Entity Linking

Genevieve Gorrell and Johann Petrak

Session Overview

- A short introduction to entity linking

- YODIE – Entity Linking with GATE

- BioYODIE – Linking of Biomedical entities with GATE

Introduction to Entity Linking

What is Entity Linking

- Entity linking is the task of identifying all mentions in text of a specific entity from a database or ontology

- Also referred to as entity disambiguation: there may be several different with the same name in the database

- Link mentions to the concept in the KB that best matches the meaning in the given context

- Do this efficiently for a KB with millions of concepts and with dozens or hundreds of concept candidates per mention

en.wikipedia.org/wiki/Shinji_Kagawa

en.wikipedia.org/wiki/Borussia_Dortmund

KAGAWA will be allowed to rejoin Borussia Dortmund in January in a swap deal which would see defender @NSubotic4 join #MUFC http://tiny.cc/4t19ux

en.wikipedia.org/wiki/Manchester_United_F.C

en.wikipedia.org/wiki/Neven_Subotić

Why are we doing it?

- Rather than just annotate the words "Berlusconi" and "Берлускони" as a Person (NER), link it to a specific ontology instance (entity)

  – Differentiate between Silvio Berlusconi, Marina Berlusconi, etc.

  – Ontologies tell us that this particular Berlusconi is a Politician, which is a type of Person. He is based in Italy, which is part of the EU. He was a prime minister, etc. This is all helpful to disambiguate and link the mention in the text to the correct entity URI in the ontology

- Having identified the unique entity enables further tasks like relation detection, semantic search a.o.

Why is it hard?

- Entity linking needs to handle:

  - Name variations (entities are referred to in many different ways, including colloquial variants)

  - Entity ambiguity (the same string can refer to more than one entity) Sometimes dozens or hundreds of possible alternatives!

  - Missing entities – there is no target entity in the entity knowledge base/database

Data Sources for EL

- Entity Linking is based on a datasource/knowledge base to which to link (or several)

- Researchers have used Wikipedia (e.g. TAC KBP, WikipediaMiner), Linked Open Data (in particular DBpedia, YAGO, and Freebase) and the UMLS metathesaurus for medical concepts (Metamap, Bio-YODIE)

- Some datasources/knowledge bases have names in several languages (Wikipedia, UMLS), but coverage is often very different between languages.

- Ideally adapt one system to different languages, different domains

Data Sources for EL

- The entity linking system can either return a matching entry from the target knowledge base or "NIL" to indicate there is no matching entry in the entity database

- Some entity linking systems make the closed world assumption (CWA) that there is always a target entity in the database

- Often still focused on entities of type PER, LOC, ORG and often focused on English documents

LOD Sources - DBpedia

- LOD = Linked Open Data: 1000s of sources available

- Dbpedia is probably the most prominent: derived from Wikipedia

- Machine readable knowledge on 4.85 million entities and topics (as of Jun 2015), including:

  - 735,000 places/locations,

  - 1,445,000 persons

  - 241,000 organisations

- For each entity there are:

  - Entity name variants (e.g. IBM, Int. Business Machines)

  - a textual abstract

  - reference(s) to corresponding Wikipedia page(s)

  - entity-specific properties (e.g. death-date of a person)

DBpedia Example

LOD Sources – UMLS Metathesaurus

- Unified Medical Language System (from the US National Library of Medicine)

- Contains over two million biomedical and health-related concepts

- Combines many thesauri ("source vocabularies")

- Much ambiguity! E.g. "unknown" is associated with 39 concepts

- The NLP "view" (subset) is more pragmatic, excluding unhelpfully ambiguous and spurious concepts

LOD Sources UMLS Metathesaurus

- "Cold" has six associated concepts

- "CUI" means concept unique identifier and is the unique code for the concept

LOD Sources – UMLS Metathesaurus

- A concept has a type (in this case "Disease or Syndrome")

- It has definitions from different vocabularies

- It also has concept relations, i.e. things it is related to. "Common Cold" is related to "Respiration Disorders" for example

| Basic View | **Report View** | Raw View |

⊕ **Concept: [C0009443] Common Cold**

⊟ **Semantic Types**

  Disease or Syndrome [T047]

⊟ **Definitions**

CSP/PT | catarrhal disorder of the upper respiratory tract, which may be viral or a mixed infection; marked by temperature, chilly sensations, and general indisposition.

MEDLINEPLUS/PT |

Sneezing, sore throat, a stuffy nose, coughing - everyone knows the symptoms of the common cold. It is prob the course of a year, people in the United States suffer 1 billion colds.

You can get a cold by touching your eyes or nose after you touch surfaces with cold germs on them. You can usually begin 2 or 3 days after infection and last 2 to 14 days. Washing your hands and staying away from pe colds.

There is no cure for the common cold. For relief, try

- Getting plenty of rest
- Drinking fluids
- Gargling with warm salt water
- Using cough drops or throat sprays
- Taking over-the-counter pain or cold medicines

However, do not give aspirin to children. And do not give cough medicine to children under four.

NIH: National Institute of Allergy and Infectious Diseases

MSH/MH | A catarrhal disorder of the upper respiratory tract, which may be viral or a mixed infection. It genera

YODIE

Yet another
Open **D**ata
Information **E**xtraction
System

YODIE Overview

- Link entities against DBpedia using DBpedia URIs

- Also adapt to other domains → BioYODIE

- Planned to make (parts) available as Open Source Software

- A typical approach

  - Entity Linking is a crowded marketplace! Lots of exciting new approaches

  - YODIE aims to provide robust choices as (eventually!) freely available, modifiable GATE components

  - Solid performance on varied input types, including tweets

  - Also a motivation and testbed for development of GATE components and plugins

YODIE Overview

- **WHAT** does YODIE do?

  - Identify potential entity mentions by matching known names of entities against text of documents/tweets/etc.

  - For each mention, get all entity candidates and information related to the entity, mention and entity/mention combination

  - Score entity candidates based on contextual fit, mention type, congruence with other potential entities etc.

  - Select candidate using a machine learning approach that learns to combine the scores to find the best entity

  - Try to identify mentions that do not refer to anything in Dbpedia (NILs)

Identify Mentions: How?

- Find KB concept labels: Gazetteer of known possible labels for all concepts

  - DBpedia instance labels

  - DBpedia name/nickname properties

    (from WP templates)

  - Labels from redirected WP pages (spelling variations)

  - Labels from WP Disambiguation pages

  - Anchor text from intra-WP links to that concept

- Additional mentions from supplemental gazeetteers (e.g. country demonyms)

- Optional additional mentions candidates from NER

Identify Mentions: How?

- Match ALL labels ignoring case with document text

- Millions of labels, even after normalization/filtering

- Often many candidate concepts/URIs (100s) per label

- Will need a lot of information for each candidate concept:
  - URI
  - Original case
  - concept type (Org, Pers, …)
  - Frequency statistics
  - ...
  => Cannot directly use a Gazetteer for all of this
  => Separate 1) finding of mention locations and
      2) getting all possible entity candidates for them

Identify Mentions: How?

- Use gazetteer to just identify mentions

- Prepare a database that maps each mention text to all the information we need.

- Prepare as much in advance as possible so we do not need to spend time on it in the pipeline

- Preparation only needed infrequently, may require a lot of computing resources

- Desired output:
  1) gazetteer of all labels/names
  2) all the information for each label/name:
     - information about the label itself (frequencies…)
     - information for all possible entities for the label

Candidate Preparation

- Sources (numbers just for EN):
  - DBpedia labels: ~10M triples
  - DBPedia properties: ~26M triples
  - WP page links: ~172M links
  - DBpedia: 1.6M triples

- Make sure URIs are normalized:
  - proper %-encoding style (varies between DBP version)
  - use IRIs not URIs everywhere

- Make sure labels are normalized

Candidate Preparation

Normalize and filter labels:

- Exclude obvious cases (hundreds of chars, "List of ...", numbers only, …)

- Normalize for case-insensitive matching
  => but remember original case!

- Canonical representation of umlauts, accents etc.

- Generate common variants, e.g. "ä" → "ae"

- Multiple white-space, punctuation

- Extract parentheses info, e.g
  "Jean Lemaire (painter)"

Label/Candidate Preparation

- Gather information per URI: class, frequency of related WP page link in WP articles, DBpedia properties …

- Gather information per label: original spelling, source (redirected page, disambiguation, canonical page, yago), frequencies, ….

- Gather information per label/URI pair: relative frequency of label used with this URI (WP page link), relative frequency of URI used with this label ("commonness")

- Combine information and generate a de-normalized representation: key/value where the key is the label and the value is an array of rich URI-information, one element for each URI.

Candidate Preparation: Example

"yorkshire" →

- original_label="Yorkshire", all_labels=["Yorkshire","The Yorkshire Mafia"], uri="dbp:The_Yorkshire_Mafia", uriByLabel=0.0, sources=["dbp_name"],....

- original_label="Yorkshire", all_labels=["Yorkshire","Yorkshire, Ohio", "Yorkshire, OH"], uri="dbp:Yorkshire,_Ohio", sources=["dbp_labels"], uriByLabel=3.310E-3, airpClass=dbpo:PopulatedPlace, …

- original_label="Yorkshire",uri="dbp:Yorkshire", all_labels=["Yorkshire","Yorks","County of Yorkshire",...], uriByLabel=0.68,airpClass=dbpo:PopulatedPlace,parentheses =["UK","England"],...

Identify Mentions

- Use ExtendedGazetteer PR to just match the labels (~9M cleaned labels, 158M on disk, ~900M memory) Creates Mention Candidate/Spot annotation

- For each matched label, look up the candidates data from a database (~21G on disk)

- Each candidate from the list is represented as a separate annotation, the fields as features in the FeatureMap

- Mention annotations link to their candidate annotations (via a feature that contains a list of all their annotation ids) (AnnotationGraph plugin)

- Subsequently, most processing happens on entire candidate lists.

Reducing Candidates – Overall Strategy

- Matching all known labels heavily over-annotates (e.g. several entities for "the") and generates lots of overlapping annotations

- Try to remove obvious spurious mentions early on: isolated stop words, inside URLs etc.

- Try to deal with overlaps early on: generally use longest match but with exceptions (determiner, possessives, locations)

- Calculate scores for remaining candidates

- Pick best candidate or decide it must be OOKB

Choosing a Candidate

- Have a reduced set of Mentions, each with its list of candidates

- Each candidate has initial features from the prepared data:

    - Label frequency n(label in WP)

    - Commonness p(url|label)

    - Link probability p(wikiLink|label)

    - List of original labels

    - Dbpedia class (type) of candidate

    - Static Page Rank

    - ...

- Additional features based on how well the candidate "fits" into the actual context will get added ...

Choosing a Candidate—What next?

- Measure how close each candidate is semantically to the text surrounding the mention: *text-based contextual similarity*: Vector Space Models, Word/Doc. Embeddings

- Measure how well each candidate fits with other candidates in the context, what is the best fit?
  → LOD-based structural similarity, relatedness, personalized page rank, ...: *coherence*

- Measure ambiguity, calculate relative scores

- Measure which kind of label matches mention and how (preferred, alternate, nickname, acronym, same case ...)

- Choose candidate based on all features/scores:
  → Machine Learning

Choosing a Candidate

- This can be presented as a machine learning classification problem

    - Each candidate becomes an instance

    - Scores etc. become features for the machine learning

    - Class is true if candidate should be selected for the mention, false otherwise ($\rightarrow$ NILs)

- => ML task is to decide if a candidate is the right choice ("true"/1) or not ("false"/0) based on features

- Need to find a good training corpus

- Need to find a good trade-off: number of mentions, size of candidate lists vs. difficulty of the learning task (very unbalanced!)

Choosing a Candidate

- ML algorithms that have been found to work:
  - Support vector classification with RBF kernel (also need probability estimates!)
  - Random Forest

- Training data: TAC data, AIDA training corpus, training portion of the Sheffield NERD Tweet corpus
  (holding back other corpora for evaluation)

- Assigns true or false to each candidate, with a probability

  - This may mean that we have none, one or multiple "true"

  - Multiple "true": use the probability assigned by the machine learner to select the best

  - None: remove / treat as NIL.
    Actual NIL (Entity not in KB) or just spurious mention

Some Performance Figures on "AIDA B" Corpus

	Prec	Recall	F1
YODIE 2015	0.62	0.65	0.64
AIDA/2013	**0.74**	0.34	0.47
AIDA/2014	0.70	**0.74**	**0.72**
Lupedia	0.50	0.24	0.32
Spotlight	0.31	0.40	0.35
TagMe	0.61	0.56	0.58
TextRazor	0.35	0.58	0.34
AGDISTIS	0.64	0.56	0.60
Zemanta	0.51	0.29	0.37

Microblog Text

- YODIE aims to be robust to varied types of input—recent research includes adding extra processing and context for tweets

- Resolve hash-tags:

  - split multiword hash-tags

  - add hash-tag expansion candidates as text

- Resolve user screen names:

  - retrieve user profile information from Twitter and add as annotated text (location, description, language …)

- Resolve linked web pages:

  - retrieve content of web page and add as text

  - filter web page content to remove boilerplate/navigation

- Additional context provides more text and entities for comparison

Some Performance Figures on Tweets

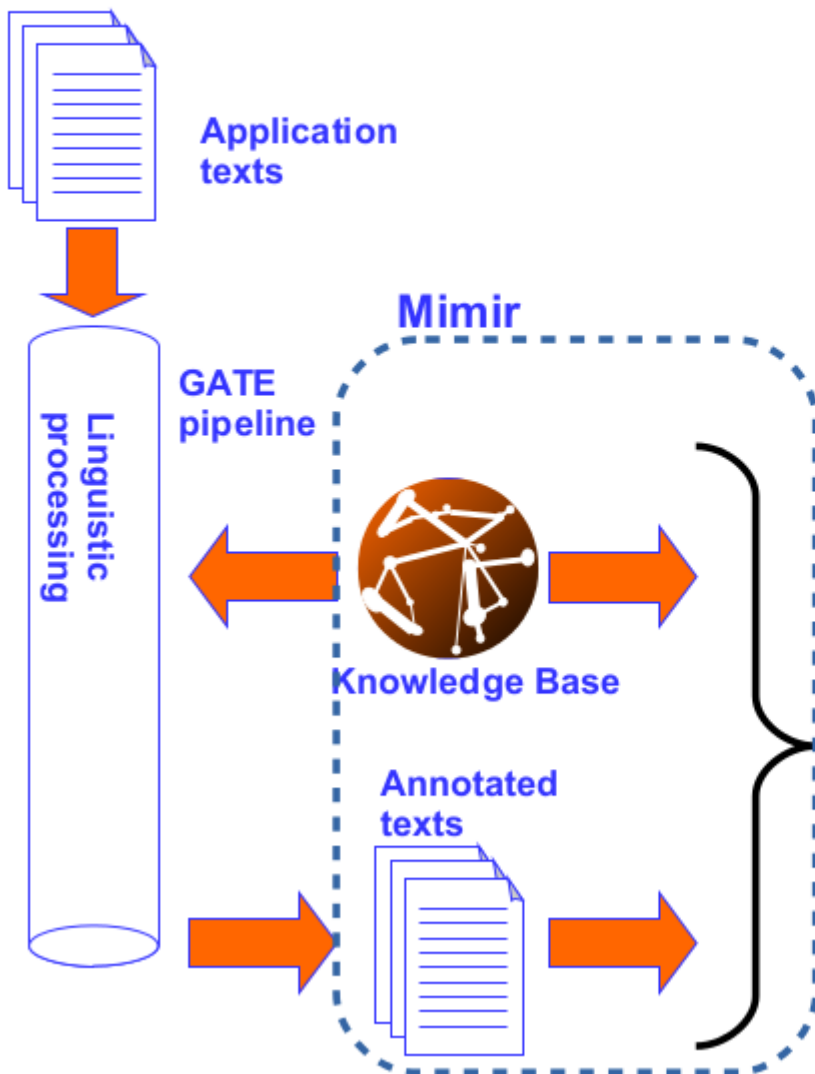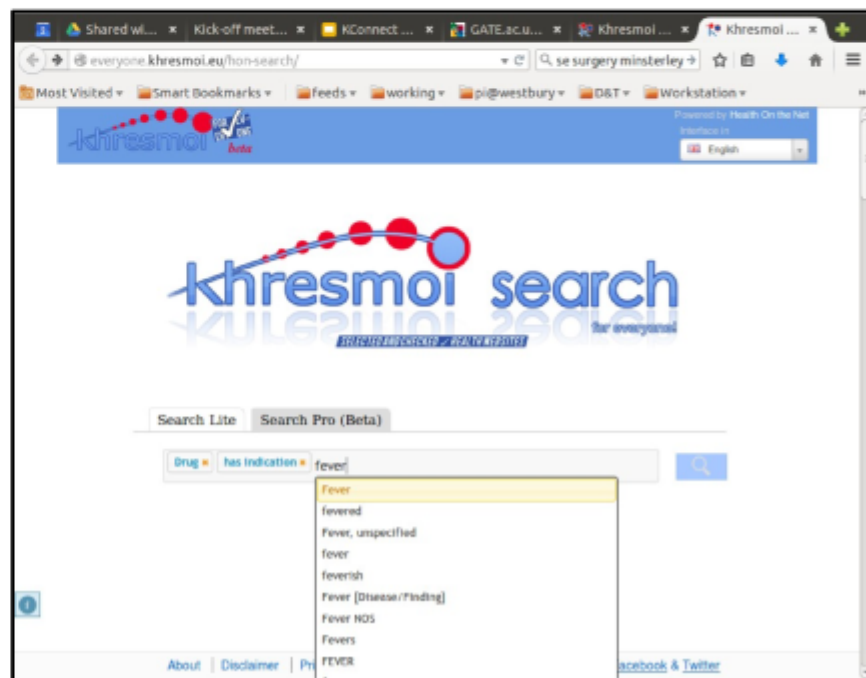|	Prec	Recall	F1
YODIE (Base)	0.44	0.55	0.49
YODIE (Exp)	0.50	0.62	**0.55**
Aida 2014	**0.59**	0.38	0.46
Lupedia	0.50	0.24	0.32
Spotlight	0.09	0.51	0.15
TagMe	0.10	**0.67**	0.17
TextRazor	0.19	0.44	0.26
Zemanta	0.48	0.56	0.52

BioYODIE

Why semantic annotation?
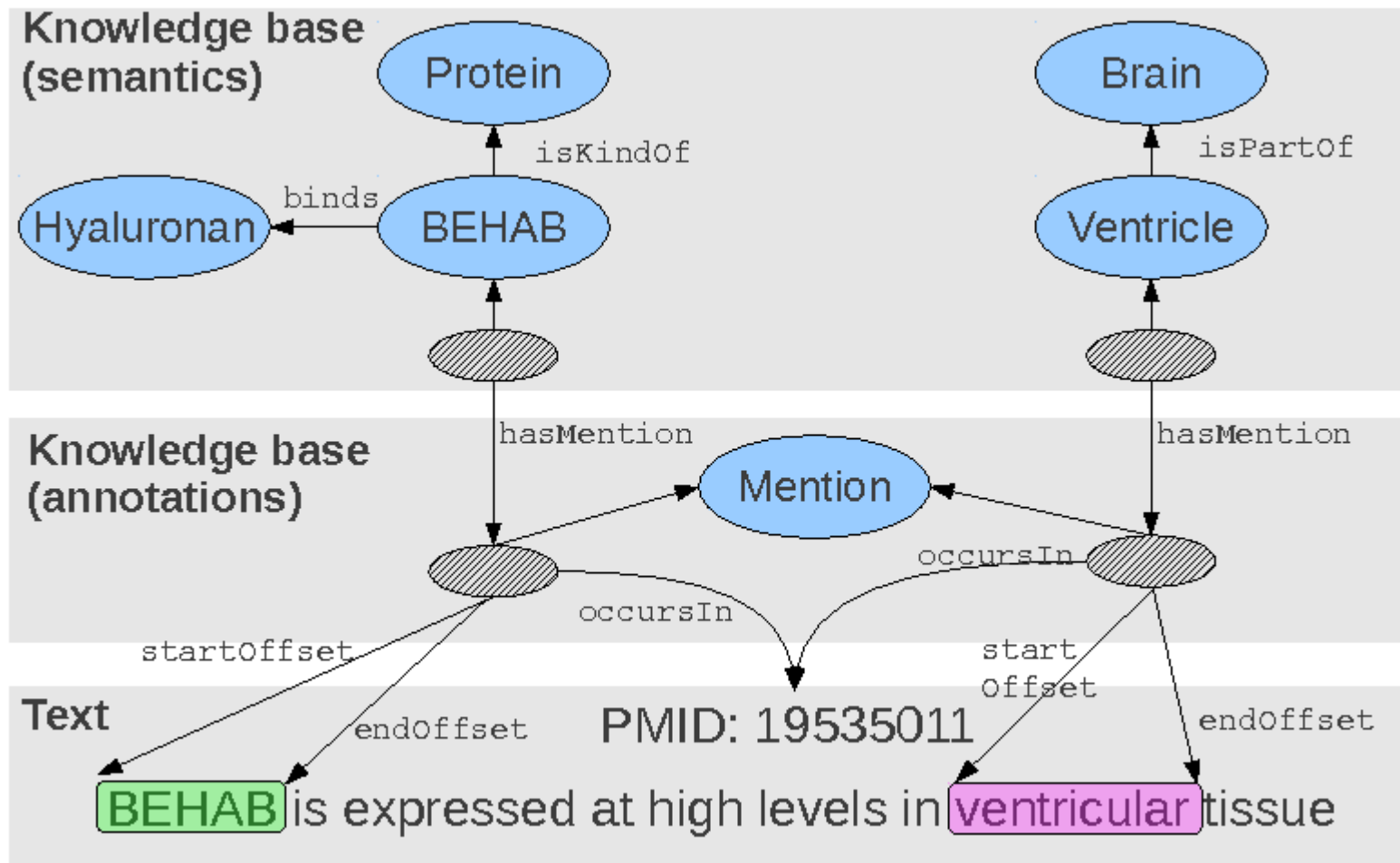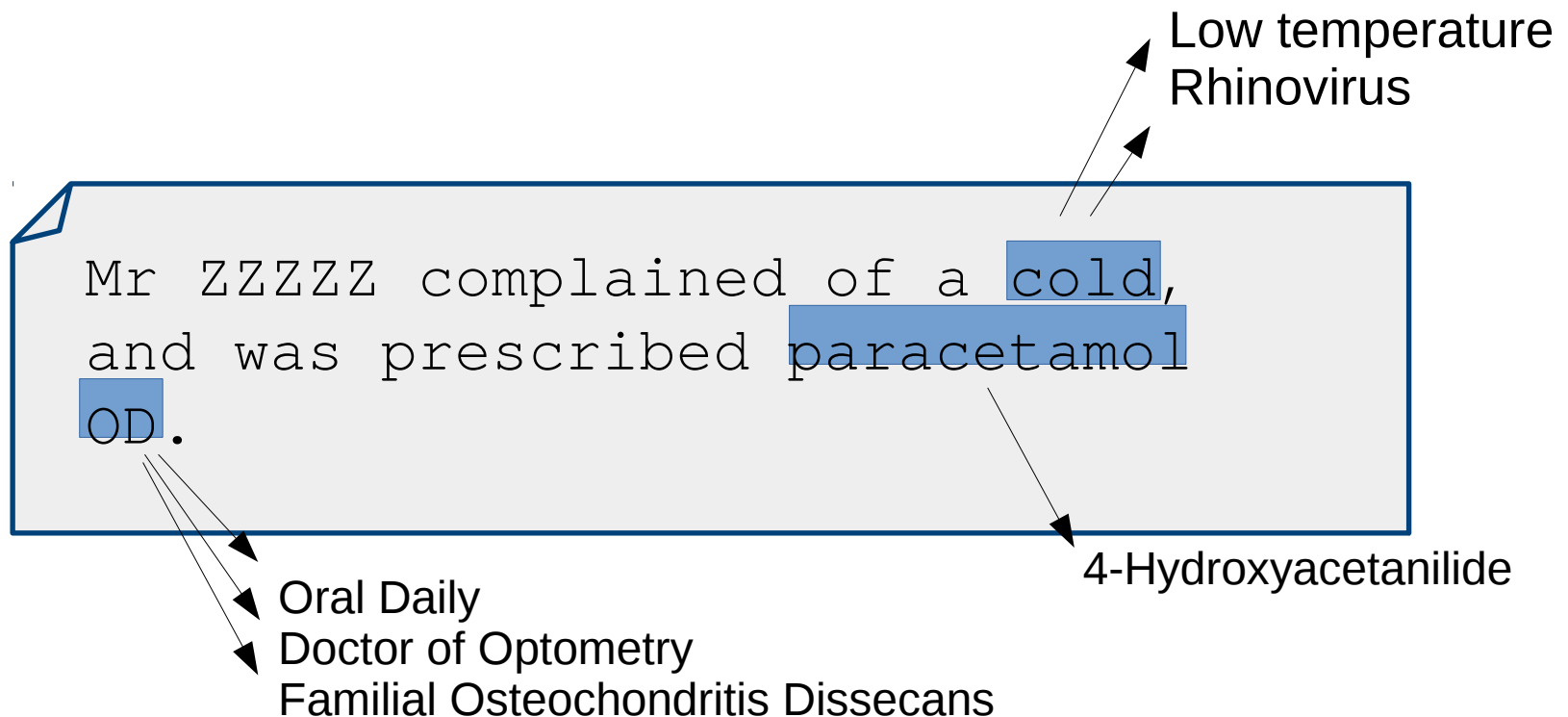
Linking text and knowledge

Find patient records that mention drugs used to treat disorders commonly associated with depression

Named Entity Linking

- You have found mentions in text, but there are several possibilities for what something is referring to

- This is the disambiguation problem

Low temperature
Rhinovirus

Mr ZZZZZ complained of a cold, and was prescribed paracetamol OD.

4-Hydroxyacetanilide

Oral Daily
Doctor of Optometry
Familial Osteochondritis Dissecans

Bio-YODIE

- Main approaches similar / identical to YODIE
- Dictionary based term identification based on names of UMLS entities of relevant types
- Retrieval of possible interpretations—any entity that can be called that
- Multiple disambiguation strategies gauging how well the candidates fit
- Scores combined to choose the best

KB: UMLS Metathesaurus

- Unified Medical Language System (from the US National Library of Medicine)

- Contains over two million biomedical and health-related concepts

- Combines many thesauri ("source vocabularies")

- Much ambiguity! E.g. "unknown" is associated with 39 concepts

- The NLP "view" (subset) is more pragmatic, but excludes important acronyms

Bio-YODIE: Finding Mentions in Text

Bio-YODIE disambiguation scoring strategies

- PageRank compiled across the UMLS co-occurrence table (compiled from MeSH co-occurrences in PubMed articles) gives a prior probability

- Prior probabilities of entities compiled from training data gives a valuable score

- Prior probabilities of an entity for a particular term is also valuable

- Contextual scores are slow and haven't been found effective in our work so far

- Scores are simply combined, as there is currently insufficient data for a machine learning approach to score combination

Comparison with other Biomedical NEL systems

	Time	Prec.	Recall	F1	Accuracy	Kappa
MetaMap	2994	0.57	0.56	0.57	0.86	0.86
MetaMapLite	**149**	**0.65**	0.55	0.59	0.88	0.88
Bio-YODIE	263	0.62	**0.61**	**0.62**	**0.89**	**0.89**

Mimir index statistics

Mentions by Type

- 🔵 Anatomy
- 🔴 Disease
- 🟠 Investigation
- 🟢 Drug

12%
13.2%
16.6%
58.1%
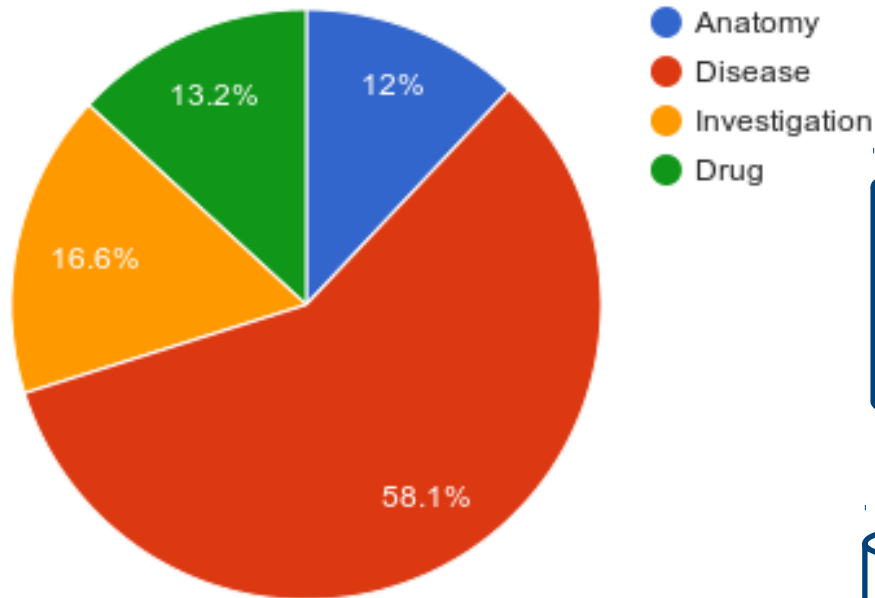
Proportion of types found on 50K test documents (patient records)

Using GCP, we (USFD and KCL) have indexed ...

15 million documents

4 million Trip documents

GATE Components for YODIE

- StringAnnotation plugin: ExtendedGazetteer
https://github.com/johann-petrak/gateplugin-StringAnnotation

- JdbcLookup plugin
https://github.com/johann-petrak/gateplugin-JdbcLookup

- ModularPipelines plugin
https://github.com/johann-petrak/gateplugin-ModularPipelines
- modularization
- configuration

- LearningFramework plugin
https://github.com/GateNLP/gateplugin-LearningFramework

- GATE ConText Wrapper (Harkema et.al. 2009)
https://github.com/GateNLP/gateplugin-ConText
https://github.com/GateNLP/gateplugin-ConText
(Negation, Experiencer, Temporal Status)

GATE Components for YODIE

- Java "Scripting" plugin
  https://github.com/johann-petrak/gateplugin-Java
  Fast withing-GUI Java prototyping / programming

- AnnotationGraph plugin
  https://github.com/johann-petrak/gateplugin-AnnotationGraphs
  Represent relations, sequences, candidate lists as
  Annotations

- Evaluation plugin
  https://github.com/johann-petrak/gateplugin-Evaluation

- [several others ….]

- GATE GCP: process large corpora

- GATE Cloud: deploy service