	0.713380098	ibm_software	0.642227352	ibm_announce	0.704842031
system_software	0.931963742	maker_sco	0.713360071	drive_idataplex	0.642201006	cloud_video	0.704631567
improve	0.929755747	applix	0.708620787	deep-pocketed_player	0.640904605	video_unit	0.703812957
sell_server	0.929618537	xensource	0.706225395	market_computer	0.640653729	ibm_insight	0.702063084
oracle	0.928216755	giant_roil	0.703248084	smb_apps	0.640520751	data_services	0.697872102
proprietary_solaris	0.927893996	unix_maker	0.701602578	highly-tested	0.640065312	wwwibm	0.696583211
rational	0.926518321	telelogic	0.701261342	intel-compatible	0.639556468	watson_service	0.695548415
antitrust_lawsuit	0.926318586	january_purchase	0.701039195	prime_competitor	0.639511287	watson_beat	0.694848537
rate	0.926261961	sun-developed	0.700815618	intel-compatible_server	0.638401508	developer_partner	0.693343222
intense	0.926035345	sun-developed_java	0.700212479	power_pc-based	0.63667053	watson_iot	0.691154718
potential	0.923175871	prominent_name	0.696930885	ibm_power-based	0.636367381	imaging_collaborative	0.690888762
solaris_server	0.922847867	java_software	0.694627345	ibm_x	0.635502517	beat_jeopardy	0.690651357
java	0.92274034	oracle_buy	0.694197714	entire_x	0.634402633	ibforbes_ibhi	0.689501226

Fig. 6.7 Similar words associated with the word "IBM" over time

points to try to learn from. Similar words being close together allows us to generalize from one sentence to a class of similar sentences. This does not just switch a word for a synonym, but rather switch a word for a word in a similar class (e.g., "the wall is blue" → "the wall is *red*"). Further, we can change multiple words (e.g., "the wall is blue" → "the *ceiling* is *red*"). This is a benefit that W provides.

Word embeddings also allow us to generalize to new combinations of words. You've seen all the words that you understand before, but you haven't seen all the sentences that you understand before. So too with neural networks. Word embedding models can automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Word embeddings exhibit an even more remarkable property: analogies between words seem to be encoded in the difference vectors between words. For example, there seems to be a constant male-female difference vector:

$$W(''woman'') - W(''man'') \simeq W(''aunt'') - W(''uncle'')$$
$$W(''woman'') - W(''man'') \simeq W(''queen'') - W(''king'')$$

Gender pronouns mean that switching a word can make a sentence grammatically incorrect. For instance, supposed that there are sentences like "*she* is the aunt" and "*he* is the uncle." (Similarly, "*he* is the King" but "*she* is the Queen." If one sees "*she* is the *uncle*," the most likely explanation is a grammatical error. If words are being randomly switched half the time, it seems pretty likely that happened here.

Mikolov et al. (2013) points out that the word embeddings learn to encode gender in a consistent way. Depending on the datasets that word embedding models

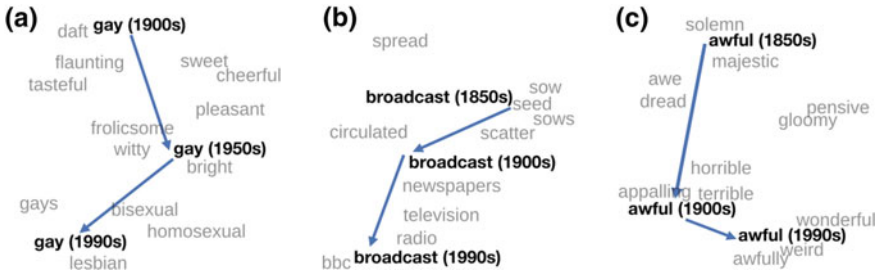

Fig. 6.8 The shift of meanings of words in HistWords. *Source* Hamilton et al. (2016)

like word2vec is built upon, there's probably a gender dimension. Same thing for singular vs plural.

HistWords is an interesting collection of tools and datasets for analyzing language change using word embeddings for historical text.[3] The semantic evolution of more than 30,000 words across four languages was modeled by historical word vectors. HistWords is maintained by a group of researchers at Stanford University, William L. Hamilton, Jure Leskovec, and Dan Jurafsky (2016). They found that the meanings of more frequently used words tend to be more stable over time than less frequently used words and that the meanings of polysemous, those words with multiple meanings, change at faster rates than others (see Fig. 6.8).

It's important to appreciate that all of these properties of W are *side effects*. This seems to be a great strength of neural networks: they learn better ways to represent data, automatically. Representing data well, in turn, seems to be essential to success at many machine learning problems. Word embeddings are just a particularly useful example of learning a representation.

There are several word embedding models that are publicly available. The most popular one is the Google news word2vec model.[4] The name of the model is called GoogleNews-vectors-negative300.bin.gz. Google published pre-trained vectors trained on part of Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described by Mikolov et al. (2013).

To use the word2vec program provided by Google, download it with svn checkout from http://word2vec.googlecode.com/svn/trunk/. Then compile word2-vec with 'make' from a Linux terminal window or linux emulator like cygwin, and then run the demo scripts: ./demo-word.sh and. /demo-phrases.sh.

[3] https://nlp.stanford.edu/projects/histwords/.

[4] https://code.google.com/archive/p/word2vec/.

Summary

Extracting entities and their relations from unstructured text is essential for text mining, topic modeling, constructing ontological structures, and deep learning. Resources such as WordNet and BabelNet play an instrumental role in a wide variety of applications. Deep learning, especially advances such as word2vec, has revitalized the interest in text analysis and document understanding. As demonstrated by the wide adoption of word2vec and distributional paradigms, the series of technical advances from LSI, LDA, to word2vec will continue to grow. With increasingly powerful and intuitive tools, one can tackle more challenging problems at a larger scale. In terms of Shneider's four-stage evolution model, quantitative studies of science as a field are likely to benefit profoundly from the stream of text modeling techniques.

References

Aggarwal CC, Zhai C (2012). Mining text data. Springer

Ahlers CB, Fiszman M, Demner-Fushman D, Lang F, Rindflesh TC (2007). Extracting semantic predication from MEDLINE citations for pharmacogenomics. In: Pacific symposium on biocomputing, pp 209–220

Aronson AR (2001) Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. In: AMIA annual symposium proceedings, pp 17–21

Basu S, Mooney RJ, Pasupuleti K, Ghosh, J (2001) Evaluating the novelty of text-mined rules using lexical knowledge. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM San Francisco, California, pp 233–238

Bengio Y, Ducharme R, Vincent P, Jauv C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155

Berry MW, Dumais ST, O'Brian GW (1995) Using linear algebra for intelligent information retrieval. SIAM Rev 37(4):573–595

Bikel DM, Schwartz RL, Weischedel RM (1999) An algorithm that learns what's in a name. Mach Learn 34:211–231

Blei DM (2012) Probablisitic topic models. Commun ACM 55(4):77–84. doi:10.1145/2133806.2133826

Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. JMLR 3:993–1022

Bui QC, Nualláin BÓ, Boucher CA, Sloot PM (2010) Extracting causal relations on HIV drug resistance from literature. BMC Bioinform 11(1):101

Chen Y, Liu F, Manderick B (2011) Extract protein-protein interactions from the literature using support vector machines with feature selection, biomedical engineering, trends, research and technologies. In: Olsztynska S (ed). ISBN: 978-953-307-514-3

Chowdhury MFM, Abacha AB, Lavelli A, Zweigenbaum P (2011) Two different machine learning techniques for drug-drug interaction extraction. In: Challenge task on drug-drug interaction extraction, pp 19–26

Collins AM, Quillian MR (1969) Retrieval time from semantic memory. J Verbal Learn Verbal Behav 8:240–247

Deerwester S, Dumais ST, Landauer TK, Furnas GW, Harshman RA (1990) Indexing by latent semantic analysis. J Am Soc Info Sci 41(6):391–407

Elberrichi Z, Rahmoun A, Bentaalah MA (2008) Using WordNet for text categorization. Int Arab J Info Technol 5(1):16–24

Feldman R, Dagan I (1995) Knowledge discovery in textual databases (KDT). KDD

Feldman R, Sanger J (2007) The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge University Press, Cambridge, UK

Finkel JR, Grenager T, and Manning C (2005) Incorporating non-local information into information extraction systems by gibbs sampling. In: ACL '05, proceedings of the 43rd annual meeting on association for computational linguistics, Association for Computational Linguistics Morristown, NJ, USA, pp 363–370

Fu Y, Bauer T, Mostafa J, Palakal M, Mukhopadhyay S (2002) Concept extraction and association from Cancer literature. Eleventh international conference info knowledge management (CIKM2002)/Fourth ACM interenational workshop web info data management (ACM WIDM 2002). McLean, VA, USA, pp 100–103

Girju R, Badulescu A, Moldovan D (2006) Automatic discovery of part-whole relations. Comput Linguist 32(1):83–135

Green SJ (1999) Building Hypertext Links By Computing Semantic Similarity. IEEE Trans Knowl Data Eng 11(5):713–730

Gruber TR (1995) Toward principles for the design of ontologies used for knowledge sharing? Int J Hum Comput Stud 43(5):907–928

Hamilton WL, Leskovec J, Jurfsky D (2016) Diachronic word embeddings reveal statistical laws of semantic change. In: Proceedings of the 54th annual meeting of the association for computational linguistics, Berlin, Germany, August 7–12, 2016, Association for Computational Linguistics, pp 1489–1501. http://aclweb.org/anthology/P16-1141

Harrington B (2009) ASKNet: automatically creating semantic knowledge networks from natural language Text, Ph.D. thesis, University of Oxford

Harrington B, Wojtinnek PR (2011) Creating a standardized markup language for semantic networks. In: Proceedings of the 5th ieee international conference on semantic computing

Hotho A, Staab S, Gerd S (2003) Wordnet improves text document clustering. In: Proceedings of the SIGIR 2003 semantic web workshop of the 26th annual international ACM SIGIR conference, Toronto, CA

Hristovski D, Friedman C, Rindflesch TC, Peterlin B (2006) Exploiting semantic relations for literature-based discovery. AMIA Annu Symp Proc 2006:349–353

Hsu MH, Tsai MF, Chen HH (2008) Combining WordNet and ConceptNet for automatic query expansion: a learning approach. In: Proceedings of the 4th asia information retrieval conference on information retrieval technology, Springer Harbin, China, pp 213–224

Huang M, Zhu X, Li M (2006) A hybrid method for relation extraction from biomedical literature. Int J Med Informatics 75(6):443–455

Hung C, Wermter S (2004) Neural network based document clustering using WordNet ontologies. Int J Hybrid Intell Syst 1(3–4):127–142

Koike A, Niwa Y, Takagi T (2005) Automatic extraction of gene/protein biological functions from biomedical text. Bioinformatics 21(7):1227–1236

Lin H, Yang Z, Li Y (2011). Protein-protein interactions extraction from biomedical literatures. biomedical engineering, trends, research and technologies. In: Olsztynska S (ed). ISBN: 978-953-307-514-3

Majoros WH, Subramanian GM, Yandell MD (2003) Identification of key concepts in biomedical literature using a modified Markov heuristic. Bioinformatics 19(3):402–407

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119

Miller GA, Beckwith R, Fellbaum C, Gross D, Miller K (1990) WordNet: an on-line lexical database. Int J Lexicogr 3(4):235–244

Navigli R, Ponzetto SP (2012) BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif Intell 193:217–250

Rindflesch TC, Fiszman M (2003) The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. J Biomed Inform 36(6):462–477

Rindflesch TC, Tanabe L, Weinstein JN, Hunter L (2000) EDGAR: extraction of drugs, genes and relations from the biomedical literature. Pac Symp Biocomput pp 517–528

Rodriguez MDB, Gomez-Hidalgo JG, Diaz-Agudo B (1997) Using WordNet to complement training information in text categorization. In: Milkov R, Nicolov N, Nikolov N (ed) Second international conference on recent advances in natural language processing (RANLP), John Benjamins Publishing, Stanford CA USA

Shehata S, Karray F, Kamel M (2007) A concept-based model for enhancing text categorization, KDD

Snow R, Jurafsky D, Ng AY (2004) Learning syntactic patterns for automatic hypernym discovery. In: Advances in neural information processing systems (NIPS 2004), Vancouver, British Columbia

Song M, Yu HJ, Han WS (2011) Combining active learning and semi-supervised learning techniques to extract protein interaction sentences. BMC Bioinform 12(Suppl):12

Stillings NA, Feinstein MH, Garfield JL, Rissland EL, Rosenbaum DA, Weisler SE, Baker-Ward L (1987) Cognitive science: an introduction. MIT Press, Cambridge, MA

Tseng YH, Lin CJ, Lin YI (2007) Text mining techniques for patent analysis. Inf Process Manage 43(5):1216–1247

Wang JZ, Taylor W (2007) Concept forest: a new ontology-assisted text document similarity measurement method. In: Proceedings of the IEEE/WIC/ACM international conference on web intelligence, IEEE Computer Society, pp 395–401

Wilkowski B, Fiszman M, Miller CM, Hristovski D, Arabandi S, Rosemblat G, Rindflesh TC (2011). Graph-based methods for discovery browsing with semantic predications. In: AMIA Annual Symposium Proceedings, pp 1514–1523

Yang H Callan J (2009) A metric-based framework for automatic taxonomy induction. In: Proceedings of the joint conference of the 47th annual meeting of the acl and the 4th international joint conference on natural language processing of the AFNLP, vol 1. Suntec, Singapore, Association for Computational Linguistics, pp 271–279

Yang Z, Lin H, Li Y (2010) BioPPISVMExtractor: a protein–protein interaction extractor for biomedical literature using SVM and rich feature sets. J Biomed Inform 43(1):88–96

Zelikovitz S, Hirsh H (2000) Improving short text classification using unlabeled background knowledge to assess document similarity. In: Proceedings of the seventeenth international conference on machine learning

Zheng HT, Kang BY, Kim HG (2009) Exploiting noun phrases and semantic relationships for text document clustering. Inf Sci 179(13):2249–2262

Zhou D, He Y (2008) Extracting interactions between proteins from the literature. J Biomed Inform 41(2):393–407

Zhou X, Zhang X, Hu X (2006) Maxmatcher: biological concept extraction using approximate dictionary lookup. In: PRICAI 2006 Aug 9-11, pp 1145–1149