CoolNELLI™

your link to knowledge

Overview of Presentation

- Goals of the project
- Problems encountered & decisions made
- Architecture of CoolNELLI™
- Evaluation criteria
- Presentation per module
- Evaluation results
- Testrun

Goal

1. In any given text, find words that an (average) internet user might want to have more information about;

The MTV Video Music Awards were overshadowed by the steamy sight of president Bush kissing Britney Spears.

Pneumonia is rare in american caimans.

Goal

1. In any given text, find words that an (average) internet user might want to have more information about;

The MTV Video Music Awards were overshadowed by the steamy sight of president Bush kissing Britney Spears.

Pneumonia is rare in american caimans.

Goal

2. Link each word to a website with relevant information;

The MTV Video Music Awards were overshadowed by the steamy sight of president Bush kissing Britney Spears.

Pneumonia is rare in american caimans.

Goal

2. Link each word to a website with relevant information;

www.MTV.com
www.bush.com

www.britneyspears.com

www.en.wikipedia.org/wiki/Pneumonia

www.en.wikipedia.org/wiki/Pneumonia

The MTV Video Music Awards were overshadowed by the steamy sight of president Bush kissing Britney Spears.

Pneumonia is rare in american caimans.

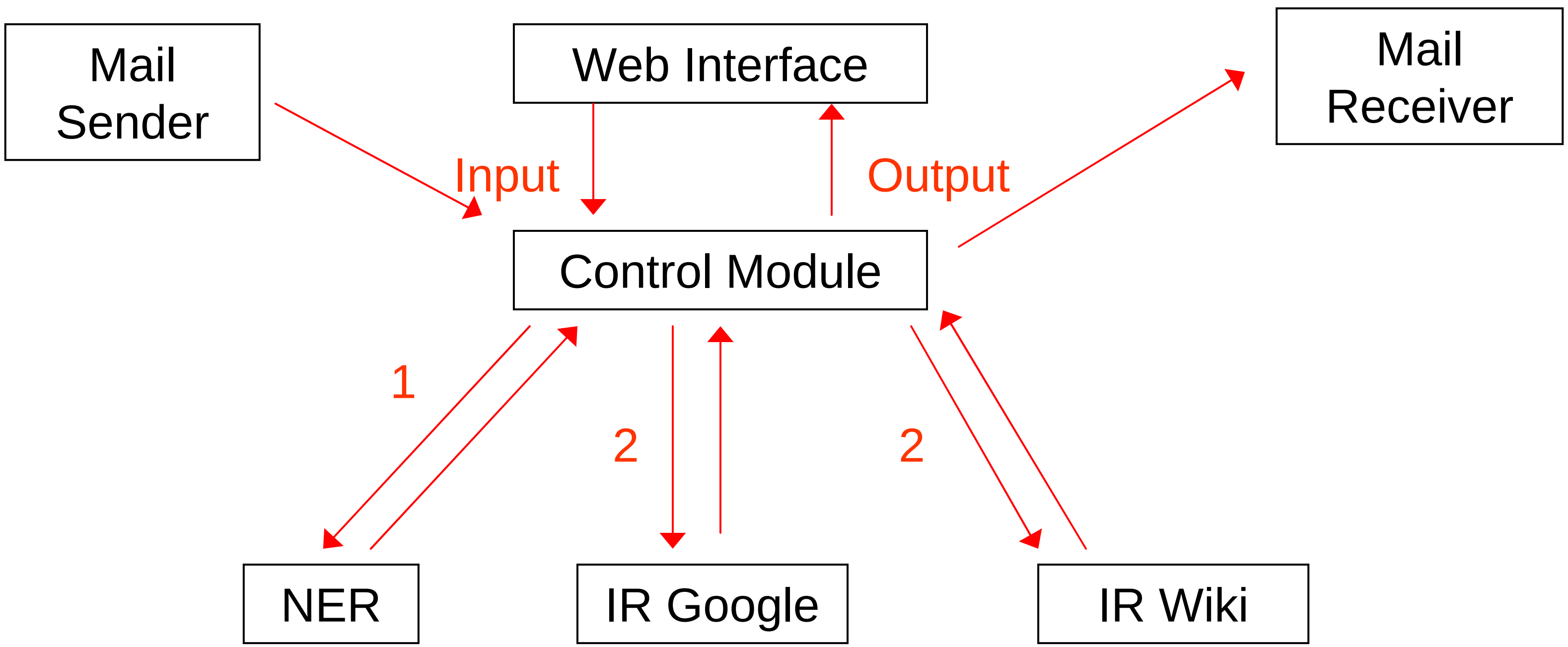
Problems

- Architecture
- Communication between modules:
  - How to mark Named Entities in text?
  - What information is needed for refining the search?
    - context
    - types

Solutions

- How to mark Named Entities in text?
  - XML schema
- What information is needed for refining the search?
  - Titles: president, chairman, doctor
  - Types: Location, Organisation

Architecture



Evaluation: Test Set

Manually, individually linked all words in a test text consisting of different genres:

- Newsreport
- MTV event announcement
- Emails
- Encyclopedia articles

Evaluation: Criteria

Compare with manually linked text on:

- Precision of links
- Recall of links
- Correct Wikipedia links (exact)
- Correct Google links (useful)

Evaluate:

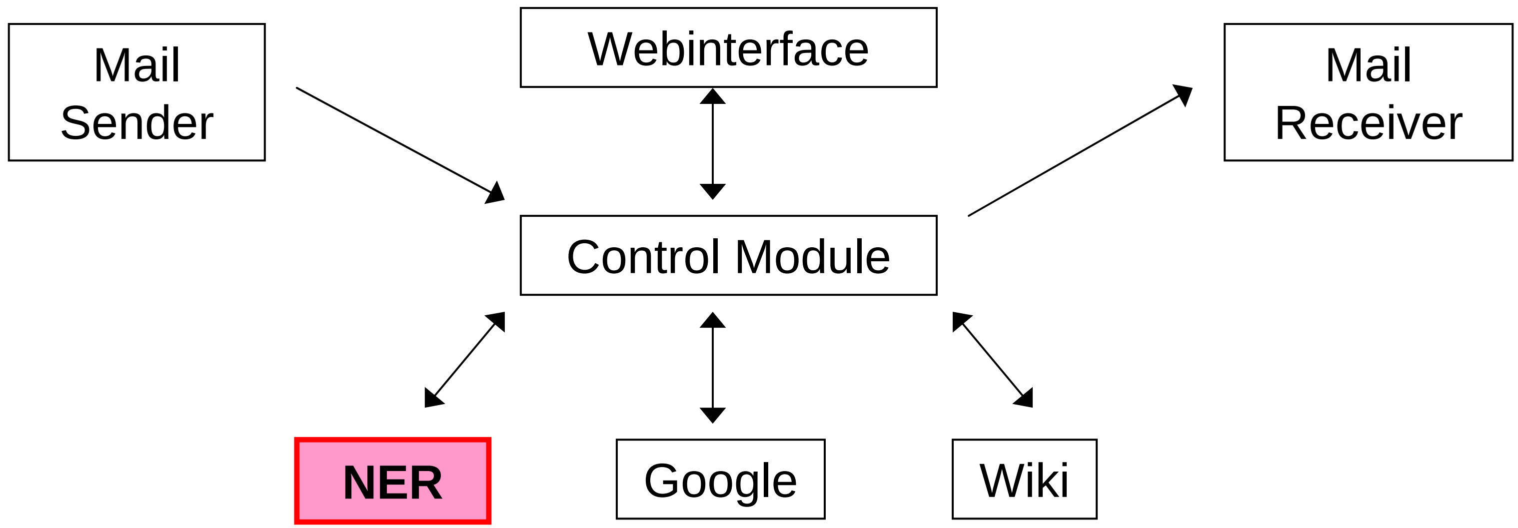
- speed
- user friendliness
- applications: email, html, plain text

Presentations per module

- Named Entity Recognition Module
- Information Retrieval Module for Google
- Information Retrieval Module for Wikipedia
- Interface / Control Module

NE Recognition

Ori Garin & Gustaaf Haan



Goal

- In any given text, find words that an (average) internet user might find useful to have more information about.
  - Find proper names: Bush
    - type: person
    - title: president
  - Find infrequent words:
    - Unknown proper names: Balkenende
    - Infrequent or unknown nouns: caiman

Problems Encountered

- How to identify a proper name?
- How to identify an infrequent word?
- How to get extra information on types and titles?

Solutions

- How to identify a proper name?
  - Parts Of Speech Tagger
    - Slow
    - Irrelevant information
    - Not very adequate
  - Capitalization
    - Hard
  - MINIPAR: built-in Named Entity Recognizer
- MINIPAR problems:
    - MINIPAR cannot process very large texts
    - technical problems

Solutions

- How to identify an infrequent word?
  - New York Times corpus: 1.400.000 words
    - Newspaper
    - Derivations are counted separately:
      - » Caiman
      - » Caimans
    - MINIPAR gives constituents
      - » No verbs: corpus too small

Solutions

- How to get extra information on types and titles?
  - MINIPAR: ‘location’,‘corpname’, ‘person’
    - Too much noise in ‘Person’:
      - Wall Street, Rocky Mountains, Shell
      - Tags: location and organisation, else: unknown
  - MINIPAR: ‘title’
    - Sometimes unclear whom the title applies to
      - Dependency trees

NER Module

1. Read text and split sentences, words
2. Split into sentences, assign offsets
3. Feed to MINIPAR, match offsets
4. Extract tags, categories, dependencies
5. Find NE chunks and relevant titles
6. Identify infrequent nouns
7. Output as XML

Extracting Sentences

- Sample input:

Kimani Jorsel Kawabanga indicated that Dr. Robert W.T.

Grant's favorite city is Boston.

- Output of Sentence Splitter:

<s> Kimani Jorsel Kawabanga indicated that Dr. Robert W. T. Grant's favorite city is Boston. </s>

MINIPAR output:

3 … **Kimani Jorsel Kawabanga** N 4 s

…

6 … **Dr.** ~ U **9** title (atts (sem (+**title**))))

…

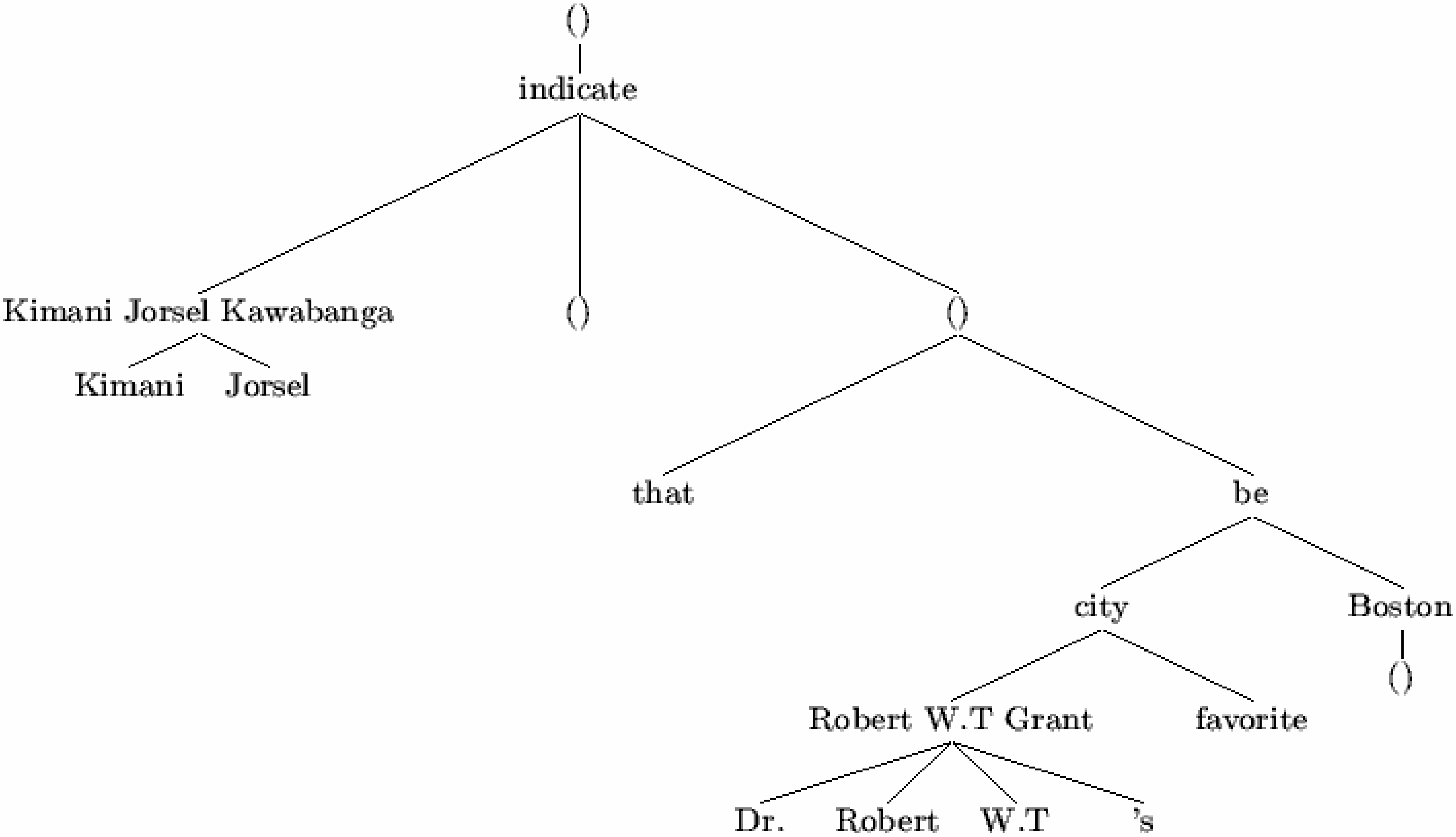
9 … **Robert W.T Grant** N 12 (atts (sem (+**person**))))

…

14 … **Boston** ~ N 13 (atts (sem (+city +**location**))))

…

Names and Dependencies



What Then?

- Extracting
- Possible tags
- Searching for titles in a limited scope
- MINIPAR chunks unknown capitalized words but doesn’t commit to their tags
- Other unknown/infrequent nouns
- The new-old distinction

Evaluation: Criteria

Compare with manually linked text on:

- Precision
- Recall
- Correctly underlined NE's
- Correct titles
- Correct types

Evaluation: Results

Compare with manually linked text on:

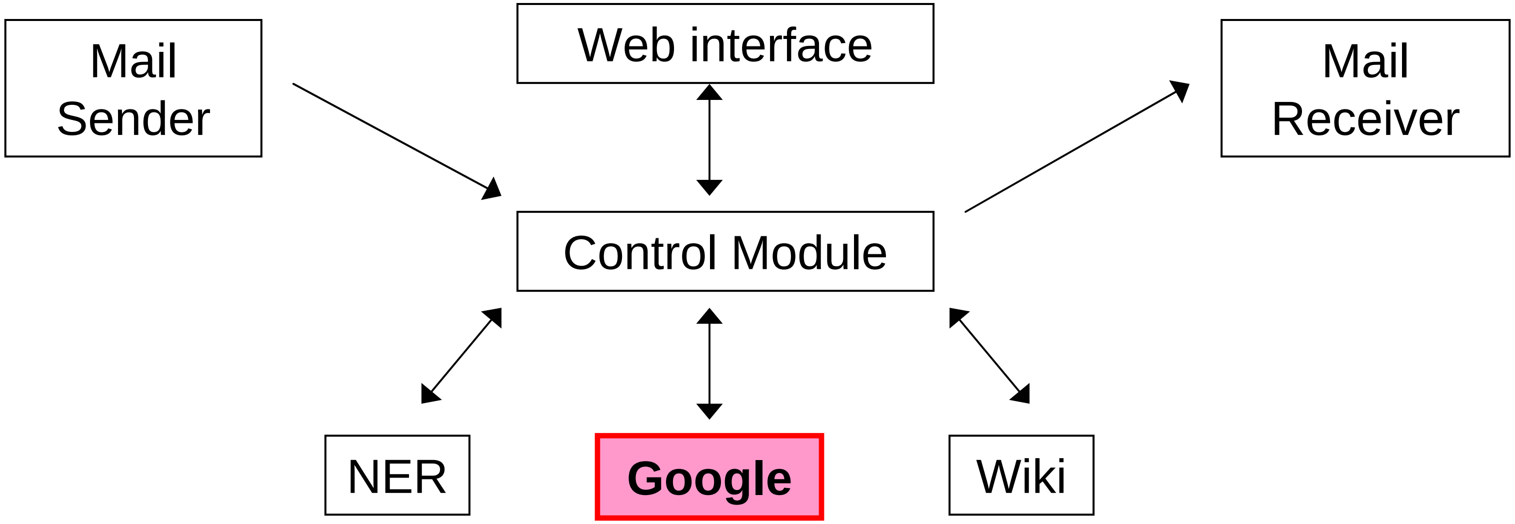
• Precision		65%
• Recall		89%
• Correctly underlined NE's		95%
• Correct titles	8%	100%
• Correct types	54%	100%

Further research

- Infrequent words:
  - derivations, verb inflections, could be handled by MINIPAR
  - a larger and more adequate corpus that contains words such as: ‘hello’, ‘damn’, ‘bro’, ‘funk’,
- Deal with definite descriptions and titles
  - “Gone With The Wind”
- Try to extract more context information

IR GOOGLE

Tim van Erven & Janneke van der Zwaan



Overview

- Task
- Approach
- Experiments
- Results
- Conclusion
- Future work

Task: introduction

- Suppose you want to find background information on Colin Powell
- Search the web (e.g. with Google)
- Look for personal homepage or website with a biography
- Select desired website

Task: specification

- Input:
  - *Plain text* extracted from website, mail, etc.
  - List of *named entities* in text with meta-information
- Output for each named entity:
  - *Link* to background info found using Google
  - Total number of Google hits

Approach: division in categories

- Proper names
- Infrequent nouns
- All other words

Approach: dedicated websites

- A dedicated website is a page that:
  - is a personal homepage, or

    gives a description of a person
  - is dedicated to a single, specific person
  - is not about the person in a specific context (so, not like The Tragedy of Colin Powell)

- First Google search result is a dedicated website in 67% of searches for persons (on test set of size 52)

Approach: outline

- Get top 10 Google search results
- For *persons*, select first *dedicated website in results*
- Otherwise, fall back to first result

Approach: dedicated website detection

- Extract *features* from website
- Train classifier to classify based on features
- Classes: *dedicated*, *not dedicated*

Approach: features of dedicated websites

- Reason: use *prior knowledge* about dedicated websites.
- Different features for each *subtype* of proper names?(person, location, organization and unknown)
- Binary versus continuous

Approach: features of dedicated websites for persons

- Name in domain, e.g. www.powell.com
- Name in path, e.g. www.usa.com/colin_powell
- Name in title tag
- Name in keywords (meta tag)
- Name in headings, e.g. <h1>Colin Powell</h1>
- Name in body text

Experiments: annotated set

- Manually annotated first 10 Google hits for 52 person names

- Train classifiers
- Test performance

Experiments

- Implementation of two different classifiers:
  - Perceptron (linear discriminant)
  - Majority vote (instance of nearest neighbour)

- Both easily implemented
- Compare results

Experiments: majority vote

- Special case of 3-nearest neighbor
- Small number of binary features (6), so small number of possible feature vectors ($2^6 = 64$)
- To classify, vote among all samples in train set that have the same feature vector
- If <3 samples with same feature vector, then include nearest neighbours in vote

Results: performance of classifiers

- Performance: $\dfrac{\text{number of websites correctly classified}}{\text{total number of websites classified}}$
- Split annoted set: train on 468 examples, test on 52 examples
- Perceptron: 59% (avg. 20 runs)
- Majority vote: 70% (avg. 100 runs)

Results: performance of IRGoo (detailed)

- Recall = $\dfrac{\text{Number of dedicated websites found}}{\text{Total number of named entities}}$
- 52 of 52 (100%)

- Precision = $\dfrac{\text{Number of dedicated websites found accurate}}{\text{Total number of dedicated websites found}}$
- 40 of 52 (76%)
  - remember: first Google hit 67%

- About the 12 inaccurate results:
  - 8 contained information about the right person (15%)
  - 4 contained completely useless information (8%)

Results: performance of IRGoo (persons)

	Recall	Precision	Total number of samples
Our train set (train on 468, test on 52)	100%	76%	52
Test text	100%	100%	1
Test mail	100%	0%	1
Test html	85%	100%	7

Results: performance of IRGoo (named entities)

	Recall	Precision	Total number of samples
Test text	9%	100%	11 (1 person)
Test mail	16%	0%	6 (1 person)
Test html	9%	100%	66 (7 persons)

Implementation

- IRGoo is a module that:
  - for *proper names that are persons,* classifies Google hits with majority vote classifier and returns first dedicated page found
  - in other cases, falls back to the first Google hit
  - reports confidence measure for URLs
  - reports Google hitcount

Conclusions

- Classifiers:
  - perceptron performs poorly
  - majority vote has quite acceptable performance

Conclusions

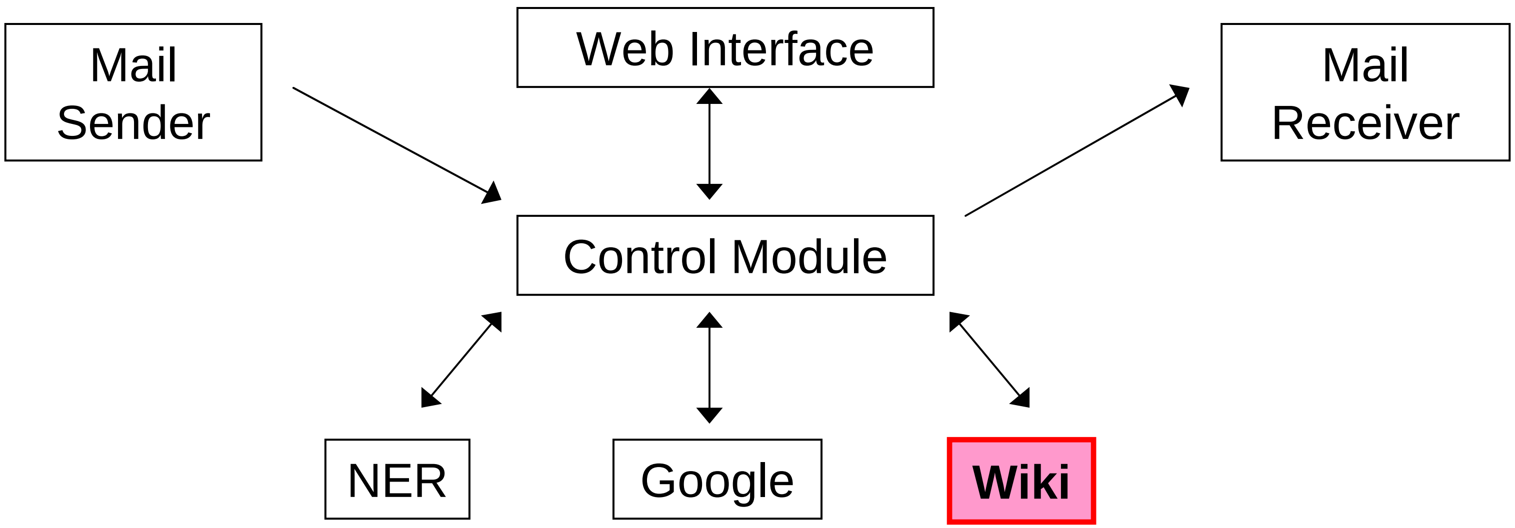
- Total solution performance very good for the very restricted problem domain it was designed for
- Even when it misclassifies pages, it usually links to something related to what was desired
- Fallback to Google for other domains, which has reasonable performance

Future work

- Find better features for discriminating between dedicated and not dedicated websites
  - automatically?
- Train classifier for other types of proper names
  - location
  - organization
  - unknown
- Train classifier for other types of named entities
  - infrequent nouns, etc.
- Improve performance

IR Wikipedia

Peter van der Meer & Paul Manchego



Wikipedia Information Retrieval (IR) module

- Named Entity Recognition (NER) module provides Named Entities
- Named Entity →Proper Name
  - returns the best Wikipedia URL
- Named Entity →Infrequent Noun
  - Return the best Wiktionary URL
  - Return a small definition

Requirements

- Correctness
- Performance
- XML communication standard
- Exchange data using STDIN/STDOUT

URL Retrieval

- Currently only a Named Entity is used to search for a matching URL.
- Four approaches can be used to retrieve an URL

Approach 1

- Wikipedia online web search engine
  - Advantages
    - Uses up to date information
  - Disadvantages
    - Slow
    - Need html parsing
    - Supports only simple queries

Approach 2

- Wikipedia downloadable database
  - Advantages
    - Better performance
    - Customized queries
  - Disadvantages
    - Database is 29MB large!!!
    - Database needs updates periodically
    - Hard to use with faculty machines

Approach 3

- Google API is used query Wiki domain
  - Advantages:
    - Google is a more advanced search engine compared to the Wikipedia engine
    - Java API is freely available, no API for Wikipedia
  - Disadvantages
    - Better performance on database approach
    - API search key has 1000 queries limit/day

Approach 4

- Wikipedia URLs and pages have a very specific format
- A Named Entity can be translated into an URL

  George W. Bush→

  http://en.wikipedia.org/wiki/George_W._Bush
- The URL is checked for correctness by looking if the Named Entity is present the <title> tag

Methods

- Our method combines the last two aproaches (3+4)
- Proper Noun
  - Expand Named Entity to URL and check <title> tag.
  - If the Named Entity is not present in the <title> then check the highest ranked Google result. Count the number of Named Entity occurrences to determine link confidence.
  - If there is still no result don’t return an URL

Methods CONT'D

- Infrequent Noun
  - Expand Named Entity to URL and check <title> tag
  - If the Named Entity is not present in the <title> then check the highest ranked Google result.
  - If there is a result also retrieve a small definition

Results

NamedEntity	Precision	
Proper Noun	0.97	**(n=39)**
InfrequentNoun	1.00	**(n=17)**

$$\text{Precision} = \frac{\#\text{Correct NE}}{\#\text{Total NE}}$$

Conclusion

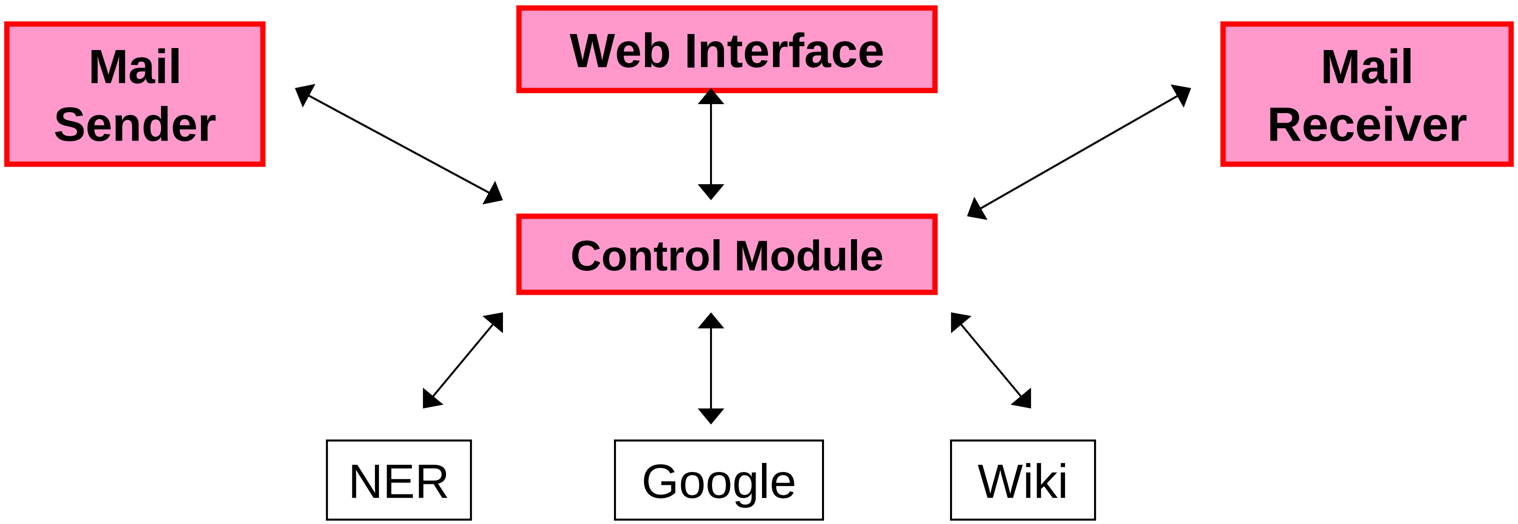
- If there is a Wikipedia link it is usually quite relevant to the Named Entity
- Named Entities are matched with relevant URLs

Future Work

- Matching an Infrequent Noun with a Wiktionary link by default can be improved
- No context is taken into account when searching for the most suitable URL
- Database approach can be looked into

Controle/interface Module

Daan Vreeswijk & Joeri Honeff



Goals

- Control module – runs the program
- User interface:
  - Web
  - E-mail

Control module

- Communicate with user interface
- Communicate with other modules
- Strip html-tags from input
- Incorporate found links in original text

Control module problems

- Communication between modules
- Speed
- Technical issues

Control Module Solutions

- Python
- Standard input and -output
- Threading
- A lot of trial and error

Unsolved problems

- Html-stripping is not perfect
- Sometimes a bit buggy

Interface

- Web:
  - Get text to link from user
  - Show linked text to user
  - Communicate with control module
- E-mail
  - Integrate in mail program
  - Communicate with control module

Interface problems

- Web:
  – Site design
  – Communication with control module
- E-mail
  – Retrieving body from message
  – Building html-mail
  – Integration in e-mail program

Interface solutions

- Web:
  - CSS for design
  - Standard input and –output
- E-mail
  - Python module for e-mail handling
  - Integrated in K-mail
- Separate control modules for e-mail and web

Evaluation – what did go well

- Web- and e-mail integration succeeded
- All modules are started correctly

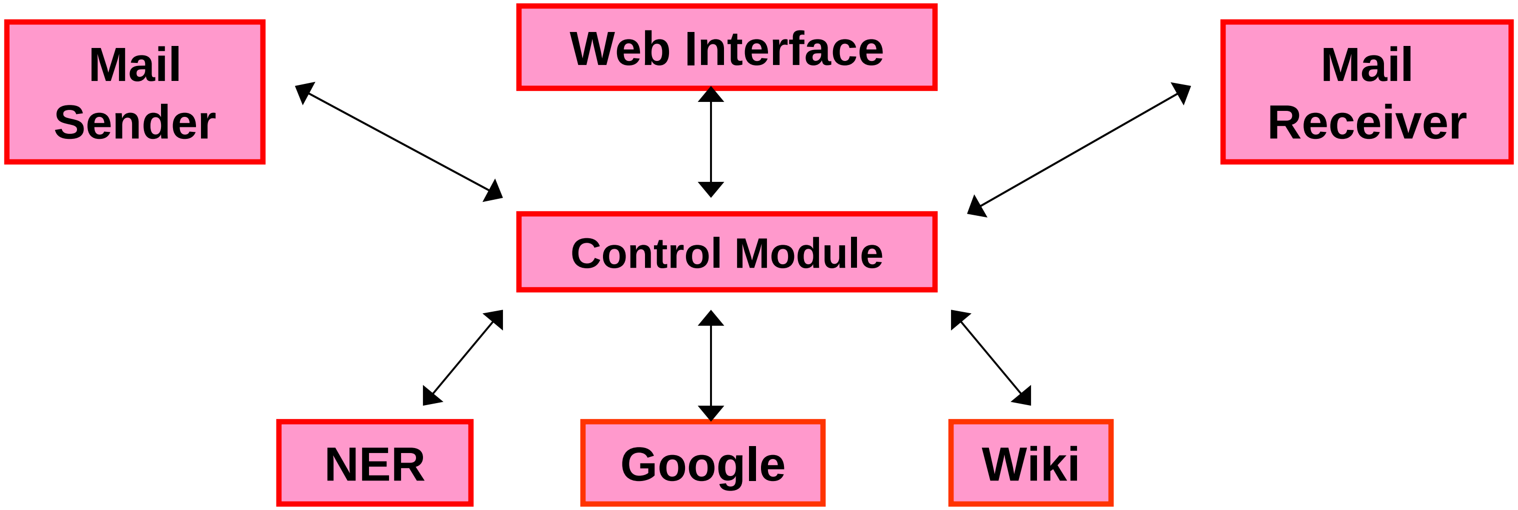
Evaluation – what didn’t go well

- Separate control modules needed
- Errors not always handled correctly

To do

- Better html-stripping
- Error-handling
- Integration of the two control modules

Evaluation



Mail Sender
Web Interface
Mail Receiver
Control Module
NER
Google
Wiki

System evaluation

- It works!
- Exact performance still to be evaluated
  - How many interesting words are getting linked?
  - How many words are linked correctly
  - Compare with manually annotated data

Future work

- Better error-handling
- More robust
- Speed enhancements (NELLI on steroids)

Discussion

- Relevance
- Practical usefulness
- Speed

Conclusion

- System generates links automatically
- Appears to link relatively well (still to be evaluated)
- Still works too slow to be really useful

Any questions?

CoolNELLI™

your link to knowledge

T E S T R U N