Università degli Studi di Pisa

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica

A Framework to compare text annotators and its applications

October 2012

Master Degree Thesis

Candidate
Marco Cornolti
cornolti@cli.di.unipi.it

Supervisor
Prof. Paolo Ferragina
ferragin@di.unipi.it
Università di Pisa

Co-Reviewer
Prof.ssa Anna Bernasconi
annab@di.unipi.it
Università di Pisa

Academic Year 2011/2012

Contents

Chapter 1

Introduction

Concerning topic-retrieval

Texts in human languages have a low logical structure and are inherently ambiguous because of this structure and the presence of polysemous terms. Nevertheless, the typical approach of Information Retrieval to manage text documents has been, up to now, based on the Bag-of-words model (BoW) [38]. In this model, texts are represented as the multi-set of terms they contain, thus discarding any possible structure or positional relations existing among the terms. Moreover, terms are interpreted as sequence of characters and mapped to independent dimensions into a huge Euclidean space, so that synonymy and polysemy issues are not taken into account at all. Because of its simplicity, BoW is at the core of most (if not all) current text retrieval systems; but anyone is aware of its obvious limitations.

Recently, some research groups tried to overcome these limitations by proposing a novel approach which consists of adding some “contextual information” to text representation, identifying meaningful mentions in the input text and linking them to their corresponding topics provided by a proper ontology. See Figure 1.1 for an example. This process is nowadays called “text annotation” and the software systems which implement it are called “topic annotators” or “topic-retrieval systems”. These systems differ to each other by the ontology used to extract the annotated concepts (E.g. Wordnet [10], CiC [24], Wikipedia, Yago 2 [16]) and by the algorithms employed to derive these annotations.

The power of these systems resides in the underlying structure which interconnects the topics attached to the texts within the ontology. The most successful systems are currently the ones based on Wikipedia, and these systems have been applied to improve the performance of IR tools on many classic problems such as: the categorization or the clustering of documents; the topic-based search over a web collection, and so on.

The success in the use of Wikipedia lies in the fact that this online encyclopedia offers free (as in freedom) and open access to a huge knowledge base that, despite not being guaranteed to be correct, provides a very high quality thanks to the process used to author Wikipedia pages [22]. Wikipedia is open to the contribute of everyone, including anonymous users, and pages authoring is collaborative. The lack of a central authority potentially leads to a low reliability, but since the review process is distributed as well, involuntary mistakes or malicious errors are quickly found and corrected [25].

Wikipedia is a huge mine of semi-structured information. First of all, Wikipedia pages can be seen as a representation of specific and unambiguous topics. Their abstract and their content give a detailed description of the topic, metadata like the hits count and the revisions give information about the popularity of the concept and how frequently its description changes, pages are categorized by a rich set of categories, and anchors of links to a Wikipedia page offer a set of commonly used synonyms for the concept the page is about. But the most interesting information lies into the structure of its graph, where the nodes are the Wikipedia pages and the edges are the links between pages: the shape of the graph can tell much about how semantically close two pages are [31, 40, 12, 15, 34].

Many topic-retrieval systems use this direct graph, and the whole information that Wikipedia offers, to solve synonymy and polysemy issues in the input text [14, 28, 5, 8, 9]. The approaches followed by the systems differ in the way this information is exploited.

Our Contribution

These systems give surprisingly good results, but the research have followed specific and target-oriented trends, leading to disuniform terminology and approaches, despite targeting the same set of problems. To address the research, it is fundamental to have a consistent framework that offers a formal base upon which it's possible to build new theories, algorithms and systems.

Moreover, there is no improvement without measuring, and literature gives a plenty of ways to determine the performance of a system. Unfortunately, the used methods are inconsistent with each other.

The aim of this thesis is to formalize such a framework, presenting both some of the problems related to topic-retrieval and a set of measures to assess the performance of the systems in solving those problems. The result of this work is a benchmarking framework software that performs the measures on the systems.

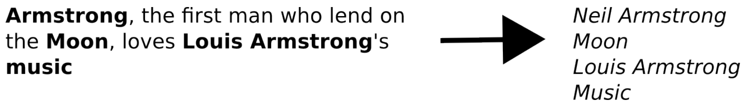

Armstrong, the first man who lend on the **Moon**, loves **Louis Armstrong**'s **music**
Neil Armstrong
Moon
Louis Armstrong
Music

Figure 1.1: An example of a topic retrieval task: from the un-structured text on the left, the unambiguous topics are extracted.

In Chapter 2, we discuss the formal framework, presenting the problems related to topic-retrieval. To solve the lack of uniformity in the measures, our contribution is the presentation in Chapter 3 of a set of metrics that can be used to fairly compare the topic annotators to each other. Despite sounding straightforward, the definition of this metrics hides non-trivial issues. The implemented benchmarking software based on these metrics is presented in Chapter 4.

In Chapter 5 a snapshot of the state-of-the-art topic annotators is given, as well as the available datasets to perform the benchmarking. Results of the benchmark of these annotators on the dataset are given in Chapter 6 and show that some systems, like TagMe, Illinois Wikifier and Wikipedia Miner, give good results with a rather low runtime, suggesting their application to large-scale datasets.

In Chapter 7 some lines of possible future development are presented.

To facilitate the reading of this thesis, all defined formulas are reported in a table in Appendix A, with a brief description.

Chapter 2

Some topic retrieval problems

> What's in a name? that which we call a rose
> By any other name would smell as sweet.
>
> – *William Shakespeare, Romeo and Juliet, Act II*

2.1 Terminology

As stated in the Introduction, literature about topic retrieval presents a wide variability of terminologies. The following terminology, that will be used in the next chapters of this thesis, is a compromise between the popularity of a term in literature, its clarity, and the avoidance of conflicts with other works that may lead to ambiguity.

- A *concept* is a Wikipedia page. It can be uniquely identified by its Page-ID (an integer value).
- A *mention* is the occurrence of a sequence of terms located in a text. It can be codified as a pair $\langle p, l \rangle$ where p is the position of the occurrence and l is the length of the sub-string including the sequence of terms.
- A *score* is a real value $s \in \mathbb{R}$, $s \in [0, 1]$ that can be assigned to an annotation or a tag. Higher values of the score indicate that the annotation (or tag) is more likely to be correct.
- A *tag* is the linking of a natural language text to a concept and is codified as the concept c the text refers to. A *tag* may have a score: a *scored tag* is encoded as a pair $\langle c, s \rangle$, where s is the score.

This thesis	Milne-Witten [33]	Han-Sun-Zhao [13]	Ferragina et al. [11]	Ratinov et al. [36]	Meij et al. [27]
concept	sense	entity	sense	Wikipedia title	concept
mention	anchor	name mention	spot	mention	
tag					annotation
annotation	link	entity linking	annotation	mapping	
score		score	ρ-score	score	

Table 2.1: Terminology used by some of the works in literature.

- An *annotation* is the linking of a mention in a natural language text to a concept. It can be codified as a pair $\langle m, c\rangle$ where m is the mention and c is the concept. An annotation may have a score: a *scored annotation* can be codified as $\langle m, c, s\rangle$, where s is the score.

Table 2.1 reports a "vocabulary" of the different terminology used in other publications.

2.2 Definition of problems

We define a set of problems related to the retrieval of concepts in natural language texts. The difference between these problems can be seen as subtle but leads to different approaches to measure the performance of a topic-retrieval system. The definition of the problems related to topic retrieval is given in Table 2.2, where each problem comes with the type of the input (that is, the type of a problem instance) and output (the type of the solution). Figure 2.1 shows some examples of input and correct output for the given problems.

2.2.1 The topic-retrieval problems and their applications

Problems presented Table 2.2 face different applications. Let's give some shallow examples. For a document clustering based on the document topics, we are interested in finding only the tags, and not the annotations of a document, hence this application would depend on the the solution of C2W and its scored variant Sc2W. Successful applications of this problem are presented in [18, 20, 39]. Rc2W, that returns concepts ordered by the likelihood that they are correct, can as well be used. Applications such as user profiling and document retrieval are as well based on these problems.

Annotations are useful for assisting human reading. Reading a text, like an article on an on-line newspaper, the meaning of some mentions could be unclear. Annotating the text adding a link from these mentions to the right

Problem	Input	Output	Description
Disambiguate to Wikipedia (D2W)	Text, Set of mentions	Set of annotations that, to each mention given as input, assign a (possibly *null*-) concept.	Given a list of mentions, find the concept expressed by each mention (*null* if the concept could not be found). This problem has been defined in [36].
Concepts to Wikipedia (C2W)	Text	Set of tags	Identify the set of concepts that are explicitly mentioned in a text.
Scored concepts to Wikipedia (Sc2W)	Text	Set of scored tags with distinct concepts	Identify the set of concepts that are explicitly mentioned in a text. Tags are assigned a score representing the likelihood that the tag is correct.
Annotate to Wikipedia (A2W)	Text	Set of annotations	Identify the relevant mentions of the text and the concepts expressed by those mentions.
Scored-annotate to Wikipedia (Sa2W)	Text	Set of scored annotations	Identify the relevant mentions of the text and the concepts expressed by those mentions. Annotations are assigned a score representing the likelihood that the annotation is correct.
Ranked-concepts to Wikipedia (Rc2W)	Text	List of tags	Identify the set of concepts that are cited in a text. The concepts are ranked by the probability that the concept is correct.

Table 2.2: Description of topic-retrieval problems

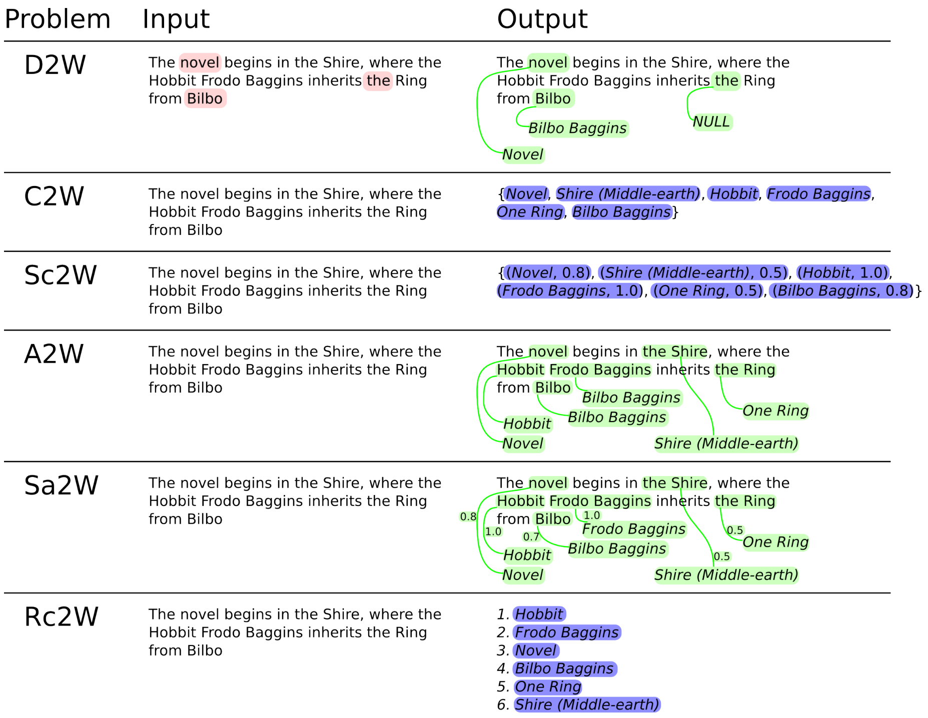


Figure 2.1: Examples of instances of the topic-retrieval problems and their correct solution. Mentions are highlighted in red, (scored) tags in blue, (scored) annotations in green. Concepts are in italics.

Reduction	Instance adaptation	Solution adaptation
A2W $\propto$ Sa2W	No adaptation.	discard the scores, take only the concepts with a score higher than a given threshold.
D2W $\propto$ A2W	Take only the text, discard the mentions.	let M be the set of mentions to disambiguate, part of the instance. Take only the annotations $\langle m, c\rangle$ of the solution such that $m \in M$. Set the concept of all other mentions in M to *null*.
Sc2W $\propto$ Sa2W	No adaptation.	Discard the mentions, take only the concepts and their score. Let $A = a_1, \cdots, a_n$ be the solution of problem Sa2W. If two annotations a_i and a_j have the same concept, discard the one with lower score.
Rc2W $\propto$ Sc2W	No adaptation.	Take only the concepts with a score higher than a given threshold. Rank the concepts by their score, discard the scores.
C2W $\propto$ Rc2W	No adaptation.	Turn the list into a set.
C2W $\propto$ A2W	No adaptation.	Discard the mentions, take only the set of concepts.

Table 2.3: Reduction between problems. $A \propto B$ means A can be reduced to B (hence B is harder than A).

Wikipedia concept would help a human understand the text. This is called *text augmenting* and lies upon the solution of A2W and its scored variant Sa2W. D2W could also be used for this application, interactively asking the human user which mentions should be annotated.

2.3 Our contribution: A hierarchy of problems

It is of key importance to note that the defined problems are strictly related to each other, and some of them can be reduced to others. Note that $A \propto B$ indicates that A reduces to B, so that an instance I_A of A can be adapted in polynomial time to an instance I_B of B and a solution S_B of B can be adapted in polynomial time to a solution S_A of A. In this case, we say that B is harder than A, since an algorithm that solves B also solves A. For a presentation of the reduction theory, see [7].

Let's give an example. If a solution S_{A2W} for the A2W problem is found, it can be adapted in polynomial time ($O(n)$ where n is the size of the output) to a solution S_{C2W} of the C2W problem, by simply discarding the mentions and leaving the retrieved concepts. The instance I_{C2W} of the problem C2W does not even need any adaptation to fit to problem A2W, since both problems have texts as instances (thus $I_{A2W} = I_{C2W}$ and the adaptation is $O(1)$).

We can safely assume the reductions presented in Table 2.3.

Keeping in mind that the reduction between problems is transitive and reflexive, this leads to a hierarchy of problems illustrated by the graph in Figure

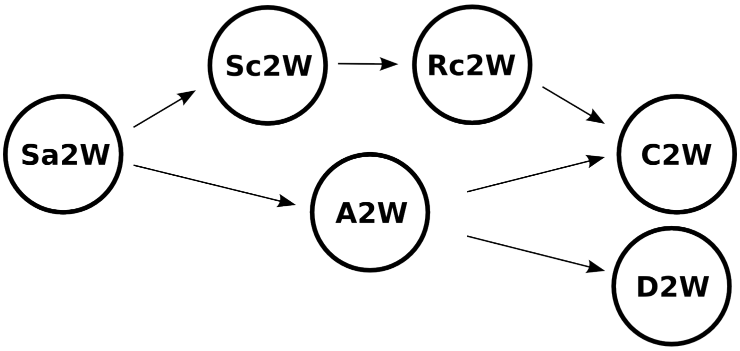


Figure 2.2: Preordering of the reductions between problems.

2.2. The most general problems is Sa2W, and all other problems reduce to it. Problems D2W and C2W are the most specific. The complete chains in the preordering of the problems are:

$$C2W \propto Rc2W \propto Sc2W \propto Sa2W$$
$$C2W \propto A2W \propto Sa2W$$
$$D2W \propto A2W \propto Sa2W$$

Throwing a rope to the next chapters, it is fundamental to point out that, for the purpose of benchmarking the annotation systems, the performance of two systems S' (solving problem P') and S'' (solving problem P'') can be fairly compared only if they respectively solve P' and P'', and there is a problem P such that $P \propto P' \ \wedge \ P \propto P''$. In this hypothesis, the evaluation can be done with respect to the ability of systems S' and S'' to solve problem P. Since the preordering of the reductions is reflexive, obviously two systems can be compared if they both solve problem P or if they solve respectively P' and P'', and $P' \propto P''$.

Keeping an eye on the graph in Figure 2.2, note that the performances of systems S' and S'' in solving problem P can be compared if and only if a reverse-path exists from problem P to P' and from problem P to P''.

Also note that, if $P' \propto P''$, a dataset giving a gold standard (i.e. an expected output) for P'' can be adapted to be used as a gold standard for P', using the same techniques presented in Table 2.3.

2.4 Conclusions

This chapter has presented the basic terminology that will be used in the following chapters. We also presented the first part of the formal framework, defining a set of problems related to topic retrieval. These problems show different features, but can be framed into a preordering representing the reduction between them. This lets two systems natively solving two different problems P' and P'' be fairly compared to each other with respect to their ability to solve a third problem P such that $P \propto P' \wedge P \propto P''$.

Chapter 3

New evaluation metrics

> We judge ourselves by what we feel capable of doing,
> while others judge us by what we have already done.
>
> – *Henry Wadsworth Longfellow, Kavanagh: A Tale.*

The issue of establishing a baseline, shared by the community, to evaluate the performance of systems that solve the problems presented in Chapter 2 is of crucial importance to help the development of new algorithms and to address the research in this field. In this chapter, some metrics for the evaluation of correctness are proposed. The aim is to establish a set of experiments that fairly evaluate the performance of a system (Section 3.1) and evaluate how similar two systems are (Section 3.4). The performance of a system mainly depends on four factors:

1. The ability of the system in recognizing the mentions (for problems Sa2W and A2W).
2. The ability of the system in assigning a set of candidate concepts to each mention (for problems Sa2W, A2W and D2W) or to the whole text (Sc2W, Rc2W, C2W);
3. The ability of the system in selecting the right concept (disambiguation);
4. The ability of the system in assigning the score (for problems Sc2W, Sa2W) or the ranking (for Rc2W) to the annotations or tags.

While Section 3.1 presents a set of metrics to evaluate the performance of the full chain of these abilities (1-4) for all problems, some of them can be tested in-depth and separately: Section 3.2 focuses the metrics on the evaluation of finding mentions (ability 1), while metrics presented in Section 3.3 evaluates the ability of finding the right concepts (abilities 2 and 3).

3.1 Metrics for correctness evaluation

Experiments are performed checking the output of the tagging systems against the gold standard given by a dataset. Of course, for each problem a different set of metrics has to be defined. This section covers all the problems presented in Chapter 2. Classical measures like the true positives, false positives, false negatives, precision and recall are generalized and built on top of a set of binary relations. These binary relations represent a match between two tags or two annotations. The necessity of the generalization comes from the need that two annotations or tags, to be considered as matching, do not need to be equal but, more generally, have to satisfy a match relation. Things will be clearer continuing the reading of the next subsections. In the meanwhile, the following definitions are given:

Definition 1 Let X be the set of elements such that a solution of problem P is a subset of X. Let $r \subseteq X$ be the output of the system for an instance I of problem P, $g \subseteq X$ be the gold standard given by the dataset for instance I and M a symmetric match relation on X. The following higher-order functions are defined:

$$\begin{array}{ll}
\text{true positives} & tp(r,g,M) = \{x \in r \mid \exists x' \in g : M(x',x)\} \\
\text{false positives} & fp(r,g,M) = \{x \in r \mid \nexists x' \in g : M(x',x)\} \\
\text{false negatives} & fn(r,g,M) = \{x \in g \mid \nexists x' \in r : M(x',x)\} \\
\text{true negatives} & tn(r,g,M) = \{x \notin r \mid \nexists x' \in g : M(x',x)\}
\end{array}$$

□

Generally, a dataset offers more than one instance. Thus, an output of a system checked against all the instances provided by a dataset consists of a list of results, one for each instance. The following commonly used metrics [26] are re-defined, generalized with the matching relation M.

Definition 2 Let $G = [g_1, g_2, \cdots, g_n]$ be the gold standard given by a dataset that contains n instances $I_1, \cdots, I_n$, given as input to a system, g_i being the gold standard for instance I_i. Let $R = [r_1, \cdots, r_n]$ be the output of the system, where r_i is the result found by the system for instance I_i. The following metrics are defined:

precision	$P(r,g,M) = \frac{\lvert tp(r,g,M)\rvert}{\lvert tp(r,g,M)\rvert + \lvert fp(r,g,M)\rvert}$
recall	$R(r,g,M) = \frac{\lvert tp(r,g,M)\rvert}{\lvert tp(r,g,M)\rvert + \lvert fn(r,g,M)\rvert}$
F1	$F_1(r,g,M) = \frac{2 \cdot P(r,g,M) \cdot R(r,g,M)}{P(r,g,M) + R(r,g,M)}$
macro-precision	$P_{macro}(R,G,M) = \frac{1}{n} \cdot \sum_{i=1}^{n} P(r_i, p_i, M)$
macro-recall	$R_{macro}(R,G,M) = \frac{1}{n} \cdot \sum_{i=1}^{n} R(r_i, g_i, M)$
macro-F1	$F1_{macro}(R,G,M) = \frac{1}{n} \cdot \sum_{i=1}^{n} F_1(r_i, g_i, M)$
micro-precision	$P_{micro}(R,G,M) = \frac{\sum_{i=1}^{n} \lvert tp(r_i,g_i,M)\rvert}{\sum_{i=1}^{n}(\lvert tp(r_i,g_i,M)\rvert + \lvert fp(r_i,g_i,M)\rvert)}$
micro-recall	$R_{micro}(R,G,M) = \frac{\sum_{i=1}^{n} \lvert tp(r_i,g_i,M)\rvert}{\sum_{i=1}^{n}(\lvert tp(r_i,g_i,M)\rvert + \lvert fn(r_i,g_i,M)\rvert)}$
micro-F1	$F1_{micro}(R,G,M) = \frac{2 \cdot P_{micro}(R,G,M) \cdot R_{micro}(R,G,M)}{P_{micro}(R,G,M) + R_{micro}(R,G,M)}$

□

Note that, if the binary relation M is the equality ($M(a,b) \Leftrightarrow a = b$), the measures presented above become the classical Information Retrieval measures.

Now that this layer of metrics have been defined, we can play on the match relation M.

3.1.1 Metrics for the C2W problem

For the C2W problem, the match relation to use is quite straightforward. The output of a C2W system is a set of tags. Keeping in mind that a tag is codified as the concept it refers to, the following definitions are given:

Definition 3 Let T be the set of all tags. A *Strong tag match* is a binary relation M_t on T between two tags t_1 and t_2. It is defined as

$$M_t(t_1, t_2) \Longleftrightarrow d(t_1) = d(t_2)$$

Where d is the dereference function (see Definition 4) □

Definition 4 Let L be the set of redirect pages, C be the set of non-redirect pages (thus $C \cap L = \emptyset$) in Wikipedia. *Dereference* is a function

$$d: \ L \cup C \cup \{\text{null}\} \mapsto C \cup \{\text{null}\}$$

such that:

$$d(p) = \begin{cases} p & \text{if } p \in C \\ p' & \text{if } p \in L \\ \text{null} & \text{if } p = \text{null} \end{cases}$$

where $p' \in C$ is the page p redirects to. □

Definition 3 and the dereference function worth an explanation. In Wikipedia, a page can be a redirect to another, e.g. "*Obama*" and "*Barrack Hussein Obama*" are redirects to "*Barack Obama*". Redirects are meant to ease the finding of pages by the Wikipedia users. Redirects can be seen as many-to-one bindings from all synonyms (pages in L) to the most common form of the same concept (pages in C). Two concepts identified by c_1 and c_2, where $c_1 \neq c_2$ but $d(c_1) = d(c_2)$ (meaning that c_1 redirects to c_2 or that c_1 and c_2 redirect to the same page c_3) represent the same concept and thus must be considered as equal.

It is obvious that the *Strong tag match* relation M_t is reflexive ($\forall x \in T.\ M_t(x,x)$), symmetric ($M_t(y,x) \Leftrightarrow M_t(x,y)$), and transitive ($M_t(x,y) \wedge M_t(y,z) \Rightarrow M_t(x,z)$).

To achieve the actual metrics for the C2W problem, the number of true/false positives/negatives, precision, recall and F1 must be computed according to Definitions 1 and 2, using $M = M_t$.

3.1.2 Metrics for the D2W problem

D2W output consists of a list of annotations, some of them possibly with *null*-concept. To compare two annotations, the following match function, as well reflexive, symmetric and transitive, is given.

Definition 5 Let A be the set of all annotations. A *Strong annotation match* is a binary relation M_s on A between two annotations $a_1 = \langle\langle p_1, l_1\rangle, c_1\rangle$ and $a_2 = \langle\langle p_2, l_2\rangle, c_2\rangle$. It is defined as

$$M_s(a_1, a_2) \Longleftrightarrow \begin{cases} p_1 = p_2 \\ l_1 = l_2 \\ d(c_1) = d(c_2) \end{cases}$$

Where d is the dereference function (see Definition 4). □

Note that in D2W it does not make sense to count the negatives, since the mentions of the annotations contained in the output are the same as the mentions given as input, and only the concepts can be either correct (true positive) or wrong (false positive). To compute the number of true and false positives, as long as the precision, functions defined in Definition 1 and 2 can be used, with $M = M_s$.

3.1.3 Metrics for the A2W problem

As in D2W, the output of a A2W problem is a set of annotations. The main difference is that in A2W the mentions are not given as input and must be found by the system.

A possible set of metrics for A2W would be analogue to those given in Definitions 1 and 2 with $M = M_s$ (Definition 5). But the *Strong annotation match* will result to be true only if the mention matches perfectly, and this approach leaves aside some cases of matches that should still be considered as right. Suppose a A2W annotator is given as input the sentence “The New Testament is the basis of Christianity”. A correct annotation returned by the annotation system could be $\langle\langle 4, 13\rangle, \textit{New Testament}\rangle$ (correctly mapping the mention “New Testament” to the concept *New Testament*). But suppose the gold standard given by the dataset was another similar and correct annotation $\langle\langle 0, 17\rangle, \textit{New Testament}\rangle$ (mapping the mention “The New Testament” to the same concept). Since the mentions differ, a metric based on the *Strong annotation match* would count one false positive and one false negative, whereas only one true positive should be counted. Definition 5 can be relaxed as described in Definition 6 to match annotations with overlapping mentions and same concept.

Definition 6 Let A be the infinite set of all annotations. A *Weak annotation match* is a binary relation M_w on A between two annotations $a_1 = \langle\langle p_1, l_1\rangle, c_1\rangle$ and $a_2 = \langle\langle p_2, l_2\rangle, c_2\rangle$. Let $e_1 = p_1 + l_1 - 1$ and $e_2 = p_2 + l_2 - 1$ be the indexes of the last character of the two mentions. The relation is defined as

$$M_w(a_1, a_2) \iff \begin{cases} p_1 \leq p_2 \leq e_1 \ \vee \ p_1 \leq e_2 \leq e_1 \ \vee \ p_2 \leq p_1 \leq e_2 \ \vee \ p_2 \leq e_1 \leq e_2 \\ d(c_1) = d(c_2) \end{cases}$$

□

A *Weak annotation match* is verified if a *Strong annotation match* is verified (annotations have equal mentions) or, more generally, if the mentions overlap. Both are verified only if the concept of the annotations is the same. Relation M_w is trivially reflexive and symmetric, but is not transitive nor anti-symmetric.

Metrics for the A2W problem can be those defined in Definition 1 and 2, with $M = M_w$ (for *Weak annotation match*) or $M = M_s$ (for *Strong annotation match*).

3.1.4 Metrics for the Rc2W problem

As pointed out in [27], since the output of a Rc2W system is a ranking of tags, common metrics like P1, R-prec, Recall, MRR and MAP [26] should be used after being adapted to the *Strong tag match* relation M_t.

3.1.5 Metrics for the Sc2W and Sa2W problems

As their non-scored version, Sc2W and Sa2W return respectively a set of annotations and a set of tags, with the addition of a likelihood score for each annotation/tag. In practice, the output of such systems is never compared against a gold standard of the same kind (in a gold standard, it's a nonsense to assign a "likelihood score" to the annotations/tags). Hence, the output of a Sc2W and Sa2W system must be adapted (see the Section 2.3 about problem reductions) to the problem for which a solution is offered by the gold standard. This introduce a threshold on the score. Metrics presented above (Definitions 1 and 2 with $M = M_t$ for Sc2W and $M \in \{M_w, M_s\}$ for Sa2W) can be used for values of the threshold ranging in $[0, 1]$.

3.2 Finding the mentions (for Sa2W and A2W)

The metrics presented above for Sa2W and A2W measure the ability of the systems to find the correct annotation, which includes, for each annotation, finding both the correct mention *and* the correct concept. But how much of the error is determined by the lack of mention recognition? To answer this question, we can use a match relation that only checks the overlap of mentions, ignoring the concept:

Definition 7 Let A be the set of all annotations. A *Mention annotation match* is a binary relation M_m on set A between two annotations $a_1 = \langle\langle p_1, l_1\rangle, c_1\rangle$ and $a_2 = \langle\langle p_2, l_2\rangle, c_2\rangle$. Let $e_1 = p_1 + l_1 - 1$ and $e_2 = p_2 + l_2 - 1$ be the indexes of the last character of the two mentions. The relation is defined as

$$M_m(a_1, a_2) \iff p_1 \leq p_2 \leq e_1 \;\vee\; p_1 \leq e_2 \leq e_1 \;\vee\; p_2 \leq p_1 \leq e_2 \;\vee\; p_2 \leq e_1 \leq e_2$$

□

3.3 Finding the concepts (for Sa2W and A2W)

Dually, it would be interesting, when comparing the result of a Sa2W/A2W problem against a gold standard, to isolate the problem of finding the right concepts (discarding the binding of the mentions to the concepts). That's exactly what is done by the metrics for the C2W problem presented above.

Hence, to isolate the measure of concept recognition, the output of a system, as long as the gold standard, have to be adapted to a solution for the C2W problem (See Table 2.3) and measured with metrics presented in 3.1.1.

Note that this is roughly equivalent, for a Sa2W or A2W output, to:

1. Adapt the Sa2W output to a A2W output choosing a score threshold under which annotations are discarded (Sa2W only);
2. Discard annotations $a = (p, l, c) \in A_d$ given for document d such that $\exists a' = (p', l', d') \in A_d \mid a' \neq a \wedge d(c) = d(c')$ (i.e. if more than one annotation have the same concept, keep only one of them);
3. Use the metrics defined in Definitions 1 and 2 with $M = M_c$, defined as:

Definition 8 Let A be the set of all annotations. A *Concept annotation match* is a binary relation M_c on A between two annotations $a_1 = \langle p_1, l_1, c_1 \rangle$ and $a_2 = \langle p_2, l_2, c_2 \rangle$. It is defined as

$$M_c(a_1, a_2) \Longleftrightarrow d(c_1) = d(c_2)$$

Where d is the dereference function (see Definition 4) □

This measure roughly reflects the performance of the candidate finding and the disambiguation process, and is fundamental for all applications in which we are interested in retrieving the concept a text is about, rather than the annotations.

3.4 Similarity between systems

The similarity between systems can be measured considering how similar their output for the text documents contained in a dataset are. In this section, the following definitions hold: let $D = [d_1, d_2, \cdots, d_n]$ be a dataset that contains n documents, given as input to two systems t_1 and t_2 that solve problem $P \in \{\text{Sa2W}, \text{Sc2W}, \text{Rc2W}, \text{C2W}, \text{A2W}, \text{D2W}\}$. Let $A = [a_1, a_2, \cdots, a_n]$ and $B = [b_1, b_2, \cdots, b_n]$ be respectively the output of t_1 and t_2, so that a_i and b_i are the solutions found respectively by t_1 and t_2 for document d_i. The type of elements in A, that is the same as elements in B, varies depending on the problem P that t_1 and t_2 solve.

3.4.1 A new similarity measure on sets

A proposed measure to check the similarity of the two sets of annotations a and b is inspired by the Jaccard similarity coefficient [21], but must take into account the possibility that two annotations match according to a match relation such as those presented in Definitions 3, 5, 6, 7 and 8, even though not being equal. The following measure is proposed:

Definition 9 Let $a \subseteq X$ and $b \subseteq X$ be two sets, and M be a reflexive and symmetric relation on set X. Similarity measure S' is defined as:

$$S'(a,b,M) = \frac{|\{x \in a \mid \exists y \in b : \ M(x,y)\}| + |\{x \in b \mid \exists y \in a : \ M(x,y)\}|}{|a| + |b|}$$

□

Note that function S' is symmetric and ranges in $[0,1]$. Important features of S' are that $S'(a,b) = 1$ if and only if, for all elements in a, there is a matching element in b, and vice-versa; $S'(a,b) = 0$ if and only if there is not one single element in a that matches with an element in b, and vice-versa. Unlike the Jaccard measure, S' is not a distance function since $S'(a,b) = 0$ does not imply $a = b$, and it does not verify the triangle inequality.

Note that for our purpose, as M, any of *Strong tag match* (for Sc2W and C2W systems whose output is a set of tags), *Weak annotation match*, *Strong annotation match*, *Mention annotation match* and *Concept annotation match* (for Sa2W, A2W, D2W systems whose output is a set of annotations) can be used, since they are all reflexive and symmetric.

3.4.2 A similarity measure on lists of sets

The similarity of two lists of sets A and B can be defined as the average of S' on the sets of the lists, giving the same importance to all the sets contained in the lists regardless of their size (S_{macro}) or as the overall "intersection" divided by the overall size, which gives more importance to bigger sets (S_{micro}). Formally:

Definition 10 Let A and B be two lists of elements $a_i, b_i \subseteq X$ and let M be a binary relation on X. The following definitions are given:

$$S_{macro}(A,B,M) = \tfrac{1}{n} \cdot \textstyle\sum_{i=1}^{n} S'(a_i, b_i, M)$$
$$S_{micro}(A,B,M) = \frac{\sum_{i=1}^{n}(|\{x \in a_i \mid \exists y \in b_i : \ M(x,y)\}| + |\{x \in b_i \mid \exists y \in a_i : \ M(x,y)\}|)}{\sum_{i=1}^{n}(|a_i| + |b_i|)}$$

□

S_{macro} and S_{micro} share the same properties as S': they range in $[0,1]$, their value is 0 if and only if, for each $i \in [1, \cdots, n]$, there is not one single element in a_i that matches with an element in b_i and vice-versa, and their value is 1 if and only if, for each $i \in [1, \cdots, n]$ and for all elements in a_i, there is a matching element in b_i, and vice-versa. If M is reflexive and $A = B$, then $S_{macro}(A,B,M) = S_{micro}(A,B,M) = 1$.

3.4.3 Combining S with M_*

Let S be any of S_{macro} or S_{macro}. The meaning of the value given by this similarity measure depends only on the match relation M it is combined with. For C2W and Sc2W, the only defined matching function is $M = M_t$. In this case, S give a measure of how many of the concepts found by t_1 and t_2 are in common.

For all problems whose output is a set of annotations (Sa2W, A2W, D2W), any match relation $M \in \{M_s, M_w, M_m, M_c\}$ can be used, with the following meaning:

- $S(A, B, M_s)$ gives the fraction of common annotations (having same concept and same mention).
- $S(A, B, M_w)$ gives the fraction of common overlapping annotations (having same concept and overlapping mention).
- $S(A, B, M_m)$ gives the fraction of common overlapping mentions found in the text.
- $S(A, B, M_c)$ gives the fraction of common concepts found in the text.

3.4.4 Measuring true positives and true negatives similarity in detail

S-measures can be used not only to check the whole output of two systems. The focus can instead be put on measuring how many of the true positives and true negatives two systems have in common, to see whether their correct spots and mistakes are similar or not. To do this, we can simply take a subset of elements of A and B representing the true positives or the false negatives.

Definition 11 Let $G = [g_1, \cdots, g_n]$ be the gold standard for a dataset, $g_i \subseteq X$ being the gold standard for instance I_i. Let $O = [o_1, \cdots, o_n]$ be the output of a system, $o_i \subseteq X$ being the output for instance I_i. Let M be a reflexive and symmetric binary relation on X. The following definitions are given:

$$T(O, G, M) = [tp(o_1, g_1, M), \cdots, tp(o_n, g_n, M)]$$
$$F(O, G, M) = [fp(o_1, g_1, M), \cdots, fp(o_n, g_n, M)]$$

Where tp and fp are the true positives and the false positives functions defined in Definition 1. □

$T(O, G, M)$ and $F(O, G, M)$ are lists containing, for each instance I_i, respectively the true positives and the false positives contained in the output o_i according to the match relation M and the gold standard g_i.

The fraction of common true positives between outputs A and B is hence given by $S(T(A, G, M), T(B, G, M), M)$ whereas the fraction of common false negatives is given by $S(F(A, G, M), F(B, G, M), M)$, where S can be either S_{micro} or S_{macro}.

3.5 Conclusions

This chapter has presented the second part constituting the formal framework employed in this thesis. The classical measures of Information Retrieval have been generalized adding a match relation M. This includes the basic measurement of true/false positives/negatives for the solution of a single instance and the F1, precision and recall measurements, in their macro- and micro- version, for a set of solutions to instances given, for example, by a dataset. M is a binary relation defined on a generic set X such that the output of a system is formed by a subset of X. Playing on M, we can focus the measurement on specific features of the comparison. For every problem, we defined a proper match relation that, combined with the defined measures, lets us evaluate the performance of a system in solving that problem. Other proposed match relations let us focus the measures on certain aspects of a system.

We also defined a way of comparing the output of two systems, as well based on a match relation. This S measure is inspired by the Jaccard similarity measure but takes as parameter a match relation M. The similarity can be restricted to the true positives or the false positives using functions T and F.

Chapter 4

The comparison framework

Comparisons are odious.

– *Archbishop Boiardo, Orlando Innamorato.*

It has been developed a benchmarking framework that runs experiments on systems that solve problems given in Chapter 2 in order to measure the performance of the systems and their similarity. The framework is based on the metrics given in Chapter 3, providing an implementation of the proposed measures and match relations. The target was to create a framework that is easily extendible with new problems, new annotation systems, new datasets, new match relations and new metrics not yet defined. This work is intended to be released to the public, as a contribution to the scientific community working on the field of topic retrieval. We would like it to become a basis for further experiments, that anyone can reproduce on its own. Distributing this work open source and with a clear documentation is a condition to let anyone assess its fairness or propose modifications to the code.

The framework is written in Java and implements the actual execution of the systems on a given dataset, the caching of the results, the measuring of the performance in terms of correctness and runtime against a given dataset, the computation of the similarity between systems, the reduction between problems (that is, given two problems P', P'' such that $P' \propto P''$, adapting an instance of P' to P'' and adapt the solution of P'' to P'). Datasets and topic-retrieval systems are implemented as plugins.

4.1 Code structure

The code of the comparison framework is organized in 8 Java packages. All package names begin with `it.acubelab.annotatorBenchmark`:

`.data` contains classes representing basic objects needed by the framework: `Annotation`, `ScoredAnnotation`, `Tag`, `ScoredTag`.

`.cache` contains the caching system for the results of the experiment. Caching is needed to avoid the repetition of experiments that may last for days, depending on the size of the dataset. The package also contains, in class `Benchmark`, the core of the framework, namely the methods to actually perform the experiments and store the results in the cache. Caching is done by simply storing the result in an object of the class `BenchmarkResults` and serializing it to a file.

`.problems` contains interfaces representing a dataset for the problems defined in Chapter 2 (such interfaces are called P`Dataset`, where P is the name of the problem, E.g. `A2WDataset`, `C2WDataset`) and the interfaces for a system solving one of them (called P`System`, E.g. `A2WSystem`, `C2WSystem`)[1]. The class hierarchy reflects the preordering of the problems given in Figure 2.2: if $P \propto Q$ then a system that solve Q can as well solve P. Speaking of interfaces, this is reflected in the fact that a system that implements interface Q`System` (which defines a method for solving an instance of Q) must also implement methods in PSystem (which defines a method for solving an instance of P), and thus Q`System` extends P`System`. The hierarchy of these classes is reported in Figure 4.2.

`.datasetPlugins` contains some implementations of P`Dataset` interfaces of package `.problems`. These classes provide standard datasets used in literature.

`.systemPlugins` contains some implementations of P`System` interfaces of package `.problems`, providing actual access to the topic-retrieval systems. These classes are the glue between the benchmarking framework and the topic-retrieval systems. Generally, in their implementation, to solve an instance of a problem, they query the system through its web service or running it locally.

`.metrics` contains the implementation of the metrics presented in Chapter 3. The abstract class `Metrics` provides the methods for finding the true/false positives/negatives and for computing precision, recall, F1 and similarity. These methods need, as parameter, an object representing a *match relation*, like those given in Section 3.1. The implementations of the *match relations* (`StrongTagMatch`, `StrongAnnotationMatch`, etc.) are as well

[1] Interfaces for problems for which there is no system (natively or not) solving it or datasets giving a gold standard for it, were not implemented in this version of the benchmarking framework. They will be added if needed.

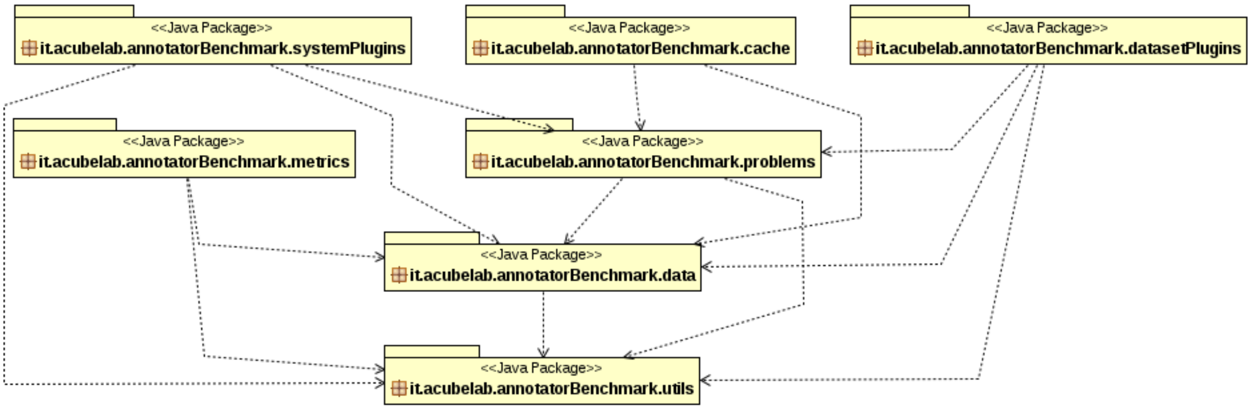

<<Java Package>>
it.acubelab.annotatorBenchmark.systemPlugins
<<Java Package>>
it.acubelab.annotatorBenchmark.cache
<<Java Package>>
it.acubelab.annotatorBenchmark.datasetPlugins
<<Java Package>>
it.acubelab.annotatorBenchmark.metrics
<<Java Package>>
it.acubelab.annotatorBenchmark.problems
<<Java Package>>
it.acubelab.annotatorBenchmark.data
<<Java Package>>
it.acubelab.annotatorBenchmark.utils

Figure 4.1: Automatically-generated UML class diagram with the packages of the annotator benchmark framework and their dependencies.

contained in this package. Classes representing a *match relation* on type `E` implement the generic interface `MatchRelation`, using `E` as type parameter.

`.utils` contains some general-purpose utilities used along the whole framework. This includes some Exceptions; some data-storing utilities; `Export`, the utility to export a dataset in XML form; `ProblemReduction` that implements the adaptations of instances and solutions needed for problem reductions; and `WikipediaApiInterface`, that provides some methods to query the Wikipedia API to retrieve information about concepts (Wikipedia pages). In particular, `WikipediaApiInterface.dereference(int)` implements the de-reference function d defined in Definition 4.

`.scripts` contains example scripts that run the benchmark on the datasets and print data in `gnuplot` and LaTeX style.

The packages form a kind of four-“layer” structure[2], depicted in Figure 4.1. At the two bottom layers lie the utility package and the data package, sparsely used by all other packages. The third layer provides the abstract part of the framework, including the interfaces and the implementation of the metrics measurement. The fourth and most concrete layer provide both the caching and the implementations of some datasets and systems.

[2] The *layer* term is abused, since packages in the upper layer may depend on all lower layers, and not only on the layer just below.

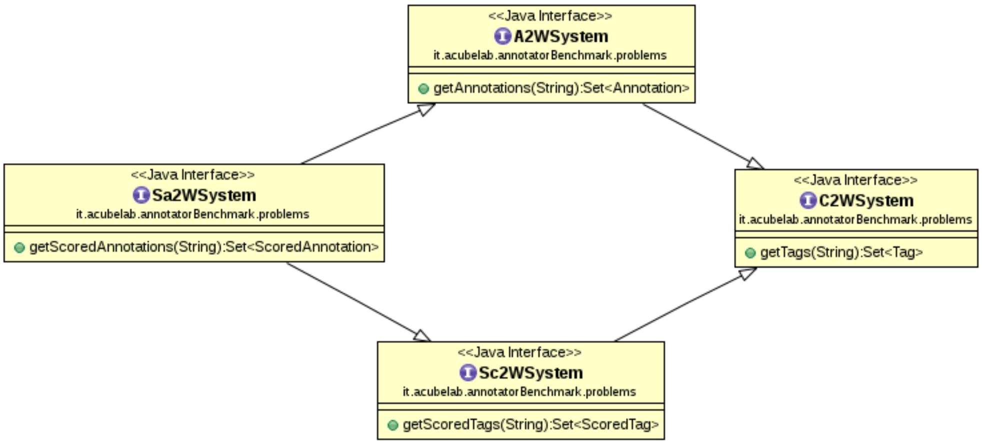

<<Java Interface>>
A2WSystem
it.acubelab.annotatorBenchmark.problems
getAnnotations(String):Set<Annotation>
<<Java Interface>>
Sa2WSystem
it.acubelab.annotatorBenchmark.problems
getScoredAnnotations(String):Set<ScoredAnnotation>
<<Java Interface>>
C2WSystem
it.acubelab.annotatorBenchmark.problems
getTags(String):Set<Tag>
<<Java Interface>>
Sc2WSystem
it.acubelab.annotatorBenchmark.problems
getScoredTags(String):Set<ScoredTag>

Figure 4.2: Hierarchy of interfaces that topic annotators must implement, reflecting the hierarchy of the problems presented in 2.2.

4.2 Running the experiments

A class with a `main()` method running the actual experiments should be located on top of all these packages. The `main()` method basically creates the objects representing the topic-retrieval system, the dataset, the match relation and the metrics, asks the cache for the result for each document of the dataset (the cache will perform the experiment if the result is not cached), and outputs the results given by the metrics. An example snippet of code follows.

```java
public class LulzMain {

  public static void main(String[] args){
    Benchmark.useCache("results.cache");
    WikipediaApiInterface api = new WikipediaApiInterface("wid.cache", "redirect.cache");
    MatchRelation<Annotation> wam = new WeakAnnotationMatch(api);
    Metrics<Annotation> metrics = new Metrics<Annotation>();
    Sa2WSystem miner = new WikipediaMinerAnnotator();
    A2WDataset aidaDs = new ConllAidaDataset("datasets/aida/AIDA-YAGO2-dataset.tsv", api);

    List<Set<ScoredAnnotation>> computedAnnotations = Benchmark.doSa2WAnnotations(miner, aidaDs);
    List<Set<Annotation>> reducedAnnotations = ProblemReduction.Sa2WToA2WList(computedAnnotations, 0.5f);
    List<Set<Annotation>> goldStandard = aidaDs.getA2WGoldStandardList();
    MetricsResultSet rs = metrics.getResult(reducedAnnotations, goldStandard, wam);

    printResults(rs);

    Benchmark.flush();
    api.flush();
  }
```

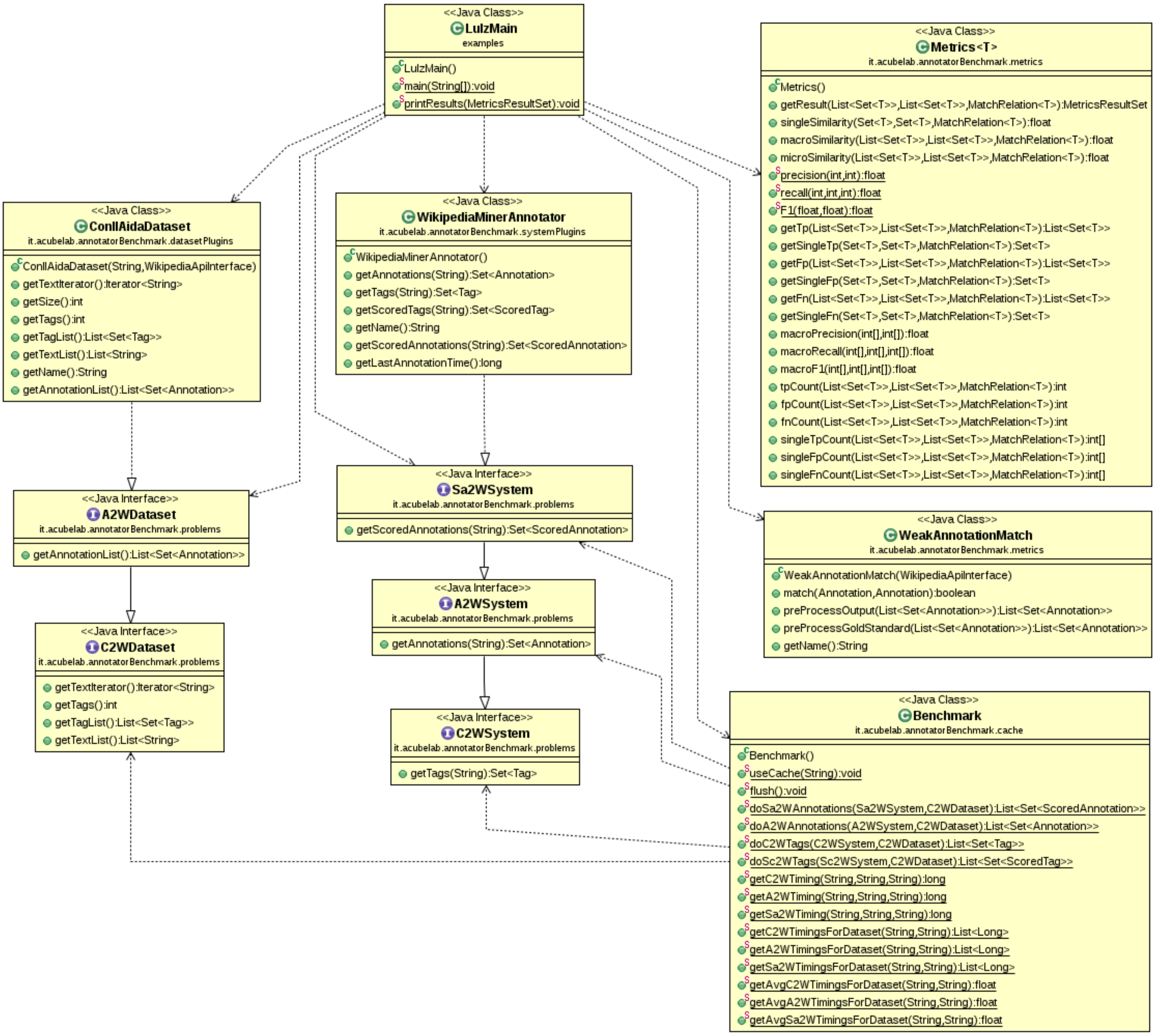

<<Java Class>>
LulzMain
examples
LulzMain()
main(String[]):void
printResults(MetricsResultSet):void
<<Java Class>>
Metrics<T>
it.acubelab.annotatorBenchmark.metrics
Metrics()
getResult(List<Set<T>>,List<Set<T>>,MatchRelation<T>):MetricsResultSet
singleSimilarity(Set<T>,Set<T>,MatchRelation<T>):float
macroSimilarity(List<Set<T>>,List<Set<T>>,MatchRelation<T>):float
microSimilarity(List<Set<T>>,List<Set<T>>,MatchRelation<T>):float
precision(int,int):float
recall(int,int,int):float
F1(float,float):float
getTp(List<Set<T>>,List<Set<T>>,MatchRelation<T>):List<Set<T>>
getSingleTp(Set<T>,Set<T>,MatchRelation<T>):Set<T>
getFp(List<Set<T>>,List<Set<T>>,MatchRelation<T>):List<Set<T>>
getSingleFp(Set<T>,Set<T>,MatchRelation<T>):Set<T>
getFn(List<Set<T>>,List<Set<T>>,MatchRelation<T>):List<Set<T>>
getSingleFn(Set<T>,Set<T>,MatchRelation<T>):Set<T>
macroPrecision(int[],int[]):float
macroRecall(int[],int[],int[]):float
macroF1(int[],int[],int[]):float
tpCount(List<Set<T>>,List<Set<T>>,MatchRelation<T>):int
fpCount(List<Set<T>>,List<Set<T>>,MatchRelation<T>):int
fnCount(List<Set<T>>,List<Set<T>>,MatchRelation<T>):int
singleTpCount(List<Set<T>>,List<Set<T>>,MatchRelation<T>):int[]
singleFpCount(List<Set<T>>,List<Set<T>>,MatchRelation<T>):int[]
singleFnCount(List<Set<T>>,List<Set<T>>,MatchRelation<T>):int[]
<<Java Class>>
ConllAidaDataset
it.acubelab.annotatorBenchmark.datasetPlugins
ConllAidaDataset(String,WikipediaApiInterface)
getTextIterator():Iterator<String>
getSize():int
getTags():int
getTagList():List<Set<Tag>>
getTextList():List<String>
getName():String
getAnnotationList():List<Set<Annotation>>
<<Java Class>>
WikipediaMinerAnnotator
it.acubelab.annotatorBenchmark.systemPlugins
WikipediaMinerAnnotator()
getAnnotations(String):Set<Annotation>
getTags(String):Set<Tag>
getScoredTags(String):Set<ScoredTag>
getName():String
getScoredAnnotations(String):Set<ScoredAnnotation>
getLastAnnotationTime():long
<<Java Interface>>
A2WDataset
it.acubelab.annotatorBenchmark.problems
getAnnotationList():List<Set<Annotation>>
<<Java Interface>>
Sa2WSystem
it.acubelab.annotatorBenchmark.problems
getScoredAnnotations(String):Set<ScoredAnnotation>
<<Java Class>>
WeakAnnotationMatch
it.acubelab.annotatorBenchmark.metrics
WeakAnnotationMatch(WikipediaApiInterface)
match(Annotation,Annotation):boolean
preProcessOutput(List<Set<Annotation>>):List<Set<Annotation>>
preProcessGoldStandard(List<Set<Annotation>>):List<Set<Annotation>>
getName():String
<<Java Interface>>
A2WSystem
it.acubelab.annotatorBenchmark.problems
getAnnotations(String):Set<Annotation>
<<Java Interface>>
C2WDataset
it.acubelab.annotatorBenchmark.problems
getTextIterator():Iterator<String>
getTags():int
getTagList():List<Set<Tag>>
getTextList():List<String>
<<Java Interface>>
C2WSystem
it.acubelab.annotatorBenchmark.problems
getTags(String):Set<Tag>
<<Java Class>>
Benchmark
it.acubelab.annotatorBenchmark.cache
Benchmark()
useCache(String):void
flush():void
doSa2WAnnotations(Sa2WSystem,C2WDataset):List<Set<ScoredAnnotation>>
doA2WAnnotations(A2WSystem,C2WDataset):List<Set<Annotation>>
doC2WTags(C2WSystem,C2WDataset):List<Set<Tag>>
doSc2WTags(Sc2WSystem,C2WDataset):List<Set<ScoredTag>>
getC2WTiming(String,String,String):long
getA2WTiming(String,String,String):long
getSa2WTiming(String,String,String):long
getC2WTimingsForDataset(String,String):List<Long>
getA2WTimingsForDataset(String,String):List<Long>
getSa2WTimingsForDataset(String,String):List<Long>
getAvgC2WTimingsForDataset(String,String):float
getAvgA2WTimingsForDataset(String,String):float
getAvgSa2WTimingsForDataset(String,String):float

Figure 4.3: Main classes and interfaces involved in running the Wikipedia Miner annotator on the Conll/AIDA dataset. Both system and dataset will be presented in Chapter 5

```
21
22   public static void printResults(MetricsResultSet rs){
23     //print the results
24   }
25 }
```

Let's have a closer look at the snippet of code above. Figure 4.3 shows the hierarchy of the classes and interfaces involved in this snippet. Lines 4-9 prepare the environment: the cache containing the results of the problems is bound to file `results.cache`. An object representing the API to Wikipedia is created and assigned to variable `api`. This object is needed by the `match` method of the `WeakAnnotationMatch` object, assigned to variable `wam`, that implements the *Weak annotation match*: as defined in Definition 6, this match relation is based on the *dereference* function, implemented by the Wikipedia API. An object representing the Wikipedia Miner annotator – discussed in Chapter 5 – is created and assigned to variable `miner`. In the implementation, this object queries the Wikipedia Miner web service passing a document as parameter and returning the resulting set of scored annotations. Furthermore, the dataset AIDA/CoNLL is created, loading the instances (i.e. documents) and the gold standard from a file.

Lines 11-14 call some methods to gather the actual output of the Wikipedia Miner system for the dataset AIDA/CoNLL. If this system has already been called for some of the text contained in the dataset, the result stored in the cache will be quickly returned instead of calling the Wikipedia Miner web service. Since Wikipedia Miner solve problem Sa2W, while the dataset represent a gold standard for problem A2W, the output of Wikipedia Miner (for each document, a set of scored annotations) must be adapted to the output of A2W, discarding the scores and taking only the annotations above a given threshold. In the code snippet, the adaption of the solution is done in line 12 and the threshold on the score is set to 0.5. In line 14 the object representing the metrics is called, passing as parameter the list of solutions adapted to A2W, the A2W gold standard given by the dataset, and the *Weak annotation match* object. The metrics object computes the actual measures like the true positives, the precision, the F1, etc. Results are returned in an object of class `MetricsResultSet` whose content is printed to the screen calling method `printResults` in line 16.

Lines 18-19 flush the cache of the results and of the Wikipedia API to a file in a permanent memory, to guarantee a faster execution if the `Main` method is executed twice.

4.3 Extending the framework

An important feature of this benchmarking framework is that it can be easily extended with new problems, new annotation systems, new datasets, new match relations and new metrics. Let's have a closer look at how an implementation should be done for each of these categories.

4.3.1 Extending with new systems

To add to the framework a new system that solves a problem A, all that is needed is to create a class that implements interface A`System`. Note that a single system may natively solve more than one problem. In this case, the class will implement one interface for each problem solved.

Suppose we want to add a new system called Cool Annotator that natively solve A, and let B be a problem such that $B \propto A$ that Cool Annotator does not natively solve. Let `CoolAnnotator` the concrete class representing the system, implementing A`System` (and thus B`System`). `CoolAnnotator` will implement a method defined in A`System` to solve problem A. The implementation of the method defined in B`System` to solve problem B should simply

1. call routine methods to adapt the instance I_B of problem B to an instance I_A of problem A. This takes polynomial time;

2. call the method defined in A`System` for solving problem A on the instance I_A, it will return solution S_A;

3. adapt S_A to a solution S_B of problem B. This takes polynomial time;

4. return S_B.

Methods for adapting the instances and the solutions from a problem to another are implemented in the class `ProblemReductions` in package `utils`. Here follows a complete example for $A =$ A2W, $B =$ T2W (thus, the reduction is T2W $\propto$ A2W). The following interfaces are involved:

Interface `problems.C2WSystem`:

```
public interface C2WSystem extends TopicSystem {
  public Set<Tag> solveC2W(String text) throws AnnotationException;
}
```

Interface `problems.A2WSystem`:

```
public interface A2WSystem extends C2WSystem{
  public Set<Annotation> solveA2W(String text) throws AnnotationException
      ;
}
```

A draft for an annotator natively solving A2W (and thus solving C2W as well), implementing the A2WSystem interface specified above, could look like this:

```java
public class CoolAnnotator implements A2WSystem{
  private long lastTime = -1;

  @Override
  public Set<Annotation> solveA2W(String text) {
    lastTime = Calendar.getInstance().getTimeInMillis();
    Set<Annotation> result = this.computeResult(text);
    lastTime = Calendar.getInstance().getTimeInMillis() - lastTime;
    return result;
  }

  @Override
  public Set<Tag> solveC2W(String text) throws AnnotationException {
    //no adaptation of the instance is needed
    Set<Annotation> tags = solveA2W(text);
    return ProblemReduction.A2WToC2W(tags);
  }

  @Override
  public String getName() {
    return "Cool Annotator";
  }

  @Override
  public long getLastAnnotationTime() {
    return lastTime;
  }

  private Set<Annotation> computeResult(String text){
    // do the actual annotations
  }
}
```

Method `solveA2W()` is the method which is called for annotating a document. In its body, it takes the time after and before the annotation process and store the difference in the `lastTime` variable returned by `getLastAnnotationTime()`. The actual annotations are done by a private method `computeResult()`.

Method `solveC2W()` gives the solution for problem C2W. Since an instance for problem C2W (a text) is also an instance for problem A2W, there is no need to adapt it. The instance is therefore solved for problem A2W calling the method `solveA2W()`. Its solution is then adapted to a solution of the C2W problem, calling `ProblemReduction.A2WToC2W()`. In its body (not listed), this method simply discards the mentions leaving the computed concepts.

4.3.2 Extending with new datasets

To add to the framework a new dataset giving a gold standard for problem A, all that is needed is to make a concrete class C implementing interface A`Dataset`. Talking of inheritance, the same logic explained in previous subsections for the system hierarchy holds for the dataset hierarchy. A complete example of two interfaces `C2WDataset` and `A2WDataset` follows. Note that C2W $\propto$ A2W (see Chapter 2).

Interface `problems.C2WDataset`:

```
public interface C2WDataset extends TopicDataset{
  public List<String> getTextInstanceList();
  public List<Set<Tag>> getC2WGoldStandardList();
  public int getTagsCount();
}
```

Interface `problems.A2WDataset`:

```
public interface A2WDataset extends C2WDataset{
  public List<Set<Annotation>> getA2WGoldStandardList();
}
```

A draft for a dataset, whose name is Lulz Dataset, implementing the `A2WDataset` interface specified above, could look like this:

```
public class LulzDataset implements A2WDataset{
  List<String> texts;
  List<Set<Annotation>> annotations;

  public LulzDataset() throws AnnotationException{
    texts = this.loadTexts();
    annotations = this.loadAnnotations();
  }

  @Override
  public int getSize() {
    return texts.size();
  }

  @Override
  public int getTagsCount() {
    int count = 0;
    for (Set<Annotation> a : annotations)
      count += a.size();
    return count;
  }

  @Override
  public Iterator<Set<Annotation>> getAnnotationsIterator() {
    return annotations.iterator();
  }

  @Override
  public List<Set<Annotation>> getA2WGoldStandardList() {
    return annotations;
  }
```

```java

  @Override
  public List<String> getTextInstanceList() {
    return texts;
  }

  @Override
  public List<Set<Tag>> getC2WGoldStandardList() {
    return ProblemReduction.A2WToC2WList(this.getA2WGoldStandardList());
  }

  @Override
  public String getName() {
    return "Lulz Dataset";
  }

  private List<String> loadTexts(){
    //load the text documents from somewhere into variable texts
  }

  private List<Set<Annotation>> loadAnnotations(){
    //load the annotations for the documents from somewhere into variable annotations
  }
```

4.3.3 Extending the hierarchy of problems

Suppose we want to add a problem A. The hierarchy of interfaces – one for each problem – presented in Figure 4.2, can be extended adding a new interface called A`System` that all systems solving A must implement, providing methods to solve an instance of A.

This interface should extend interface B`System` of package `problems` if and only if $B \propto A$. In this hypothesis, a class implementing A`System` representing a system that solves the harder-problem A, must also implement the methods defined in B`System` (therefore, it must also be able to solve the easier problem B, as by the hypothesis $B \propto A$).

Moreover, the hierarchy of interfaces for datasets, that is isomorphic to the hierarchy for the systems interfaces, can be extended in an analogous way, adding interface A`Dataset` that extends B`Dataset` if and only if $B \propto A$.

Let $B \propto A$. As explained in the previous subsections, to let a system natively solving A solve B as well, a polinomial-time algorithm to adapt an instance of B to an instance of A and a polinomial-time algorithm to adapt the solution of A to a solution of B must be implemented. Two methods performing these two adaptions must be implemented in a class extending `ProblemReduction` in package `utils`.

A complete example follows. Suppose we want to add Ab2W (defined in Definition 13 and further detailed in Chapter 7) to the problem hierarchy. Ab2W is the problem of finding both mentioned concepts expressed in a text (like in Sa2W) *and* the concepts expressed in a text, even though being not mentioned. For an example, see Chapter 7. The solution of the problem is thus formed by two sets: s_a is the set of scored annotations (for mentioned concepts) and s_t is the set of scored tags (for non-mentioned concepts). It's trivial that Sa2W $\propto$ Ab2W: the instance needs no adaptation and the solution of Ab2W can be adapted to Sa2W simply discarding s_t and keeping s_a.

The new interface representing a system that solve Ab2W would look like the following. Method `getAb2WOutput` returns a solution for problem Ab2W:

```java
public interface Ab2WSystem extends Sa2WSystem{
  public Pair<Set<ScoredAnnotation>, Set<ScoredTag>> getAb2WOutput(String text);
}
```

The adaptation of the solution is trivially implemented in a method of a class extending `ProblemReduction`:

```java
public class Ab2WProblemsReduction extends ProblemReduction{
  public static Set<ScoredAnnotation> Ab2WToSa2W(Pair<Set<ScoredAnnotation>, Set<ScoredTag>> ab2wSolution){
    return ab2wSolution.output1;
  }
}
```

Here follows the draft of a class representing a system that solves Ab2W. The class implements `Ab2WSystem`, and thus must provide an implementation for the method `getAb2WOutput`. The actual solution, computed by the private method `computeMentionedAnnotations`, is obviously not listed in the example. Since Sa2W $\propto$ Ab2W, the class also implements method `solveSa2W`, that provides a solution for Sa2W, computed calling the method implemented in `Ab2WProblemsReduction`. All other problems P that this system does not solve natively but such that $P \propto$ Ab2W, have an analogous method.

```java
public class LolAbstractAnnotator implements Ab2WSystem{
  private long lastAnnotation = -1;

  @Override
  public Pair<Set<ScoredAnnotation>,Set<ScoredTag>> getAb2WOutput(String text) {
    lastAnnotation = Calendar.getInstance().getTimeInMillis();
    Pair<Set<ScoredAnnotation>,Set<ScoredTag>> res =  computeAb2wSolution(text);
    lastAnnotation = Calendar.getInstance().getTimeInMillis()-lastAnnotation;
    return res;
  }

  private Pair<Set<ScoredAnnotation>,Set<ScoredTag>> computeAb2wSolution(String text) {
    ... //compute the solution and return it.
  }

  @Override
  public Set<ScoredAnnotation> solveSa2W(String text) {
    return Ab2WProblemsReduction.Ab2WToSa2W(getAb2WOutput(text));
  }

  @Override
  public Set<ScoredTag> solveSc2W(String text) {
    return Ab2WProblemsReduction.Sa2WToSt2W(solveSa2W(text));
  }

  @Override
  public Set<Annotation> solveA2W(String text) {
    return Ab2WProblemsReduction.Sa2WToA2W(solveSa2W(text));
  }

  @Override
  public Set<Tag> solveC2W(String text) {
    return Ab2WProblemsReduction.A2WToC2W(solveA2W(text));
  }

  @Override
  public String getName() {
    return "Lol Abstract Annotator";
  }

  @Override
  public long getLastAnnotationTime() {
    return lastAnnotation;
  }
}
```

4.3.4 Implementation of the metrics

To introduce the extension of the metrics, we first have to explain how the classes involved in the metrics measurement ineract with each other.

Package `metrics` contains class `Metrics` which implements the measures defined in Chapter 3 (precision, recall, F1, etc). All methods for computing such metrics take as argument a match relation M, implemented as an object of type `MatchRelation`. For each *match relation* M given in Chapter 3, there is a class implementing the interface `MatchRelation` that provides an implementation of M in the method `match`.

Class `Metrics` is generic in that it has type variable `T` such that the measures are computed over systems that return sets of objects of type `T` (E.g. `T` can be `Tag`, `Annotation`, etc.). Therefore, measures like micro- and macro- F1, recall and precision are performed for lists of sets of `T`-objects (a set for each instance given by a dataset). Also interface `MatchRelation` has generic type variable `E`, such that the match test is done on elements of type `E`. Of course, to use a `MatchRelation<E>` with `Metrics<T>`, it must be `T` = `E`.

In other words, to assess the overall consistency, it must be true that:

- Match relation M is defined on elements of set X (E.g. X can be the set of all annotations);
- Metrics are measured employing M as match relation, thus the tp, fp, fn, F1, recall and precision measures are performed over subsets of X, while micro- and macro- F1, precision and recall are performed over lists of subsets of X (E.g. M can be the *Weak annotation match* M_w);
- In the framework, elements of X are represented as objects of class `T` (E.g. `T` is class `Annotation`);
- The match relation M is represented as an object of a class implementing `MatchRelation<E>`, let this object be assigned to variable `matchRelation`;
- The metrics are represented as an object of a class `Metrics<T>`, let this object be assigned to variable `metricsComputer`;
- `T`=`E`;

In this scenario, actual measurements using match relation M can be run calling the methods of the `metricsComputer` and passing `matchRelation` as argument.

Interface `MatchRelation` also declares methods `preProcessOutput` and `preProcessGoldStandard`, which are called by all methods of class `Metrics` that implement the measurements, before running the measurements. They should be used if certain metrics need to adapt the output or to perform optimization tasks[3].

Before showing the classes, some preparatory speculations must be done. Some gold standards, as well as the output of some annotators, may contain, for a document, annotations with overlapping or nested mentions. For example, the sentence *"A cargo ship is sailing"* could contain both annotations $a_1 = \langle\langle 2, 10\rangle$, Cargo ship$\rangle$ and $a_2 = \langle\langle 8, 4\rangle$, Ship$\rangle$. Since some annotation systems return overlapping annotations while other don't, comparing the output of an annotation system which contains overlapping annotations against a gold standard that doesn't contain overlapping annotations (or vice-versa) would be unfair. For the sake of simplicity, before comparing the output of an annotator against a gold standard, in the current implementation of the match relations, both are pre-processed and scanned for overlapping mentions: if two annotations overlap, then only the one with the longest mention is kept, while the other is discarded. The choice of keeping the longest mention is motivated by the assumption that longer mentions refer to more specific – and thus more relevant – concepts (see

[3] If no pre-processing has to be done, these methods should simply return the data given as parameter.

the example of “Cargo ship” against “Ship”). Note that, using metrics based on *Weak annotation match*, annotations with longer mentions are more likely to result as a true positive.

Here follows the listing of (parts of) some classes involved in the extension of the metrics.

Class `metrics.Metrics` implements the measures presented in Chapter 3:

```java
public class Metrics <T> {

  public MetricsResultSet getResult(List<Set<T>> outputOrig, List<Set<T>> goldStandardOrig, MatchRelation<T> m) {
    List<Set<T>> output = m.preProcessOutput(outputOrig);
    List<Set<T>> goldStandard = m.preProcessGoldStandard(goldStandardOrig);

    int tp = tpCount(goldStandard, output, m);
    int fp = fpCount(goldStandard, output, m);
    int fn = fnCount(goldStandard, output, m);
    float microPrecision = precision(tp, fp);
    float microRecall = recall(tp, fp, fn);
    float microF1 = F1(microRecall, microPrecision);
    int[] tps = singleTpCount(goldStandard, output, m);
    int[] fps = singleFpCount(goldStandard, output, m);
    int[] fns = singleFnCount(goldStandard, output, m);
    float macroPrecision = macroPrecision(tps, fps);
    float macroRecall = macroRecall(tps, fps, fns);
    float macroF1 = macroF1(tps, fps, fns);

    return new MetricsResultSet(microF1, microRecall, microPrecision, macroF1, macroRecall, macroPrecision, tp, fn, fp);
  }

  public static float precision(int tp, int fp){
    return tp+fp == 0 ? 1 : (float)tp/(float)(tp+fp);
  }

  public static float recall(int tp, int fp, int fn){
    return fn == 0 ? 1 : (float)tp/(float)(fn+tp);
  }

  public static float F1(float recall, float precision){
    return (recall+precision == 0) ? 0 : 2*recall*precision/(recall+precision);
  }

  public List<Set<T>> getTp(List<Set<T>> expectedResult, List<Set<T>> computedResult, MatchRelation<T> m){
    List<Set<T>> tp = new Vector<Set<T>>();
    for (int i=0; i<expectedResult.size(); i++){
      Set<T> exp = expectedResult.get(i);
      Set<T> comp = computedResult.get(i);
      tp.add(getSingleTp(exp, comp, m));
    }
    return tp;
  }

  ...

}
```

Class `metrics.WeakAnnotationMatch` implements the *Weak Annotation Match* M_w defined in Definition 6. Generic type `T` is set to `Annotation`, since M_w is defined on the set of annotations. In the constructor, the interface to the Wikipedia API is passed. This is needed to implement the *dereference function*. The body of method `match` implements the match relation M_w: two annotations match if they have the same (dereferenced) concept *and* their mentions overlap. In methods `preProcessOutput` and `preProcessGoldStandard`, both the system output and the gold standard are searched for internal nested or overlapping annotations, that are discarded according to the speculations previously discussed. Moreover, the information about the concepts contained in the dataset and in the system output (including their possible redirect page) is pre-fetched from Wikipedia, to let the `match` relation work on cached data, avoiding a call to the Wikipedia API for each match test.

Class `metrics.WeakAnnotationMatch`[4]:

```java

package it.acubelab.annotatorBenchmark.metrics;

...

public class WeakAnnotationMatch implements MatchRelation<Annotation>{
  private WikipediaApiInterface api;

  public WeakAnnotationMatch(WikipediaApiInterface api){
    this.api = api;
  }

  @Override
  public boolean match(Annotation t1, Annotation t2) {
    return (api.dereference(t1.getConcept()) == api.dereference(t2.
         getConcept())) &&
        t1.overlaps(t2);
  }

  @Override
  public List<Set<Annotation>> preProcessOutput(List<Set<Annotation>>
       computedOutput) {
    Annotation.prefetchRedirectList(computedOutput, api);
    List<Set<Annotation>> nonOverlappingOutput = new Vector<Set<
         Annotation>>();
    for (Set<Annotation> s: computedOutput)
      nonOverlappingOutput.add(Annotation.deleteOverlappingAnnotations(s)
          );
    return nonOverlappingOutput;
  }

  @Override
```

[4] In this listing, the treating of exceptions has been removed for the sake of clarity.

```
  public List<Set<Annotation>> preProcessGoldStandard(List<Set<Annotation>> goldStandard) {
    Annotation.prefetchRedirectList(goldStandard, api);
    List<Set<Annotation>> nonOverlappingGoldStandard = new Vector<Set<Annotation>>();
    for (Set<Annotation> s: goldStandard)
      nonOverlappingGoldStandard.add(Annotation.deleteOverlappingAnnotations(s));
    return nonOverlappingGoldStandard;
  }

  ...

}
```

4.3.5 Extending with a new match relation

To add a new match relation M_* on objects of type X, a class called `MatchRelationName` should be created implementing `MatchRelation<XObj>`, where `XObj` is the class representing elements of X. `MatchRelationName` could also extend a class implementing `MatchRelation<XObj>`, to increase code reusage, reimplementing only a subset of its methods.

The following listing gives an example of a match relation on tags. The match occurs if and only if the semantic closeness of the two tags, according to a certian function, is greater than 0.5. The class `CloseTagMatch` extends `StrongTagMatch`, and thus implements `MatchRelation<Tag>`, reusing the implementation of methods `preProcessOutput` and `preProcessGoldStandard` provided by `StrongTagMatch`.

```
public class CloseTagMatch extends StrongTagMatch implements MatchRelation<Tag>{

  public CloseTagMatch(WikipediaApiInterface api) {
    super(api);
  }

  @Override
  public boolean match(Tag t1, Tag t2) {
    return closeness(t1, t2) > 0.5;
  }

  private float closeness(Tag t1, Tag t2){
    float closeness = ...
    return closeness;
  }
}
```

Note that, as explained in Chapter 3, all match relations must be reflexive and symmetric. It is up to the user to implement the `match` method properly. Breaking this requirement results in inconsistent measures.

4.3.6 Extending with a new measure

To extend the metrics adding new measures, it is enough to extend class `Metrics` with a new class implementing the new measures.

Here follows a case study: the standard F1 measure is implemented in `Metrics`, but F1 can be generalized adding a β parameter in the form:

$$F_{micro}(R, G, M, \beta) = (1 + \beta^2) \cdot \frac{P_{micro}(R, G, M) \cdot R_{micro}(R, G, M)}{\beta^2 \cdot P_{micro}(R, G, M) + R_{micro}(R, G, M)}$$

Lower values of β give more importance to the precision, while higher values give more importance to the recall. Setting $\beta = 1$ we obtain the F1 measure.

Suppose we want to add this measure to the set of metrics. A new class implementing this measure would look like this:

```java
public class GeneralizedF1Metrics<T> extends Metrics<T> {

  public static float genF1(float recall, float precision, float beta){
    float betaPow = beta*beta;
    return (recall+precision == 0) ? 0 : (1+betaPow)*recall*precision/(
        betaPow*recall+precision);
  }

  public float computeMicroF1(List<Set<T>> outputOrig, List<Set<T>>
      goldStandardOrig, float beta, MatchRelation<T> m) throws
      IOException{
    List<Set<T>> output = m.preProcessOutput(outputOrig);
    List<Set<T>> goldStandard = m.preProcessGoldStandard(goldStandardOrig
        );
    int tp = tpCount(goldStandard, output, m);
    int fp = fpCount(goldStandard, output, m);
    int fn = fnCount(goldStandard, output, m);
    float microPrec = precision(tp, fp);
    float microRecall = recall(tp, fp, fn);

    return genF1(microRecall, microPrec, beta);
  }
}
```

4.4 Conclusions

This chapter presented an implementation of a benchmarking framework, that makes it possible to run an experiment for a system solving one of the problems defined in Chapter 2, measuring its performance with the metrics defined in Chapter 3. In some cases, the code of the benchmarking framework reflects the formal framework defined in the previous chapters.

The benchmarking framework is easily extendible adding new systems, new datasets, new problems and new metrics. This chapter provides detailed examples on how to do that.

Chapter 5

Datasets, Systems and Wikipedia

Every gun makes its own tune.

– *The Good, the Bad and the Ugly.*

There are several systems that solve problems related to the retrieval of topics, most of them developed for research purposes according to different philosophies. This chapter presents the features of a subset of them, selected because they employ interesting and original techniques or give particularly interesting results. All the reviewed systems use Wikipedia not only to refer to its pages as unambiguous topics, but also as a knowledge base to extract the information from. For this reason, this chapter starts with an introduction to the features of the online encyclopedia. To perform the experiments, some datasets have been developed providing a set of instances and the gold standard solutions. These datasets are discussed in the last section of this chapter. In the next chapter are reported the results given by the benchmarking framework for all these systems and datasets.

5.1 Wikipedia and its graph

The Wikipedia online encyclopedia is probably the most important collaborative project ever created on the Internet. Supported by the Wikimedia Foundation, a U.S.-based non-profit organization, and based upon the MediaWiki software, that lets any user edit the encyclopedia pages even with a small technical knowledge, it comes up in 285 languages.

All versions of Wikipedia are entirely released under the Creative Commons Attribution/ Share-Alike 3.0 license, that lets anyone freely distribute its con-

Name	Institution	Addressed problem	Notes & key references
TagMe	University of Pisa	Sa2W	[11]
Illinois Wikifier	University of Illinois at Urbana-Champaign	Sa2W, D2W	[36]
CMNS	University of Amsterdam	C2W	[27]
Wikipedia Miner	University of Waikato	Sa2W	[33, 30, 32]
AIDA	Max Planck Institute for Informatics	Sa2W, D2W	[41, 17]

Table 5.1: Problems natively addressed by the annotation systems.

tents and the derived works. Dumps of the whole encyclopedia are downloadable in open formats. This open approach makes it easy to manipulate the data and to automatically process it.

The english version of Wikipedia, the biggest of all language editions, has about 10 million visits per hour and counts 4 million pages. The active editors (users that made at least 5 edits in a month) are 33,680 and the encyclopedia is growing with 1,067 new articles per day[1]. The Wikipedia graph is a Small-world network, with an average shortest path from any node to another of just 4.5 steps. In the graph, there are more than 70 million edges (an average of about 17.5 outgoing edges per node).

An edge from page p_1 to page p_2 suggests some kind of semantic relation between the two concepts expressed by the two pages, since p_1, at a certian point of the text, cites the concept expressed by p_2. Unfortunately, this does not always indicate an actual semantic proximity, because concepts cited in a text may be not that relevant for the text. For example, the page about *Shoe*s has a direct link to *Bone*[2], even though the two concepts are not strongly related. Given the Small-world property of the graph, this loose correlation gets milder increasing the length of the path in the graph: starting from the page about *Astronomy in medieval Islam*, in just two steps, the page about *Jimi Hendrix* can be reached. A mutual direct link between two pages indicates a stronger semantic relation, and many other *relatedness functions* have been developed to address the task of finding how semantically close two pages are [31, 36].

5.2 Topic-retrieval systems

The state-of-the-art topic retrieval systems and the problem they address natively are presented in Table 5.1.

[1] Data for July 2012, taken from `http://stats.wikimedia.org/EN/`

[2] The anchor is contained in the sentence "*The foot contains more bones than any other single part of the body*".

5.2.1 Overview of the algorithmic features of the topic-retrieval systems

The topic-retrieval systems, while addressing the same area of problems, exploit different techniques.

TagMe searches the input text for mentions picked up by the set of Wikipedia page titles, anchors and redirects. Each mention is associated to a set of candidate concepts the mention may refer to. The disambiguation is done exploiting the structure of the Wikipedia graph, trying to bind the mentions to concepts that are related to each other, using the *relatedness measure* [31] introduced in the Wikification algorithm, that takes into account the amount of common in-going and outgoing links between the two pages. TagMe disambiguation is enriched with a *voting scheme*, in which all possible bindings between mentions and concepts express a vote for the others, and the combination with highest vote average is selected.

Wikipedia Miner is an implementation of the Wikification algorithm presented in [33], one of the first approaches to the disambiguation to Wikipedia. This system performs disambiguation *before* the identification of mentions. Disambiguation is done with a machine-learning approach that trains with links taken from Wikipedia pages (thus created by Wikipedia users). Semantic relatedness between pages is not computed with the *relatedness measure* of the original Wikification algorithm but is as well machine-learned.

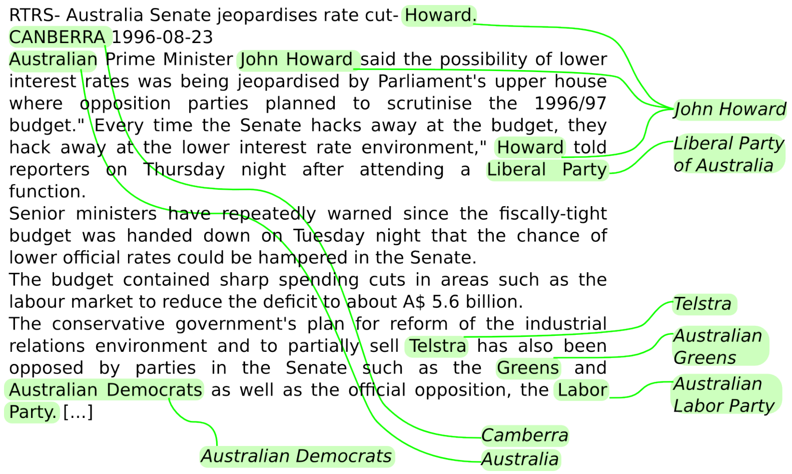
AIDA searches for mentions using the Stanford NER Tagger and uses the YAGO2 knowledge base [16], which provides a catalog of concepts and the relationships among concepts, including their semantic distance. AIDA disambiguation comes in three variants:

PriorOnly Mentions are bound to the concept that is most commonly bound in the knowledge base.

LocalDisambiguation Uses the local similarity disambiguation technique, that disambiguates each mention independently, without enforcing a semantic coherence among the mentions.

CocktailParty YAGO2 is used to perform a collective disambiguation of the mentions: using a graph-based approach, the mapping between mentions and concepts that preserves the highest coherence between each other is iteratively found.

Illinois Wikifier as TagMe, the input text is searched for mentions extracted by Wikipedia anchors and titles using the Illinois NER system [35]. The



RTRS- Australia Senate jeopardises rate cut- Howard.
CANBERRA 1996-08-23
Australian Prime Minister John Howard said the possibility of lower interest rates was being jeopardised by Parliament's upper house where opposition parties planned to scrutinise the 1996/97 budget." Every time the Senate hacks away at the budget, they hack away at the lower interest rate environment," Howard told reporters on Thursday night after attending a Liberal Party function.
Senior ministers have repeatedly warned since the fiscally-tight budget was handed down on Tuesday night that the chance of lower official rates could be hampered in the Senate.
The budget contained sharp spending cuts in areas such as the labour market to reduce the deficit to about A$ 5.6 billion.
The conservative government's plan for reform of the industrial relations environment and to partially sell Telstra has also been opposed by parties in the Senate such as the Greens and Australian Democrats as well as the official opposition, the Labor Party. [...]

Figure 5.1: Part of an example document (news story) given in the AIDA/-CoNLL dataset with its annotations, constituting the gold standard for this document.

disambiguation problem is formulated as an optimization problem. As in the other systems, a global approach is adopted, which instead of disambiguating each mention at a time, tries to disambiguate them all together, preserving the highest coherence. Illinois Wikifier uses an original relatedness measure between Wikipedia pages based on NGD (Normalized Google similarity distance) and PMI (Pointwise mutual information).

CMNS is meant to treat very short documents (e.g. Twitter posts). It generates a ranked list of candidate concepts for all N-grams in the input text. The list is created through lexical matching and language modeling. The disambiguation is done with a method based on supervised machine learning that takes as input a set of documents and, for each document, a set of annotations done by a human.

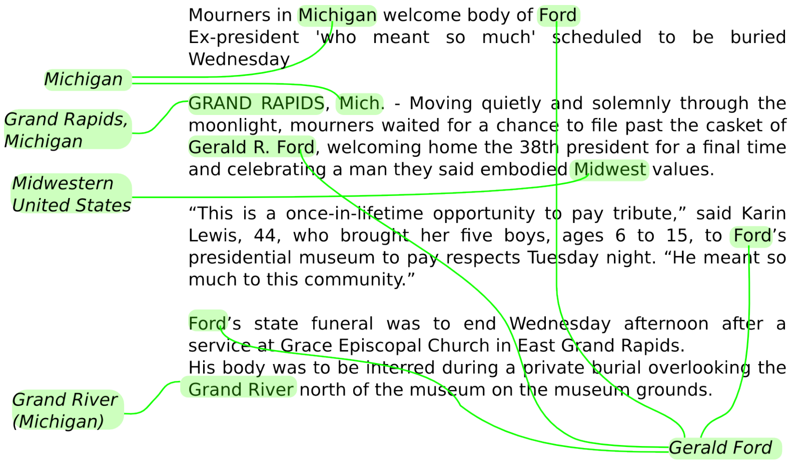
5.3 Available datasets

A noteworthy publication of a new system always comes along with test results on peculiar datasets to assess its performance. Unfortunately, each system is tested on different datasets and with different tricks and measures. Table 5.2 give a description for some of the published datasets, that were implemented in the benchmarking framework.

Some extract of documents given by the datasets as instances, together with the gold standard proposed by the dataset, are given in Figures 5.1, 5.2, 5.3, 5.4 and 5.5.

Dataset	Description	Published in
AIDA/CoNLL	Contains a subset of the the original CoNLL 2003 entity recognition task dataset. The documents are taken from the Reuters Corpus V1 and consists of news stories. A quite large subset of mentions (though not all of them), including the most important ones, are annotated. Topics are annotated at each occurrence of a mention.	[17]
MSNBC	Contains newswire text in English from MSNBC news network. Only important topics are annotated and all occurrences of mentions that refer to those topics are annotated.	[8]
AQUAINT	Contains a subset of the original AQUAINT corpus, consisting of newswire text data in English. Not all occurrences of the mentions are annotated: if more than one mention in a document refers to the same concept, only the first mention is actually annotated. Moreover, only the mentions that refer to topics considered important are annotated. This reflects the Wikipedia-style linking.	[33]
Meij	Contains microblog messages publicly available on twitter. Maximum document length is 140 characters. All topics contained in each tweet are tagged.	[27]
KDD	Contains search engine queries. Most topics are annotated.	[37]

Table 5.2: Description of available datasets.



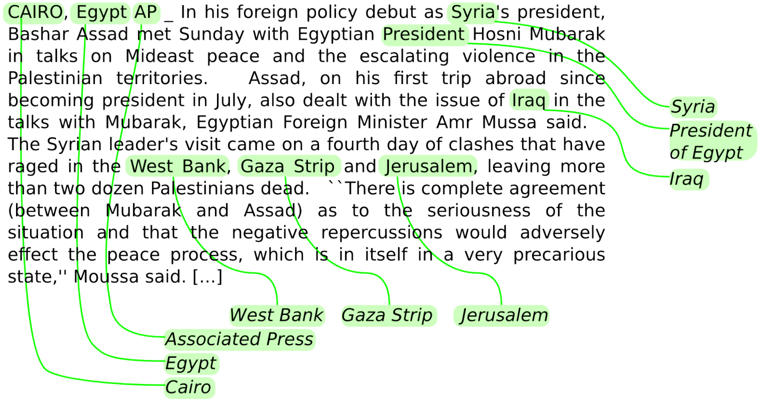
Mourners in Michigan welcome body of Ford
Ex-president 'who meant so much' scheduled to be buried Wednesday

GRAND RAPIDS, Mich. - Moving quietly and solemnly through the moonlight, mourners waited for a chance to file past the casket of Gerald R. Ford, welcoming home the 38th president for a final time and celebrating a man they said embodied Midwest values.

“This is a once-in-lifetime opportunity to pay tribute,” said Karin Lewis, 44, who brought her five boys, ages 6 to 15, to Ford’s presidential museum to pay respects Tuesday night. “He meant so much to this community.”

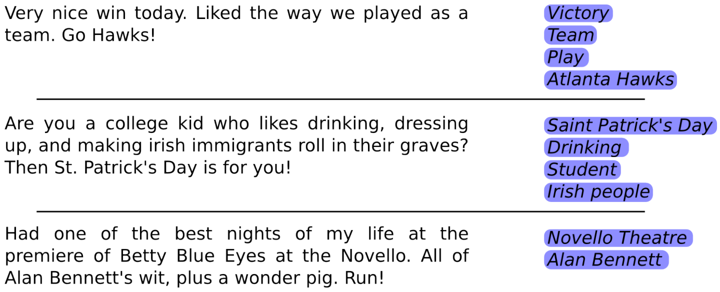
Ford’s state funeral was to end Wednesday afternoon after a service at Grace Episcopal Church in East Grand Rapids.
His body was to be interred during a private burial overlooking the Grand River north of the museum on the museum grounds.

Figure 5.2: Part of an example document (news story) given in the MSNBC dataset with its annotations, constituting the gold standard for this document.



CAIRO, Egypt AP _ In his foreign policy debut as Syria's president, Bashar Assad met Sunday with Egyptian President Hosni Mubarak in talks on Mideast peace and the escalating violence in the Palestinian territories. Assad, on his first trip abroad since becoming president in July, also dealt with the issue of Iraq in the talks with Mubarak, Egyptian Foreign Minister Amr Mussa said. The Syrian leader's visit came on a fourth day of clashes that have raged in the West Bank, Gaza Strip and Jerusalem, leaving more than two dozen Palestinians dead. ``There is complete agreement (between Mubarak and Assad) as to the seriousness of the situation and that the negative repercussions would adversely effect the peace process, which is in itself in a very precarious state,'' Moussa said. [...]

Figure 5.3: Part of an example document (news story) given in the Aquaint dataset with its annotations, constituting the gold standard for this document.

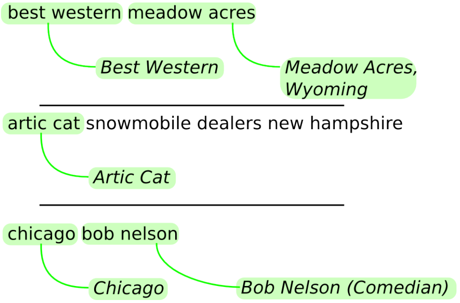


Very nice win today. Liked the way we played as a team. Go Hawks!

Are you a college kid who likes drinking, dressing up, and making irish immigrants roll in their graves? Then St. Patrick's Day is for you!

Had one of the best nights of my life at the premiere of Betty Blue Eyes at the Novello. All of Alan Bennett's wit, plus a wonder pig. Run!

Figure 5.4: Three example documents (twitter messages) given in the Meij dataset with their tags, constituting the gold standard for these documents.



best western meadow acres

artic cat snowmobile dealers new hampshire

chicago bob nelson

Figure 5.5: Three example documents (search engine queries) given in the KDD dataset with their annotations, constituting the gold standard for these documents.

Dataset	Native Problem	Docs	Ann (Tags)	Distinct Topics	Avg. Ann/-Doc	Avg. length	Ann. fre-quency
AIDA/CO-NLL	A2W	1393	27815	5591	20.0	1130	56^{-1}
MSNBC	A2W	20	658	279	32.9	3316	101^{-1}
AQUAINT	A2W	50	727	572	14.5	1415	98^{-1}
Meij	Rc2W	502	812	567	1.6	80	50^{-1}
KDD	A2W	1596	674	557	0.4	24	60^{-1}

Table 5.3: Column *Native Problem* indicates the problem for which the dataset offers a set of instances and their gold standard solutions. *Docs* is the number of documents (instances) contained in the dataset. *Ann (Tags)* is the number of overall annotations or tags in the dataset. *Distinct Topics* is the number of distinct topics that appear in the dataset. *Avg. Ann/Doc* is the average number of annotations or tags per document. *Avg. length* is the average length of a document in the dataset, in characters. *Ann. frequency* is the frequency of annotations or tags per character (a value of n^{-1} indicates an average of one annotation or tag every n characters).

Some datasets include annotations to concepts that no longer exist as pages in the current version of Wikipedia. This may happen if a page have been deleted or if its name was changed, and the dataset refers to an older version of Wikipedia. In the implementation of the benchmarking system, annotations with non-existing concepts are discarded.

Table 5.3 gives figures about the datasets. These figures help understanding the nature of the datasets: AIDA/CO-NLL, MSNBC and AQUAINT address about the same kind of documents (news stories written in good english and good punctuation), the first being the most rich and complete, the others containing longer documents (with less annotations). Meij is focused on twitter messages (short text often with abbreviations and poor punctuation) and is quite rich as well. KDD is focused on search engine queries (few keywords).

5.4 Comparing two systems for a given dataset

Figure 5.6 shows how two topic retrieval systems can be compared to each other and the datasets they can be compared against. Keeping an eye at the graph, the performance of systems S' and S'' in solving problem P can be compared if and only if a reverse-path exists from problem P to system S' and from problem P to system S'', and there is a dataset D representing a gold standard for problem $P'|P \propto P'$. In this case, the performance of systems S' and S'' can be measured with respect to their ability to solve the instances of problem P provided by dataset D. E.g.: The only way of comparing CMNS and Illinois

	Illinois	CMNS	Wikipedia Miner	AIDA
TagMe	Sa2W*, Sc2W, A2W, Rc2W, D2W, C2W	Rc2W, C2W	Sa2W*, Sc2W, A2W, Rc2W, D2W, C2W	Sa2W*, Sc2W, A2W, Rc2W, D2W, C2W
Illinois		Rc2W, C2W	Sa2W*, Sc2W, A2W, Rc2W, D2W, C2W	Sa2W*, Sc2W, A2W, Rc2W, D2W*, C2W
CMNS			Rc2W, C2W	Rc2W, C2W
Wikipedia Miner				Sa2W*, Sc2W, A2W, Rc2W, D2W, C2W

Table 5.4: Comparability of the topic-retrieval systems for each problem. Problems addressed natively by both systems are marked with a *.

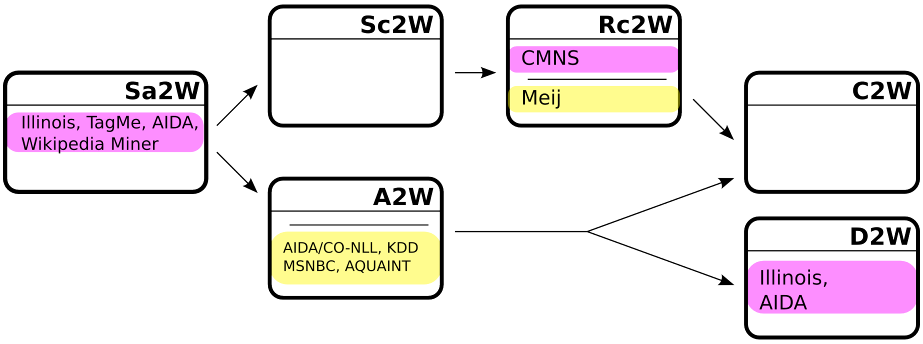


Figure 5.6: Framing of the systems and datasets in the problem hierarchy. Each system and each dataset is associated to the problem(s) it addresses natively. Systems are highlighted in pink, while datasets in yellow.

Wikifier is to compare their ability to solve problem C2W, and the comparison can be done with respect to their ability to solve the instances provided by any of the described datasets, since all of them can be adapted to give an instance and a gold standard for the C2W problem, while Illinois Wikifier and TagMe can be compared to each other for their ability to solve A2W, for example using dataset AQUAINT.

The comparability between systems derived from the graph in Figure 5.6 is presented in the matrix in Table 5.4.

5.5 Conclusions

In this chapter, we presented some of the publicly available systems, selected for their originality, that solve problems related to topic retrieval. For each systems, the problem(s) natively solved is reported, together with its main features. Since the reviewed systems are all based upon Wikipedia as a knowledge basis, we briefly presented some features of this online encyclopedia.

Some publicly available datasets, representing a set of instances and a gold standard for some problems, have been reviewed, together with their main features. These datasets can be used to perform experiments and check the ability of a system in solving the problems for the instances given by the dataset.

Chapter 6

Experimental Results

E quindi uscimmo a riveder le stelle.

– Dante Alighieri, La Divina Commedia, Canto XXXIV

Let's put all the pieces together: in this chapter, benchmarking results given by the framework illustrated in Chapter 4 are presented and commented. Experiments were run benchmarking the topic retrieval systems with the datasets presented in Chapter 5. The measures of performance and similarity are those proposed in Chapter 3. Problems presented in Chapter 2 and their reductions are the underlying basis for the analysis.

6.1 Setting up the experiments

6.1.1 Employed software and APIs

The benchmark has been done using the most up-to-date APIs or software made available by their authors. TagMe has been tested using the publicly available APIs [3] in September 2012; experiments on AIDA have been performed using the AIDA RMI Service updated on 30/07/2012 [1]; experiments on Wikipedia Miner have been performed querying the API publicly available at [4] in September 2012; Illinois Wikifier has been downloaded from [2] in August 2012 and run locally.

6.1.2 Performed experiments

Three experiments, each exploring the ability of the systems on certian kinds of datasets, have been set up. Configurations for the experiments are given in Table 6.1.

Configuration	Target	Systems	Datasets	Problem	Employed match relations
Experiment 1	News stories	All but CMNS	AIDA/CO-NLL, AQUAINT, MSNBC	A2W	M_s, M_w, M_e, M_c
Experiment 2	Tweets	All but CMNS	Meij	C2W	M_t
Experiment 3	Queries	All but CMNS	KDD	A2W	M_c

Table 6.1: Configuration for the performed experiments. Columns *Systems* and *Datasets* report respectively the systems that have been benchmarked in the experiment and the datasets on which the systems are run. Column *Problem* gives the problem for which the experiments are performed, that is the most general problem that reduces to all the problems natively solved by the systems *and* to the problem for which the dataset gives a set of instances and their gold standard.

Experiment 1: News

In Experiment 1, the correctness evaluation is focused on the A2W problem, the most general problem that all annotators (except CMNS) can solve and for which most datasets are available.

All tested systems (Illinois Wikifier, TagMe, AIDA, Wikipedia Miner) natively solve the Sa2W problem, and hence their output must be adapted to the A2W problem, for which datasets AIDA/CO-NLL, AQUAINT and MSNBC give a gold standard. As described in Table 2.3, the reduction A2W $\propto$ Sa2W introduces a variable: the threshold on the score. By consequence, the experiments are performed for values of the threshold ranging from 0 to 1.

Experiment 2: Tweets

Illinois Wikifier, TagMe, AIDA and Wikipedia Miner were also tested for their ability to deal with short twitter messages, given by dataset Meij. Unfortunately, CMNS could not be tested because it has not yet been published.

Since Meij is a C2W dataset, the system output must be adapted using reduction C2W $\propto$ A2W $\propto$ Sa2W (that is, put a threshold on the scores and take only the annotations with a higher score, then discard the mentions and the scores, leaving only a set of distinct concepts).

It is important to note that only TagMe has been built with the aim of annotating short text fragments, while Illinois Wikifier, AIDA and Wikipedia Miner target documents written in correct english. Still, this test checks the flexibility of these systems.

Experiment 3: Search engine queries (a preliminary study)

KDD dataset containing search engine queries is for problem A2W, and can be given as input to Illinois Wikifier, TagMe, AIDA and Wikipedia Miner. Nonetheless, in this case, we are not interested in computing annotations, but only in concepts. That's why the only match function adopted is the *Concept annotation match* defined in Definition 8.

In this case, none of the systems were designed to address this kind of documents, made only of keywords. To work on this dataset, major modifications should be applied to the reviewed systems. A study more focused on this task is presented in [19]. For systems are used as they come, the results are extremely poor, though showing some interesting results about their flexibility.

6.2 Results for Experiment 1: news

6.2.1 Finding the annotations

Note that each of the Sa2W annotators give a different meaning to the score, so it does not make sense to compare the performance of two annotators for the same threshold. The question that can instead be answered is: If an annotator knew its optimal threshold on the score, which annotator would perform best?

To answer this question, the metrics presented in Chapter 3 have been measured. The interesting data is the maximum value of $F1_{micro}(R_t, G, M_w)$ reached varying the threshold t, where M_w is the *Weak annotation match*, G is the gold standard, R_t is the result of the system adapted as an output of the A2W problem using t as threshold. This measure for the tagging systems is sketched out for each dataset by charts in Figures 6.1, 6.2 and 6.3.

The most significant results are those obtained with the AIDA/CO-NLL dataset, since it is the largest with 1.393 documents and 27.815 annotations. The dataset has an average document length of 1130 characters, as described in Table 5.3.

Maximizing micro-F1

Results of metrics based on *Weak annotation match* are reported in Table 6.2 for all datasets. Results for each system and each dataset are given for the threshold t^* that maximize the $F1_{micro}(R_t, G, M_w)$, where R_t is the result adapted from Sa2W to A2W keeping only annotations with a score higher than t, G is the gold standard given by the dataset, and M_w is the *Weak annotation match.* t^*

is obtained varying the threshold $t \in [0,1]$. In other words, we have

$$t^* = \arg\max_{t\in[0,1]} F1_{micro}(R_t, G, M_w)$$

Results based on the *Strong annotation match*, thus for the thresholds that maximize the $F1_{micro}(R_t, G, M_s)$ are reported in Table 6.3 only for the AIDA/CO-NLL dataset, mainly to show the difference between the two match relations (results over other datasets are coherent). Metrics based on the *Strong annotation match* give slightly lower results, especially for annotators such as TagMe, Illinois Wikifier and Wikipedia Miner, that seem to return a significant set of annotations with mentions that don't coincide with those in the dataset, even though overlapping with them.

The best annotator for AIDA/CO-NLL in terms of micro-F1 is TagMe, followed by Illinois Wikifier, Wikipedia Miner, and then AIDA. For AIDA, the best algorithm is clearly CocktailParty, followed by Local and PriorityOnly. Note that AIDA have a high precision and a low recall - this is probably caused by its poor ability to recognize the mentions (see Section 6.2.2): few mentions are recognized, probably the clearest ones, but the concept associated to them is mostly correct.

Datasets AQUAINT and MSNBC are less meaningful, but still give an idea of the performance for longer documents. TagMe gives the best results. For AQUAINT (avg. length: 1415 characters), Wikipedia Miner reaches the second position followed by Illinois Wikifier and then AIDA. For MSNBC, the dataset with longest documents (avg. length: 3316 characters), the second most performing is AIDA with the Local algorithm, followed by Illinois Wikifier and Wikipedia Miner.

Precision, recall and the scoring function

Charts in Figures 6.1, 6.4 and 6.5 respectively show the $F1_{micro}(R_t, G, M_w)$, $P_{micro}(R_t, G, M_w)$ and $R_{micro}(R_t, G, M_w)$ measures obtained testing the annotators against the AIDA/CO-NLL dataset, for $t \in [0,1]$.

As expected, for all the annotators, for values of $t \simeq 1$ the F1 drops, since the recall drops to 0 (too many correct annotations are discarded increasing the number of false negatives), as Figure 6.5 clearly shows.

It is less straightforward to interpret the behavior of the precision shown in Figure 6.4. For Wikipedia Miner and AIDA-PriorityOnly, increasing the threshold, the precision grows as expected, while the other annotators, for a certain threshold, experience a counter-intuitive loss of precision. First of all, it must be taken into account that the number of positives for such high thresholds are very small, thus the precision is heavily affected by a small number of false

Dataset	Annotator	Best Threshold	$F1_{micro}$	P_{micro}	R_{micro}	tp	fp	fn
MSNBC (len 3316)	TagMe 2	0.172	**0.511**	0.507	**0.516**	335	326	314
	Illinois Wikifier	0.469	0.408	0.338	**0.515**	335	656	315
	AIDA-Local	0.000	0.450	**0.767**	0.318	207	63	443
	AIDA-CocktailParty	0.000	0.441	0.752	0.312	203	67	447
	AIDA-PriorityOnly	0.000	0.359	0.611	0.254	165	105	485
	Wikipedia Miner	0.672	0.191	0.191	0.191	124	526	525
AQUAINT (len 1415)	TagMe 2	0.172	**0.506**	**0.471**	**0.547**	398	447	329
	Illinois Wikifier	0.516	0.341	0.285	0.425	309	776	418
	AIDA-Local	0.000	0.223	0.380	0.158	115	188	612
	AIDA-CocktailParty	0.000	0.216	0.366	0.153	111	192	616
	AIDA-PriorityOnly	0.000	0.219	0.373	0.155	113	190	614
	Wikipedia Miner	0.531	0.469	0.365	0.657	478	833	249
AIDA/CO-NLL (len 1130)	TagMe 2	0.219	**0.552**	0.587	0.520	14475	10172	13338
	Illinois Wikifier	0.453	0.544	0.492	**0.607**	16897	17450	10922
	AIDA-Local	0.000	0.469	0.725	0.346	9624	3644	18191
	AIDA-CocktailParty	0.000	0.470	**0.728**	0.347	9662	3606	18153
	AIDA-PriorityOnly	0.000	0.406	0.629	0.300	8343	4927	19472
	Wikipedia Miner	0.594	0.520	0.495	0.547	15227	15537	12587

Table 6.2: The main results about correctness is shown in this table. Column $F1_{micro}$ indicates the maximum micro-F1 measure computed as $F1_{micro}(R_{t^*}, G, M_w)$, where t^* is the threshold that maximizes the micro-F1. Column *Best Threshold* indicates t^*. Columns P_{micro} and R_{micro} indicate the micro-precision and micro-recall for the same threshold t^*. Columns *tp*, *fp*, *fn* respectively show the overall true positives, false positives and false negatives for the results of the dataset, still for threshold t^*.

Dataset	Annotator	Best Threshold	$F1_{micro}$	P_{micro}	R_{micro}	tp	fp	fn
AIDA/CO-NLL (len 1130)	TagMe 2	0.219	**0.536**	0.562	0.512	14243	11115	13572
	Illinois Wikifier	0.469	0.499	0.423	**0.610**	16956	23150	10859
	AIDA-Local	0.000	0.467	0.723	0.345	9595	3673	18220
	AIDA-CocktailParty	0.000	0.469	**0.726**	0.346	9632	3636	18183
	AIDA-PriorityOnly	0.000	0.405	0.627	0.299	8316	4954	19499
	Wikipedia Miner	0.594	0.474	0.413	0.556	15472	22034	12343

Table 6.3: Results based on *Strong annotation match* (Only for AIDA/CONLL dataset). Column $F1_{micro}$ indicates the maximum micro-F1 measure computed as $F1_{micro}(R_{t^*}, G, M_s)$. For a description of the other columns, see Table 6.2.

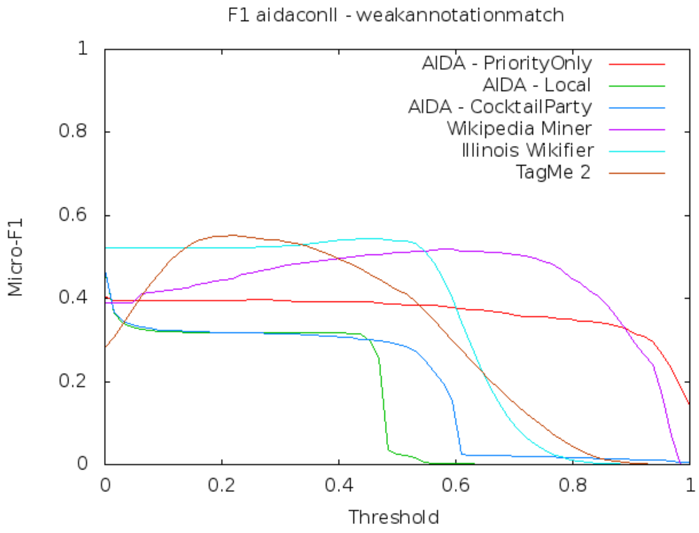


Figure 6.1: Micro-F1 measure for Sa2W systems on dataset AIDA/CO-NLL, for threshold $t \in [0, 1]$.

positives. Anyhow, this behavior demonstrates that a certain number of wrong annotations have a high score. Note that this behavior does not have a big influence on the F1 measure, since for these thresholds the recall is very close to 0, and so is the F1.

The behaviour of the precision and recall in Figures 6.4 and 6.5 give an evidence of the performance of an important feature of an annotation system. As said in the introduction of Chapter 3, one factor which the performance of a system depends upon how good the system is in assigning the score to the annotations. This issue should be inspected more in-depth, but some assumptions can be made: Let $p(a)$ be the probability that annotation a is correct, and let $s(a)$ be the scoring function. The ideal scoring function is $s^*(a) = p(a)$. A good scoring function should obviously assign higher scores to annotations that are more likely to be correct, thus it should be $s(a') \geq s(a'') \Leftrightarrow p(a') \geq p(a'')$. In this scenario, functions describing precision and recall would be respectively monotonically non-decreasing and monotonically non-increasing.

While for the recall chart in Figure 6.5 this property seems to be respected, the precision chart given in Figure 6.4 indicates a failure for the scoring functions of AIDA-Local, AIDA-CockailParty, Illinois Wikifier and TagMe, that, in some cases, assign a high score to wrong annotations, while the scoring functions of Wikipedia Miner and AIDA-PriorityOnly seem to perform better.

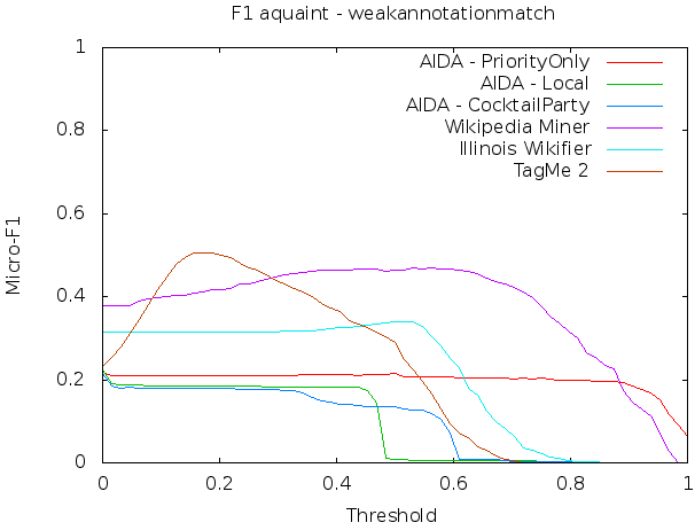


Figure 6.2: Micro-F1 measure for Sa2W systems on dataset AQUAINT, for threshold $t \in [0, 1]$.

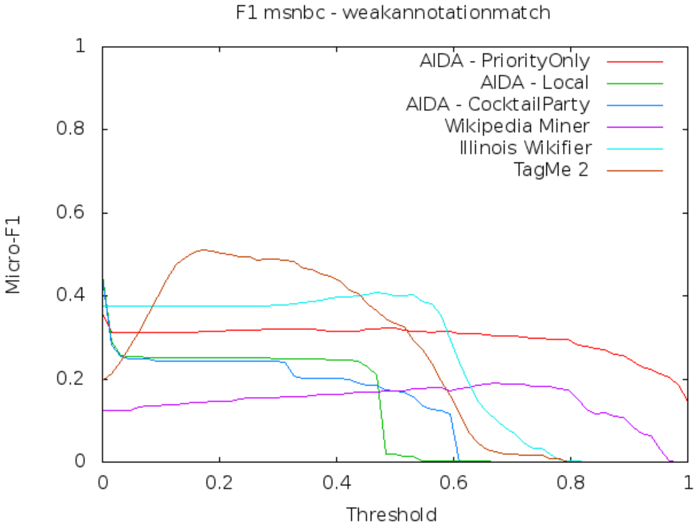


Figure 6.3: Micro-F1 measure for Sa2W systems on dataset MSNBC, for threshold $t \in [0, 1]$.

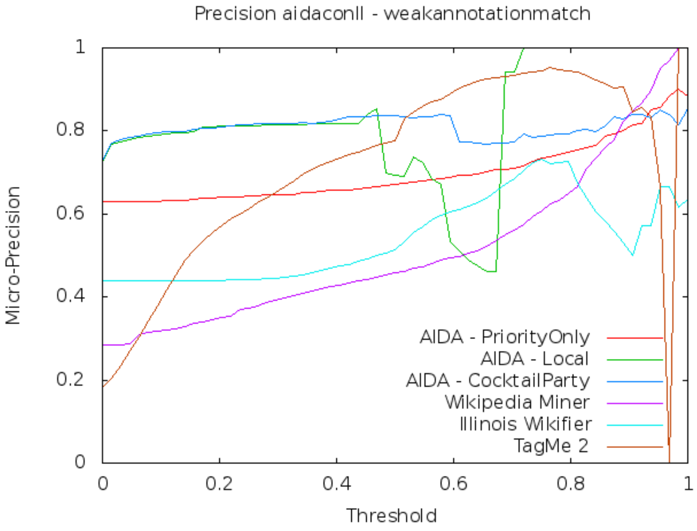


Figure 6.4: Micro-Precision measure for Sa2W systems on dataset AIDA/CO-NLL, for threshold $t \in [0, 1]$.

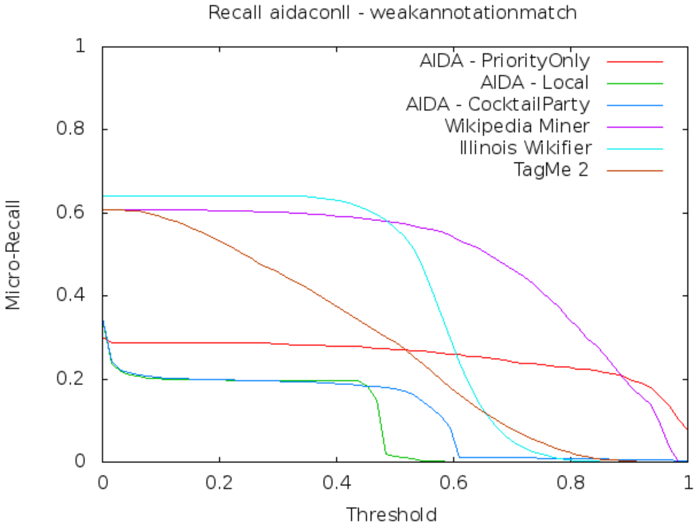


Figure 6.5: Micro-recall measure for Sa2W systems on dataset AIDA/CO-NLL, for threshold $t \in [0, 1]$.

Dataset	Annotator	Best Threshold	$F1_{micro}$	P_{micro}	R_{micro}	tp	fp	fn
AIDA/CO-NLL (len 1130)	TagMe 2	0.203	**0.710**	0.734	0.687	19011	6875	8657
	Illinois Wikifier	0.391	0.680	0.593	**0.798**	22263	15308	5634
	AIDA-Local	0.000	0.603	**0.933**	0.445	12381	887	15428
	AIDA-CocktailParty	0.000	0.603	**0.933**	0.445	12380	888	15429
	AIDA-PriorityOnly	0.000	0.603	**0.933**	0.445	12380	890	15429
	Wikipedia Miner	0.484	0.682	0.607	0.779	21610	13976	6139

Table 6.4: Results based on *Mention annotation match* (Only for AIDA/CONLL dataset). Column $F1_{micro}$ indicates the maximum micro-F1 measure computed as $F1_{micro}(R_{t^*}, G, M_m)$. For a description of the other columns, see Table 6.2.

6.2.2 Finding the mentions

As illustrated in Section 3.2, the sub-task of finding the mentions can be measured using the same micro-F1, micro-recall and micro-precision measures combined with the *Mention annotation match* relation $M = M_m$.

The value of $F1_{micro}(R_{t^*}, G, M_m)$ (for the threshold t^* that maximizes it) for the experiment ran on the AIDA/CO-NLL dataset is given for each system in Table 6.4 and basically shows the performance of the NER algorithms used by the annotation systems. Methods of the AIDA system use the same NER system (Stanford NER tagger) and obviously have the same result, that is the lowest one in terms of F1 but provides a very high precision. Illinois Wikifier (that uses the Illinois Named Entity Tagger) shows similar results to Wikipedia Miner. TagMe has a higher precision but a lower recall.

6.2.3 Finding the concepts

Keeping in mind Section 3.3, the sub-task of finding the concepts can be measured with the same metrics used in the other tests, combined with the *Concept annotation match* $M = M_c$.

The value of $F1_{micro}(R_{t^*}, G, M_c)$ (for the threshold t^* that maximizes it) for the experiment ran on the AIDA/CO-NLL dataset is given for each system in Table 6.5. TagMe outperforms the other annotators in terms of F1 and recall, while the highest precision is achieved by AIDA-CocktailParty, that has a very low recall. The CocktailParty algorithm seems to be the best in the choice of the AIDA algorithms, being slightly better than Local and significantly better that PriorityOnly.

Dataset	Annotator	Best Threshold	$F1_{micro}$	P_{micro}	R_{micro}	tp	fp	fn
AIDA/CO-NLL (len 1130)	TagMe 2	0.297	**0.642**	0.657	**0.629**	10480	5476	6190
	Illinois Wikifier	0.531	0.567	0.531	0.609	10149	8976	6521
	AIDA-Local	0.000	0.541	0.773	0.416	6932	2035	9738
	AIDA-CocktailParty	0.000	0.542	**0.793**	0.411	6859	1795	9811
	AIDA-PriorityOnly	0.000	0.473	0.655	0.370	6164	3249	10506
	Wikipedia Miner	0.594	0.532	0.489	0.582	9708	10132	6962

Table 6.5: Results based on *Concept annotation match* (Only for AIDA/CONLL dataset). Column $F1_{micro}$ indicates the maximum micro-F1 measure computed as $F1_{micro}(R_{t^*}, G, M_c)$. For a description of the other columns, see Table 6.2.

6.2.4 Output similarity

As described in Section 3.4, the outputs of two annotators can be compared to understand how similar they are. This is measured with the S_{micro} and S_{macro} measures given in Definition 10, combined with any match relation, depending on what we want to check the similarity of, as described in Subsection 3.4.3.

Similarity of the annotations

In Table 6.6 are reported values of the similarity measure $S_{macro}(A_{t'}, B_{t''}, M_w)$ defined in Definition 10, where A and B are the output of each pair of annotators and $A_{t'}$ and $B_{t''}$ are the output keeping only annotations with a score above the thresholds that maximize the micro-F1 (those reported in Table 6.2). These sets include true positives and false positives found by the tagging systems. As match relation M, the *Weak annotation match* M_w has been used. In other words, the $S_{macro}(A_{t'}, B_{t''}, M_w)$ measure given in Table 6.6 represents the similarity of the best output of two systems, including both true and false positives.

As said in Subsection 3.4.4, the fraction of true positives can be measured in detail. Table 6.7 is analogous to Table 6.6 except that the similarity

$$S_{macro}(T(A_{t'}, G, M_w), T(B_{t''}, G, M_w), M_w)$$

is not computed on the whole output, but is restricted to the true positives using the T function defined in Definition 11. Hence, this measure shows how many of the correct annotations two systems have in common.

As expected, the highest similarities are between the methods of the AIDA system, that all use the same NER algorithm. A similarity around 50% is also shown between Illinois Wikifier, Wikipedia Miner and TagMe, while these three

	TagMe 2	Illinois Wikifier	AIDA-Local	AIDA-CocktailParty	AIDA-PriorityOnly	Wikipedia Miner
TagMe 2	1.000	**0.407**	0.325	0.318	0.342	**0.533**
Illinois Wikifier		1.000	0.246	0.245	0.240	**0.545**
AIDA-Local			1.000	**0.921**	**0.821**	0.306
AIDA-CocktailParty				1.000	**0.808**	0.300
AIDA-PriorityOnly					1.000	0.317
Wikipedia Miner						1.000

Table 6.6: Similarity between the annotation systems, given as the S_{macro} measure on the best outputs for the AIDA/CONLL dataset.

	TagMe 2	Illinois Wikifier	AIDA-Local	AIDA-CocktailParty	AIDA-PriorityOnly	Wikipedia Miner
TagMe 2	1.000	**0.600**	0.444	0.435	0.463	**0.686**
Illinois Wikifier		1.000	0.430	0.427	0.431	**0.691**
AIDA-Local			1.000	**0.963**	**0.899**	0.471
AIDA-CocktailParty				1.000	**0.884**	0.464
AIDA-PriorityOnly					1.000	0.490
Wikipedia Miner						1.000

Table 6.7: Similarity between the annotation systems, given as the S_{macro} measure on the true positives of the best outputs for the AIDA/CONLL dataset.

systems are rather dissimilar to the AIDA system. Results in Table 6.7 enhances those in Table 6.6.

Similarity of the mentions

To check the similarity of the systems in recognizing the mentions, the same S measure has been computed using $M = M_m$, that is, the *Mention annotation match* is used as match relation. Here, the measure

$$S_{macro}(T(A_{t'}, G, M_m), T(B_{t''}, G, M_m), M_m)$$

represents how much of the correct output have overlapping annotations. $A_{t'}$ and $B_{t''}$ are selected using the thresholds reported in Table 6.4, that maximize the recognition of mentions. Moreover, $A_{t'}$ and $B_{t''}$ are restricted to the true

	TagMe 2	Illinois Wikifier	AIDA-Local	AIDA-CocktailParty	AIDA-PriorityOnly	Wikipedia Miner
TagMe 2	1.000	**0.700**	0.498	0.498	0.498	**0.754**
Illinois Wikifier		1.000	0.503	0.502	0.503	**0.819**
AIDA-Local			1.000	**0.999**	**1.000**	0.545
AIDA-CocktailParty				1.000	**0.999**	0.545
AIDA-PriorityOnly					1.000	0.545
Wikipedia Miner						1.000

Table 6.8: Similarity between the annotation systems in recognizing the mentions, given as the S_{macro} measure on the outputs that maximize the F1 for the recognition of mentions. Results for the AIDA/CONLL dataset.

positives using the T function defined in Definition 11.

Results are shown in Table 6.8. The mention recognition for the AIDA algorithm is the same, and thus show a similarity of 1. Wikipedia Miner shares around 75% of mentions with TagMe and Illinois Wikifier, while AIDA have only about 50% of mentions in common with the other systems. These data seem to indicate that a big part of the performance of a system in solving problem Sa2W and A2W is determined by the way mentions are recognized in the text.

Similarity of the concepts

The similarity for the concepts,

$$S_{macro}(T(A_{t'}, G, M_m), T(B_{t''}, G, M_m), M_m)$$

is reported in Table 6.9. $A_{t'}$ and $B_{t''}$ are selected using the thresholds reported in Table 6.4, that maximize the recognition of concepts. Moreover, $A_{t'}$ and $B_{t''}$ are restricted to the true positives using the T function defined in Definition 11. Hence, values reported in Table 6.9 show how many of the correct concepts are in common between two systems.

The similarity between the AIDA algorithms is still the highest, and all similarities are above 50%, indicating that more than half of the retrieved concepts are in common. A rather high similarity is shown between Wikipedia Miner, TagMe and Illinois Wikifier.

	TagMe 2	Illinois Wikifier	AIDA-Local	AIDA-CocktailParty	AIDA-PriorityOnly	Wikipedia Miner
TagMe 2	1.000	**0.675**	0.526	0.525	0.525	**0.750**
Illinois Wikifier		1.000	0.541	0.542	0.534	**0.720**
AIDA-Local			1.000	**0.968**	**0.918**	0.569
AIDA-CocktailParty				1.000	**0.904**	0.564
AIDA-PriorityOnly					1.000	0.580
Wikipedia Miner						1.000

Table 6.9: Similarity between the annotation systems in recognizing the concepts, given as the S measure on the outputs that maximize the F1 for the recognition of concepts. Results for the AIDA/CONLL dataset.

6.3 Results for Experiment 2: tweets

The main difference between news stories and tweets is their size and the different register. Poor verbosity, many abbreviations and bad grammars make it hard for the NER systems to recognize the mentions and find the associated concepts. Hence, results on microblogging posts are lower than those on news stories.

6.3.1 Finding the concepts

How do the reviewed systems deal with short microblogging messages? Values in Table 6.10 report the measures for the threshold t^* that maximizes the $F1_{micro}(R_{t^*}, G, M_c)$. TagMe and Wikipedia Miner prove to perform well with the highest micro-F1 and recall. Illinois Wikifier give lower results, especially in terms of recall, while AIDA, in all its variants, has a poor performance, though having the highest precision. This is probably due to the lack of mention recognition.

6.3.2 Output similarity

Table 6.11 gives the similarity of the output for the system as the

$$S_{macro}(T(A_{t'}, G, M_t), T(B_{t''}, G, M_t), M_t)$$

measure. $A_{t'}$ and $B_{t''}$ are selected with the thresholds t' and t'' given in Table 6.10, and restricted to the true positives using the T function defined in Definition 11. Hence, figures in Table 6.11 give the amount of common correct

Dataset	Annotator	Best Threshold	$F1_{micro}$	P_{micro}	R_{micro}	tp	fp	fn
Meij (len 80)	TagMe 2	0.125	**0.481**	0.473	**0.489**	397	443	415
	Illinois Wikifier	0.406	0.417	0.510	0.353	287	276	525
	AIDA-Local	0.000	0.243	0.529	0.158	128	114	684
	AIDA-CocktailParty	0.000	0.243	0.529	0.158	128	114	684
	AIDA-PriorityOnly	0.281	0.277	**0.667**	0.175	142	71	670
	Wikipedia Miner	0.297	0.477	0.487	0.467	379	399	433

Table 6.10: Results based on *Strong tag match* for the Meij dataset. Column $F1_{micro}$ indicates the maximum micro-F1 measure computed as $F1_{micro}(R_{t^*}, G, M_t)$. For a description of the other columns, see Table 6.2.

	TagMe 2	Illinois Wikifier	AIDA-Local	AIDA-CocktailParty	AIDA-PriorityOnly	Wikipedia Miner
TagMe 2	1.000	**0.747**	0.631	0.635	0.632	**0.809**
Illinois Wikifier		1.000	0.705	0.709	0.716	**0.790**
AIDA-Local			1.000	**0.995**	**0.955**	0.651
AIDA-CocktailParty				1.000	**0.959**	0.653
AIDA-PriorityOnly					1.000	0.654
Wikipedia Miner						1.000

Table 6.11: Similarity between the tagging systems in recognizing the mentions, given as the S measure on the outputs that maximize the F1 for the recognition of mentions. Results for the Meij dataset.

concepts found by the annotators on the Meij dataset.

There is a significant amount of common concepts between all the annotators. AIDA-Local and AIDA-CocktailParty find almost the same set of concepts, very similar to the output of AIDA-CocktailParty. There is also a high similarity between TagMe, Illinois Wikifier and Wikipedia Miner.

6.4 Results for Experiment 3: queries

Search engine queries have little in common with natural language texts: made of keywords, they have no grammar, no punctuation and no structure at all, since they are usually meant to follow the bag-of-words model. Reviewed annotation systems were not built to process such kind of documents, therefore, the preliminary study that can be done give odd results.

Dataset	Annotator	Best Threshold	$F1_{micro}$	P_{micro}	R_{micro}	tp	fp	fn
KDD (len 24)	TagMe 2	0.250	**0.343**	**0.281**	0.439	296	757	378
	Illinois Wikifier	0.484	0.069	0.058	0.083	56	902	618
	AIDA-Local	0.453	0.003	0.200	0.001	1	4	673
	AIDA-CocktailParty	0.266	0.003	0.167	0.001	1	5	673
	AIDA-PriorityOnly	0.953	0.003	0.200	0.001	1	4	673
	Wikipedia Miner	0.313	0.222	0.148	**0.444**	299	1726	375

Table 6.12: Results based on *Concept annotation match* for the KDD dataset. Column $F1_{micro}$ indicates the maximum micro-F1 measure computed as $F1_{micro}(R_{t^*}, G, M_t)$. For a description of the other columns, see Table 6.2.

6.4.1 Finding the concepts

Values in Table 6.12 report the measures for the threshold t^* that maximizes the $F1_{micro(R_{t^*},G,M_c)}$. Even without any adaptation for the peculiar task, TagMe and Wikipedia Miner give surprisingly good results, proving their flexibility and their capacity to adapt to tasks of different nature. Illinois Wikifier gives visibly lower results. AIDA finds almost no concepts.

6.5 Experiments about runtime

Another important feature of the annotation systems is their performance in terms of the time needed to find the annotations. Such measure is dependent on the input text, the system, and the problem natively solved by the system. It does also depend on the problem the result is adapted to, but the time to perform the adaptation is negligible and dependent on the framework implementation, and thus is not included in the measures. Since in the previous experiments, all tested systems solve Sa2W, a comparable runtime can be given for each system run over each dataset.

It is important to point out that it's hard to have an accurate measure of the runtime, since the systems run on different servers, some faster than others, and in some cases the network latency affects the measure. The aim was to record the duration of the whole process, which –though systems were queried as black boxes– supposedly includes the tokenization of the input text; the recognition of mentions; the mapping from each mention to a set of Wikipedia concepts the mention may refer to; the choice of the right concept (disambiguation); the assignment of a score to the annotation.

For TagMe and Illinois Wikifier, which run locally on the same computer, the

Dataset	Annotator	Average Time
MSNBC (len 3316)	Illinois Wikifier	5651ms
	TagMe 2	1897ms
	AIDA-Local	19726ms
	AIDA-CocktailParty	45434ms
	AIDA-PriorityOnly	685ms
	Wikipedia Miner	1960ms
AIDA/CO-NLL (len 1130)	Illinois Wikifier	3157ms
	TagMe 2	748ms
	AIDA-Local	12525ms
	AIDA-CocktailParty	31190ms
	AIDA-PriorityOnly	320ms
	Wikipedia Miner	1021ms
AQUAINT (len 1415)	Illinois Wikifier	2657ms
	TagMe 2	878ms
	AIDA-Local	10173ms
	AIDA-CocktailParty	23246ms
	AIDA-PriorityOnly	329ms
	Wikipedia Miner	1058ms
KDD (len 24)	TagMe 2	11ms
	Illinois Wikifier	106ms
	AIDA-Local	128ms
	AIDA-CocktailParty	68ms
	AIDA-PriorityOnly	15ms
	Wikipedia Miner	49ms
Meij (len 80)	TagMe 2	152ms
	Illinois Wikifier	2267ms
	AIDA-Local	1402ms
	AIDA-CocktailParty	1307ms
	AIDA-PriorityOnly	31ms
	Wikipedia Miner	66ms

Table 6.13: The average time needed by the annotation systems to annotate the documents contained in the datasets. In the first column, the average length of the documents contained in the dataset is reported.

runtime has been recorded as the difference between the system time after and before the library call; For AIDA, the time has been recorded as that returned by the method `getOverallRuntime()` of the RMI Service, and thus does not include network latency; For Wikipedia Miner, that is accessed through a web service, the time has been recorded as the difference between the system time after and before the query, and the timing has been calibrated subtracting the time needed to process an empty query.

Despite the heterogeneity and roughness of the methods used to record the timings, the figures differ enough to give a meaningful idea of how fast the annotation systems work. Data for each system and dataset is given in Table 6.13.

Results show that TagMe and Wikipedia Miner have similar runtime per-

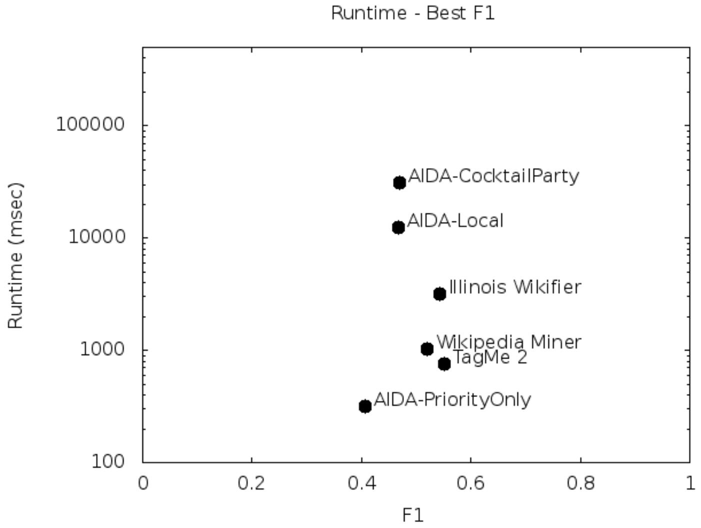


Figure 6.6: Average runtime (in log-scale) for dataset AIDA/CO-NLL and best achieved F1 measures (metrics based on *Weak annotation match*).

formance, suggesting their deployment in large-scale data processing. Illinois Wikifier is about 3 times slower. AIDA with PriorityOnly algorithm is the fastest one (this is due to the trivial disambiguation process) while the other methods, Local and CocktailParty are way slower, with the first being about 10 times and the latter being 30 times slower than TagMe and Wikipedia Miner. For all of them, the time needed to find the annotations looks linear in the size of the text – which is good. Chart in Figure 6.6 compares the runtime and the achieved best F1 measure for the systems when performing tests on the AIDA/CO-NLL dataset.

6.6 Conclusions

This chapter have reported the results given by the benchmarking framework applied on the systems and datasets presented in the previous chapters. Three experiments were run, each focusing on a different kinds of datasets: the news stories, the twitter messages and the search engine queries. Commenting the results, we highlighted some features and some weaknesses of the systems.

All systems can be improved but, at least for news and tweets, results are encouraging. TagMe seems to outperform the other systems in each of the three experiments, though other systems show interesting results. AIDA seems to have a main weakness in the recognition of mentions.

For the queries, all systems clearly need a major adaptation. TagMe and Wikipedia Miner incredibly give meaningful results even on this kind of dataset. Despite not being satisfactory, results encourage the application of these systems on search engine queries.

The similarity of the system outputs have also been measured, showing a rather high similarity between TagMe, Illinois Wikifier and Wikipedia Miner, while AIDA seem to find rather distinct annotations.

The measured runtime show the fundamental property that the time needed to annotate a document is linear in the size of the document. This makes it possible, especially for faster systems like TagMe, Wikipedia Miner and Illinois Wikifier, to apply them to large-scale datasets.

Chapter 7

Future Developments

Gutta cavat lapidem non vi, sed saepe cadendo

– *Latin proverb.*

This thesis gives a contribution to a line of research that is moving its first steps of development, and forecasting its future advancement is a hard job of imagination. Though, some lines of expansion can be sketched out. First of all, new algorithms and systems can be developed to enhance the performance in solving the topic-retrieval problems. Moreover, the benchmark needs a clear documentation provided in Javadoc that has not yet been done, and will probably need to be maintained reflecting the state of the art measures, even fixing the bugs its code certainly hides. On the theoretical side, the measures presented in Chapter 3 can be expanded to inspect certain aspects of a system performance.

7.1 Definition of new problems

The problems defined in Chapter 2 are not the only problems related to topic retrieval. New problems can be formulated, exploring new horizons for this research area.

7.1.1 Assigning a relevance to a topic

Most of the systems presented in Chapter 5 assign to the annotations a likelihood score, expressing the probability that the annotation is correct, but none of them deal with the task of finding the relevance of the topics, i.e. how important the topic is to describe the content of the input text. This information would be crucial to assist text categorization, text clustering and the document retrieval.

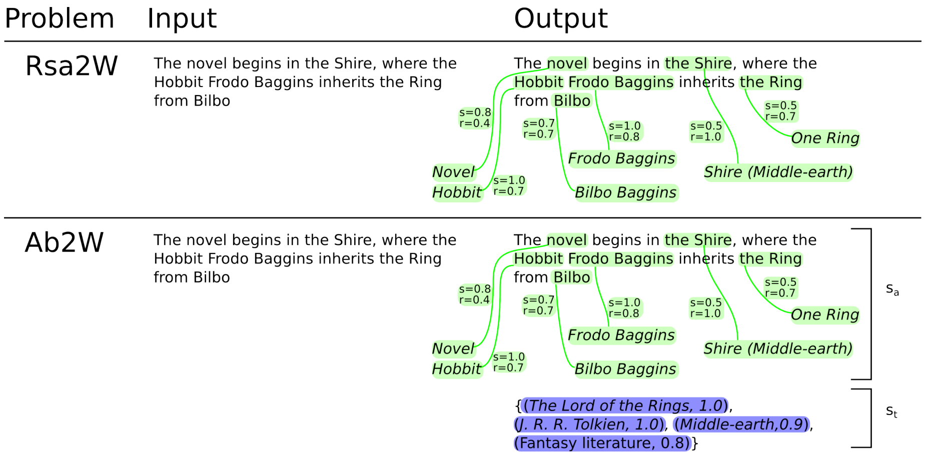

Problem
Input
Output
Rsa2W
The novel begins in the Shire, where the Hobbit Frodo Baggins inherits the Ring from Bilbo
The novel begins in the Shire, where the Hobbit Frodo Baggins inherits the Ring from Bilbo
s=0.8 r=0.4
s=0.7 r=0.7
s=1.0 r=0.8
s=0.5 r=1.0
s=0.5 r=0.7
s=1.0 r=0.7
Novel
Hobbit
Frodo Baggins
Bilbo Baggins
Shire (Middle-earth)
One Ring
Ab2W
The novel begins in the Shire, where the Hobbit Frodo Baggins inherits the Ring from Bilbo
The novel begins in the Shire, where the Hobbit Frodo Baggins inherits the Ring from Bilbo
s=0.8 r=0.4
s=0.7 r=0.7
s=1.0 r=0.8
s=0.5 r=1.0
s=0.5 r=0.7
s=1.0 r=0.7
Novel
Hobbit
Frodo Baggins
Bilbo Baggins
Shire (Middle-earth)
One Ring
sa
{(The Lord of the Rings, 1.0), (J. R. R. Tolkien, 1.0), (Middle-earth,0.9), (Fantasy literature, 0.8)}
st

Figure 7.1: Examples of instances of the Ab2W and RelA2W problems and their correct solution. Scored tags are highlighted in blue, scored annotations in green. Concepts are in italics.

Definition 12 *Relevance scored annotations to Wikipedia (Rsa2W)* is the problem of identifying the set of annotations in a text and assign them both a likelihood score s and a relevance score r indicating how relevant the topic is to describe the contents of the text. The solution consists of a set of tuples $\langle m, c, s, r\rangle$ where m is the mention (a pair (p, l) where p is the index of the character where the mention begin and l is the length of the mention), c is the concept, s is the likelihood score and r is a real number $r \in \mathbb{R}, r \in [0, 1]$ indicating the relevance of the concept to describe the topics of the input text. □

An example instance of this problem and its solution is given in Figure 7.1.

7.1.2 Finding relevant non-mentioned topics

Another natural expansion of the problems presented in Chapter 2 can be the definition of a new problem, Ab2W.

Definition 13 *Abstraction to Wikipedia (Ab2W)* is the problem of identifying the set of concepts that are mentioned in the input text, assigning them both a relevance score and a likelihood score, plus the concepts the text expresses, even though they are not mentioned in the text (i.e. they are implicitly expressed).

A solution of an instance of this problem is a pair (s_a, s_t) consisting of two sets: the first including scored annotations (for the concepts that are explicitly mentioned in the text), the latter including scored tags (for the concepts that are implicitly expressed by the text). □

An example instance of this problem and its solution is given in Figure 7.1.

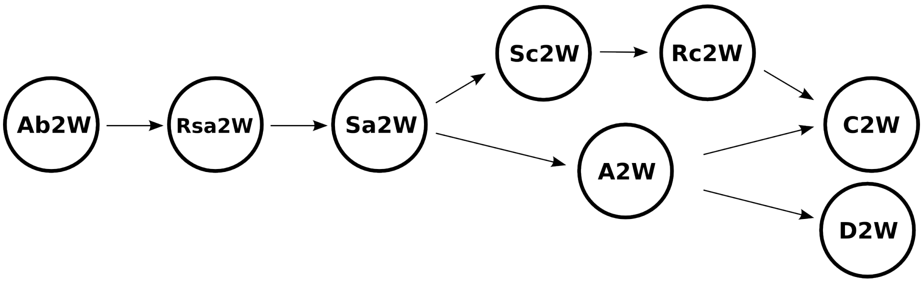


Figure 7.2: The new preordering of the problem reductions, including Ab2W and Rsa2W

7.1.3 Framing in the problem hierarchy

To properly expand the hierarchy of problems presented in Section 2.3, note that Sa2W $\propto$ Rsa2W $\propto$ Ab2W, since a solution to the Ab2W problem can be adapted in $O(()\,1)$ to a solution of the Rsa2W problem discarding the set s_t (containing scored tags of concepts not mentioned in the text) and keeping the set s_a (containing scored annotations of concepts mentioned in the text), while a solution to the Rsa2W problem can be adapted in $O(()\,n)$ to a solution of the Sa2W problem discarding, for each annotation $\langle m, c, s, r\rangle$, the relevance r, leaving just a set of scored annotations $\langle m, c, s\rangle$.

Expanding the preordering of the problems, Ab2W is located on the very left, since it is the hardest and most general, and all other problems reduce to it. The new hierarchy is given in the graph in Figure 7.2 and is described by the following chains:

$$C2W \propto Rc2W \propto Sc2W \propto Sa2W \propto Rsa2W \propto Ab2W$$
$$C2W \propto A2W \propto Sa2W \propto Rsa2W \propto Ab2W$$
$$D2W \propto A2W \propto Sa2W \propto Rsa2W \propto Ab2W$$

7.1.4 Metrics for the new problems

To measure the correctness of the relevance assigned by a system to an annotation, the concepts of the annotations could be ranked by their relevance score, and new metrics to measure the quality of the ranking should be defined, based on classical measures like MRR and MAP.

Metrics to measure the performance of a system solving Ab2W could be based on metrics defined in Definition 2 with $M = M_t$ for the set of tags s_t, and $M = M_w$ for the set of annotations s_a.

7.2 Chimera: mixing systems together

Some systems seem to perform better for some aspects, other may be better for others. Since to solve a problem P some systems share the same steps, implementing them differently, would it be possible to use an implementation for a step given by a system and the implementation for the next step of another system?

Let's consider a case study. Both TagMe, Illinois Wikifier and AIDA, to solve the A2W problem, basically follow these steps:

1. Find the mentions;
2. Associate a set of candidates to each mention;
3. Select a candidate for each mention (disambiguation).

The first step is done by a NER system. For Illinois Wikifier and AIDA, the implementation for problem A2W includes all three steps, while the implementation for D2W simply excludes the first step (mentions are given in the input, though there is no need to find them).

It would be interesting, for example, to perform step 1 using the set of mentions found by TagMe and perform steps 2 and 3 with Illinois Wikifier or AIDA. A way of doing this, given an instance I of problem A2W (a text document), is to run TagMe on I, convert the output of TagMe to an instance of the D2W problems (the new instance I' would be the same text provided by I with the addition of the mention found by TagMe), and solve I' with AIDA's or Illinois Wikifier's native D2W implementation.

The new system, mixing TagMe and AIDA or TagMe and Illinois Wikifier, would be able to natively solve Sa2W.

Chimera, a plugin for the benchmarking framework, could be developed in such direction.

7.3 Conclusions

In this last chapter, we sketched out the possible future lines of development. A new problem, Ab2W, has been defined. This problem is particularly meaningful for some applications like document clustering, involving the retrieval of topics that are not specifically mentioned in a text, though being highly relevant to describe the topics the text is about.

Another fundamental task, that helps all applications, is that of finding the relevance of an annotation, i.e. the importance of its topic for describing the contents of the document.

Chimera, an expansion of the benchmarking framework, aiming to mix systems features, has been outlined.

Appendices

Appendix A

Formulary

To facilitate the reading, in Table A.1 is reported a list of formulas defined in this thesis along with a brief description.

Formula	Defined in	Short description
$tp(r, g, M)$	Def. 1, p. 14	The set of *true positives* (correctly retrieved elements) for an instance.
$fp(r, g, M)$	Def. 1, p. 14	The set of *false positives* (incorrectly retrieved elements) for an instance.
$fn(r, g, M)$	Def. 1, p. 14	The set of *false negatives* (incorrectly non-retrieved elements) for an instance.
$tn(r, g, M)$	Def. 1, p. 14	The set of *true negatives* (correctly non-retrieved elements) for an instance.
$P(r, g, M)$	Def. 2, p. 14	*Precision* (fraction of the retrieved elements that are correct) for an instance.
$R(r, g, M)$	Def. 2, p. 14	*Recall* (fraction of the correct elements that were retrieved) for an instance.
$F1(r, g, M)$	Def. 2, p. 14	*F1* measure (mix of precision and recall) for an instance.
$P_{macro}(R, G, M)$	Def. 2, p. 14	*Average precision* for a set of instances.
$R_{macro}(R, G, M)$	Def. 2, p. 14	*Average recall* for a set of instances.
$F1_{macro}(R, G, M)$	Def. 2, p. 14	*Average F1* for a set of instances.
$P_{micro}(R, G, M)$	Def. 2, p. 14	*Overall precision*, not counting the belonging of an element to any particular instance.
$R_{micro}(R, G, M)$	Def. 2, p. 14	*Overall recall*, not counting the belonging of an element to any particular instance.
$F1_{micro}(R, G, M)$	Def. 2, p. 14	*Overall F1*, not counting the belonging of an element to any particular instance.
$d(p)$	Def. 4, p. 15	*Dereference* function: given a page, return its non-redirect concept.
$M_t(t_1, t_2)$	Def. 3, p. 15	*Strong tag match* binary relation: match occur if two tags have the same (dereferenced) concept.
$M_s(a_1, a_2)$	Def. 5, p. 16	*Strong annotation match* binary relation: match occur if two annotations have the same (dereferenced) concept and equal mention.
$M_w(a_1, a_2)$	Def. 6, p. 17	*Weak annotation match* binary relation: match occur if two annotations have the same (dereferenced) concept and overlapping mention.
$M_m(a_1, a_2)$	Def. 7, p. 18	*Mention annotation match* binary relation: match occur if two annotations have overlapping mentions.
$M_c(a_1, a_2)$	Def. 8, p. 19	*Concept annotation match* binary relation: match occur if two annotations have the same (dereferenced) concept.
$S'(a, b, M)$	Def. 9, p. 20	*Similarity* measure over two solutions of an instance.
$S_{macro}(A, B, M)$	Def. 10, p. 20	Average *similarity* for two lists of solutions.
$S_{micro}(A, B, M)$	Def. 10, p. 20	*Overall similarity* for two lists of solutions, counting matches independently on the solution they belong to.
$T(O, G, M)$	Def. 11, p. 21	Mapping of function *tp* over the elements of lists O, G.
$F(O, G, M)$	Def. 11, p. 21	Mapping of function fp over the elements of lists O, G.

Table A.1: Formulary reporting the formulas defined in the thesis.

Bibliography

[1] Aida rmi service. http://www.mpi-inf.mpg.de/yago-naga/aida/#rmiservice.

[2] Ccg - software: Illinois wikifier. http://cogcomp.cs.illinois.edu/page/software_view/33.

[3] Tagme - api guide. http://www.test.org/doe/.

[4] Wikipedia miner: Using the web services. http://wikipedia-miner.cms.waikato.ac.nz/wiki/Wiki.jsp?page=Using%20the%20web%20services.

[5] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 9–16, Trento, Italy, 2006.

[6] Soumen Chakrabarti, Kriti Puniyani, and Sujatha Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 717–726, New York, NY, USA, 2006. ACM.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Press, 2001.

[8] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *In Proc. 2007 Joint Conference on EMNLP and CNLL*, pages 708–716, 2007.

[9] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 277–285, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[10] Christiane Fellbaum. Wordnet. In Roberto Poli, Michael Healy, and Achilles Kameas, editors, *Theory and Applications of Ontology: Computer Applications*, pages 231–243. Springer Netherlands, 2010.

[11] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1625–1628, New York, NY, USA, 2010. ACM.

[12] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[13] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 765–774, New York, NY, USA, 2011. ACM.

[14] Xianpei Han and Jun Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 215–224, New York, NY, USA, 2009. ACM.

[15] Xianpei Han and Jun Zhao. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 50–59, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[16] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 229–232, New York, NY, USA, 2011. ACM.

[17] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Conference on Empirical Methods in Natural Language Processing, Edinburgh, Scotland, United Kingdom 2011*, pages 782–792, 2011.

[18] Jian Hu, Lujun Fang, Yang Cao, Hua-Jun Zeng, Hua Li, Qiang Yang, and Zheng Chen. Enhancing text clustering by leveraging wikipedia semantics. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 179–186, New York, NY, USA, 2008. ACM.

[19] Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 471–480, New York, NY, USA, 2009. ACM.

[20] Xiaohua Hu, Xiaodan Zhang, Caimei Lu, E. K. Park, and Xiaohua Zhou. Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 389–396, New York, NY, USA, 2009. ACM.

[21] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[22] Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. He says, she says: conflict and coordination in wikipedia. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 453–462, New York, NY, USA, 2007. ACM.

[23] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 457–466, New York, NY, USA, 2009. ACM.

[24] Douglas B. Lenat. Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38, November 1995.

[25] Jun Liu and Sudha Ram. Who does what: Collaboration patterns in the wikipedia and their impact on article quality. *ACM Trans. Manage. Inf. Syst.*, 2(2):11:1–11:23, jul 2011.

[26] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Sch&#252;tze. *Introduction to Information Retrieval.* Cambridge University Press, 1 edition, July 2008.

[27] Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference*

on Web search and data mining, WSDM '12, pages 563–572, New York, NY, USA, 2012. ACM.

[28] Rada Mihalcea. Using Wikipedia for Automatic Word Sense Disambiguation. In *North American Chapter of the Association for Computational Linguistics (NAACL 2007)*, 2007.

[29] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM.

[30] David Milne. *Applying Wikipedia to Interactive Information Retrieval.* PhD thesis, Department of Computer Science, The University of Waikato, 2010.

[31] David Milne and Ian H. Witten. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *AAAI 2008*.

[32] David Milne and Ian H. Witten. An open source toolkit for mining wikipedia. In *Special Issue of the Artificial Intelligence Journal on "Artificial Intelligence, Wikipedia and Semi-Structured Resources"*.

[33] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA, 2008. ACM.

[34] Elahe Rahimtoroghi and Azadeh Shakery. Wikipedia-based smoothing for enhancing text clustering. In *Proceedings of the 7th Asia conference on Information Retrieval Technology*, AIRS'11, pages 327–339, Berlin, Heidelberg, 2011. Springer-Verlag.

[35] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[36] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[37] Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. Piggyback: using search engines for robust cross-domain named entity recognition. In *Proceedings of the 49th Annual Meeting of the Association*

for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, pages 965–975, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[38] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513 – 523, 1988.

[39] Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. Topical clustering of search results. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 223–232, New York, NY, USA, 2012. ACM.

[40] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1419–1424. AAAI Press, 2006.

[41] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. AIDA: an online tool for accurate disambiguation of named entities in text and tables. *PVLDB*, 4(12):1450–1453, 2011.

[42] Yiping Zhou, Lan Nie, Omid Rouhani-Kalleh, Flavian Vasile, and Scott Gaffney. Resolving surface forms to wikipedia topics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1335–1343, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Acknowledgments...

I would like to thank my supervisor, Paolo Ferragina, whose brilliance and open mind stimulated the development of this work, as long as Daniele Vitale and Ugo Scaiella for their willingness in working together with me. Thanks to my co-reviewer Anna Bernasconi for her precious advice.

During the development of this thesis, I interacted with many people around the world, finding great and prompt cooperation. Thanks to Ian Witten (University of Waikato), Edgar Meij (University of Amsterdam), Mark Sammons and Lev-Arie Ratinov (University of Illinois at Urbana-Champaign), Edwin Lewis-Kelham (Max Planck Institute for Informatics).

I'd also like to thank the developers of the open-source software and services that assisted the work of this thesis, most notably `gnuplot`, `inkscape`, LaTeXand `Eclipse`. Special thanks to the Wikimedia Foundation, the Wikipedia project, and all its silent and industrious contributors, for having embarked the great task of letting anyone benefit from the free sharing of knowledge.

None of this would have been possible without the constant support (both scientific and humane) of those I have around in the everyday life: my family, my friends, my flat-mates, and all those I share the will to change the world with. Thank you for encouraging me in believing I am doing something worthy.

Thanks to the ice cream for containing sugar, to the penguins, kittens, wild boars, elves and orcs, polynomials, mangoes, bicycles, and, above all, biscuits.