Entity Linking

Genevieve Gorrell and Johann Petrak

Session Overview

- Introduction to entity linking
- YODIE—how we approach entity linking
- Demos
  - YODIE
  - Some taggers by other groups that are integrated into 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
- 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)
  - Missing entities – there is no target entity in the entity knowledge base/database

Candidate ambiguity is high = tough task

	TAC-KBP				
	PER	LOC	ORG	UKN	TOTAL
Entities	89	361	141	274	865
Avg. number of tokens	1.91	1.20	2.12	1.87	1.78
Candidate URIs	9,427	9,553	9,502	14,649	43,131
Avg. number cand. URIs	105.02	26.46	67.39	53.46	49.86
Unambig. candidates	3	10	3	43	59

Data Sources for EL

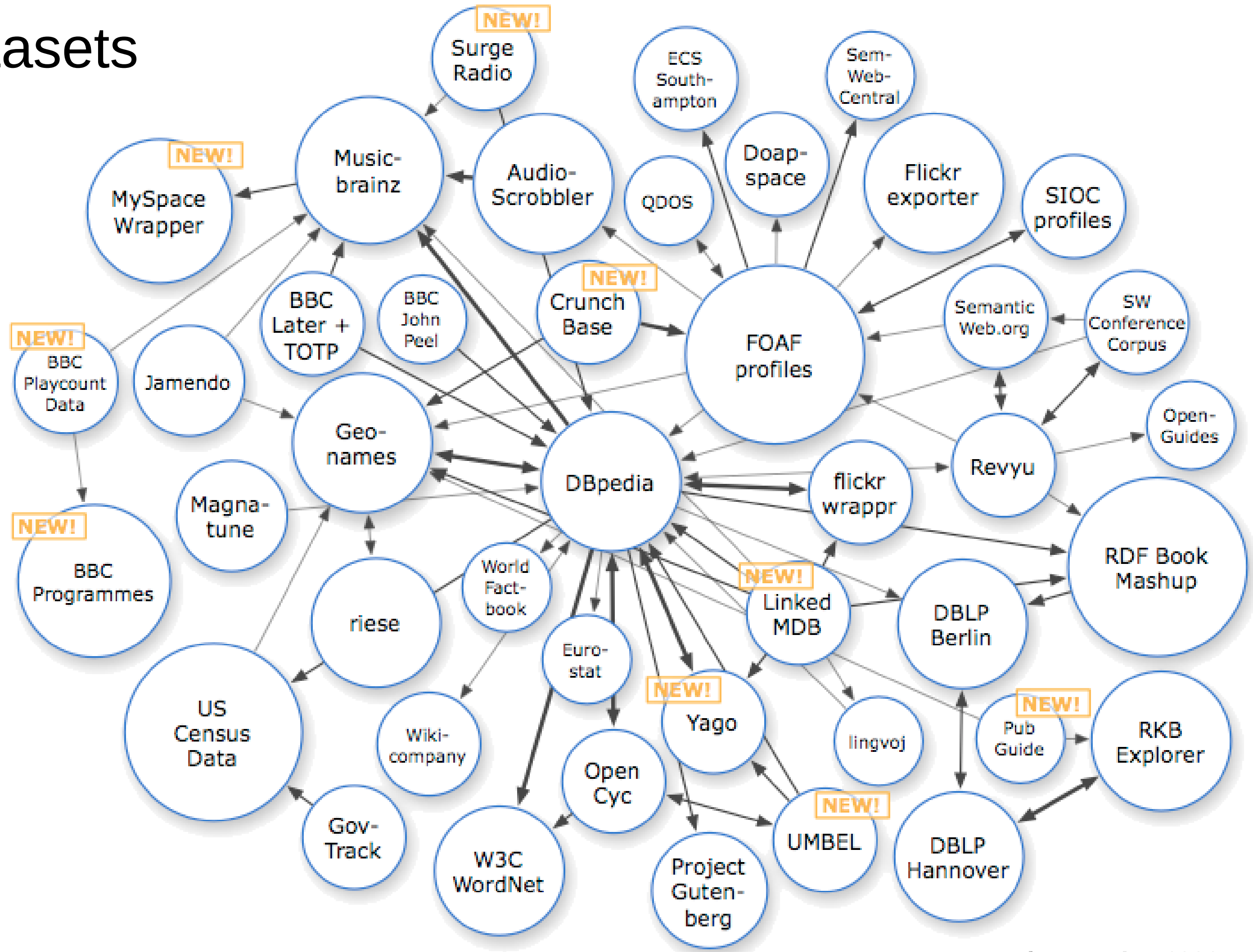
- Entity Linking is based on a datasource/knowledge base to which to link (or several)
- Researchers have used Wikipedia (e.g. TAC KBP, WikipediaMiner), Linked Open Data (in particular DBpedia, YAGO, and Freebase) and the UMLS metathesaurus for medical concepts

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

Linked Open Data: 2008

45 Datasets

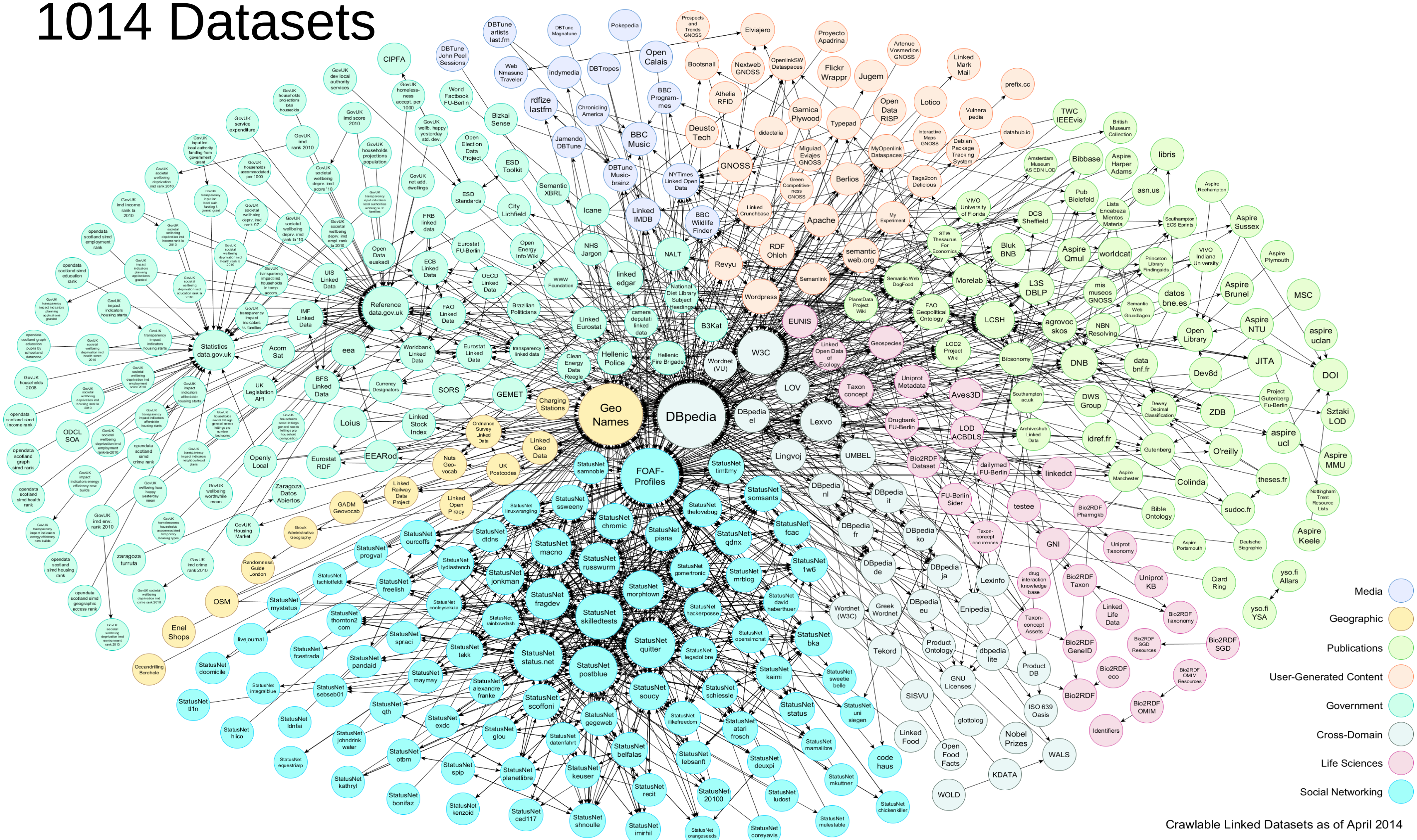


As of September 2008

GATE

Linked Open Data: 2014

1014 Datasets



Crawlable Linked Datasets as of April 2014

LOD Sources - DBpedia

- 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

D About: Thames Barrier

dbpedia.org/page/Thames_Barrier

dbpedia Thames B

About: Thames Barrier

An Entity of Type : _Feature, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

DBpedia

The Thames Barrier is the world's second-largest movable flood barrier and is located downstream of central London, United Kingdom. Its purpose is to prevent London from being flooded by exceptionally high tides and storm surges moving up from the sea. It needs to be raised (closed) only during high tide; at ebb tide it can be lowered to release the water that backs up behind it.

. . .

owl:sameAs	▪ http://cs.dbpedia.org/resource/Bariéry_na_Temži ▪ http://de.dbpedia.org/resource/Thames_Barrier ▪ http://fr.dbpedia.org/resource/Barrière_de_la_Tamise ▪ http://it.dbpedia.org/resource/Thames_Barrier ▪ http://sws.geonames.org/2636058/ ▪ freebase:Thames Barrier
geo:geometry	▪ POINT(0.0367 51.4977)
geo:lat	▪ 51.497700 (xsd:float)
geo:long	▪ 0.036700 (xsd:float)

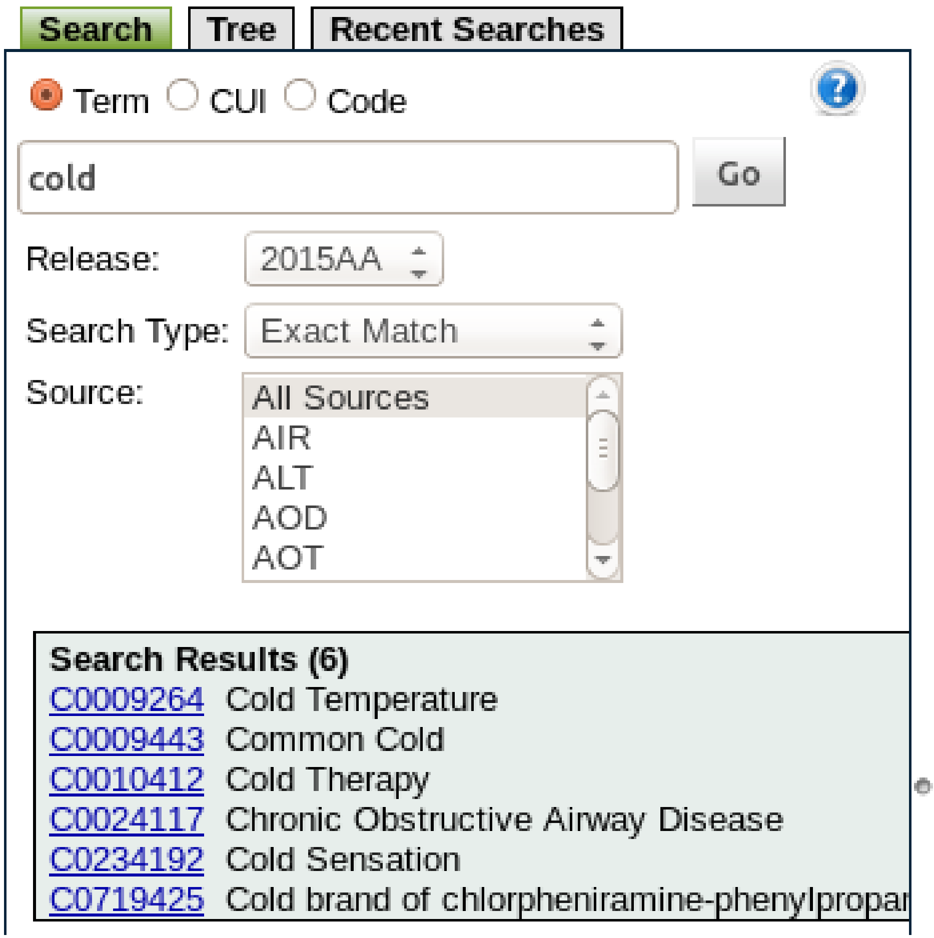
Links to GeoNames And Freebase

Latitude & Longitude

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

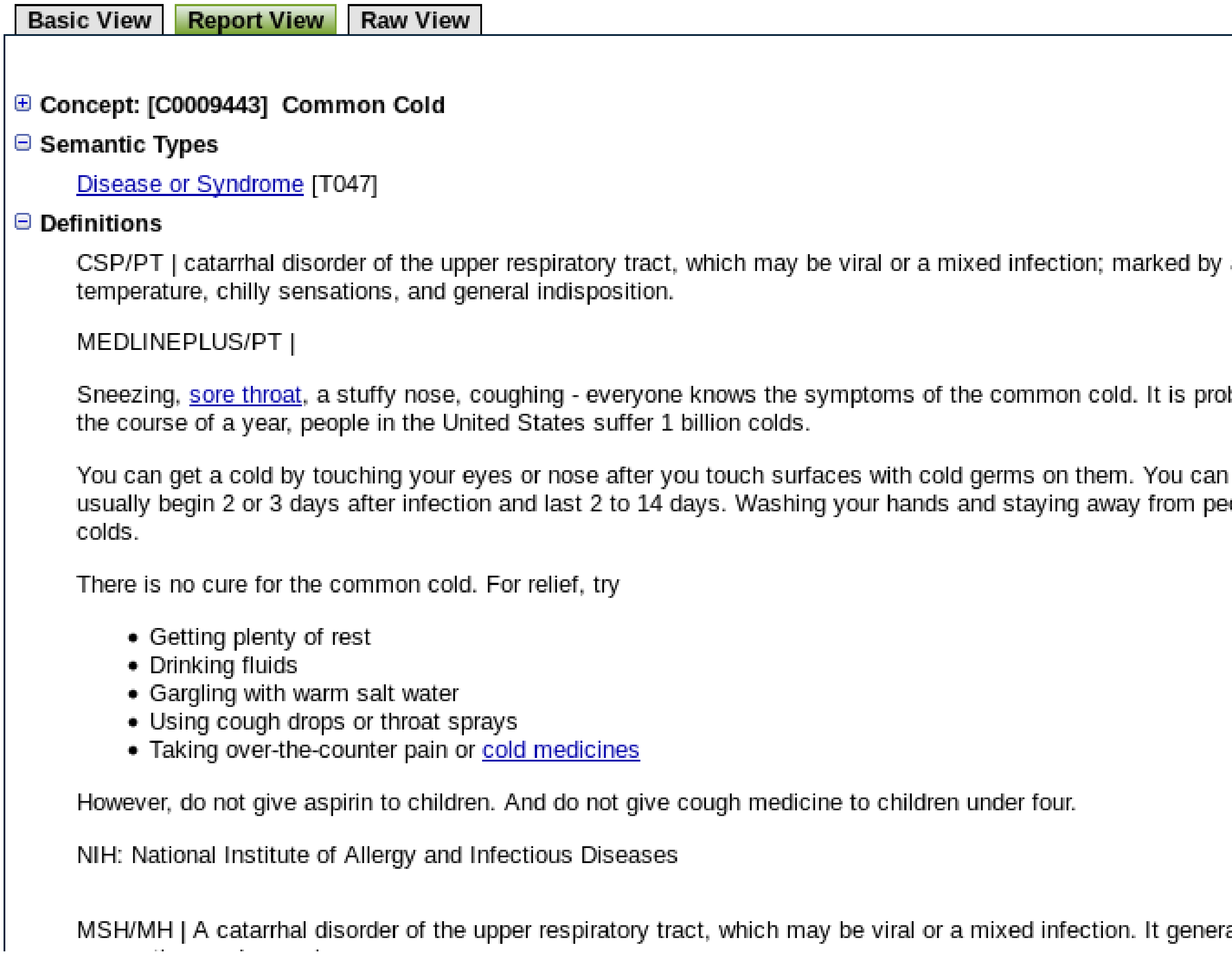
LOD Sources UMLS Metathesaurus



- “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 a
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
- 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
  - Cross-domain capability—porting to UMLS in progress!

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)
    - YAGO labels of mapped YAGO instances
    - Labels from redirected WP pages (spelling variations)
    - Labels from WP Disambiguation pages
    - Anchor text from intra-WP links to that concept
  - Additional mentions from NER which may be variations of known multi-word labels

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

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: all the information for each mention text (label)

Candidate Preparation

- Sources (numbers just for EN):
  - DBpedia labels: ~10M triples
  - YAGO labels: ~27M triples (including non-mappable)
  - DBPedia properties: ~26M triples
  - WP page links: ~172M links
  - DBpedia/Airpedia types: 1.6M/10M 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)”

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 de-normalized data from a key/value store (~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 rubbish early on
- Try to deal with obvious cases early on
- Try to deal with overlaps early on
- Calculate scores for remaining candidates
- Pick best candidate or decide it must be OOKB

=> when to filter or when to just calculate a score?
=> how to decide based on scores and other features?
→ Ongoing Research!

Reducing Mentions & Candidates

- Reduce based on POS tags: require at least one proper noun (plus some special cases like country adjectivals)
  - Problem: wrong POS tag will do unrecoverable harm,
    but POS tags will often be wrong in Tweets, Title-case text..
- Reduce based on known patterns, e.g.
  - [Mention1], [Mention2] with Mention1 containing location candidates, Mention2 containing location candidates and some pairs of candidates related, e.g. “Yorkshire, Ohio”
  - [Mention1] ([Mention2]) with Mention1 a multi word term and Mention2 something that looks like an acronym, e.g. “Inter-Services Intelligence (ISI)”
- Reduce based on known overlap patterns, e.g. a mention of a person name within a longer mention of a person name
- Filter candidates based on target types (e.g. PER, ORG, LOC, Product)
- ….

Choosing a Candidate

- Now we have a good quality candidate list
- 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
  - ...
- Additional features based on how well the candidate “fits” into the actual context will get added ...

Choosing a Candidate—What next?

- Add contextual scores—rate candidate for likelihood of appearing in context
  - Measure how close each candidate is semantically to
    - the text surrounding the mention: *text-based contextual similarity*
    - other entities mentioned in the context: LOD-based structural similarity, relatedness: *coherence*
  - Semantic similarity of candidate and ANNIE named entity type
- Choose candidate based on all features/scores

Evaluating Scores—Baselines

- Picking candidate at random: F1 = 0.229
- Picking candidate for which the URI was used most often for the label in Wikipedia (=”*commonness*”): F1 = 0.521
  - Commonness Wikipedia is a good prior and a strong baseline
  - Other scores contribute additional kinds of information and can increase the score even if they would give worse results if used alone
- (Results calculated on whole of Sheffield NERD Tweet corpus, available at https://gate.ac.uk/applications/sheffield-nerd-tweet-corpus.zip)

Text-based Contextual Similarity

- Each candidate for a mention represents a DBpedia concept describing that entity
- DBpedia entity has different kinds of information about the entity (usually originally from WP templates):
  Datatype Properties (Literals), Object Properties (other concepts)
- We can infer something about the likelihood of that candidate being correct by comparing the context in which the mention appears with the literal/textual information of the Dbpedia concept: short abstract, birthplace label, country label, ...
- How to do this? Vector space approaches

Text-based Contextual Similarity

- Bag-of-word vectors are constructed from
  - the context window of the mention
  - the text/literals of the DBpedia concept for each candidate
- A semantic space is prepared in advance from DBpedia short abstracts
  - Encodes word co-occurrences
  - TF-IDF transform emphasises the most discriminatory words
- Context and candidate vectors are mapped through the semantic space to re-weight and expand with related terms
- Similarity (cosine) of the resulting vectors is calculated

How Useful?

	Precision	Recall	F1
Abstracts Only	0.201	0.421	0.272
All Textual Fields	0.221	0.379	0.279

- Overall result is much lower than commonness but significantly higher than choosing a random candidate
- This information is not correlated to commonness and so it can be combined with commonness to achieve a score higher than is possible using commonness alone
- This is a typical approach though other systems vary in the the details (what text from dbpedia, semantic space ...)

YODIE Scores—Coherence

- Related candidates are more likely to be correct
- Different ways of deciding if candidates are related
- We use two; a DBpedia-based metric (LOD-based structural similarity) and a Wikipedia-based one (relatedness)

KAGAWA will be allowed to rejoin Borussia Dortmund in January in a swap deal which would see defender @NSubotic4

Julie Kagawa (Author)

Shinji Kagawa (Footballer)

Kyoko Kagawa (Actress)

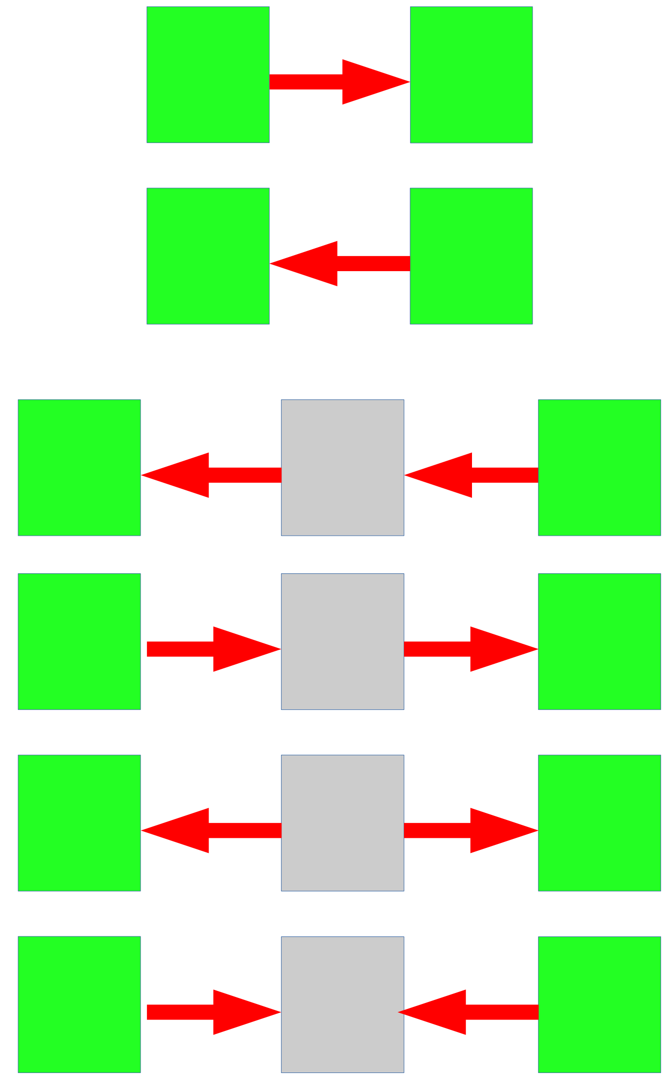
Footballers

Neven Subotic (Footballer)

Nash Subotic (CEO, Westpac)

LOD-based Structural Similarity

- DBpedia contains relations such as who is the president of a country, or who starred in a particular film
- Relation types
    - Direct connection—inbound
    - Direct connection—outbound
    - Indirect:
        - unidirectional—both inbound
        - unidirectional—both outbound
        - shared parent
        - shared child



LOD-based Structural Similarity

- DBpedia relations are higher quality but sparser
  - C_c is current candidate
  - n is number of mentions in context window (entire tweet)
  - m is number of candidates on the entity
  - R is count of relationships between candidates in DBpedia
  - w is inverse square of degrees of separation in relationship
  - d is distance between mentions in characters

$$s(C_c) = \sum_{i=1}^{n} \left(\frac{\sum_{j=0}^{m} w\, R(C_c, C_{ij})}{d_{(c,i)}} \right)$$

Relatedness (Milne & Witten 2008)

- Popular Wikipedia-based score
  - A is set of Wikipedia pages that link to a
  - B is set of Wikipedia pages that link to b
  - W is all pages in Wikipedia

$$r(a,b)=\frac{\log(max(|A|,|B|))-\log(|A\cap B|)}{\log(|W|)-\log(min(|A|,|B|))}$$

Coherence—How Useful?

	Precision	Recall	F1
LOD-based	0.416	0.267	0.326
Relatedness	0.236	0.244	0.240

- Again, these are helpful metrics providing new information, that can supplement what we have
- LOD-based coherence may have some bias toward common entities, meaning though the score is higher in absolute terms, it contributes less in conjunction with commonness

Further Features Also Included ..

- Class match
    - ANNIE is used to determine what class of entity we expect at that point in the text
    - We can then calculate, for each candidate, whether that candidate would be a class match to ANNIE
  - String and POS features for this and adjacent tokens were evaluated but expensive and require larger training corpus to be useful
  - Commonness metrics
    - Calculated in advance for all entities in DBpedia
  - Death date, link frequency etc ..

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 or false for whether it is the right answer
- ML task is to decide if a candidate is right or not based on features

Choosing a Candidate

- We use a probabilistic SVM
  - Chosen for superior performance in comparison trials
  - RBF kernel, high cost parameter
  - Achieves 0.96 accuracy, 0.85 kappa on task of identifying correct candidates (but note this is a subtask only)
- Training data is three years of TAC data plus AIDA training corpus and the training portion of the Sheffield NERD Tweet corpus
- Assigns correct or incorrect to each candidate, with a probability
  - This may mean that we have multiple corrects, or none
  - Multiple corrects—use the probability assigned by the machine learner to select the best
  - No corrects—remove. Disambiguation stage responsible for removing spurious mentions

Choosing a Candidate

- This is a typical approach
- Others have also tried “learning to rank” rather than evaluating each candidate on its own merits
  - Tries to decide which candidates are better than which other candidates
- Graph-based disambiguation also popular
  - Attempts to make a congruent choice for all mentions simultaneously

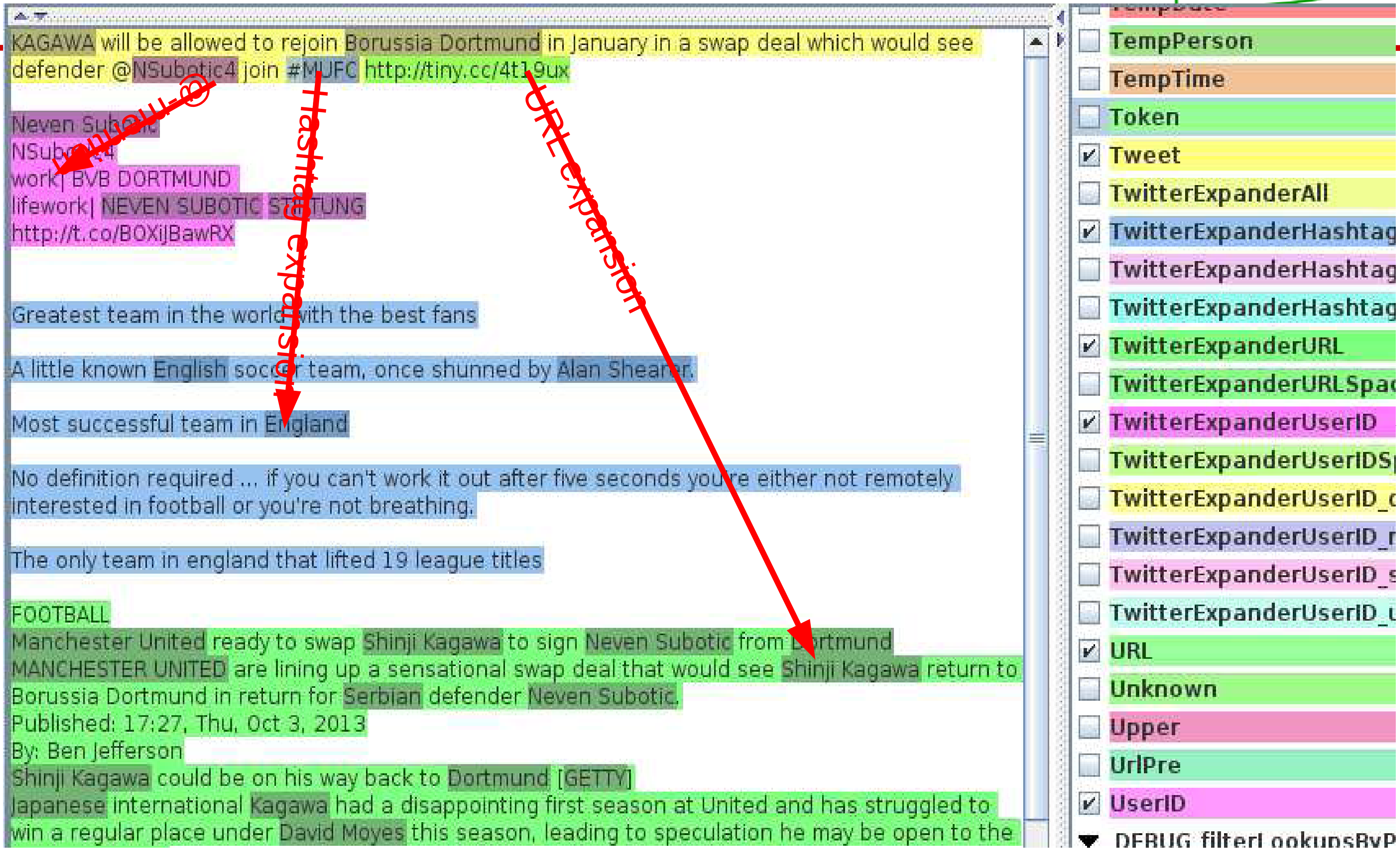
Some Performance Figures on “AIDA B” Corpus

	Prec	Recall	F1
YODIE	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

- This corpus is newswire text

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

How much do Tweet Expansions Help?

	Precision	Recall	F1	Accuracy
Base	0.442	0.550	0.490	0.550
Id	0.444	0.557	0.494	0.557
Id + Proj	0.444	0.642	0.525	0.642**
Url	0.452	0.568	0.504	0.568*
Hash	0.446	0.559	0.496	0.559*
Id + Proj + Url	0.452	**0.660**	0.536	**0.660****
All	**0.495**	0.623	**0.552**	0.623**

- Significant improvements in accuracy, in excess of 10% are obtainable, and 6% in F1 (significance not calculated for F1)
- Note that final accuracy for overall task is different to ML classification accuracy. Here, it means accuracy in getting the referent correct. Earlier it meant deciding for all the candidates if they were correct or not, correctly!
- (Figures calculated on test portion only of Sheffield NERD Tweet corpus, so not directly comparable to those given earlier)

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

- Evaluation results vary a lot depending on test corpus characteristics and task definition (nils? Nil clustering? What types are included? What tweet expansions are available?)
- If we include the tweet expansion work, we get a good result on tweets
  – Though note that the other systems might do better too if they used that information!

Demos!