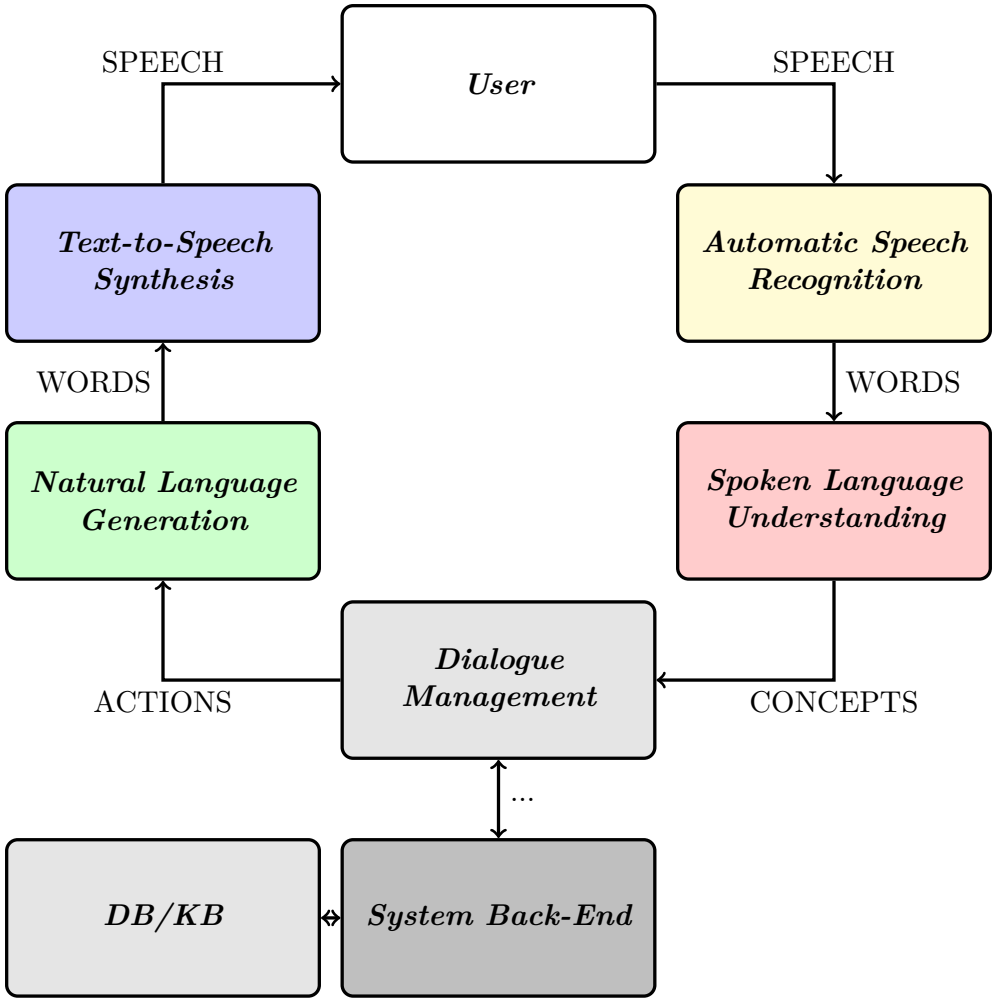
Language Understanding Systems

Spoken Dialogue System Baseline

Evgeny A. Stepanov

SISL, DISI, UniTN
evgeny.stepanov@unitn.it

Spoken Dialogue System



Lab Objectives

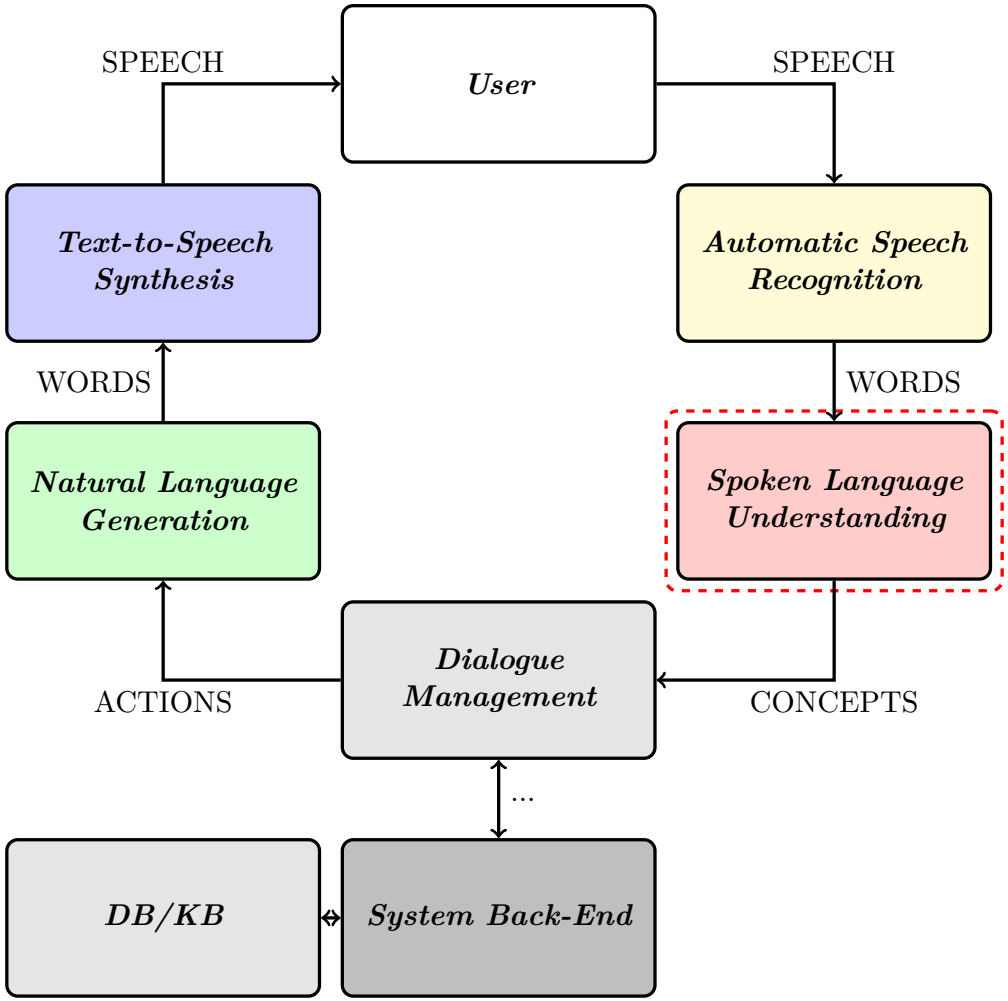
- from raw text (or processed ASR output) to DB results
  - Parsing SLU output
  - Utterance Classification
  - SQL Query Construction
  - DB Querying
- Required 'new' tools
  - Install MySQL & populate DB
  - Install `fstprintstrings`

Outline

Section 1

Spoken Language Understanding

SDS Back-End


SPEECH
User
SPEECH
Text-to-Speech Synthesis
Automatic Speech Recognition
WORDS
WORDS
Natural Language Generation
Spoken Language Understanding
Dialogue Management
ACTIONS
CONCEPTS
...
DB/KB
System Back-End

Subsection 1

Simple PHP

Executing External Command

`http://php.net/manual/en/book.exec.php`

- exec
- shell_exec

exec

exec – Execute an external program

```
string exec ( string $command [, array &$output [, int &$return_var ]] )
```

Parameters

- **command** The command that will be executed.
- **output** If the **output** argument is present, then the specified *array* will be filled with every line of output from the command.
- **return_var** If the **return_var** argument is present along with the **output** argument, then the return status of the executed command will be written to this variable.

shell_exec

shell_exec – Execute command via shell and return the complete output as a *string*

```
string shell_exec ( string $cmd )
```

Parameters

- **cmd** The command that will be executed.

Executing External Command

`http://php.net/manual/en/ref.filesystem.php`

- mkdir
- fopen
- fwrite
- fclose

JSON Functions

`http://php.net/manual/en/ref.json.php`

- json_decode
- json_encode

Subsection 2

Extracting Concepts from SLU

Provided Classes

- FstSlu.php
  - Wrapper for FST-based models
- SluResults.php
  - extracts $concepts from SLU output file/array
- FstUtilities.php
  - set of wrapper functions for fst tools
  - use as a 'black box'

Subsection 3

Utterance Classification

Utterance Classification

- SLU Concepts → user **provided** information (slots)
- Utterance classification → user **requested** information (intent)

Provided Classes

- FstClassifier.php
  - simple FST-based Naive Bayes classifier
  - returns $class (see example.php)
  - use as a 'black box'
- FstUtilities.php
  - set of wrapper functions for fst tools
  - use as a 'black box'

Subsection 4

Confidence Scores

Confidence Measures

- What is confidence measure?
- A confidence measure (CM) is a number between 0 and 1 that is applied to ASR/SLU output, which gives an idea of how confident we are that the output is correct.

Classifier Confidences

- Naive Bayes Classifier (provided) outputs posterior probabilities for each class.
- Posterior probability is the conditional probability assigned after observation.
- It can be used directly as a confidence measure (after cost to probability conversion).
- Provided by `fstprintstrings`

FST/LM Confidences

Similar to Naive Bayes

```
3 2 star 0                7.31074905
2 1 of   0                3.47148705
1 0 thor B-movie.name     9.43759155
0                         1.55078459
```

Remember that these are negative log probabilities

PHP Class Internally

- sum weights/costs along the path (`fstprintstrings`)
- convert to probability as e^{-x}, where x is the sum
- normalize between 0 and 1 as $\frac{p_i}{P}$, where p_i is the probability for an output label/sequence, and P is the sum of all label/sequence posterior probabilities

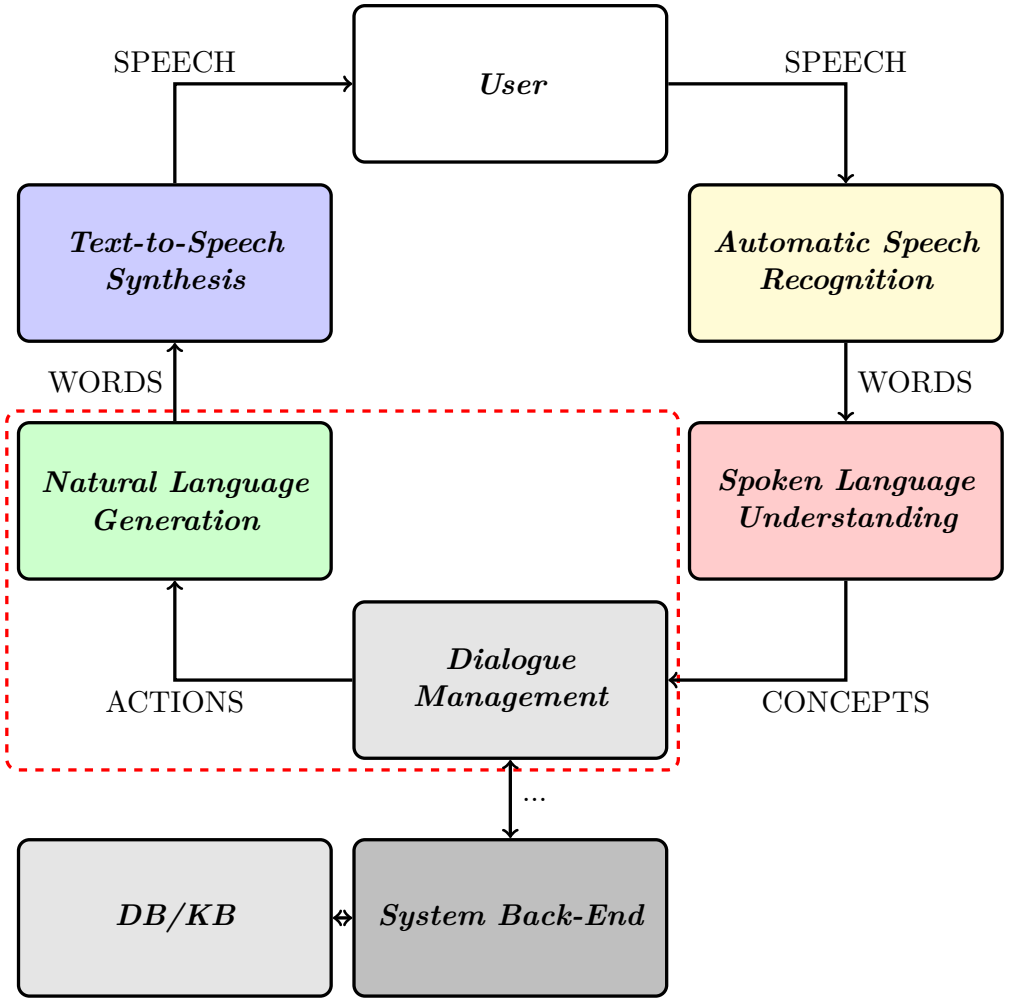
Exercises

- Analyze provided scripts
- Replace required models by your own
- Experiment with n-best options
- Retrieve confidences for n-best list
- **Combine with ASR confidences**
  - *consult lecture slides*
  - or just multiply them (?)

Section 2

Dialogue Manager

SDS Back-End


SPEECH
User
SPEECH
Text-to-Speech Synthesis
Automatic Speech Recognition
WORDS
WORDS
Natural Language Generation
Spoken Language Understanding
Dialogue Management
ACTIONS
CONCEPTS
...
DB/KB
System Back-End

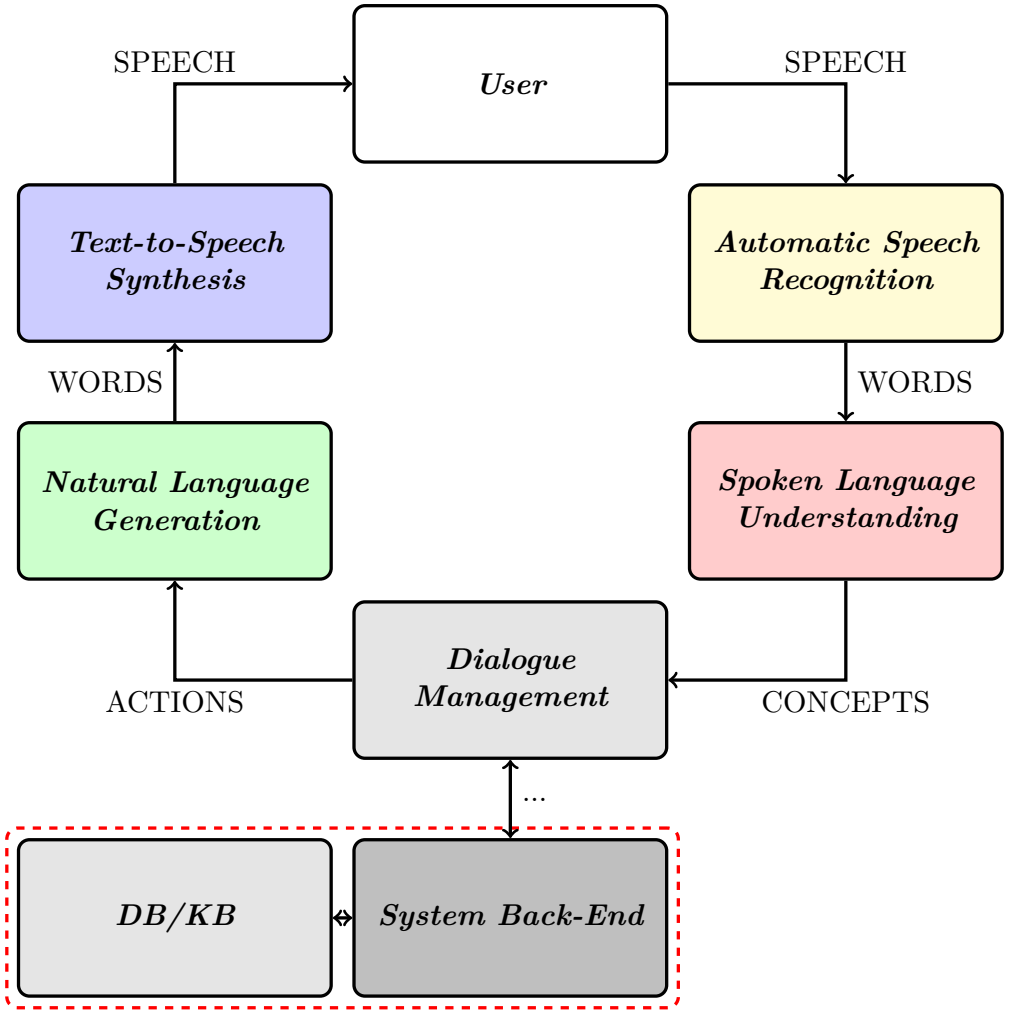
SDS Back-End

- TO BE DEVELOPED as Project 2
- See example.php for conditional statements

Section 3

System Back-End

SDS Back-End


SPEECH
User
SPEECH
Text-to-Speech Synthesis
Automatic Speech Recognition
WORDS
WORDS
Natural Language Generation
Spoken Language Understanding
Dialogue Management
ACTIONS
CONCEPTS
...
DB/KB
System Back-End

Back End

The SDS is usually interfaced with some external software (Back-End): DataBase (DB), Knowledge Base (KB) or an expert system (ES)

Back End

The SDS is usually interfaced with some external software (Back-End): DataBase (DB), Knowledge Base (KB) or an expert system (ES)

Thus, Spoken Language Understanding and Dialogue Management internal representations have to be converted to the *domain-specific* format of the DB/KB/ES: e.g. SQL, SPARQL, STRIPS, etc.

Back End

The SDS is usually interfaced with some external software (Back-End): DataBase (DB), Knowledge Base (KB) or an expert system (ES)

Thus, Spoken Language Understanding and Dialogue Management internal representations have to be converted to the *domain-specific* format of the DB/KB/ES: e.g. SQL, SPARQL, STRIPS, etc.

Informational SDS → provides information from DB/KB

Provided Classes

- Slu2DB.php
  - Given SLU concepts & utterance classifier results
  - Construct SQL query
- QueryDB.php
  - Connect & Query DB
  - Returns query results as array

Section 4

SDS Development Considerations

SDS Development Considerations

- Data Base / Knowledge Base Considerations
  - What information is available?
  - What is the ontology (if any)?
  - How to access information in DB/KB?
- Spoken Language Understanding Considerations
  - Tailored towards the task
  - Ontology $\approx$ DB/KB ontology
- Coverage
  - What information from DB/KB to provide?
  - What users ask the most?
    - User Study

Ontology (from Wikipedia)

In computer science and information science, an ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular **domain** of discourse.
An ontology compartmentalizes the variables needed for some set of computations and establishes the relationships between them.

Movie Domain

Data Base / Knowledge Base Considerations

- What information is available?
  - Information about movies, actors, etc.
- What is the ontology?
- How to access information in DB/KB?

Coverage: NL-SPARQL Data Set

Query Complexity

Complexity	Count	%
0 entity	435	10%
1 entity	3,665	83%
2 entities	311	7%
3 entities	11	0.3%

Some user questions are not supported by KB: e.g. ‘trailer’, ‘review’...

Coverage: NL-SPARQL Data Set

Query Types

Type	Count	%
Total	3,987	100%
movie by name	295	7%
movie by actor	283	7%
movie by X	1312	33%
count movie	21	0.5%
person by name	151	4%
actor by movie	246	6%
X by movie	1,277	32%
Cumulative	3,585	90%

Coverage: NL-SPARQL Data Set

Query Types

Type	Count	%
Total	3,987	100%
movie by name	295	7%
movie by actor	283	7%
movie by X	1312	33%
count movie	21	0.5%
person by name	151	4%
actor by movie	246	6%
X by movie	1,277	32%
Cumulative	3,585	90%

Just by covering single entity & 7 types of queries we cover 90% of valid user data!