3.2 INFORMATION EXTRACTION

Populating Knowledge Bases

Manual creation of knowledge bases is expensive

Can we produce them automatically?

Idea: Extract knowledge from documents

Challenge: Knowledge is encoded in natural language

Objectives

- Automated or accelerated creation of knowledge bases
- Support for structured search on documents

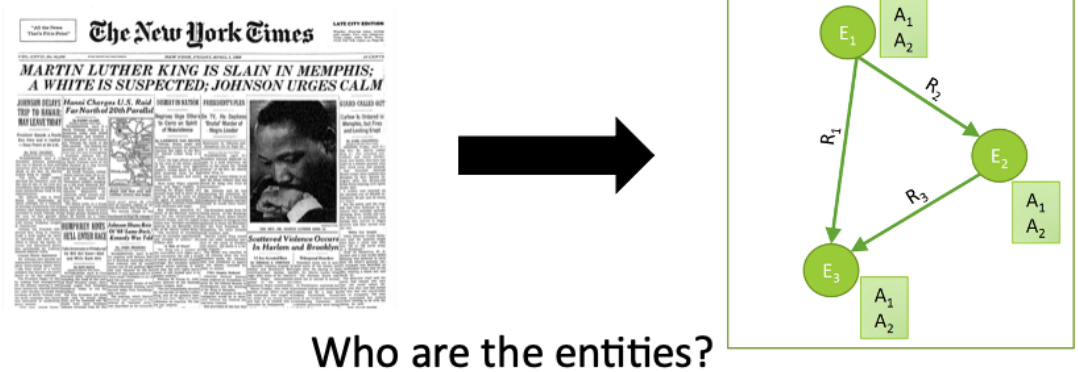
Traditionally knowledge bases are created manually, either by experts (e.g., WordNet) or by crowd-sourcing (e.g., WikiData). This is expensive. In the case of WordNet, it took tens of years to construct the knowledge base, in the case of WikiData (resp. WikiPedia) we all know about the notorious difficulty to finance this endeavor. Therefore, an interesting question is whether such knowledge bases could not be automatically constructed.

For automatic construction we can exploit data that is digitally available, e.g., all documents accessible on the Web. These documents encode massive human knowledge in natural language. The challenge is to extract such knowledge by analyzing natural language text, which is not an easy problem.

The results would, however, be immensely useful. First, we could create massive knowledge bases in a nearly automated way, and furthermore these knowledge bases could be used to annotate documents, for supporting more expressive and precise searches and analyses.

Information Extraction

From text to knowledge graphs



Who are the entities?

What are their attributes?

How are they related?

For investigating knowledge extraction, more commonly called information extraction, from textual content, we can consider the different constituents of a knowledge graph separately: entities, attributes and relationships. We will now introduce methods for extracting entities and then for establishing relationships among entities and with attributes.

3.2.1 Key Phrase Extraction

Idea: key phrase extraction is "the automatic selection of important and topical phrases from the body of a document" (Turney, 2000)

- Document summarization, search and indexing
- Document classification and opinion mining

EPFL is one of the two **Swiss Federal Institutes of Technology**. With the status of a **national school** since 1969, the **young engineering school** has grown in many dimensions, to the extent of becoming one of the most **famous European institutions** of **science and technology**. Like its **sister institution** in **Zurich**, **ETHZ**, it has three **core missions**: **training**, **research** and **technology transfer**. Associated with several specialised **research institutes**, the two **Ecoles Polytechniques** (**Institutes of Technology**) form the **EPF domain** , which is directly dependent on the **Federal Department of Economic Affairs, Education and Research (EAER)**.

A first type of information extraction method is key phrase extraction. Key phrase extraction aims at identifying words and phrases that are typical for a document and characterize important concepts that occur in a document. Key phrase extraction has been developed to support document summarization, where the key phrase gives an overview of the key concepts of a document. Key phrase extraction supports also document search and indexing, where key phrases are used to index documents. Moreover, key phrases can also provide useful features for document classification, i.e., key phrase extraction can be considered as a feature selection method. In the example text we see the possible outcome of key phrase extraction, with all identified key phrases marked in bold.

Keyphrase Extraction Methods

Approach: generate candidate phrases and rank them

Candidate phrases

- Remove stopwords
- Use word n-grams
- Consider part-of-speech tags (POS)

Baseline ranking approach

- rank candidate phrases of the document according to their tf-idf value

Advanced approaches

- Use of many structural, syntactic features of the documents
- Use of external resources, such as Wikipedia, Wordnet
- Use of transformer models

The basic approach to key phrase extraction uses principles well known from information retrieval. As in information retrieval stopwords are excluded. Key phrase candidates can be all word n-grams in the remaining text. Furthermore, part-of-speech tags can be used to further select the candidates, e.g., for excluding all verb phrases.

In order to assess whether a candidate phrase is characteristic for a document, tf-idf ranking is a possible approach. As in IR a candidate phrase is considered as relevant for a document if it is at the same time frequent and specific. Apart from that, many heuristics have been developed to refine this approach, considering additional features of the document. For example, a phrase in the title or a header could be considered as more relevant. External knowledge bases could be used to check whether a phrase corresponds to a commonly known concept. Recently, also transformer models have been applied for the task of key phrase extraction.

Use of Keyphrase Extraction

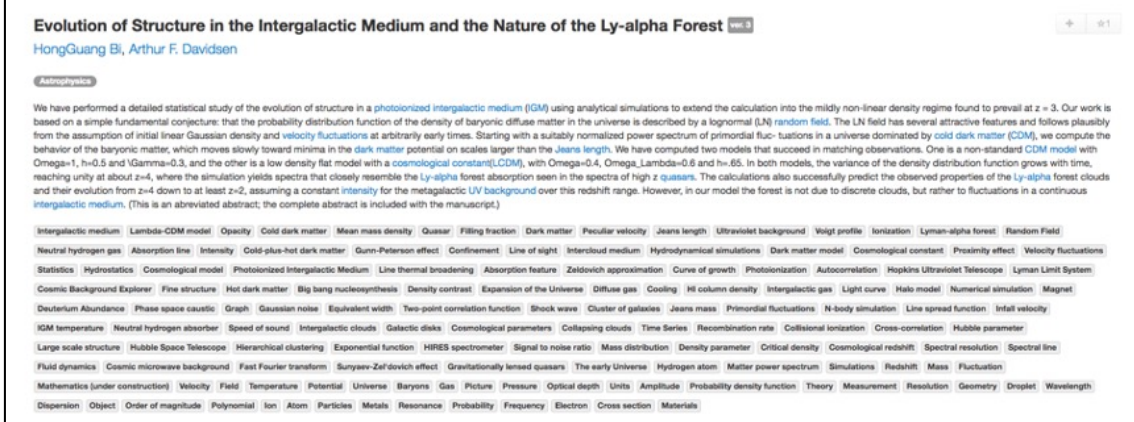
Creation of domain-specific thesaurus and taxonomy

Key phrase extraction can be used as an initial step to create a domain-specific thesaurus or taxonomy. In the example, we see a thesaurus that has been constructed for the food domain, based on key phrase extraction. For the phrase “junk food” many synonyms and near-synonyms have been identified. In practice a human expert would not be able to extract reliably all different types of mentions of such a concept.

As a result, this allows to increase recall for retrieval of documents that refer to this concept.

Use of Keyphrase Extraction

Document classification and search



Evolution of Structure in the Intergalactic Medium and the Nature of the Ly-alpha Forest ver. 3

HongGuang Bi, Arthur F. Davidsen

Astrophysics

We have performed a detailed statistical study of the evolution of structure in a photoionized intergalactic medium (IGM) using analytical simulations to extend the calculation into the mildly non-linear density regime found to prevail at z = 3. Our work is based on a simple fundamental conjecture: that the probability distribution function of the density of baryonic diffuse matter in the universe is described by a lognormal (LN) random field. The LN field has several attractive features and follows plausibly from the assumption of initial linear Gaussian density and velocity fluctuations at arbitrarily early times. Starting with a suitably normalized power spectrum of primordial fluc- tuations in a universe dominated by cold dark matter (CDM), we compute the behavior of the baryonic matter, which moves slowly toward minima in the dark matter potential on scales larger than the Jeans length. We have computed two models that succeed in matching observations. One is a non-standard CDM model with Omega=1, h=0.5 and \Gamma=0.3, and the other is a low density flat model with a cosmological constant(LCDM), with Omega=0.4, Omega_Lambda=0.6 and h=.65. In both models, the variance of the density distribution function grows with time, reaching unity at about z=4, where the simulation yields spectra that closely resemble the Ly-alpha forest absorption seen in the spectra of high z quasars. The calculations also successfully predict the observed properties of the Ly-alpha forest clouds and their evolution from z=4 down to at least z=2, assuming a constant intensity for the metagalactic UV background over this redshift range. However, in our model the forest is not due to discrete clouds, but rather to fluctuations in a continuous intergalactic medium. (This is an abreviated abstract; the complete abstract is included with the manuscript.)

Intergalactic medium | Lambda-CDM model | Opacity | Cold dark matter | Mean mass density | Quasar | Filling fraction | Dark matter | Peculiar velocity | Jeans length | Ultraviolet background | Voigt profile | Ionization | Lyman-alpha forest | Random Field | Neutral hydrogen gas | Absorption line | Intensity | Cold-plus-hot dark matter | Gunn-Peterson effect | Confinement | Line of sight | Intercloud medium | Hydrodynamical simulations | Dark matter model | Cosmological constant | Proximity effect | Velocity fluctuations | Statistics | Hydrostatics | Cosmological model | Photoionized Intergalactic Medium | Line thermal broadening | Absorption feature | Zeldovich approximation | Curve of growth | Photoionization | Autocorrelation | Hopkins Ultraviolet Telescope | Lyman Limit System | Cosmic Background Explorer | Fine structure | Hot dark matter | Big bang nucleosynthesis | Density contrast | Expansion of the Universe | Diffuse gas | Cooling | HI column density | Intergalactic gas | Light curve | Halo model | Numerical simulation | Magnet | Deuterium Abundance | Phase space caustic | Graph | Gaussian noise | Equivalent width | Two-point correlation function | Shock wave | Cluster of galaxies | Jeans mass | Primordial fluctuations | N-body simulation | Line spread function | Infall velocity | IGM temperature | Neutral hydrogen absorber | Speed of sound | Intergalactic clouds | Galactic disks | Cosmological parameters | Collapsing clouds | Time Series | Recombination rate | Collisional ionization | Cross-correlation | Hubble parameter | Large scale structure | Hubble Space Telescope | Hierarchical clustering | Exponential function | HIRES spectrometer | Signal to noise ratio | Mass distribution | Density parameter | Critical density | Cosmological redshift | Spectral resolution | Spectral line | Fluid dynamics | Cosmic microwave background | Fast Fourier transform | Sunyaev-Zel'dovich effect | Gravitationally lensed quasars | The early Universe | Hydrogen atom | Matter power spectrum | Simulations | Redshift | Mass | Fluctuation | Mathematics (under construction) | Velocity | Field | Temperature | Potential | Universe | Baryons | Gas | Picture | Pressure | Optical depth | Units | Amplitude | Probability density function | Theory | Measurement | Resolution | Geometry | Droplet | Wavelength | Dispersion | Object | Order of magnitude | Polynomial | Ion | Atom | Particles | Metals | Resonance | Probability | Frequency | Electron | Cross section | Materials

sciencewise.info

This is an example of using key phrase extraction for scientific documents in physics. Many highly specific concepts are identified automatically and allow a scientist to more precisely filter and search the documents. You can try this out on your own at sciencewise.info.

3.2.2 Named Entity Recognition (NER)

Task: Find and classify names of people, organizations, places, brands etc. that are mentioned in documents

organization organization

EPFL is one of the two **Swiss Federal Institutes of Technology**. With the status of a national school since 1969, the young engine georegion has grown in many dimensions, to the extent of becoming city ost famous **European** institutions of science and technology. Like its sister institution in **Zurich**, **ETHZ**, it has three core missions: training, research and technology transfer. Associated with several specialised research institutes, the two **Ecoles Polytechniques** (**Institutes of Technology**) form the **EPF domain** , which is directly dependent on the **Federal Department of Economic Affairs, Education and Research (EAER)**.

EPFL is located in **Lausanne** in **Switzerland**, on the shores of the largest lake in **Europe**, **Lake Geneva** and at the foot of the **Alps** and **Mont-Blanc**. Its main campus brings together over 11,000 persons, students, researchers and staff in the same magical place.

Named entity recognition is a more specific task than key phrase extraction. In NER the objective is to identify phrases that are names of specific types of entities, such as people, organizations or places. This, again, is very useful for document classification and search, but also a steppingstone to extract more complex knowledge, in particular statements relating different entities, as we will see later.

Named Entity Recognition (NER)

Uses of NER

- Named entities can be indexed, linked, etc.
- Sentiment can be attributed to companies or products
- Information extraction can use named entities as anchors

Commercial tools available

- Reuters' OpenCalais, AlchemyAPI (now IBM)
- Python libraries: NLTK NER, Spacy

NER has many commercial applications, e.g., for marketing or studying public perception, by linking volume of communication, sentiment and popularity to specific entities, such as products, companies or organizations. Thus, there exist manycommercial tools that offer this type of service.

NER as Sequence Labelling Task

Sequence of tags, indicating whether a word is inside (I) or outside of an entity (O)

The occurrences of entities (can be) typed

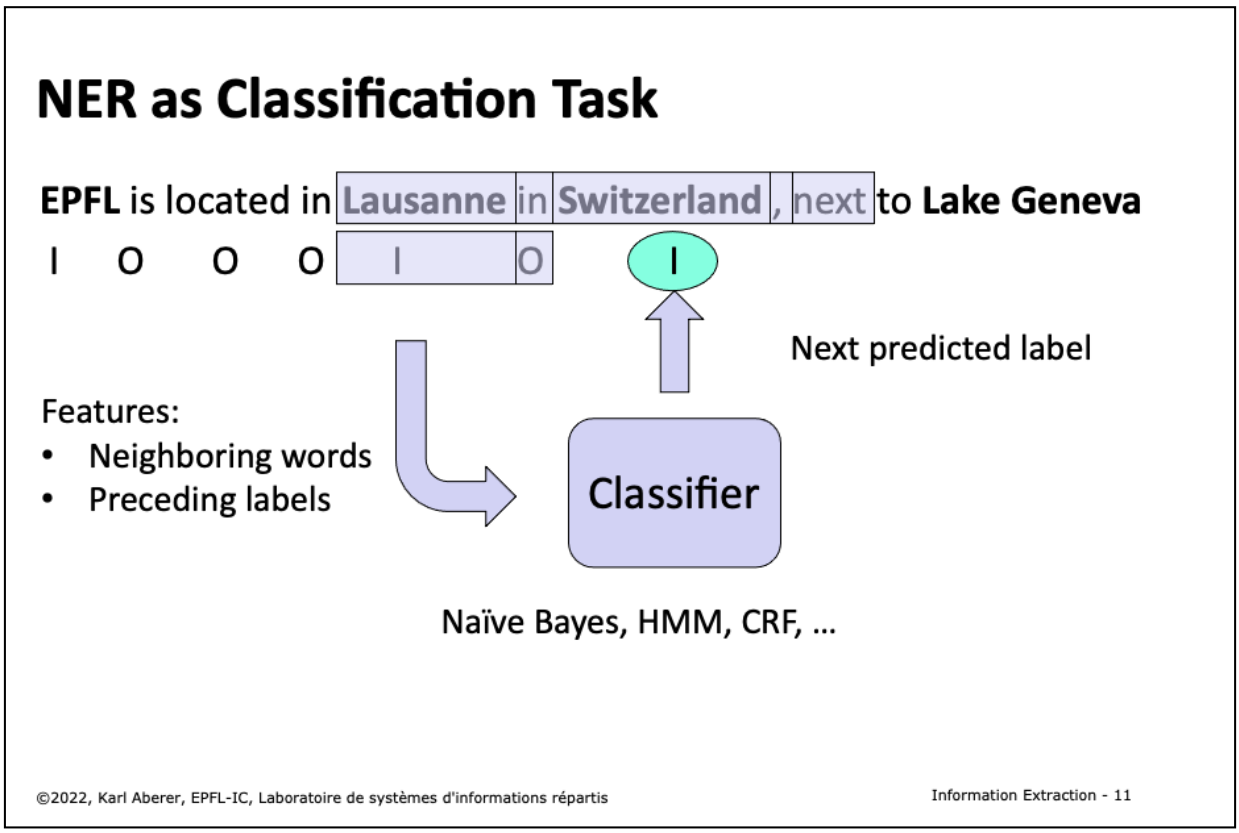
EPFL is located in **Lausanne** in **Switzerland** , next to **Lake Geneva**

I	O	O	O	I	O	I	O	O	O	I	I
ORG				GEO		GEO				GEO	GEO

A classification problem!

The basic task of NER is to detect whether a word belongs to an entity name or not. Furthermore, when an entity name is detected, it can be classified according to the type of the entity, e.g., an organisation (ORG), a location (GEO), a person etc.

When analyzing a text, NER is thus a classification problem, where for each word it needs to be decided whether it is inside or outside of an entity name. More detailed classifications, whether a word is the beginning or end of entity name, can also be performed. Note that in this context also punctuation marks are considered as words, as they may carry important information on the presence of an entity.



Given that NER can be considered as a classification problem, we must decide on two questions. First, which are the input features for the classifier, and second, which is the classification algorithm to be used. As for the input features, typically the neighborhood of a word is considered. In this neighborhood we find other words, which can be used as directly as features and from which several derived features can be produced. The classifier classifies words while reading the words in the sequence they appear in the document. Therefore, one special kind of feature that is used in NER are the labels that have been produced by the classifier for words preceding the word to be classified. Even though in principle any classification algorithm could be applied, e.g., Naïve Bayes, specific sequence-oriented classifiers (HMM, CRF) can have better performance.

Features used in NER

EPFL is located in **Lausanne** in **Switzerland** , next to **Lake Geneva**

Features of “Lausanne”:

Word and neighboring words:	Lausanne, in
Part-of-speech tags (POS):	POS(Lausanne) = NN
Prefixes and Suffixes:	prefix(Lausanne, 3) = Lau
Word shape:	WS(Lausanne) = Xxxxxxxx
Short wordshape:	SWS(Lausanne) = Xx

Here we see a list of typical features that are used in named entity recognition. Some of them are quite specific to the task. For example, part-of-speech tags can be helpful as they allow to distinguish noun phrases (NN) which are typical for entities. Pre- and suffixes are another interesting feature. For example, words ending in “land” would often be locations. Learning this fact can help to generalize the classification to new terms that would contain such a suffix. For entities, in particular in English, also the word shape is an important feature, as usually proper names start with capital letters, or acronyms consist of capital letters only.

Exploiting Context

When deciding the entity type exclusively on local context, important information may be missed

- The release of *Harry Potter and the Philosopher's Stone* in 2001 was **Watson's** debut screen performance.
- Although the system is primarily an IBM effort, **Watson's** development involved faculty and graduate students

Idea: consider a model that takes into the account the sequential structure of language and exploits sentence context

If only features derived from the local context from the immediate neighborhood of a word to be classified are used, important context information can be missed. This motivated the use of classification models that exploit the larger context of a word in the text, like the complete sentence in which the word occurs.

Generative Probabilistic Model

Sequence of words (known): $W = (w_1, w_2, w_3, \ldots, w_n)$

Sequence of labels (unknown): $E = (e_1, e_2, e_3, \ldots, e_n)$

Assume the text is produced by a probabilistic process:

$$P(E, W)$$

Find the most probable model

$$\underset{E}{\operatorname{argmax}}\, P(E|W)$$

Bayes Law

$$\underset{E}{\operatorname{argmax}}\, P(E|W) = \underset{E}{\operatorname{argmax}}\, P(E)P(W|E)$$

We introduce now a classification approach that has been used for NER and exploits the sequential nature of natural language. The approach belongs to the class of generative probabilistic models, like the one we have introduced for information retrieval.

The basic model assumes that there exists a (unknown) probability distribution $P(E, W)$ that correlates sequences of words with the corresponding sequences of entity labels. The classification task is then to identify for a given sequence of words, the most probable sequence of labels. Using Bayes law, we can reformulate this, by decomposing the conditional probability $P(E|W)$ into the product of two probability distributions. $P(E)$ is a model describing of the probability of different labels to occur, and $P(W|E)$ is a model describing of how words correlate with labels.

Approximation

Label transition probabilities (bigram model)

$$P(E) = P(e_1, \ldots, e_n) \approx \prod_{i=2,\ldots,n} P_E(e_i|e_{i-1})$$

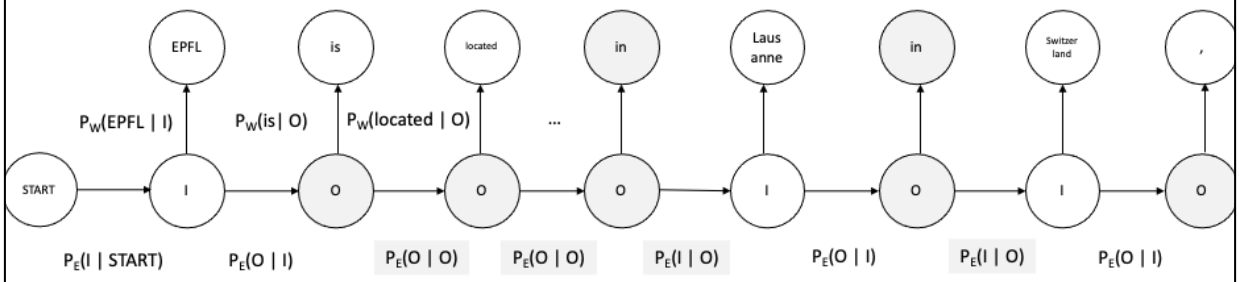
Word emission probabilities

$$P(W|E) \approx \prod_{i=1,\ldots,n} P_W(w_i|e_i)$$

As it is not possible to estimate the complete probability distribution functions $P(E)$ and $P(E|W)$, we approximate them by making independence assumptions. We assume that the probability of a label to occur, depends only on the previous label. This corresponds to a bigram model, generalizing the unigram model we have introduced in probabilistic information retrieval. For a word we assume that its probability of occurrence depends only on the label it received. Thus, the two probability functions decompose into products of simpler functions that we can estimate.

Hidden Markov Model (HMM)

Graphical representation of the approximate probabilistic model



Maximum Likelihood Estimation

$P_E(I \mid O) = 2 / 4$, $P_W(\text{in} \mid O) = 2 / 5$

We can represent approximate model of the probability distribution graphically as a Markov Model, where we indicate which probabilistic variables depend on which others. More precisely, it is a Hidden Markov Model (HMM). The hidden labels E are unknown, and their probabilities need to be estimated from the words that can be observed.

This approach for estimating probabilities of hidden variables with HMMs can be applied to other sequence labelling tasks. For example, it can be used to learn part-of-speech tags and the types of the entities.

Learning the Model

To learn the conditional probabilities from a document collection using Maximum Likelihood Estimation requires only counting

$$\text{e.g., } P_E(I|O) = 2/4\,, PW\,(in|O) = 2/5$$

Smoothing: Unseen words might only accidentally miss in the training data of length n :

$$P_{WS}(w_i|e_i) = \lambda\, P_W(w_i|e_i) + (1-\lambda)\,\frac{1}{n}$$

For labels no smoothing is needed, as all labels occur in the training data

Given a document collection we can estimate the probabilities $P_E(e_i|e_{i-1})$ and $P_W(w_i|e_i)$. As in probabilistic information retrieval we use maximum likelihood estimation. For example, for estimating the probability $P_E(e_i|O)$ we count the total number of occurrences of O, and then compute the ratio between the cases where the preceding label is e_i with the total number of occurrences of O.

As in probabilistic information retrieval, we need to consider the issue of sparsity of words in the training set. It might be the case that a specific word does not occur together with a given label in the training data, whereas this word still might be related to the label in general. Therefore, smoothing is applied, where the smoothing parameter depends on the size of the training data. The more data is available, the less the likelihood that a word-label pair that is likely to occur is not found in the training data.

For estimating the probability of labels to occur, no smoothing is required, as the number of labels is very small and thus all pairs of labels combinations are likely to occur in the training data.

Using the Model

For a given sequence of words W find the most likely values for the labels E

$$\underset{E}{\operatorname{argmax}}\, P(E|W)$$

Brute force search: compute for all possible sequences $E = (e_1, e_2, e_3, \ldots, e_n)$ the probability $P(E|W)$ and then take the maximum

Complexity $O(2^n)$ → unfeasible for longer sequences

Once the parameters of the HMM model have been derived from the training data, they can be used to estimate the most likely values of labels for an unknown sequence of words. One possibility is to apply brute-force search by computing the probability of each possible label sequence using the model. For longer sequences this becomes computationally intractable as the number of possible sequences grows exponentially.

Observation

$$\underset{E}{\operatorname{argmax}} P(E|W)$$

$$= \underset{E}{\operatorname{argmax}} \prod_{i=2,\ldots,n} P_E(e_i|e_{i-1}) \prod_{i=1,\ldots,n} P_W(w_i|e_i)$$

$$= \underset{E}{\operatorname{argmax}} P_E(e_n|e_{n-1})\, P_W(w_n|e_n)$$

$$\underset{E}{\operatorname{argmax}} \underbrace{\prod_{i=2,\ldots,n-1} P_E(e_i|e_{i-1}) \prod_{i=1,\ldots,n-1} P_W(w_i|e_i)}_{\text{Independent of the choice of } e_n}$$

We made an independence assumption on the probabilities of the elements of a label sequence, where a label depends only on its predecessor in the sequence. We can exploit this independence assumption to simplify the computation of the sequence probability. The choice of the the last label in the sequence that maximizes the overall probability is independent of the choices of the other labels in the sequence maximizing the overall probability. Using this property simplifies the computation of the sequence probability significantly. Note tha

Viterbi Algorithm

Let $\pi(k, v)$ be the maximum probability a sequence of length k can achieve with last label v

Then

$$\pi(k, v) = \max_u \pi(k-1, u)\ P_E(v|u)\ P_W(w_k|v)$$

$$\pi(0, *) = 1$$

This is a dynamic programming algorithm
→ Viterbi algorithm

Using the independence assumption described before, we can iteratively compute the sequence of labels that maximizes the probability, using in each step the label sequence found so far. The maximum probability $\pi(k, v)$ that a label sequence of length k can achieve, if the last label is v can be computed from the maximum probabilities known for shorter sequences and the parameters of the probabilistic model.

This algorithm is a simple version of Viterbi's algorithm. In its general form a random variable can depend on several earlier random variables in the earlier sequence, resulting in a dynamic programming algorithm.

An HMM model would not be an appropriate approach to identify

A. Named Entities
B. Part-of-Speech tags
C. Concepts
D. Word n-grams

Which statement is correct?

A. The Viterbi algorithm works because words are independent in a sentence
B. The Viterbi algorithm works because it is applied to an HMM model that makes an independence assumption on the word dependencies in sentences
C. The Viterbi algorithm works because it makes an independence assumption on the word dependencies in sentences
D. The Viterbi algorithm works because it is applied to an HMM model that captures independence of words in a sentence

3.2.3 Information Extraction (IE)

Task: Extract statements from text
→ creation of knowledge graphs

EPFL is one of the two **Swiss Federal Institutes of Technology**. With the status of a national school since 1969, the young engineering school has grown in many dimensions, to the extent of becoming one of the most famous **European** institutions of science and technology. Like its sister institution in **Zurich**, **ETHZ**, it has three core missions: training, research and technology transfer. Associated with several specialised research institutes, the two **Ecoles Polytechniques** (**Institutes of Technology**) form the **EPF domain** , which is directly dependent on the **Federal Department of Economic Affairs, Education and Research (EAER)**.

EPFL is located in **Lausanne** in **Switzerland**, on the shores of the largest lake in **Europe**, **Lake Geneva** and at the foot of the **Alps** and **Mont-Blanc**. Its main campus brings together over 11,000 persons, students, researchers and staff in the same magical place.

Taking the analysis of documents one step further, we now consider the extraction of statements from natural language text, as is illustrated in the example. Statements connect entities through relationships. For example, the first statement expresses that EPFL is part of a larger organization, the Swiss Federal Institutes of Technology. The notion of statement we use here corresponds exactly to the notion of statement we introduced earlier with RDF.

Sample Statements

EPFL is one of the two **Swiss Federal Institutes of Technology**

EPFL - IS-A - Swiss Federal Institute of Technology

its sister institution in **Zurich, ETHZ**

EPFL - RELATED-TO - ETHZ

EPF domain , which is directly dependent on the **Federal Department of Economic Affairs**

EPF Domain - DEPENDS-ON - FDEA

EPFL is located in **Lausanne**

EPFL – LOCATED-IN - Lausanne

Lake Geneva and at the foot of the **Alps**

EPFL – LOCATED-IN - Alps ? Lake Geneva – LOCATED-IN – Alps?

Looking more closely at some of the statements we can extract from the text, we make the following observations. First, statements are always anchored in two entities, thus they link two entities by a relationship. Second, the relationships can carry different meanings, which in natural language are typically expressed in verbs. Third, the extraction of statements can be ambiguous. In the last example, when looking at the original text, the implied meaning is that EPFL is close to the alps. When looking only at the local context of ”Lake Geneva” and “Alps”, one might make also the incorrect inference that the statement is about Lake Geneva located in the alps. Accidentally this is not a wrong statement, but not the one that is intended in the text. So, statement extraction can be a tricky task due to the ambiguity and complexity of human language.

Typed Statements

EPFL – PART-OF – Swiss Federal Institute of Technology

Type: ORG – PART-OF – ORG

EPFL – RELATED-TO – ETHZ

Type: ORG – RELATED-TO – ORG

EPF Domain – PART-OF – FDEA

Type: ORG – PART-OF – ORG

EPFL – LOCATED-IN - Lausanne

Type: ORG – LOCATED-IN – LOC

EPFL – LOCATED-IN - Alps ? Lake Geneva – LOCATED-IN – Alps?

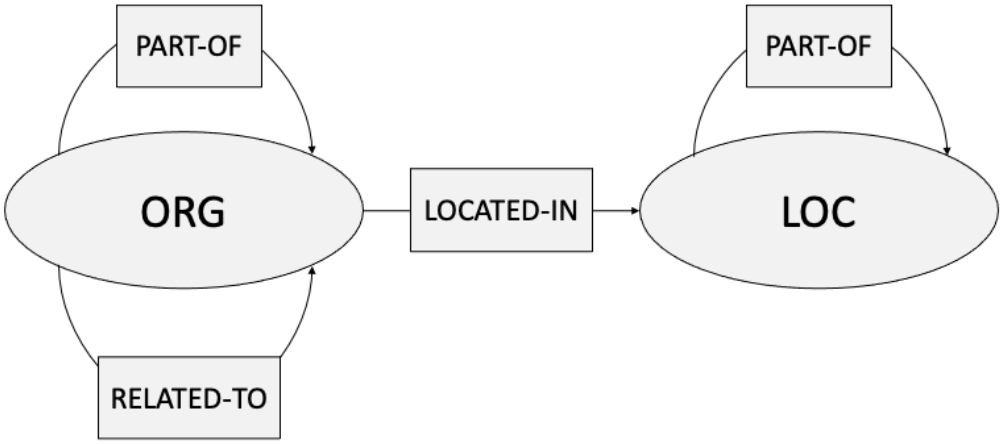
Type: ORG – LOCATED-IN – LOC

Type: LOC – LOCATED-IN – LOC

With NER we can extract entities of given types. Using entity types, we can also introduce statements of given types. The types in a statement can concern its three components, the subject, the object and the predicate. This allows to exclude statements that are meaningless, like a location being part of a person. It also helps in the task of detecting statements, since only entity pairs and predicates that match the type constraints need to be considered.

Note that the same entity types can be related by predicates of different types. For example, two universities could be related by relationships “perform joint research” and “exchange students”. When using a model for knowledge graphs, such as RDF, the types of the statements would be specified in the schema, e.g., RDF Schema.

Statement Schema



This is an example of a possible schema that constrains the type of statements to be considered.

Approaches to Information Extraction

- Hand-written patterns
- Supervised machine learning
- Bootstrapping
- Distant supervision
- Matrix Factorization

Information extraction is a central problem in interpreting and using text data, and has therefore attracted a lot of interest. Many different methods have been developed, of which we will now introduce some of the most important examples. Note that even in the recent time transformer models are successfully applied to this task as well, the earlier models have still their interest as they are often computationally less expensive and incorporate relevant ideas on how to perform information extraction that are also of interest for developing mode complex models.

3.2.3.1 Hand-Written Patterns

Early approach from **Hearst (1992)**

- "Agar is a substance prepared from a mixture of **red algae, such as Gelidium**, for laboratory or industrial use"

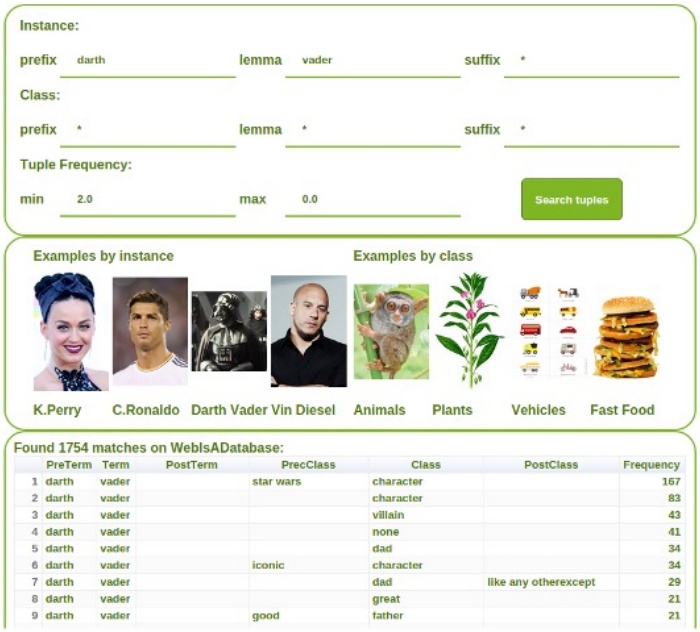
Patterns to detect IS-A relationships:

```
"Y such as X ((, X)* (, and|or) X)"
"such Y as X"
"X or other Y"
"X and other Y"
"Y including X"
"Y, especially X"
```


An early approach to information extraction is based on the observation that in natural language a relationship is often expressed in a regular fashion. This observation has been exploited to extract specific relationships, such as ISA, by using regular expression patterns. Hearst was one of the first to introduce this approach, and the method is still being used till today. The performance of the method depends on the quality and diversity of patterns used.

Web isa Database

Large scale extraction of IS-A relationships from web documents



	PreTerm	Term	PostTerm	PrecClass	Class	PostClass	Frequency
1	darth	vader		star wars	character		167
2	darth	vader			character		83
3	darth	vader			villain		43
4	darth	vader			none		41
5	darth	vader			dad		34
6	darth	vader		iconic	character		34
7	darth	vader			dad	like any otherexcept	29
8	darth	vader			great		21
9	darth	vader		good	father		21

http://webdatacommons.org/isadb/

Web is a DB is an example a large-scale effort to extract IS-A relationships from Web data using Hearst patterns. The patterns are more complex than the ones initially introduced by Hearst, but the basic idea is the same.

More General Hand-Written Patterns

Idea: relations often hold between specific entity types

- located-in (ORGANIZATION, LOCATION)
- founded (PERSON, ORGANIZATION)
- cures (DRUG, DISEASE)

First, perform Named Entities Recognition

Use typed pattern: ORG is located in LOC, LOC

EPFL is located in **Lausanne**, **Switzerland**

The idea of Hearst that has been successfully applied for detecting ISA relationships, can be extended to patterns of a more general type, for extracting statements for other types of relationships. This approach exploits the fact that certain relationships can only hold among certain types of entities. Thus, in a first step a named entity recognition is performed, and subsequently the patterns are searched for which the matching types of entities can be found.

Summary Hand-Written Patterns

Advantages

- Rules tend to be high-precision
- Can be tailored to specific domains

Disadvantages

- Human patterns are often low-recall
- A lot of effort to think of all possible patterns

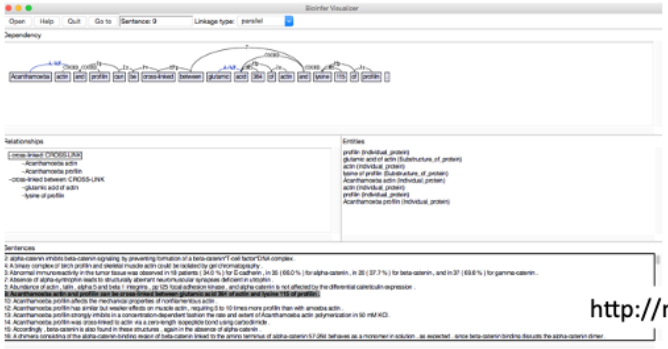
Hand-written patterns are in general a very reliable method for information extraction. However, it suffers from low recall, and it is very difficult to conceive all possible patterns by a human expert. Therefore, more automated methods are often preferrable.

3.2.3.2 Supervised Learning for IE

Approach: train a classifier on labeled data

Creating a training set

- Choose relevant named entities and their relations
- Manually label relations among entities (positive examples)



http://mars.cs.utu.fi/BioInfer/

A supervised learning approach for information extraction requires training data. Producing such a training set is labor-intensive. For example, in the domain of life sciences major efforts have been undertaken. In the example shown, more than 2000 sentences have been manually annotated, by identifying both the entities and their relationships.

Classifiers for Information Extraction

Two-step approach

- A **filtering classifier** (e.g., Naïve Bayes), to detect whether a relation exists among the entities
- A **relation-specific classifier** detecting the relation label

Training the classifiers

- Extract named entities in the document corpus using NER
- Detect pairs of entities, e.g., in the same sentence
- Use unlabeled entity pairs as negative examples

Provided manually annotated training data, classifiers for information extraction can be trained. The standard approach is to train two types of classifiers, one that detects whether a relationship exists among two entities, and a second one that determines the type of the relationship. The use of a filtering classifier for detecting relationships can speed up the classification task and allows to use of different features for the two tasks.

For training the classifiers, one first extracts all entity pairs using NER that occur in the same context, for example, in the same sentence. If the pairs are not annotated, they can be taken as negative examples. Then the classifier is trained using features extracted from the context of the occurrences of the entities.

Features Used in Information Extraction

EPFL is located in **Lausanne** in **Switzerland**, next to **Lake Geneva**

Features for mention (M1, M2) = (Lausanne, Lake Geneva):

BOW and bigrams in the sentence:	is, located, in, located in, Lausanne, in, next to …
BOW and bigrams in between the mentions:	in, Switzerland, next, to, next to
Headwords* of mentions, their concatenation:	Lausanne, Geneva, Lausanne-Geneva
Words in positions:	M1-1: in, M1+1: in, M2-1: to
Types of entities	LOC, LOC
Stemmed version of the words	
Syntactic features	
…	

*headword = entry in a dictionary

Diverse text features can be used for training a classifier for information extraction. After detecting entity pairs using NER, the so-called mentions, the features for relation extraction can use features of the both entities as well as of the text surrounding them and in between.

When we have identified two occurrences of named entities in a sentence, also called two mentions (M1, M2), then we can identify the following features related to the two mentions:

- The bag of words and bigrams found within the whole sentence
- Separate from that the BOW and bigrams in between the mentions
- The headwords, which are words that are found in a standard dictionary
- Words in specific position with respect to the mentions
- Stemmed versions of all the words above
- The type of entities used
- Syntactic features, extracted with part-of-speech analysis

Syntactic Features

Parse Tree

```
(S (NP EPFL)
        (VP is
                (VP located
                        (PP in
                                (NP Lausanne))
                        (PP in (NP Switzerland ,
                                (PP next to
                                        (NP Lake Geneva)))))))
```

Features:

Sequence between entities: PP NP PP NP

The syntactic features, or part-of-speech tags can be exploited in various ways. In the simplest case only the sequence of POS tags in between the mentions is used. More complex features can be constructed as well, e.g., the navigation path between the mentions in the parse tree. For trying out POS tagging you can try: https://www.link.cs.cmu.edu/link/submit-sentence-4.html

Which is true?

A. Hand-written patterns are in general more precise than classifiers
B. Hand-written patterns cannot exploit syntactic features
C. Supervised classifiers do not require any human input
D. Supervised classifiers can only detect typed statements

3.2.3.3 Bootstrapping

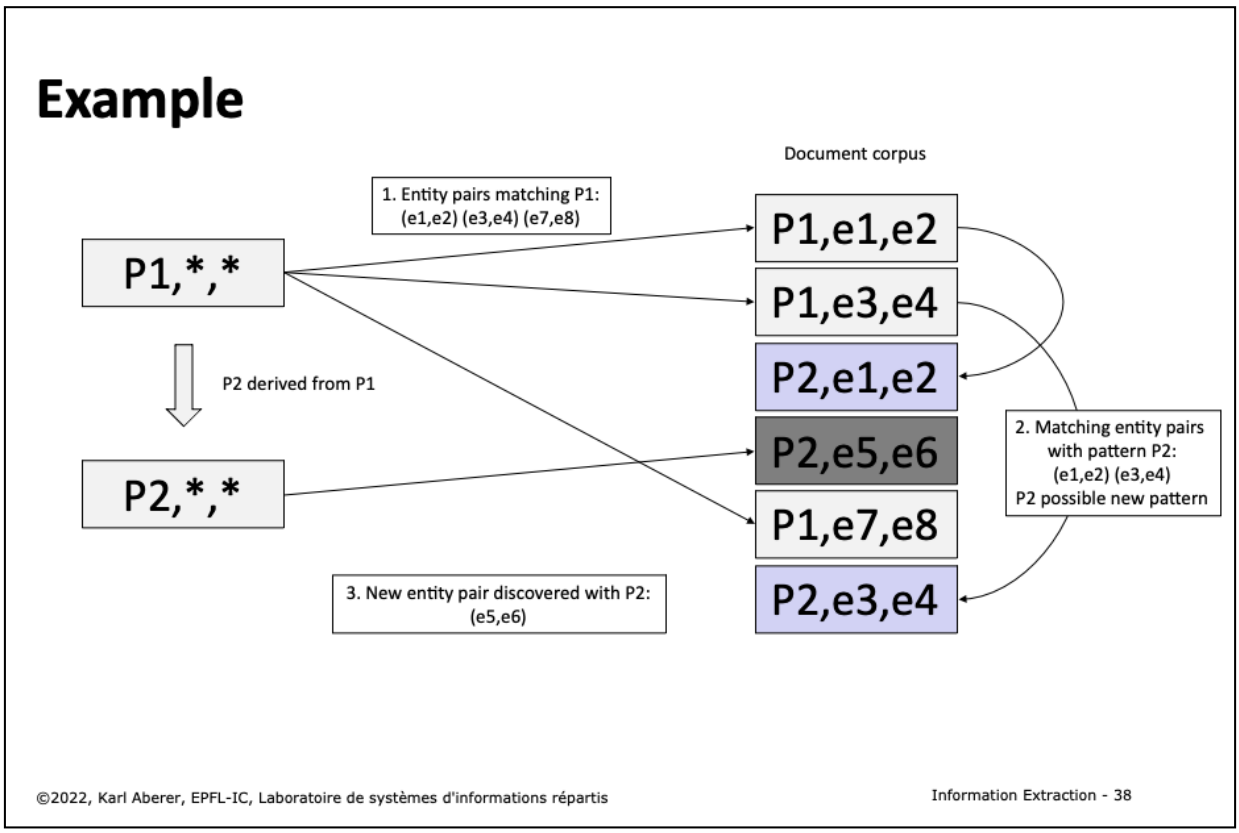
No training data, but a few high-precision patterns

Approach:

- Find entity pairs that match the pattern
- Find sentences containing those entity pairs
- Generalize the entities in those sentences
- Generate new patterns

One of the big problems with supervised approaches to information extraction is the scarcity of training data. One way to avoid the use of training data for information extraction is called bootstrapping. The basic idea is that on a few high-precision patterns, such as the Hearst patterns, one generalizes these patterns by analyzing a large text collection.

The approach is to first find entity pairs using the high precision patterns. Then using those entity pairs sentences containing the same entity pairs are searched. The assumption is that with large likelihood these sentences express the same type of relationship, just in a different syntactic representation. Thus, such sentences can be considered as text templates for expressing the relationship from which new patterns for detecting the relationship can be derived.



This figure illustrates the different steps of how new patterns are detected and applied to find new relationships. P1 is a known pattern for detecting the relationship. It is used to detect matching entity pairs. Then, in a second step, such pairs are searched. The text associated with these pairs can then be used to infer new pattern P2.

Example

Pattern: **LOC** is located in **LOC**

- Mumbai **is located in** India
- Adelaide **is located in** Southern Australia
- Sriharikota **is located in** Nellore

Search for entity pairs (Mumbai, India)

- **Mumbai** is **India's** top destination
- **Mumbai** hotels, **India**

New patterns

- **LOC** is **LOC**'s top destination
- **LOC** hotels, **LOC**

This example illustrates the approach. We start with a simple, but precise pattern, LOC is located in LOC, which we use to find occurrences of entity pairs. Then we search for those entity pairs and find other sentences mentioning them. These are then generalized to patterns by replacing the entity occurrences by their types.

Problem: Semantic Drift

Example: The pattern

LOC hotels, **LOC**

matches also

… Geneva hotels, Lausanne hotels …

➔ Geneva is located in Lausanne?

A potential problem when using this approach is that the new patterns that are found are potentially misleading and may create inaccurate results. The example shows an instance of such a problem.

Confidence

Assume we have a confirmed set of pairs of mentions M

- A new pattern should also match many of those

$Hits_p$ = number of pairs in M that a new pattern matches

$Finds_p$ = total number of pairs that a new pattern matches

Confidence that a new pattern finds many relevant mentions

$$Conf(p) = \frac{Hits_p}{Finds_p} \log(Finds_p)$$

In order to contain the problem of inferring too many inaccurate patterns, a confidence metric can be used. With this metric the confidence into a new pattern increases, when the fraction of matched entity pairs that are already confirmed to be correct is large. Otherwise said, if a new patterns frequently matches, but only few of the matched entity pairs are known to be in the intended relation, the confidence in the new pattern will be low.

Confidence in Statements

A statement s may match a set of patterns P

- Each pattern with a given confidence

Assuming confidence is a probability

Probability that the statement s is correct

$$Conf(s) = 1 - \prod_{p \in P} (1 - Conf(p))$$

Only statements with sufficient confidence are used to infer new patterns

When inferring new patterns with bootstrapping, statements that have been spotted in the text involving the relevant entities are considered as new templates. In order to avoid semantic drift, one can be selective about which statements to consider for creating new patterns, by considering confidence of the rules that match the specific statement. This can be done by using the confidence of patterns and considering it like probabilities. The product of the values of $(1 - Conf(p))$ can then be considered like the probability of all patterns being wrong, and therefore the complement being the probability of at least one pattern being right.

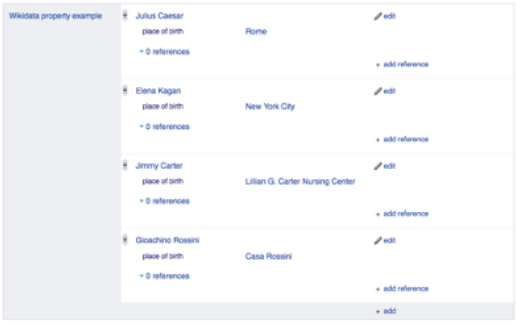
3.2.3.4 Distant Supervision

No Training Data? **Idea**: use existing knowledge bases to collect training data for building a classifier

- Combines advantages of bootstrapping with supervised learning

Example: learning PLACE-OF-BIRTH

WikiData has many positive examples!



Another approach to address the problem of lacking training data is to use knowledge bases that contain large number of acts, such as WikiData. The facts recorded in such knowledge bases can be used to spot corresponding statements in text. Such statements can then be used as text patterns for fact representation, that can be provided as training data to a classification algorithm for information extraction.

Linking Text to Knowledge Bases

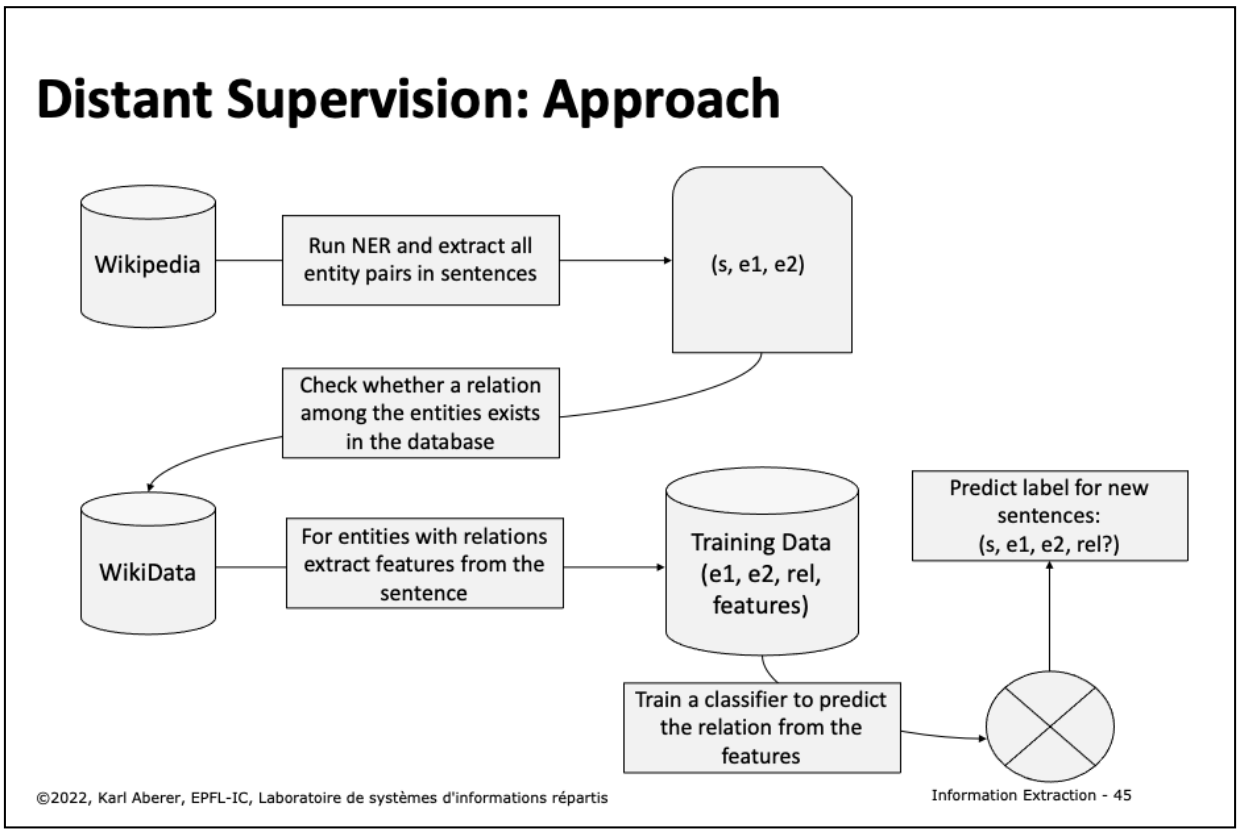
Using Entity extraction, entity mentions in text can be linked to corresponding mentions in a knowledge base

John was born in **Liverpool**, to **Julia** and **Alfred Lennon**

Entities		Text Features			Relation from knowledge base	
Entity 1	Entity 2	PER was born in LOC	PER was born to PER	PER and PER	Birthplace(X,Y)	Married(X,Y)
John Lennon	Liverpool	x			?	
John Lennon	Julia Lennon		x			
John Lennon	Alfred Lennon		x			
Julia Lennon	Alfred Lennon			x		?
Barack Obama	Hawaii	x			x	
Barack Obama	Michelle Obama			x		x

Barack Obama was born in **Hawaii. Barack** and **Michelle Obama …**

In this figure we illustrate of how entity pairs identified in text can be linked to the same entity pairs found in a knowledge base. This allows to infer the meaning of syntactic patterns (e.g., ”was born in”) and relate such syntactic patterns to the corresponding relationship (e.g., birthplace). In the example, the system can first learn from the knowledge base text patterns for known facts (about Barack Obama), and then use those patterns to infer new facts from text (about John Lennon). The matrix shown in this figure we call the entity-pairs/relation matrix.



The figure illustrates the steps in the distant supervision method. First, it starts by doing NER over a large document collection, such as WikiPedia. This produces a set of sentences s that contain entity pairs (e1, e2). Having those entity pairs, it searches in the knowledge base, in this example WikiData, whether relations among the entity pairs exist. For those entity pairs, for which a relation is confirmed in the knowledge base, it can generate an instance for the training data. The training data consists of the entity pair, the type of relations and features extracted from the text. The result is a training data set that can be used to train a classifier to predict relations for new documents. The classifier extracts from a new document the entity pairs and feeds them together with the text features into the classifier to obtain the type of relationship.

Features for Distant Supervision

Use conjunctions of standard IE features as sentence features

- Match only if all individual features match
- High precision, but low recall features!
- Feasible, since training set is large

Complex features resemble to templates used in rule-based approaches

With distant supervision it is possible to obtain many training samples. Therefore, it is possible to use a large number of features for classification. One way to obtain a large number of features is to consider each conjunction if individual text features as a separate feature. Such features resemble then to patterns that are used in rule-based approaches. They have typically low recall, but higher precision.

Example

Example sentences

- **Mumbai** hotels, **India**
- **Geneva** hotels**, Lausanne** hotels

Individual features

F1: M1=LOC, F2: M2=LOC, F3: between={hotels} , F4: after={}
F1: M1=LOC, F2: M2=LOC, F3: between={hotels} , F4: after={Switzerland}

Complex Features

CF1: M1=ORG and M2=LOC and between={hotels} and after={}
CF2: M1=ORG and M2=LOC and between={is, located, in} and after={hotels}

the two sentences have two different, unrelated complex features

Here we illustrate the use of complex features. Assume we have a set of basic features, like those that we have identified for supervised information extraction. F1 to F6 are examples of such features. These features could as such be supplied to a classification algorithm, e.g., a Naïve Bayes classifier. Given that the features of the two examples are very similar, only F5 is different, the classifier would probably decide that the relation represented by the two sentences would be the same.

Using the basic features and combining via a conjunction into one complex features, we would obtain two different complex features CF1 and CF2. A classifier that does not further consider the internal structure of these two features would now clearly distinguish the two cases.

Which is true?

A. Distant supervision requires rules for bootstrapping
B. Classifiers produced with distant supervision are more precise than rules
C. Distant supervision can help to detect rules

3.2.3.5 Matrix Factorization

Using the same data as for distant supervision

- Entity pairs from text, linked to relations from knowledge bases

Instead of learning a classifier, create low-dimensional representations for entity pairs and relations

Use those representations to

- Link text patterns to relation types and identify similar text patterns
- Extract relations from text

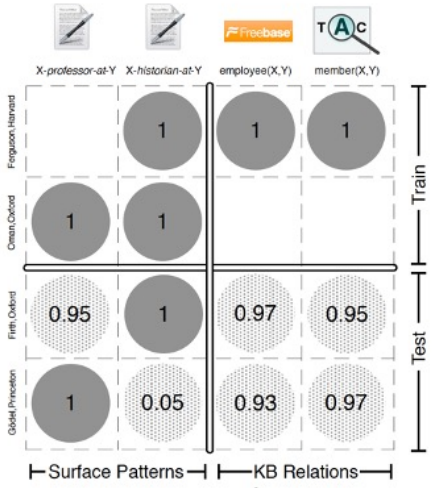
Distant supervision aims at generating classifiers for relations that are based on syntactic features. Based on the same data as used for distant supervision we can also create representations for relationships by mapping them to low-dimensional vectors, as we did for words with word embeddings. This is the idea of using matrix factorization to the entity-pairs/relation matrix.

Matrix Representation

Create a matrix with

- Entity pairs as rows
- Relation types as columns
  - Relations from text patterns
  - Relations from knowledge base

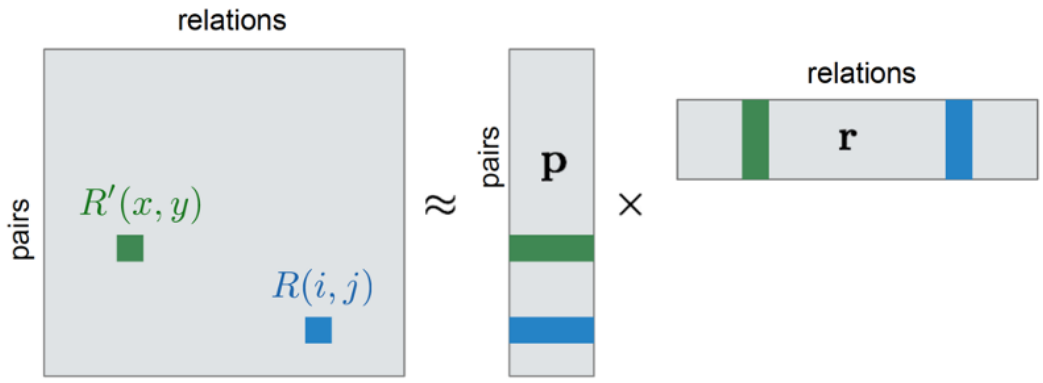
The entity-pair/relation matrix is a sparse matrix (like in recommender systems)



The intention of using matrix factorization is align text patterns and relationships that have a related meaning in a latent space. This can help both in identifying text patterns that correspond to relationships and to directly extract those relationships from text.

The entity-pair/relation matrix is a sparse matrix. Thus, the situation is comparable to the one we encountered in matrix factorization for recommender systems. Algebraic factorization does not work. On the other hand, using matrix factorization based on SGD could not only create low-dimensional representations for relationships, but also help to detect new relationships. The idea is like the one used in recommender systems to estimate missing ratings.

Matrix Factorization



$R(i,j)$ positive examples of facts

$R'(x,y)$ negative examples of facts

By factorizing the matrix, we construct two factor matrices, one with low-dimensional representations of entity pairs, and one with low-dimensional representations of relationships. An entry R(i,j) in the entity-pair/relation matrix is then represented as the product of the corresponding representation of the entity pair and the relation from the factor matrices. It corresponds to a statement, the one that puts the two entities from the entity pair into the corresponding relation. The entries correspond to the probability that the statement is correct. For known facts, for which an entry in the original matrix exists, the factorized matrix should have high values. Unknown facts, for which no entry exits in the original matrix, are considered as negative examples of facts.

Bayesian Personalized Ranking (BPR)

Idea: give observed true facts higher ranking than unobserved (true or false) facts

Approach: create ranked pairs f^+ and f^-

Objective Function

$$\sum_{f^+,f^-} \log \sigma(\theta_{f^+} - \theta_{f^-})$$

where

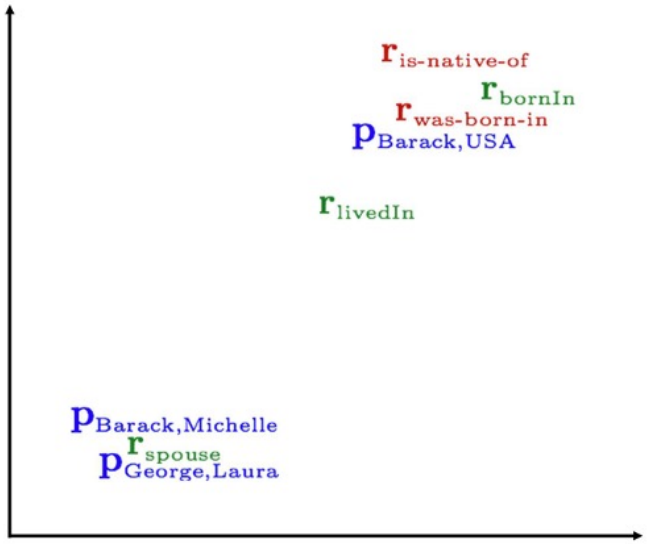
$$\theta_f = \boldsymbol{p} \cdot \boldsymbol{r}$$

Maximize with Stochastic Gradient Descent

Assuming the absence of knowledge on a fact, means that the fact is wrong, is an assumption that is too strong. It could as well be that the fact is correct, but we do not have the knowledge. This is like the situation in recommender systems, where a missing rating does not necessarily imply that the rating is negative.

To better adjust to this situation an alternative to matrix factorization is a method called Bayesian Personalized Ranking. It is based on an alternative loss function for SGD. The loss function is constructed by choosing pairs of positive and negative examples of facts and maximizing their difference. More accurately, the logarithm of the sigmoid of the difference is used as optimization objective. Intuitively this objective function says that a known fact is ranked higher than an unknown fact but does not exclude the possibility that the unknown fact is also a correct fact.

Relation Embeddings



The matrix factorization computed using BPR allows to map entity pairs and relationships in the same low-dimensional space. As illustrated in this figure, this allows to cluster both entity pairs and relations that correspond to similar relationships. This allows to infer new relationships for existing entity pairs, as well as new syntactic patterns for relationships.

Exploiting Relation Embedding Similarity

Similar in embedding space

Similar in embedding space

Entities		Relationship pattern	
Entity 1	Entity 2	COM owns part in COM	COM buys stake in COM
Renault	Nissan	x	x
BMW	Rover		x
Volkswagen	Porsche		
Ford	Toyota		

Possible inferences:

- BMW owns part in Rover (similarity of relationship)
- Volkswagen owns part in Porsche (similarity of entity pair)

This example illustrates of how the method could be used to extract relationships for previously unknown syntactic patterns, as well as relations among entity pairs that are no previously known.

Question

When searching for an entity e_{new} that has a given relationship r with a given entity e

A. We search for e_{new} that have a similar embedding vector to e
B. We search for e_{new} that have a similar embedding vector to e_{old} which has relationship r with e
C. We search for pairs (e_{new}, e) that have similar embedding to (e_{old}, e)
D. We search for pairs (e_{new}, e) that have similar embedding to (e_{old}, e) for e_{old} which has relationship r with e

Summary

Information extraction
- Populating knowledge bases and fact databases
- Taxonomy induction

Pattern-based approaches
- High precision, low recall, work intensive

Supervised learning
- Low precision, high recall, work intensive

Hybrid methods: bootstrapping, distant supervision, matrix factorization

Unknown types of relationships: open information extraction

We have introduced the different methods of information extraction. These methods aim at extracting relationships of which the type is known, i.e., with specific entity types and a known meaning of the relationship. A more general problem in information extraction is so-called open information extraction. It is used for extracting statements from large collections of documents, such as the Web, where the nature of relations is not known a priori.

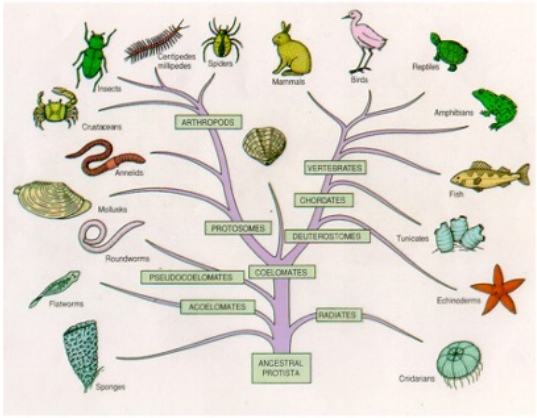
3.2.4 Taxonomy Induction

Information extraction

- Extract **isolated** facts from documents, e.g., *lion ISA animal*

Taxonomy induction

- Extract **related** facts from documents, e.g., classification of animals



Information extraction concerns the extraction of isolated facts, such as ISA relationships. Taxonomy induction aims at extracting related facts and organizing them in a structured knowledge graph, e.g., a hierarchical taxonomy. It is a special case of the more general ontology induction, which organizes knowledge using arbitrary relationships.

Use of Taxonomies

Hyponyms (subordinate terms) can inherit properties from hypernyms (more general terms)

- Due to transitivity of ISA, no need to learn inferred facts

No unique taxonomies

- Depending on the perspective and application different taxonomies may be useful:

A tiger and a puppy are both Mammals and hence belong close together in a typical taxonomy, but tiger is a WildAnimal (in the perspective of Animal-Function) and a JungleDweller (in the perspective of Habitat), while a puppy is a Pet (as function) and a HouseAnimal (as habitat), which would place them relatively far from one another

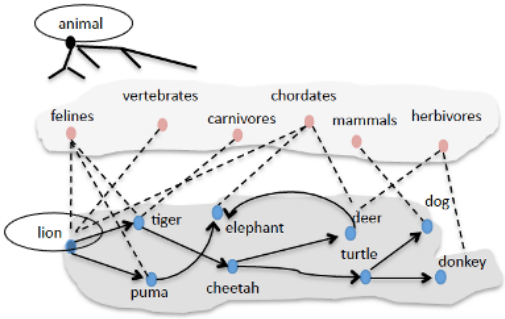
One of the advantages of taxonomy induction (and more generally ontology induction) is the possibility to perform inferences on the extracted knowledge. For example, in an ISA hierarchy, lower-level nodes in the hierarchy can inherit properties from higher-level nodes, thus these extra facts need not to be learnt separately.

One of the challenges in taxonomy induction is the fact that there is no notion of “correct” taxonomy. A taxonomy strongly depends in its intended use and on the specific perspective of the user on the domain. Integrating all possible perspectives into one single global taxonomy would not be feasible, as likely no agreement on a common view can be reached, and not useful, as the resulting taxonomy would potentially be too complex.

Taxonomy Induction Task

Starting from a root concept and a basic concept

1. Learn relevant terms and their hypernym / hyponym relationships
2. Filter out erroneous terms and relations
3. Induce a taxonomy structure



We introduce in the following one specific approach to taxonomy induction. It was one of the first approaches proposed. It starts from the assumption that one general concept (e.g., animals) and at least one basic concept of the taxonomy (e.g., lion) are provided as input. From this starting point the task is to identify more relevant concepts, represented as terms, and establish the hypernym and hyponym relationships, . The approach also filters out erroneous terms and relations and finally induces an overall taxonomy structure. The figure illustrates the process: at the bottom level first additional related terms are identified. Then intermediate concepts, more abstract than the basic concepts are found. From this data finally the taxonomy is induced.

Learning Terms

Template approach: double-anchored patterns

- Given a **root concept c** (e.g., animal) and a **seed s** (e.g., lion)
- Hyponym pattern, detecting instances:
  $P_i(c, s, X)$ = **c** such as **s** and X
- Hypernym pattern, using known instances $\mathbf{t_1}$ and $\mathbf{t_2}$, detect classes:
  $P_c(t_1, t_2, X)$ = X such as $\mathbf{t_1}$ and $\mathbf{t_2}$

The basic idea is to learn terms and relationships by querying a Web search engine. This is a very simple, but powerful tool to learn about the meaning of words. For querying, language patterns that capture hypernym and hyponym relationships are used. This is analogous to the use of Hearst patterns for extracting ISA relationships from text.

In a first phase the objective is to find more relevant terms. For this so-called double anchored patterns are used. that relate one known term to another unknown term of the same class of concepts. To start this search at least one term and one class need to be known. One example of such a pattern would be “c such as s and X”. Using such patterns new terms can be collected iteratively by applying the pattern to the terms known so far.

In a second phase, once many terms are known, new names of classes can be searched. For this purpose, hypernym patterns, like “X such as $\mathbf{t_1}$ and $\mathbf{t_2}$” can be used.

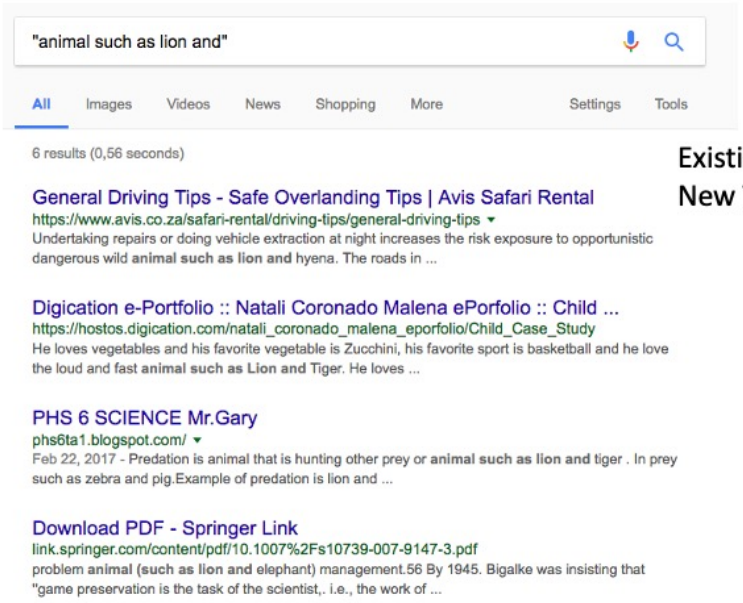
Finding Hyponyms

Iteratively harvest new terms using a Web search engine

```
T = {s}; w(t) = 0
while T changes
  for all t in T: submit P_i(c, t, X) to search engine
       if result is not empty
            add to T all new terms t_new found
                 in position X in a result
            w(t)++
```


This is the algorithm for finding new terms. While performing the search the algorithm keeps track with w(t) of the number of times a term t has been leading to the discovery of another term.

Example



Existing Term: lion
New Terms: hyena, tiger, elephant

This is an example of how terms related to an initial term “lion” would be found using double-anchored patterns. The example illustrates that the method works surprisingly well.

Finding Hypernyms

```
C = {c}; H = {}; w(t_i,t_j,h) = 0
for all t_1, t_2 in T with w(t_i) > 0:
     submit P_c(t_1, t_2, X) to search engine
     add new term h found in position X to C
     add t_1 ISA h and t_2 ISA h to the hypernym relations H
     w(t_1,t_2,h)++
```

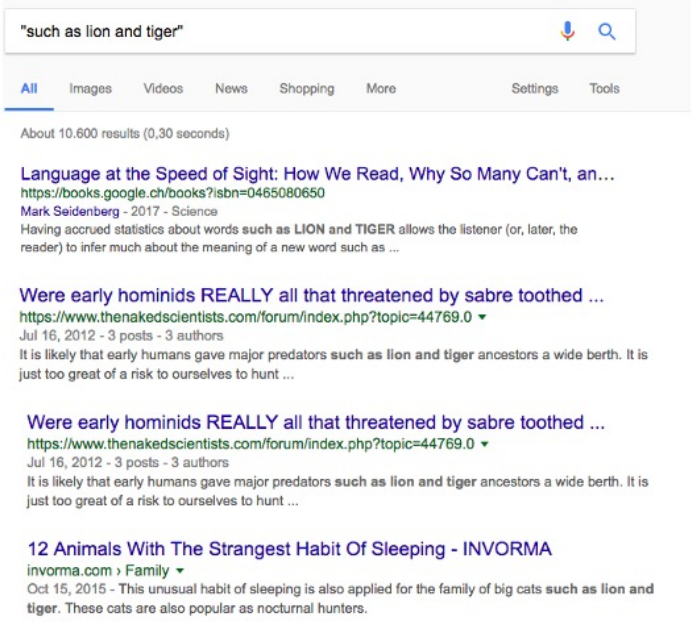
Filtering

- rank concepts h by $\sum_{t_1 t_2} w(t_1, t_2, h)$
- keep top concepts

Once the search for terms is completed, hypernyms can be searched by using the hypernym pattern with term pairs that have been found in the first step. Only terms that have been producing additional terms in the first phase are considered, using the condition w(t)>0, to avoid the addition of noisy concept names. The algorithm keeps track of how often a specific pair has produced a concept name.

Once the search for concept names is completed, a filtering step is performed. The concepts found are ranked by the number of times they have been discovered starting from different term pairs and only the highest ranked concepts are retained.

Example

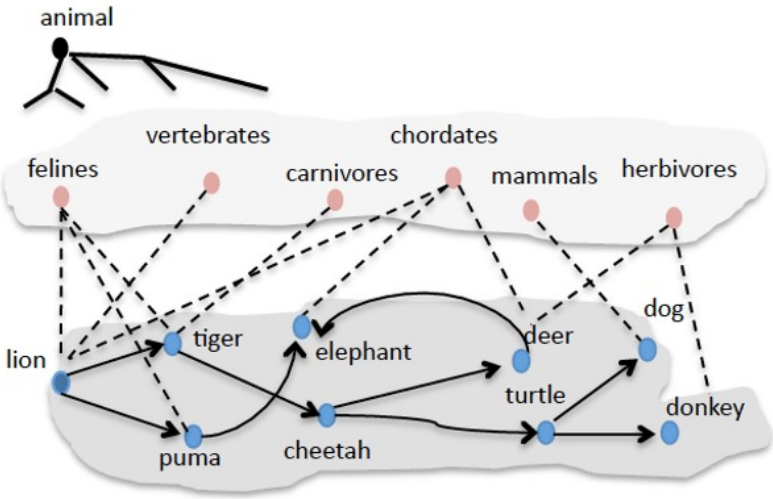


New Classes: predators, big cats,

But also: words, …

Here a few examples of how higher-level concepts related to the basic concepts “lion” and “tiger” can be found. Note that this step can easily produce noisy terms, therefore filtering is important.

Example Result



As a result of the two previous steps, we obtain basic data consisting of basic concepts T and higher-level concepts C, together with hypernym relationships indicated by dotted arrows.

Inducing Hypernym Graph

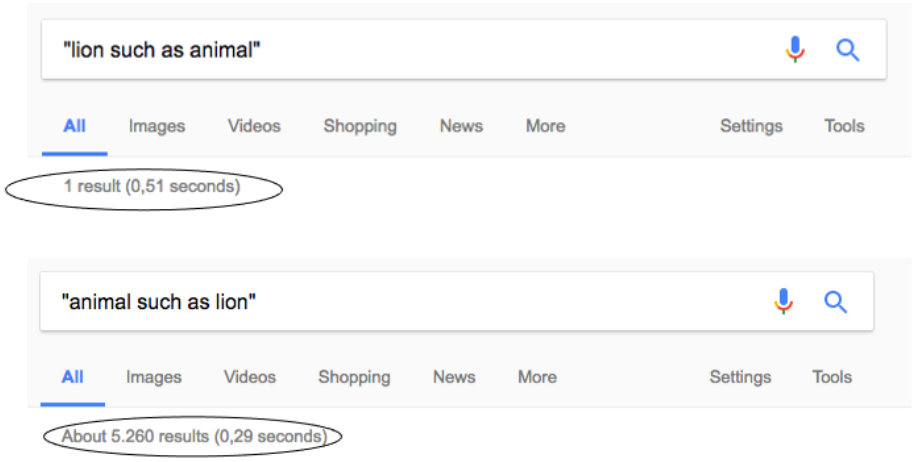
Many possible relationships among concepts have likely not been discovered

```
For each pair t1, t2 in C
     Construct query q1 = h(t1, t2) and q2 = h(t2, t1)
     with Hearst pattern h(X, Y) for ISA,
     e.g., h(X, Y) = "X such as Y"
     Submit query to search engine, count number of result
     If #results(q1) < #results(q2)
          then add t1 ISA t2 to H
          else add t2 ISA t1 to H
```

Result: A directed hypernym graph H

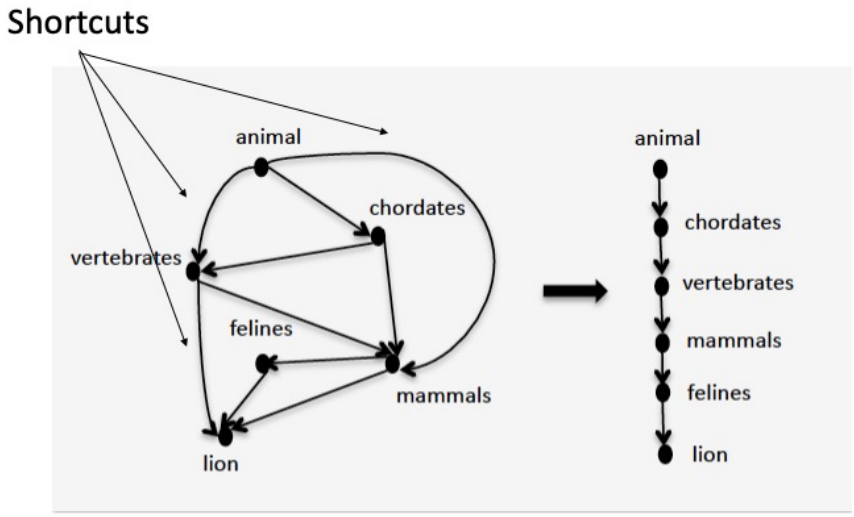
In the steps performed so far, many possible relationships among higher-level concepts and basic concepts, and among different higher-level concepts may not have been discovered. For finding those again queries are posed to a search engine. For each pair of terms designating a concept, using a query the algorithm tests whether they are in a hypernym relationship. For this purpose, standard Hearst patterns for the ISA relationship are used, such as “X such as Y”, “X are Y that”, “X including Y”, “X like Y”, “such X as Y”.

Example



This example shows how powerful this method is to test a direction of a relationship.

Example



Graph H: may contain redundant paths

After adding all possible relationships to the set of concepts, we obtain a graph that contains many redundant paths. Many of the paths correspond to the transitive closure of other relationships. Hor example, the algorithm would have found that animal ISA chordate, and chordate is a vertebrate, but also that animal is a vertebrate. In a last step such redundant relationships are removed.

Cleaning the Hypernym Graph

1. Determine all **basic concepts**
   - Not hypernym of another concept
2. Determine all **root concepts**
   - Have no hypernyms
3. For each basic concepts - root concept pair:
   - Select all hypernym paths that connect them
4. Choose the longest hypernym paths for the final taxonomy

For removing the redundant relationships, both root and basic concepts are identified and all paths that connect pairs of them. By choosing for each of those pairs the longest paths, and dismissing the others, all relationships that are part of the transitive closure are removed. Note that multiple longest paths may exist, since the taxonomy can have a lattice structure.

If t has no Hypernym ..

A. It is a root concept
B. It cannot match c such as t and X
C. It is identical to the initial root concept
D. It is a basic concept

References

Lecture partially based on

- Dan Jurafsky and James H. Martin, Speech and Language Processing (3rd ed. Draft), Chapter 21 https://web.stanford.edu/~jurafsky/slp3/
- Jay Pujara and Sameer Singh, Mining Knowledge Graphs from Text, Tutorial, https://kgtutorial.github.io

References

- Mintz, Mike, et al. "Distant supervision for relation extraction without labeled data." *ACL 2009*.
- Riedel, S., Yao, L., McCallum, A., & Marlin, B. M. Relation extraction with matrix factorization and universal schemas. *ACL 2013*.