Обработка естественного языка (NLP) в .NET

(С примерами на F# и C#)

Сергей Тихон

@sergey_tihon

Сергей Тихон

- Solution Architect в EPAM Systems
- Microsoft F# MVP (VS & Dev Tech)
- Автор F# Weekly (sergeytihon.com)
- Open Source энтузиаст
  - Stanford.NLP.NET
  - OpenNLP.NET
  - SwaggerProvider
  - … (fsprojects)
- https://twitter.com/sergey_tihon
- Беларусь, Минск

Про что этот доклад

1. **Разберемся что такое NLP, какие задачи решаются и где они встречаются**

```
T NlpTask<T> (string text) {
    return ...;
}
```

Про что этот доклад

1. Разберемся что такое NLP, какие задачи решаются и где они встречаются
2. **Разберем эталонные архитектуры (reference architecture)**
   - Промышленного документного поиска
   - Чат-ботов

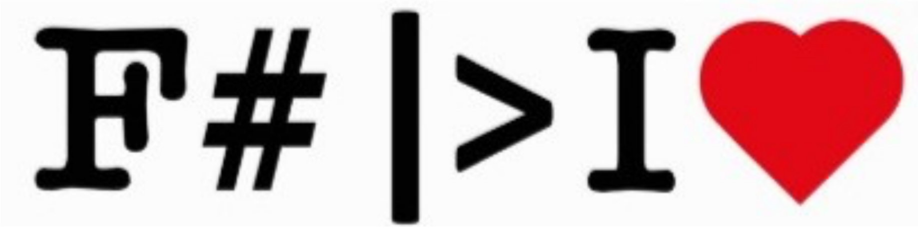
Про что этот доклад

1. Разберемся что такое NLP, какие задачи решаются и где они встречаются
2. Разберем эталонные архитектуры (reference architecture)
   - Промышленного документного поиска
   - Чат-ботов
3. **Разберем доступные нам инструменты**
   - .NET Framework
   - Облачный API (MS LUIS, IBM Watson, Google)
   - Популярные NuGet пакеты (Stanford NLP, OpenNLP и другие)

Про что этот доклад

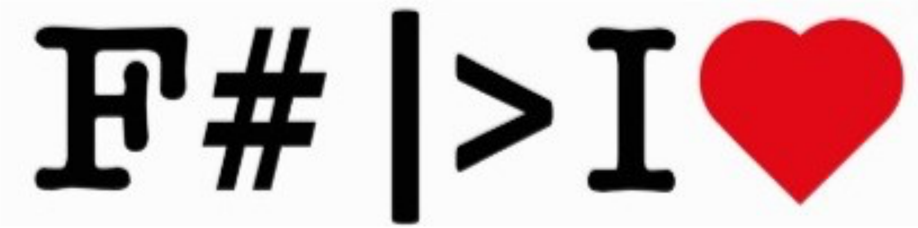
1. Разберемся что такое NLP, какие задачи решаются и где они встречаются
2. Разберем эталонные архитектуры (reference architecture)
   - Промышленного документного поиска
   - Чат-ботов
3. Разберем доступные нам инструменты
   - .NET Framework
   - Облачный API (MS LUIS, IBM Watson)
   - Популярные NuGet пакеты (Stanford NLP, OpenNLP и другие)
4. **Примеры будут на F#**

Про что этот доклад



1. Разберемся что такое NLP, какие задачи решаются и где они встречаются
2. Разберем эталонные архитектуры (reference architecture)
   - Промышленного документного поиска
   - Чат-ботов
3. Разберем доступные нам инструменты
   - .NET Framework
   - Облачный API (MS LUIS, IBM Watson)
   - Популярные NuGet пакеты (Stanford NLP, OpenNLP и другие)
4. Примеры будут на F#
5. **Не будет примеров работы с русским языком**

Про что этот доклад



1. Разберемся что такое NLP, какие задачи решаются и где они встречаются
2. Разберем эталонные архитектуры (reference architecture)
   - Промышленного документного поиска
   - Чат-ботов
3. Разберем доступные нам инструменты
   - .NET Framework
   - Облачный API (MS LUIS, IBM Watson)
   - Популярные NuGet пакеты (Stanford NLP, OpenNLP и другие)
4. Примеры будут на F#
5. Не будет примеров работы с русским языком
6. **Но каждый узнает что-то новое!**

Русский язык

NLPub

Каталог NLPub организован по вики-принципу и содержит сведения об инструментах, ресурсах, методах и алгоритмах, необходимых для успешного построения систем автоматической обработки русского языка
https://nlpub.ru/Обработка_текста

Yandex

- **MyStem** - морфологический анализ текста на русском языке https://tech.yandex.ru/mystem/
- **Томита-парсер** - извлечения структурированных данных из текста на естественном языке https://tech.yandex.ru/tomita/
- **Linguistic technologies (API)**
  - Translate API
  - Dictionary API
  - SpeechKit
  - Predictor API

Промышленный документный поиск

Enterprise document search

Документы хранятся в разных системах

Confluence

SUBVERSION

SharePoint

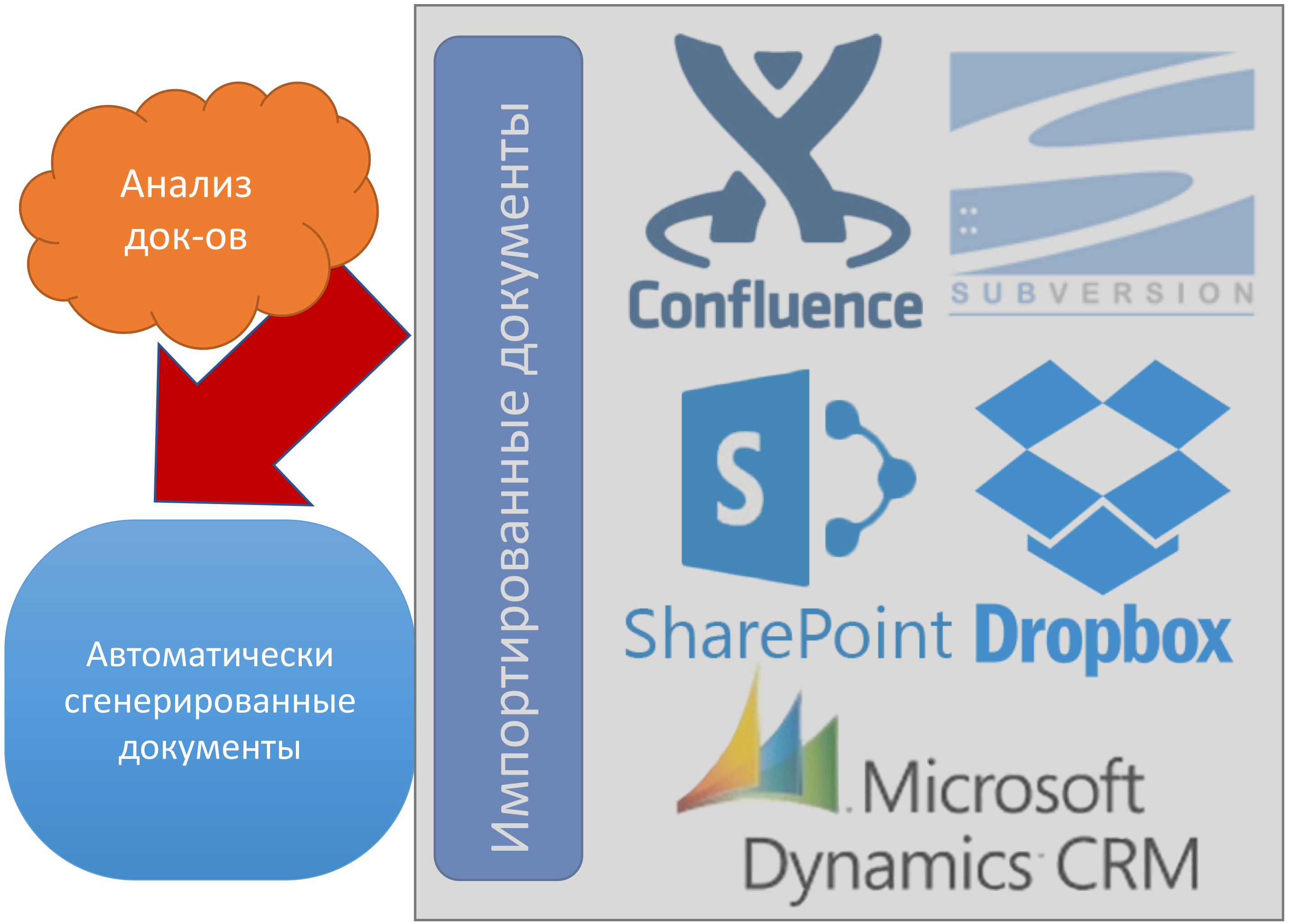
Dropbox

Microsoft Dynamics CRM

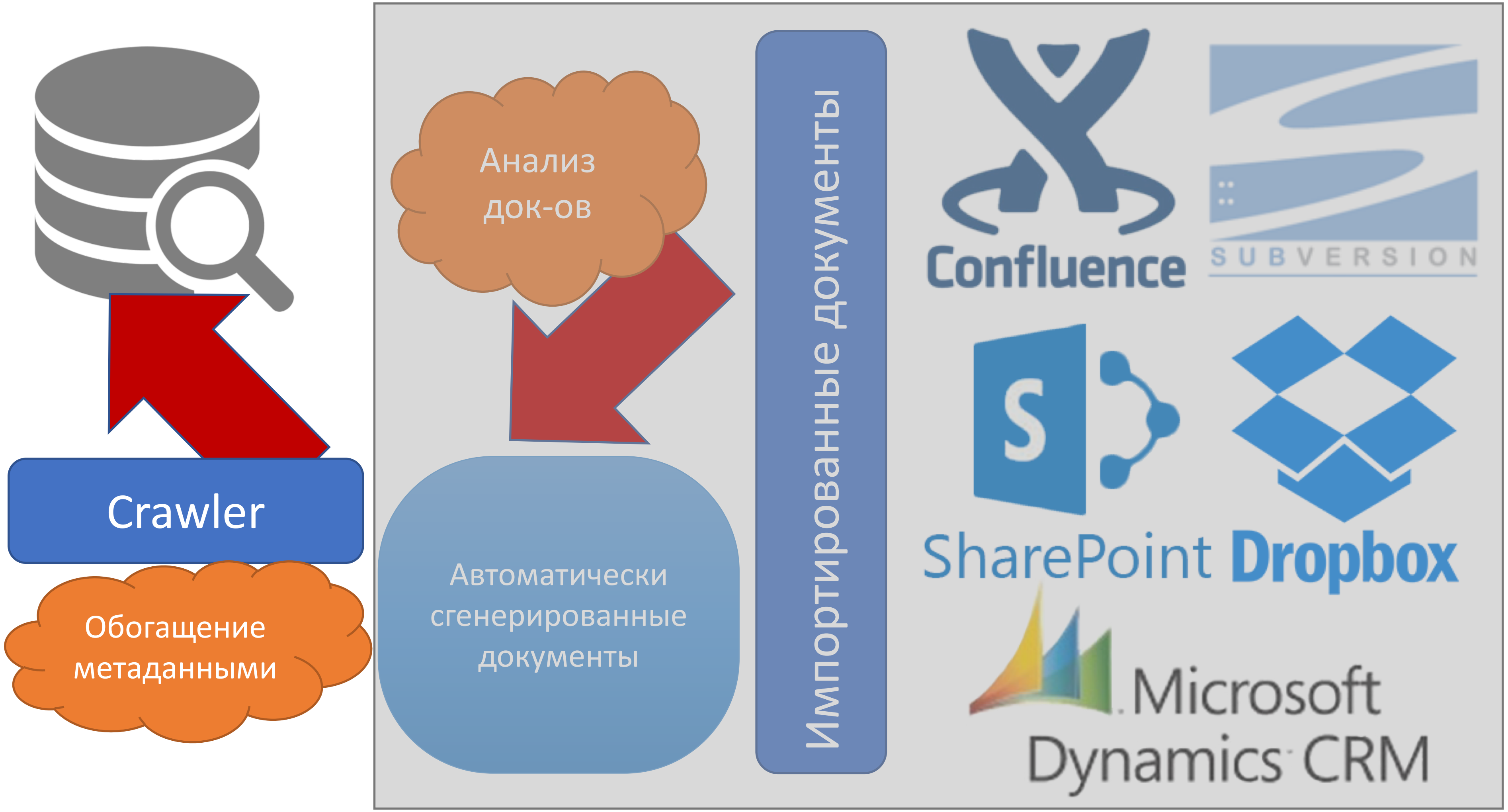
Собираем документы вместе

Импортированные документы

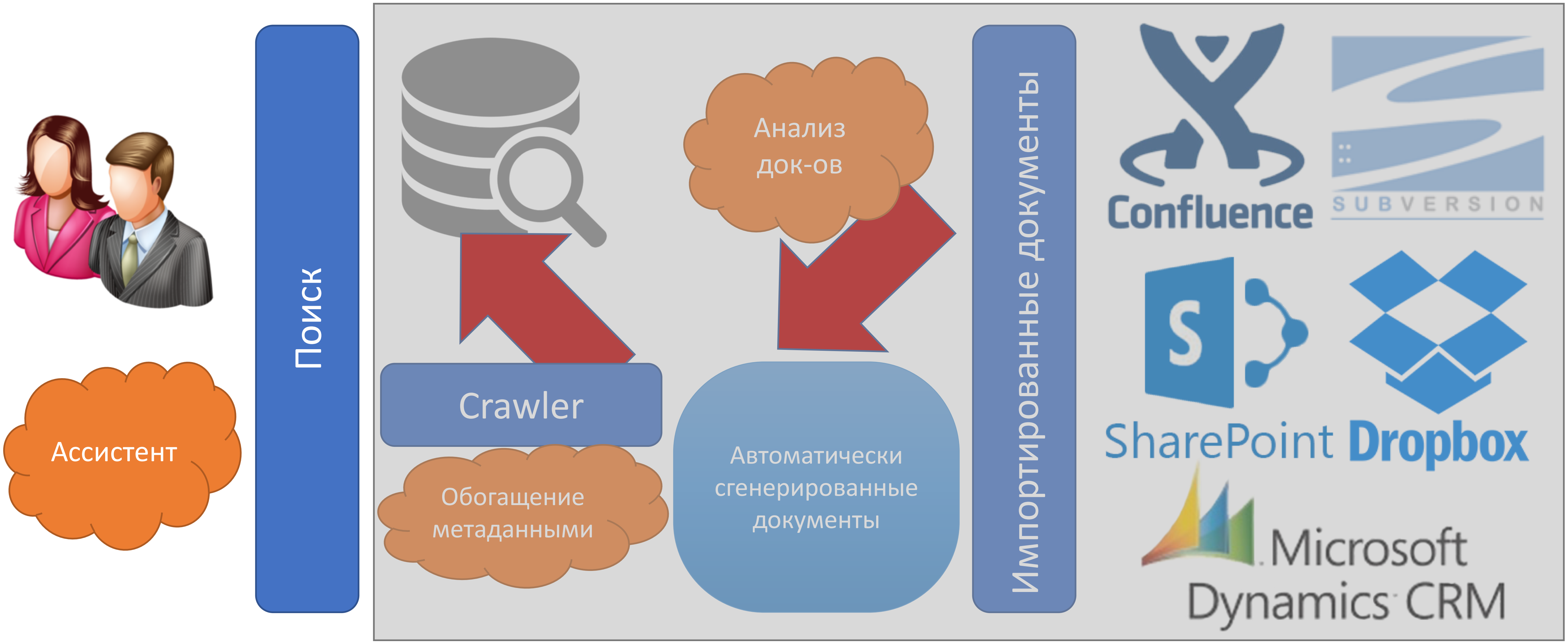
Извлекаем части для переиспользования



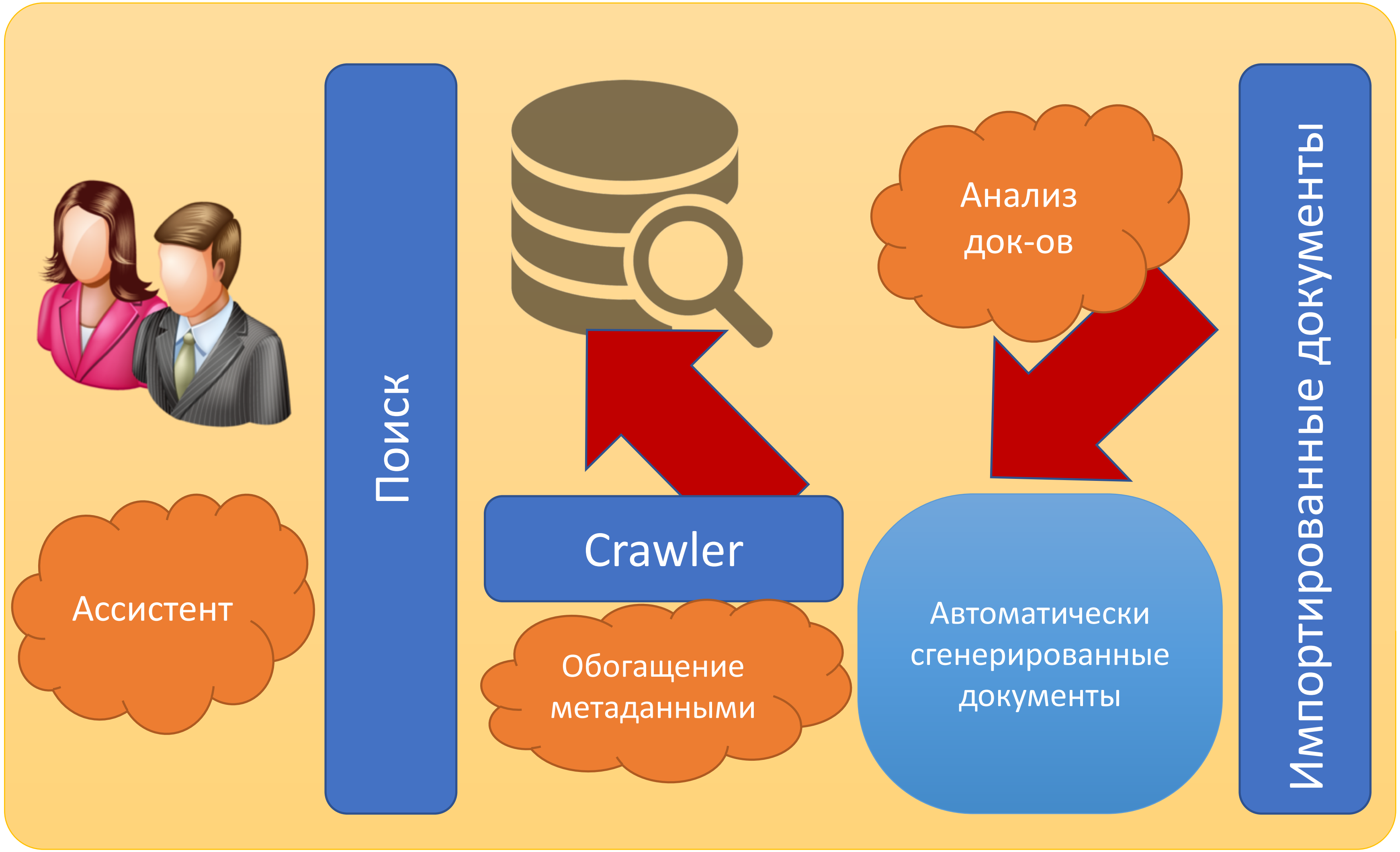
Индексируем и обогащаем метаданными



Добавляем ассистента пользователю



Промышленный документный поиск



Отличия от веб-поиска (Yandex/Google)

1. **Поиск по документам а не по веб страницам**
   a) Много повторяющегося текста и дубликатов (нету page / site rank)
   b) Большие документы (презентации достигают 2Гб)
   c) Сложности с онлайн превью

Отличия от веб-поиска (Yandex/Google)

1. Поиск по документам а не по веб страницам
   a) Много повторяющегося текста и дубликатов (нету page / site rank)
   b) Большие документы (презентации достигают 2Гб)
   c) Сложности с онлайн превью

2. **“Мало” пользователей (по сравнению с Yandex/Google)**
   a) Сложно обучаться на поведении пользователей
   b) Узкоспециализированные запросы
   c) Приходится обучатся на контенте и сильно зависеть от процессов компании и данных в разных системах.

А так же...

Поиск

Google

Yandex

Bing

Bai du 百度

Чат-боты

wit.ai

api.ai

semanticmachines

Персональные ассистенты

alexa

Hi. I'm Cortana.

Привет, я Алиса

@sergey_tihon

Задачи в NLP

Задачи в NLP

- **Sentence Segmentation** – разбить текст на предложения
- **Tokenization** – разбить предложение на токены (слова + символы)
- **Part-Of-Speech Tagging** – определить часть речи (слова в предложении)
- **Stemming** – найти базовую форму слова (например убрать временную форму)
- **Named Entity Recognition** – распознать какие токены являются известными сущностями (Имя, Дата, Место)
- **Speech Synthesis** – синтез речи (текст –> речь)

Задачи в NLP

- **Syntactic Parsing** – построить синтаксическое дерево
- **Sentiment Analysis** – определить эмоцию в тексе (на сколько юзер доволен или недоволен)
- **Relationship Extraction** – найти отношения между словами в предложении (объект – субъект)
- **Text Similarity** – посчитать на сколько похожи два текста
- **Information Extraction** – извлечение структурированной информации из текста
- **Speech Recognition** – распознавание речи (речь –> текст)

Задачи в NLP

- **Machine Translation** – автоматический машинный перевод текста или аудио
- **Natural Language Generation** – генерация текста (описать что нарисовано на картинке)
- **Automatic Summarization** – сгенерировать текст описывающий увиденное (описание к видео)
- **Question Answering** – понять и найти ответ на вопрос
- **Conversation Interfaces** – система взаимодействующая с юзером через диалог (чат-боты)

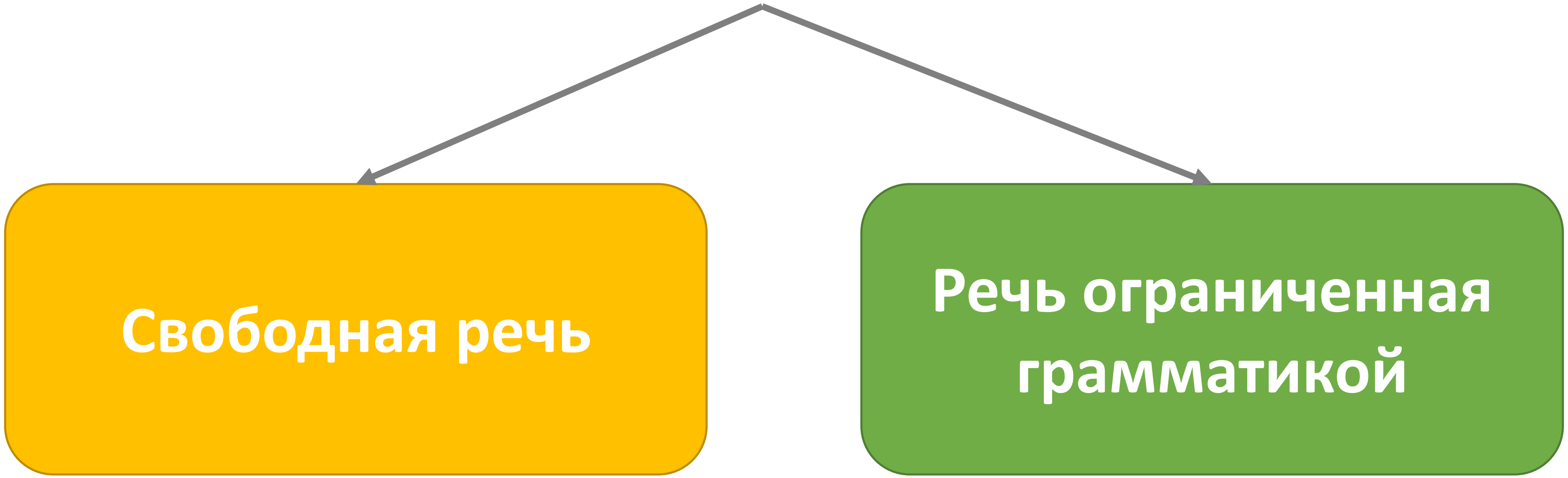
.NET Framework

Синтез речи (Speech Synthesis)

```
#r "System.Speech.dll"
open System.Speech.Synthesis

let synth = new SpeechSynthesizer()
synth.SelectVoiceByHints(VoiceGender.Female)

synth.Speak("Hello DotNext !!!")
```

System.Speech.dll присутствует в .NET начиная с версии **3.0** (т.е. с 2006г)

Распознавание речи (Speech Recognition)



Грамматика распознаваемого языка – это набор правил или ограничений, которые определяют, что механизм распознавания может распознать как “значимый ввод”.

Распознавание речи (Speech Recognition)

```
#r "System.Speech.dll"
open System.Speech.Recognition

let keywords = Choices()
keywords.Add([|"DotNext"|])

let recognizer = new SpeechRecognizer()
recognizer.LoadGrammar(
    Grammar(keywords.ToGrammarBuilder()))

recognizer.SpeechRecognized.AddHandler(
    fun _ e -> printfn "Recognized '%s'" e.Result.Text)
```

System.Speech vs Microsoft.Speech

System.Speech.dll	Microsoft.Speech.dll
Native code API (C++)	Managed code API (C#)
Часть OS (Windows Vista+)	Устанавливается отдельно
Не распространяется с приложением	Можно распространять вместе с приложением
Работает с грамматиками (Grammars) или со свободной речью	Нужно создавать грамматики (Grammars)
Тренируется под юзера	Не нуждается в тренировке

https://msdn.microsoft.com/en-us/magazine/dn857362.aspx

Облачный API

Microsoft Cognitive Services

Google Cloud Platform

IBM Watson

Baidu 百度

Dialogflow

wit.ai

AlchemyAPI™

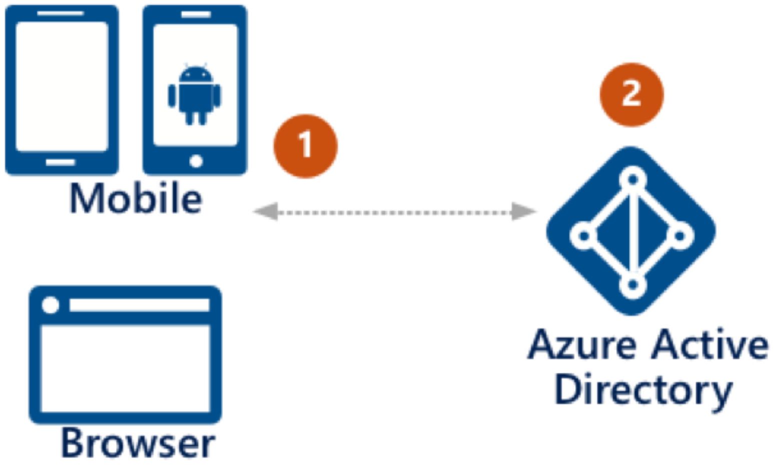
Чат-боты

Microsoft
Cognitive Services

Архитектура информационного чат-бота

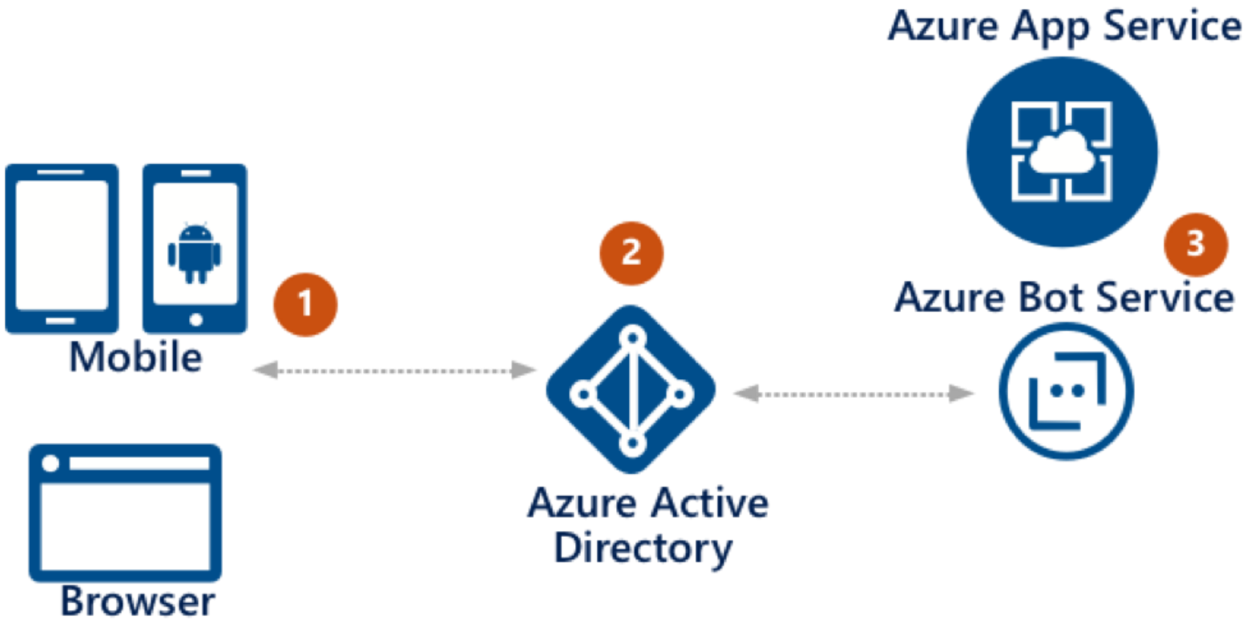
1. Пользователь начинает общение с ботом

Архитектура информационного чат-бота



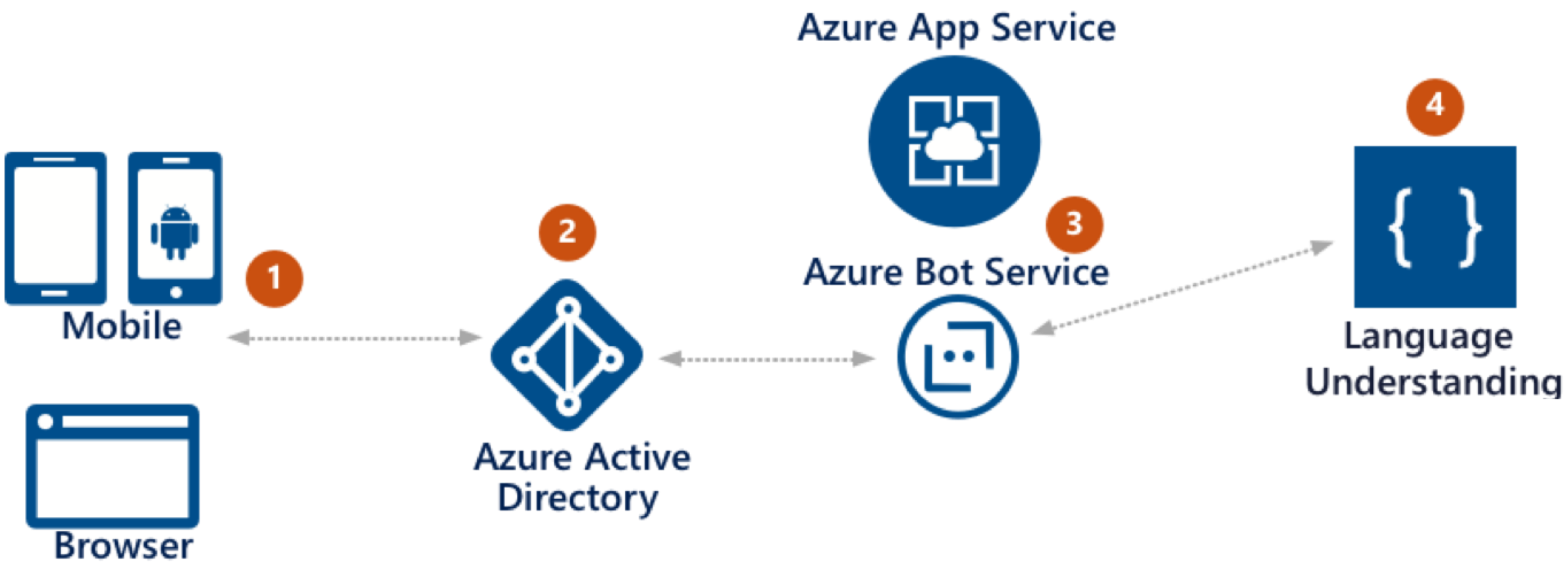
1. Пользователь начинает общение с ботом
2. Авторизуем\узнаем пользователя

Архитектура информационного чат-бота



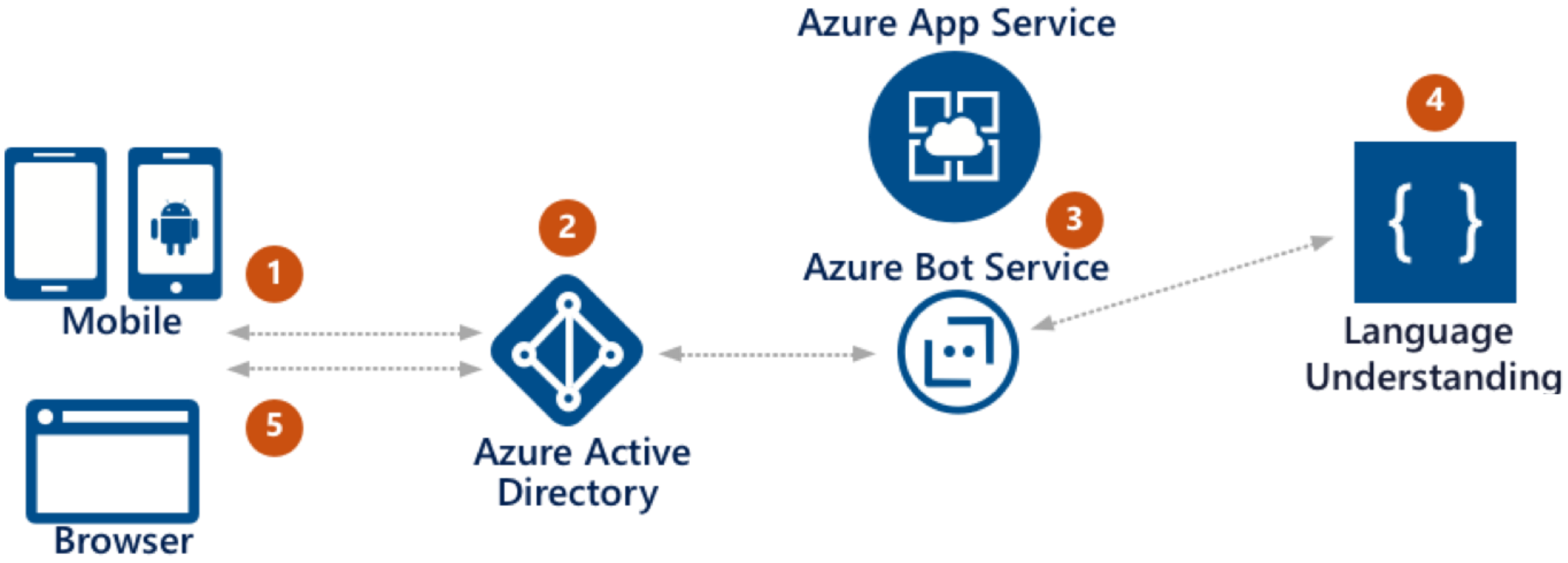
1. Пользователь начинает общение с ботом
2. Авторизуем\узнаем пользователя
3. Пользователь задает вопрос

Архитектура информационного чат-бота



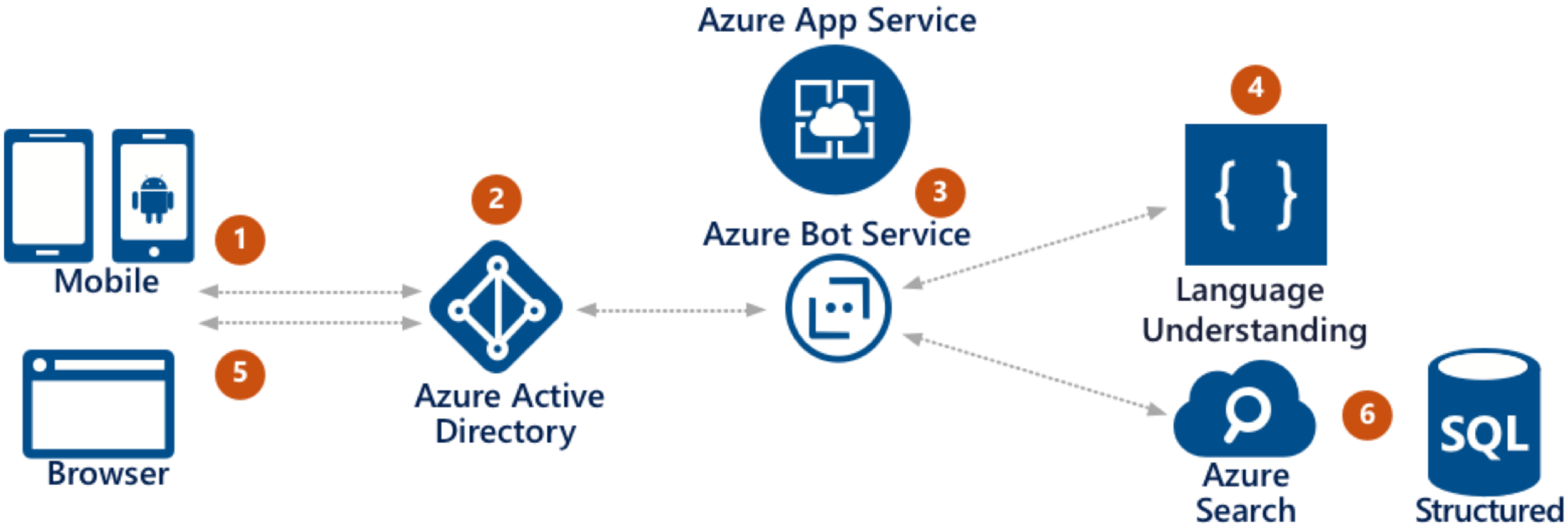
1. Пользователь начинает общение с ботом
2. Авторизуем\узнаем пользователя
3. Пользователь задает вопрос
4. Когнитивные сервисы распознают **намеренье пользователя**

Архитектура информационного чат-бота



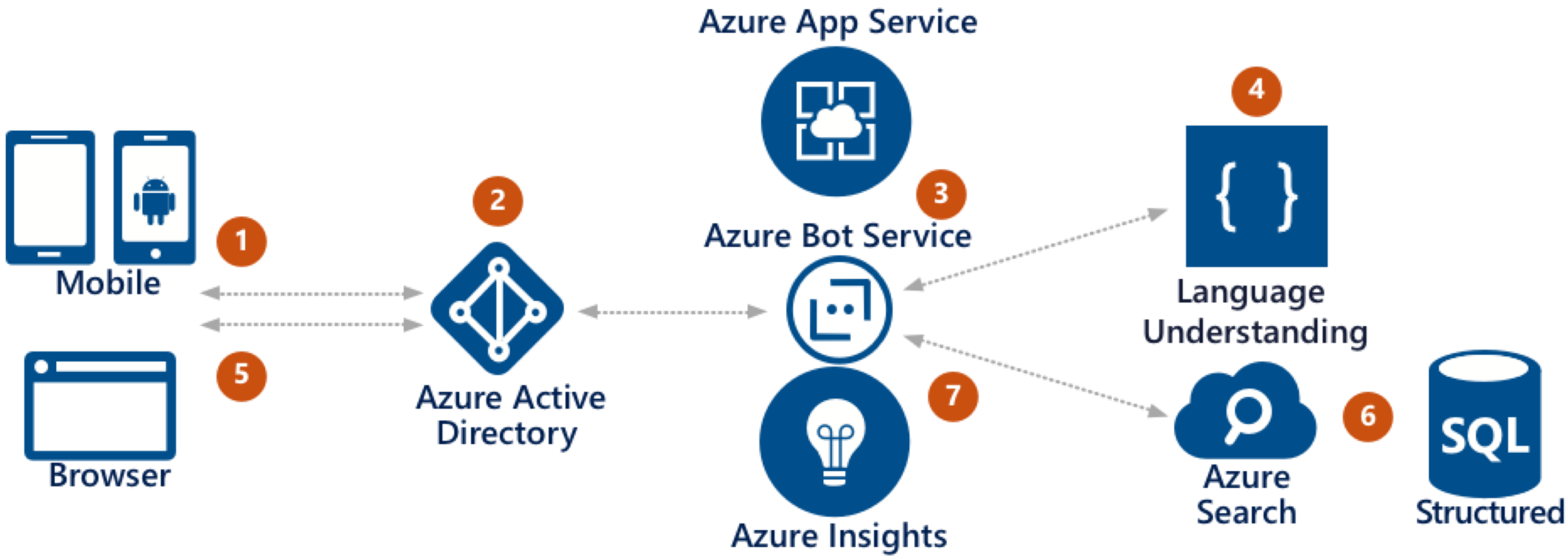
1. Пользователь начинает общение с ботом
2. Авторизуем\узнаем пользователя
3. Пользователь задает вопрос
4. Когнитивные сервисы распознают **намеренье пользователя**
5. В случае неоднозначности, намеренье уточняется

Архитектура информационного чат-бота



1. Пользователь начинает общение с ботом
2. Авторизуем\узнаем пользователя
3. Пользователь задает вопрос
4. Когнитивные сервисы распознают **намеренье пользователя**
5. В случае неоднозначности намеренье уточняется
6. Бот делает запрос за данными

Архитектура информационного чат-бота



1. Пользователь начинает общение с ботом
2. Авторизуем\узнаем пользователя
3. Пользователь задает вопрос
4. Когнитивные сервисы распознают **намеренье пользователя**
5. В случае неоднозначности намеренье уточняется
6. Бот делает запрос за данными
7. Собираем телеметрию чтобы следить за поведением и улучшать бота

https://dev.botframework.com

Что самое важное в чат-боте?

Что самое важное в чат-боте?

Удобство использования

Что самое важное в чат-боте?

Удобство использования

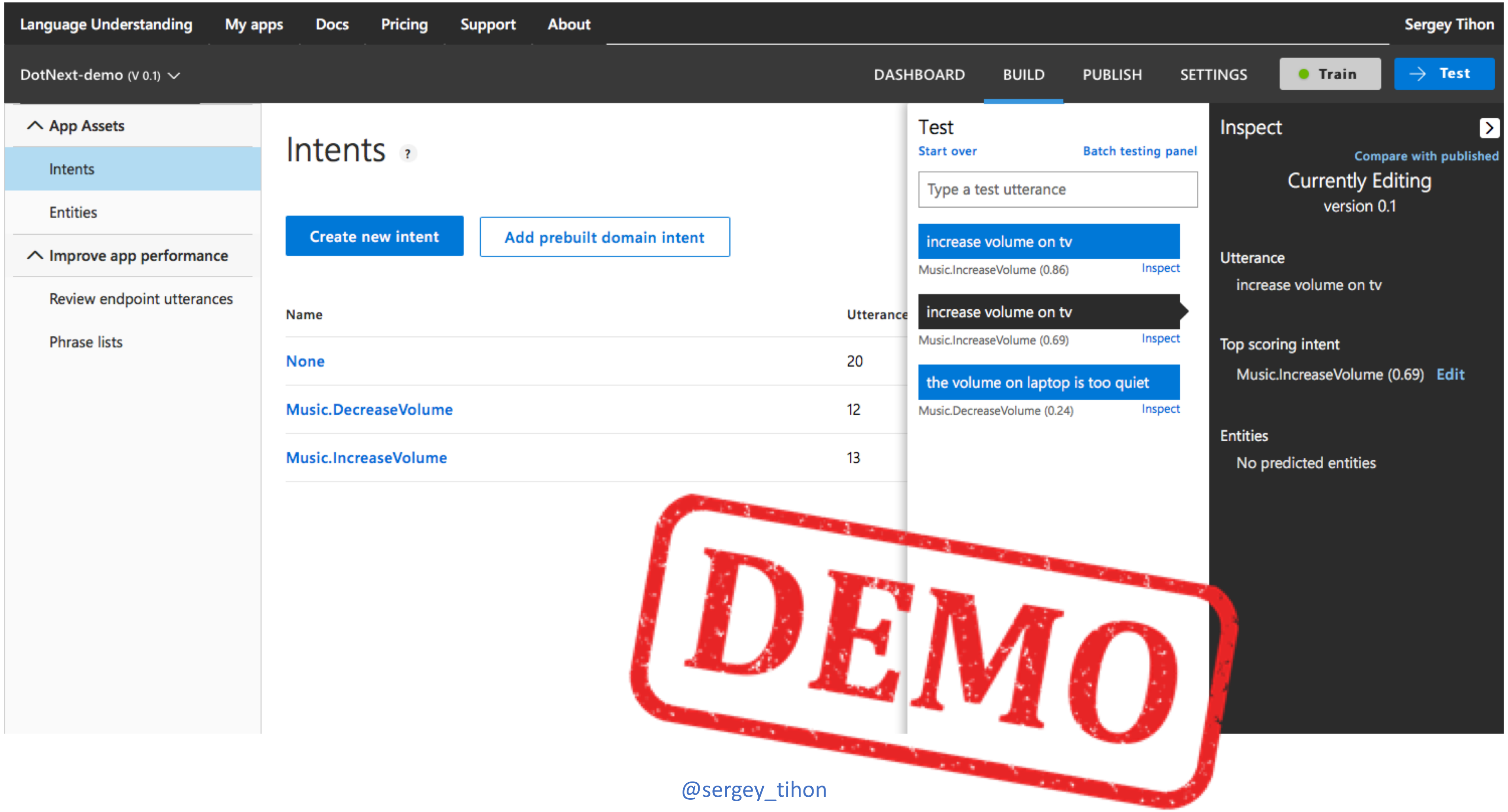
- Бот должен **ХОРОШО** понимать пользователя

Что самое важное в чат-боте?

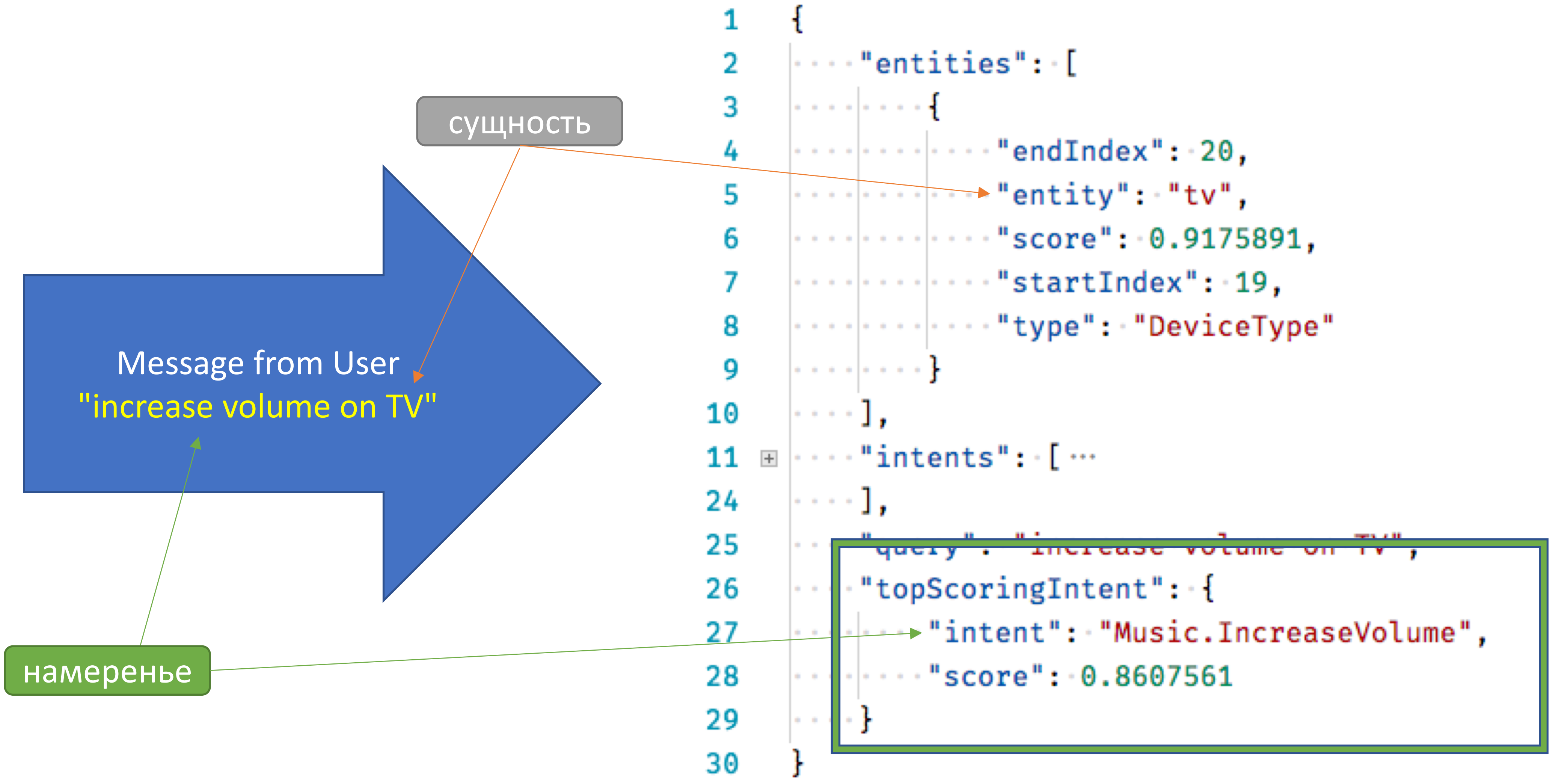
Удобство использования

- Бот должен **ХОРОШО** понимать пользователя
- Выполнить задачу используя бота должно быть **БЫСТРЕЕ** чем без него.

Language Understanding



Что такое намеренье (intent)?



Что такое намеренье (intent)?


```
{
    "entities": [
        {
            "endIndex": 20,
            "entity": "tv",
            "score": 0.9175891,
            "startIndex": 19,
            "type": "DeviceType"
        }
    ],
    "intents": [ ...
    ],
    "query": "increase volume on TV",
    "topScoringIntent": {
        "intent": "Music.IncreaseVolume",
        "score": 0.8607561
    }
}
```

Conversation

Conversation

Virtual Agent

Language

Language Translator

Natural Language Classifier

Speech

Speech to Text

Text to Speech

Аналог LUIS.ai

IBM Watson

Vision

Visual Recognition

Empathy

Personality Insights

Tone Analyzer

Discovery

Discovery

Natural Language Understanding

Discovery News

Knowledge Studio

Что есть в NuGet?

```
#I "../packages/FSharp.Data/lib/net45/"
#r "FSharp.Data.dll"

let [<Literal>] BaseUrl =  // string
    "https://api-v2v3search-0.nuget.org/query?q=nlp&take=100prerelease=false"
type NuGetSearch = FSharp.Data.JsonProvider<BaseUrl>

let rec search skip = async {  // int → Async<FSharp.Data.JsonProvider<...>.Datum []>
}

search 0
▷ Async.RunSynchronously
▷ Seq.sortByDescending (fun x→x.TotalDownloads)
▷ Seq.take 10
▷ Seq.iteri (fun i x →
    printfn "%02d. %s (%d)" (i+1) x.Title x.TotalDownloads)
```

01. Stanford.NLP.CoreNLP (74887)
02. Syn.Bot (30506)
03. AboditNLP (28543)
04. AboditUnits (21475)
05. Stanford.NLP.POSTagger (18541)
06. Stanford.NLP.NER (18256)
07. Stanford.NLP.Parser (16880)
08. OpenNLP (15033)
09. Language Detection (11171)
10. Autofac.Extras.AboditNLP (10978)

LanguageDetection

.NET порт [Language Detection Library for Java]

- Языковые профили на основе данных из Wikipedia
- Определяет язык текста используя наивный байесовский фильтр
- 99%+ точность для 53 языков

THE APACHE® SOFTWARE FOUNDATION

```
let detector = LanguageDetector() // LanguageDetector
detector.AddLanguages("ru", "en")
detector.Detect("Мы на конференции DotNext")

detector.AddAllLanguages()
detector.Detect("Tøffelhelt")
detector.Detect("Æggekage")
```

01. Stanford.NLP.CoreNLP (74887)

02. Syn.Bot (30506)

03. AboditNLP (28543)

04. AboditUnits (21475)

05. Stanford.NLP.POSTagger (18541)

06. Stanford.NLP.NER (18256)

07. Stanford.NLP.Parser (16880)

08. OpenNLP (15033)

09. Language Detection (11171)

10. Autofac.Extras.AboditNLP (10978)

Abodit NLP

Abodit NLP это проект Ian Mercer'a. Проект начинался как персональный, с целью соединить системы умного дома с чат-системами.

Вместо задач общего понимания текста, фокусируется на задачах распознавания токенов и сопоставления их с реальными объектами с которым можно взаимодействовать (комнаты, датчики, системы управления и т.д.)

http://nlp.abodit.com

01. Stanford.NLP.CoreNLP (74887)

02. Syn.Bot (30506)

03. AboditNLP (28543)

04. AboditUnits (21475)

05. Stanford.NLP.POSTagger (18541)

06. Stanford.NLP.NER (18256)

07. Stanford.NLP.Parser (16880)

08. OpenNLP (15033)

09. Language Detection (11171)

10. Autofac.Extras.AboditNLP (10978)

Syn.Bot – OSCOVA & SIML

Автономный фреймворк для разработки ботов (с поддержкой .NET Standard, Xamarin, Mono).

Позволяет создавать ботов которые работают офлайн.

Поддерживают:

- диалоги
- распознавание намеренья
- распознавание сущностей

[Synthetic Intelligence Network]() a brand of [REVARN™ Cybernetics]().

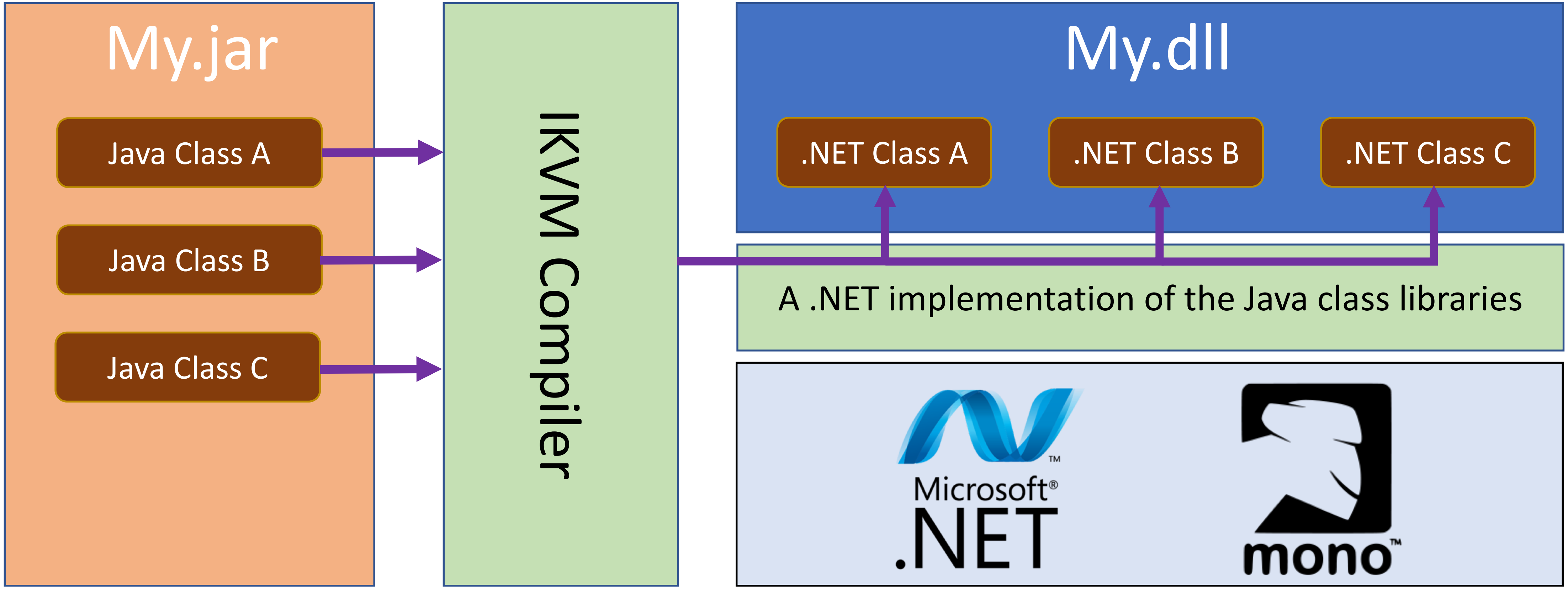
01. Stanford.NLP.CoreNLP (74887)
02. Syn.Bot (30506)
03. AboditNLP (28543)
04. AboditUnits (21475)
05. Stanford.NLP.POSTagger (18541)
06. Stanford.NLP.NER (18256)
07. Stanford.NLP.Parser (16880)
08. OpenNLP (15033)
09. Language Detection (11171)
10. Autofac.Extras.AboditNLP (10978)

Stanford NLP Group

https://nlp.stanford.edu

Stanford CoreNLP is written in **Java**; recent releases require **Java 1.8+**

IKMV.NET Compiler. http://www.ikvm.net



Терминология

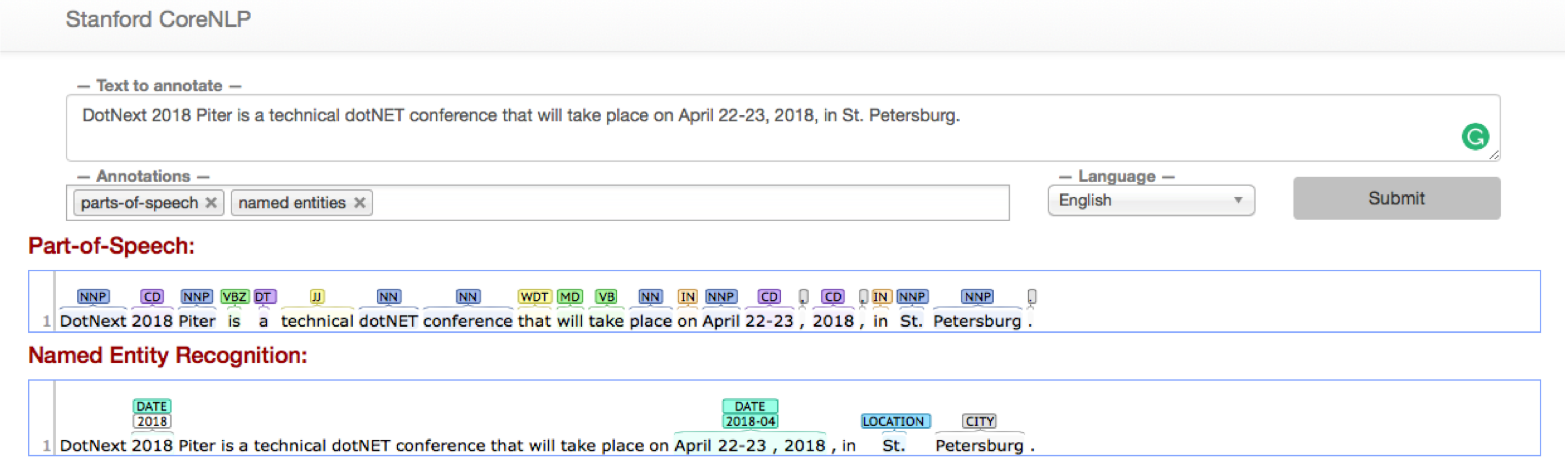
Аннотация – метаданные, ассоциированные с объектом (текстом, предложением, словом)

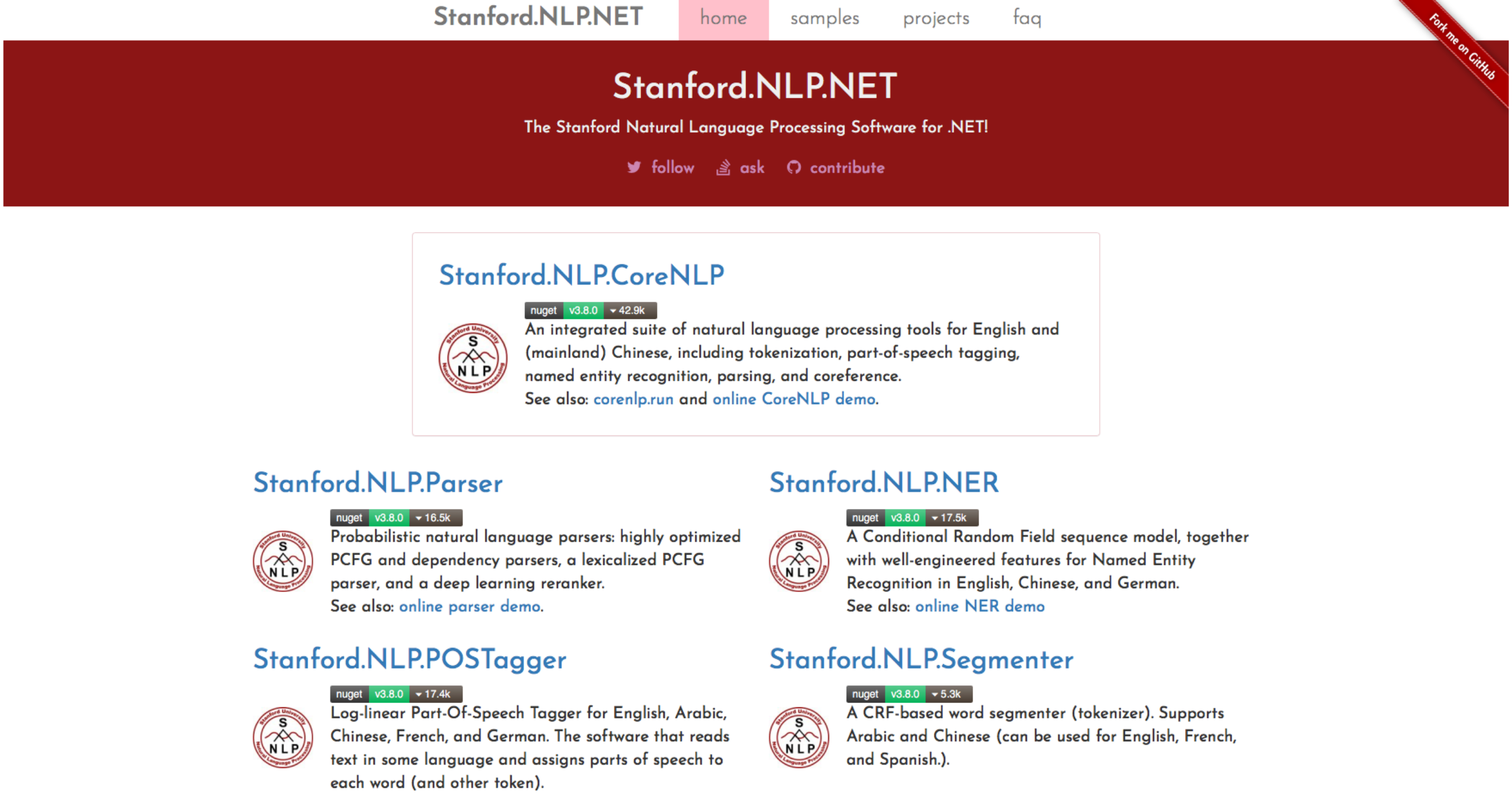
Аннотатор – объект который добавляет одну или несколько аннотаций к объекту (может иметь зависимости на другие аннотации)

Pipeline – последовательность аннотаторов

Stanford CoreNLP Server - http://corenlp.run

DotNext 2018 Piter is a technical dotNET conference that will take place on April 22-23, 2018, in St. Petersburg.





http://sergey-tihon.github.io/Stanford.NLP.NET/

Примеры или какой пакет выбрать?

Stanford CoreNLP

- What is Stanford.NLP.CoreNLP?
- Simple CoreNLP
- CoreNLP Server
- Building and using a Stanford CoreNLP pipeline
- SUTime - recognizing and normalizing time expressions

Related links:

- Sentiment analysis of tweets
- Sentiment Analysis in F#
- Splitting sentences in C# using Stanford.NLP
- Stanford CoreNLP is available on NuGet for F#/C# devs

// Подсказка: Вот этот! ;)

// P.S. Но не больше одного за раз !!!

~~Stanford Parser~~

- ~~What is Stanford.NLP.Parser?~~
- ~~Getting started with Stanford Parser~~

~~Relevant links:~~

- ~~Using a Java Framework with F#: The Stanford Parser for NLP~~
- ~~FSharp.NLP.Stanford.Parser justification or StackOverflow questions understanding.~~
- ~~NLP: Stanford Parser with F# (.NET).~~

~~Stanford Named Entity Recognizer (NER)~~

- ~~What is Stanford.NLP.NER?~~
- ~~Getting started with Stanford NER~~

~~Related links:~~

- ~~Extracting named entities in C# using the Stanford NLP Parser~~

~~Stanford Log-linear Part-Of-Speech (POS) Tagger~~

- ~~What is Stanford.NLP.POSTagger?~~
- ~~Getting started with Stanford POS Tagger~~

~~Stanford Word Segmenter~~

- ~~What is Stanford.NLP.Segmenter?~~
- ~~Getting started with Stanford Word Segmenter~~

Простой API для CoreNLP

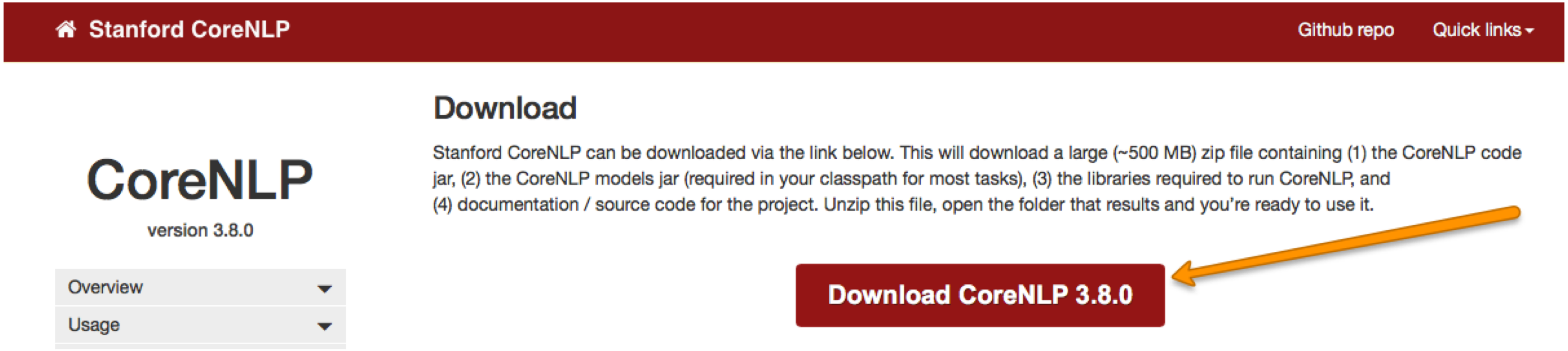
Функциональность	Аннотация	Класс	Метод
Sentence Splitting	ssplit	Document	**.sentences**() / **.sentence**(int)
Coreference Resolution	dcoref	Document	**.coref**()
Tokenization	tokenize	Sentence	**.words**() / **.word**(int)
Part of Speech Tagging	pos	Sentence	**.posTags**() / **.posTag**(int)
Lemmatization	lemma	Sentence	**.lemmas**() / **.lemma**(int)
Named Entity Recognition	ner	Sentence	**.nerTags**() / **.nerTag**(int)
Constituency Parsing	parse	Sentence	**.parse**()
Dependency Parsing	depparse	Sentence	**.governor**(int) / **.incomingDependencyLabel**(int)
Natural Logic Polarity	natlog	Sentence	**.natlogPolarities**() / **.natlogPolarity**(int)
Open Information Extraction	openie	Sentence	**.openie**() / **.openieTriples**()

Простой API для CoreNLP

Функциональность	Аннотация	Класс	Метод
Sentence Splitting	ssplit	Document	**.sentences**() / **.sentence**(int)
Coreference Resolution	dcoref	Document	**.coref**()
Tokenization	tokenize	Sentence	**.words**() / **.word**(int)
Part of Speech Tagging	pos	Sentence	**.posTags**() / **.posTag**(int)
Lemmatization	lemma	Sentence	**.lemmas**() / **.lemma**(int)
Named Entity Recognition	ner	Sentence	**.nerTags**() / **.nerTag**(int)
Constituency Parsing	parse	Sentence	**.parse**()
Dependency Parsing	depparse	Sentence	**.governor**(int) / **.incomingDependencyLabel**(int)
Natural Logic Polarity	natlog	Sentence	**.natlogPolarities**() / **.natlogPolarity**(int)
Open Information Extraction	openie	Sentence	**.openie**() / **.openieTriples**()

Не очень-то просто на самом деле

Запуск CoreNLP Server локально



 # Запуск сервера используя все jar файлы текущей директории

 # В корневой директории распакованного zip архива, используя Java8

java -mx4g –cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer –port 9000 –timeout 15000

https://stanfordnlp.github.io/CoreNLP/corenlp-server.html

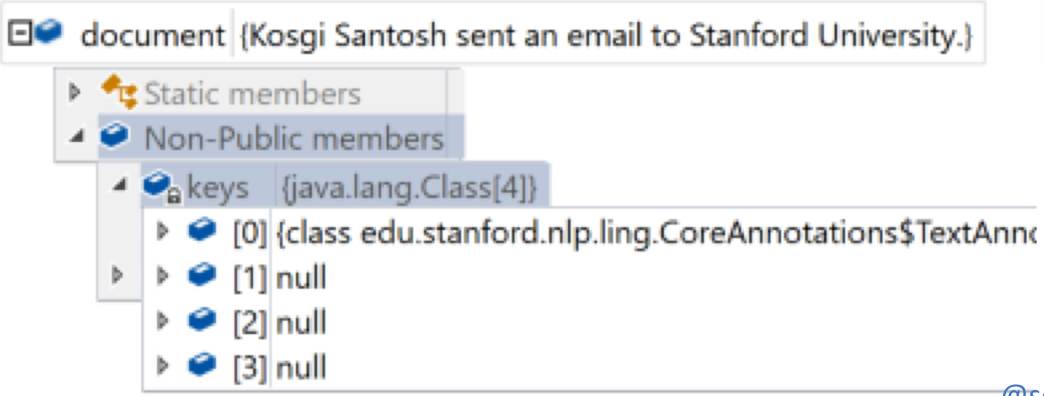
CoreNLP Server REST API

```fsharp
type StanfordResp = FSharp.Data.JsonProvider<"04_CoreNlp_sample_resp.json">

let http = new HttpClient() // HttpClient
let call text = async { // string -> Async<FSharp.Data.JsonProvider<...>.Root>
    let uri = Uri("""http://localhost:9000/?properties={"annotators": "tokenize,ssplit,pos,lemma,ner,parse,dcoref","outputFormat": "json"}""")
    use body = new StringContent(text, Encoding.UTF8)
    let! resp = http.PostAsync(uri, body) |> Async.AwaitTask
    let! str = resp.Content.ReadAsStringAsync() |> Async.AwaitTask
    return StanfordResp.Parse str
}

let resp =  // FSharp.Data.JsonProvider<...>.Root
    call "the quick brown fox jumped over the lazy dog"
    |> Async.RunSynchronously

for sentence in resp.Sentences do
  for token in sentence.Tokens do
    printfn "%s - POS='%s'" token.Word token.Pos
```

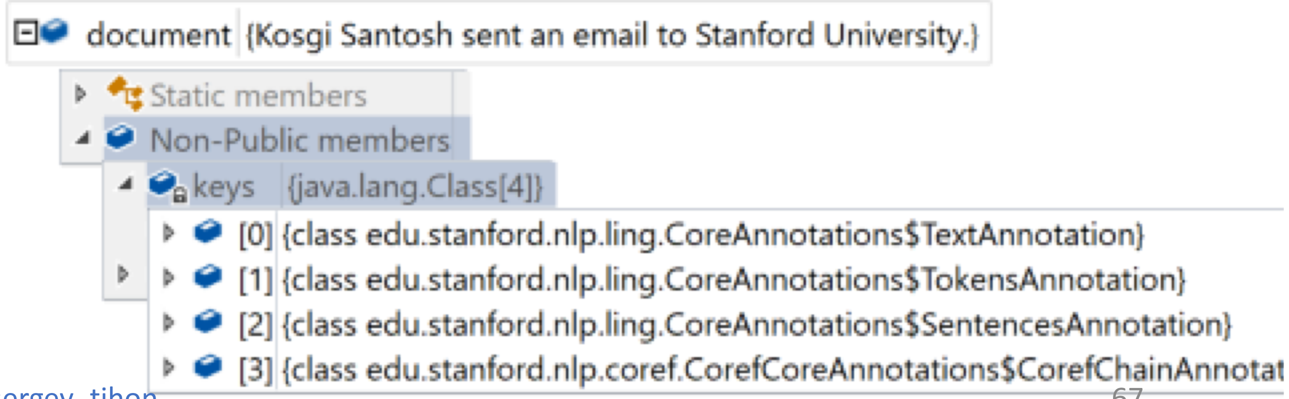
Клиент для Core NLP Server

```
// Конфигурируем сервер, описываем annotation pipeline
var props = new java.util.Properties();
props.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse, dcoref");
var pipeline = new StanfordCoreNLPClient(props, "http://localhost", 9000, 2);

var text = "Kosgi Santosh sent an email to Stanford University.";
// Создаем пустую Annotation, которая содержит только текст для аннотации
Annotation document = new Annotation(text);
// пропускаем текст через pipeline чтобы получить все аннотации
pipeline.annotate(document);
```

До pipeline.annotate:

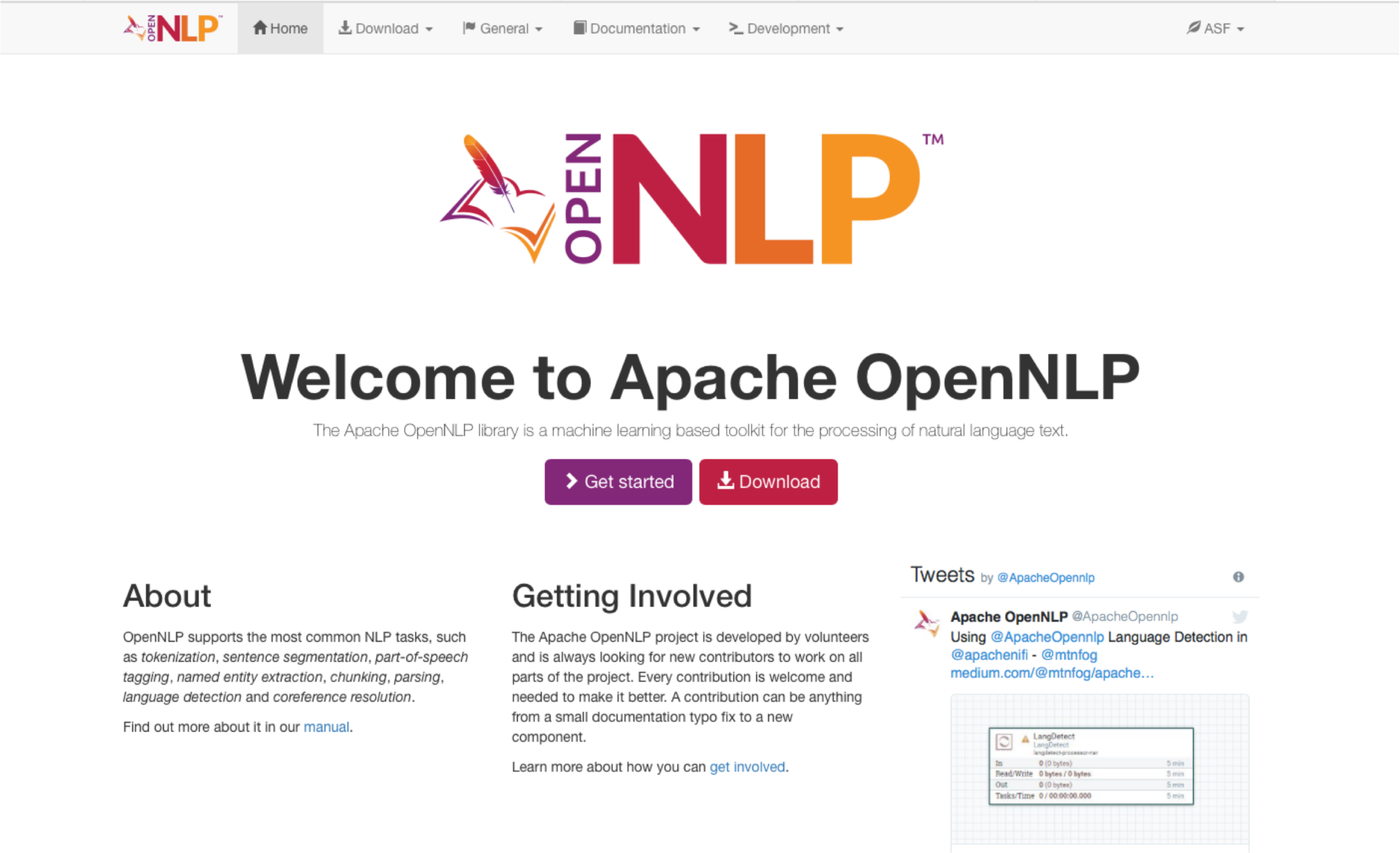


После pipeline.annotate:



01. Stanford.NLP.CoreNLP (74887)
02. Syn.Bot (30506)
03. AboditNLP (28543)
04. AboditUnits (21475)
05. Stanford.NLP.POSTagger (18541)
06. Stanford.NLP.NER (18256)
07. Stanford.NLP.Parser (16880)
08. OpenNLP (15033)
09. Language Detection (11171)
10. Autofac.Extras.AboditNLP (10978)

OpenNLP



https://opennlp.apache.org

Описание:

- Language Detector
- Sentence Detector
- Tokenizer
- Name Finder (NER)
- Document Categorizer
- Part-of-Speech Tagger
- Lemmatizer
- Chunker
- Parser
- Coreference Resolution

Мануалы по тренировке новых моделей!

Open NLP для .NET

Knuppe.SharpNL

Независимое переписывание Apache OpenNLP на C#:

- Поддержка .net standard 2.0
- Обратно совместима с моделями от
- Переписывается вручную (без автоматических тулов)

https://github.com/knuppe/SharpNL

OpenNLP.NET

Хорошо:

- Полностью совпадает по функционалу с Java версией (1.9.1)

Плохо:

- Нет поддержки .net core
- Зависит от IKVM.NET

https://github.com/sergey-tihon/OpenNLP.NET

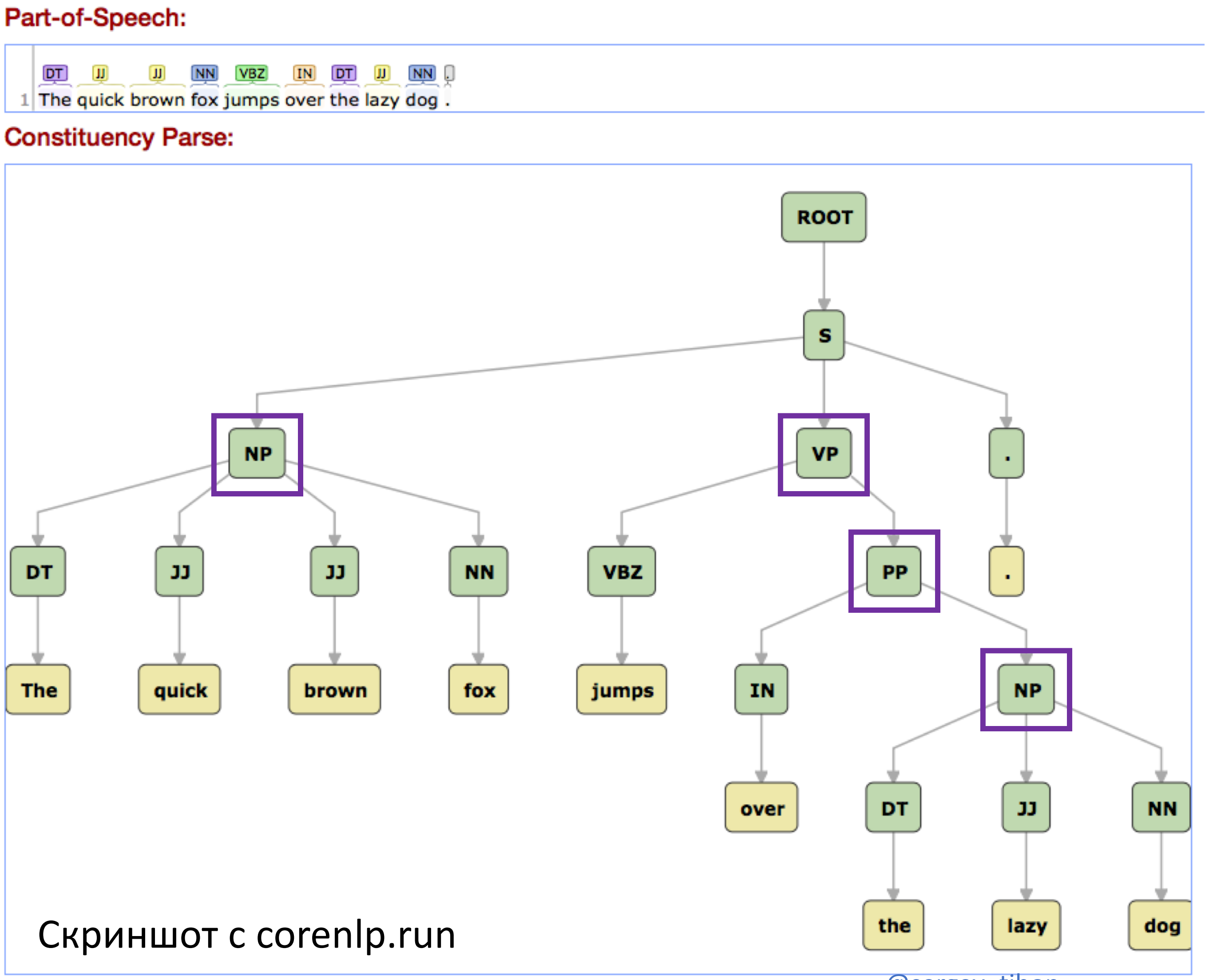
OpenNLP.NET Парсер

```
let parser =  // Parser
    let fileName = Common.PaketFiles.``opennlp.sourceforge.net``.``en-parser-chunking.bin``
    use fileInStream  = new FileInputStream(fileName)
    let parserModel = new ParserModel(fileInStream)
    ParserFactory.create(parserModel)

let sentence = "The quick brown fox jumps over the lazy dog ." // string
let topParses = ParserTool.parseLine(sentence, parser, 1) // Parse []

let tree = topParses.[0] // Для первого предложения // Parse

let treeStr = // string
    let sb = new java.lang.StringBuffer()
    tree.show(sb)
    sb.ToString()
printfn "%s" treeStr
```

Standard Output

```
"(TOP (NP (NP (DT The) (JJ quick) (JJ brown) (NN fox) (NNS jumps)) (PP (IN over) (NP (DT the) (JJ lazy) (NN dog))) (. .)))"
```

Дерево разбора предложения



Скриншот с corenlp.run

Penn Treebank II tag set

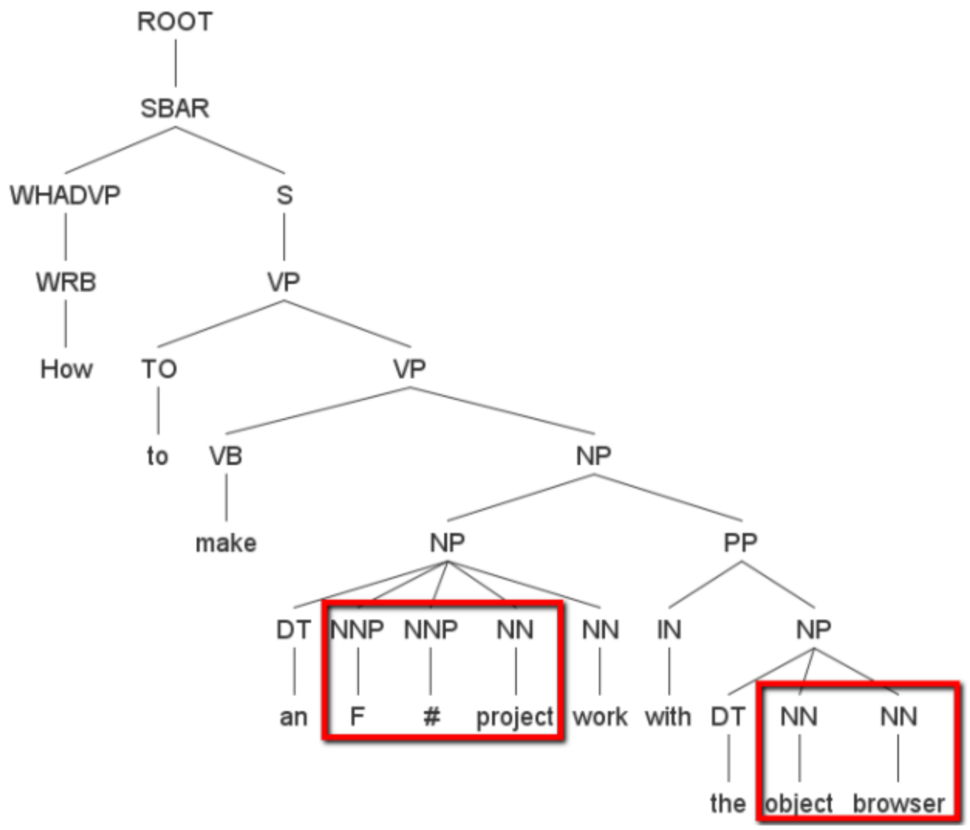
Part-of-speech tags (Части речи)

- DT – determiner (определяющее слово)
- JJ – adjective (прилагательное)
- NN – noun (существительно)
- VBZ - verb, 3rd person singular present (глагол, 3ее лицо, ед. число)
- IN - conjunction, subordinating or preposition (союз, предлог)

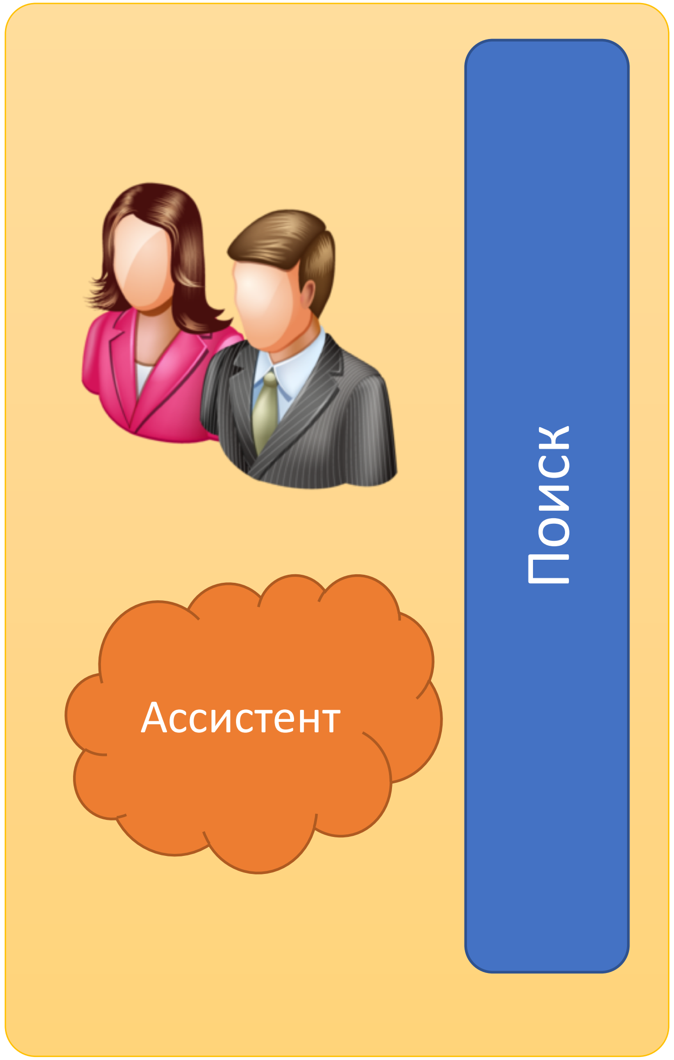
Chunk tags (Групповые теги)

- NP – noun phrase (словосочетание)
- PP – prepositional phrase (фраза с предлогом)
- VP – verb phrase (глагольная фраза)

Понимание вопросов



1. Получить вопрос
2. Построить синтаксическое дерево
3. Выделить ключевые слова и фразы
4. Построить запрос в поисковую систему



http://sergey-tihon.github.io/Stanford.NLP.Fsharp/so_questions.html

Making Tree Kernels practical for Natural Language Learning
http://www.aclweb.org/anthology/E06-1015

Обучение модели распознавания токенов (TokenNameFinderModel, NER)

```
// The training data should contain at least 15000 sentences to create a model which performs well
let train (inputFile:string) =  // string → TokenNameFinderModel
    let factory =
        { new InputStreamFactory with
            member __.createInputStream () =
                new FileInputStream(inputFile) :> InputStream }
    let lineStream = new PlainTextByLineStream(factory, StandardCharsets.UTF_8)
    use sampleStream = new NameSampleDataStream(lineStream)
    let nameFinderFactory = new TokenNameFinderFactory()

    let trainingParameters = new TrainingParameters();
    //trainingParameters.put(TrainingParameters.ITERATIONS_PARAM, "5");
    //trainingParameters.put(TrainingParameters.CUTOFF_PARAM, "200");

    NameFinderME.train ("en", "person", sampleStream, trainingParameters, nameFinderFactory)
```

Итого

Советы

1. **Проверяйте наличие облачного API решающего вашу задачу**
    1. Проще всего использовать
    2. Цена зависит от количества запросов
    3. Нужно оправлять данные за пределы организации
    4. Чаще всего модели не настраиваются под ваши данные

Советы

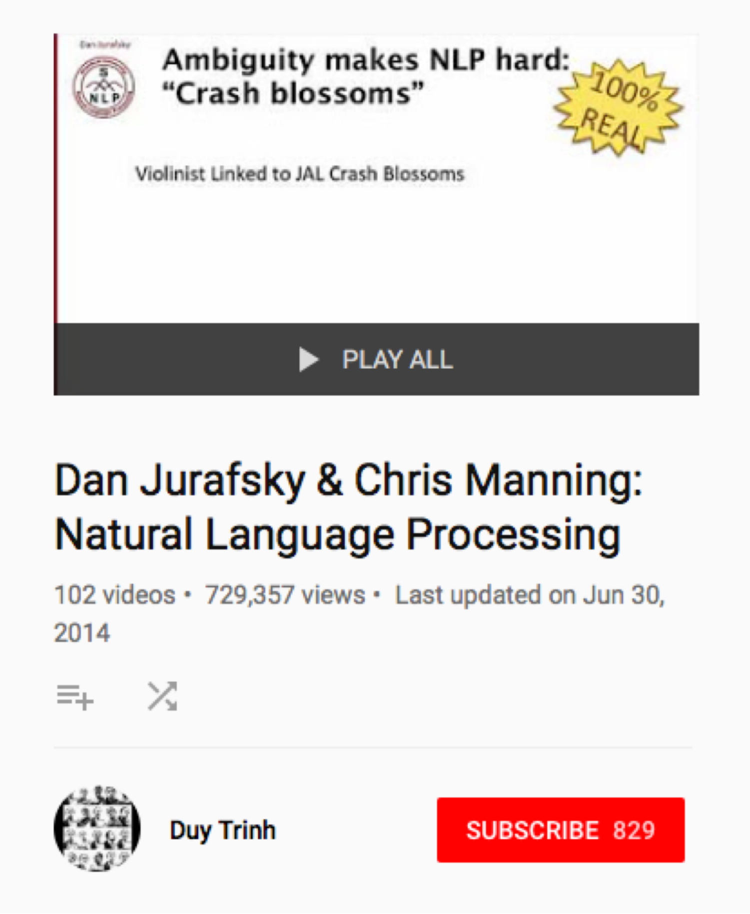
1. Проверяйте наличие облачного API решающего вашу задачу
    1. Проще всего использовать
    2. Цена зависит от количества запросов
    3. Нужно оправлять данные за пределы организации
    4. Чаще всего модели не настраиваются под ваши данные
2. **Прототипируйте / проверяйте идеи со Stanford NLP**
    1. Качественные модели
    2. Pipeline обработки данных расширяем, достаточно много примеров
    3. Лицензия GLP v3 (платно для коммерческого использования)

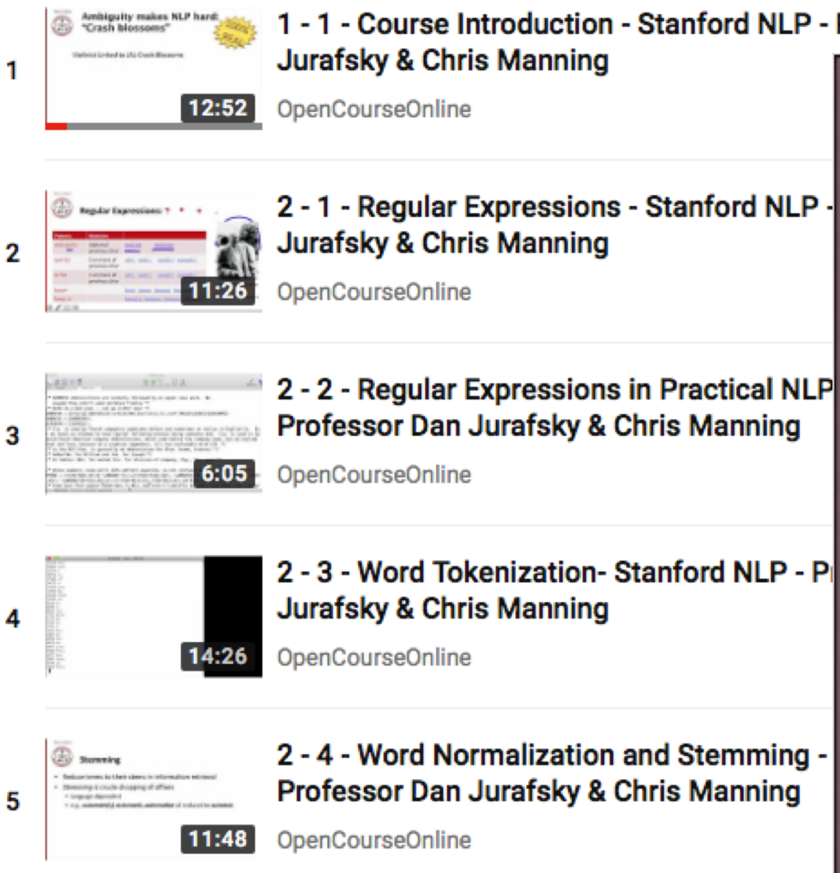
Советы

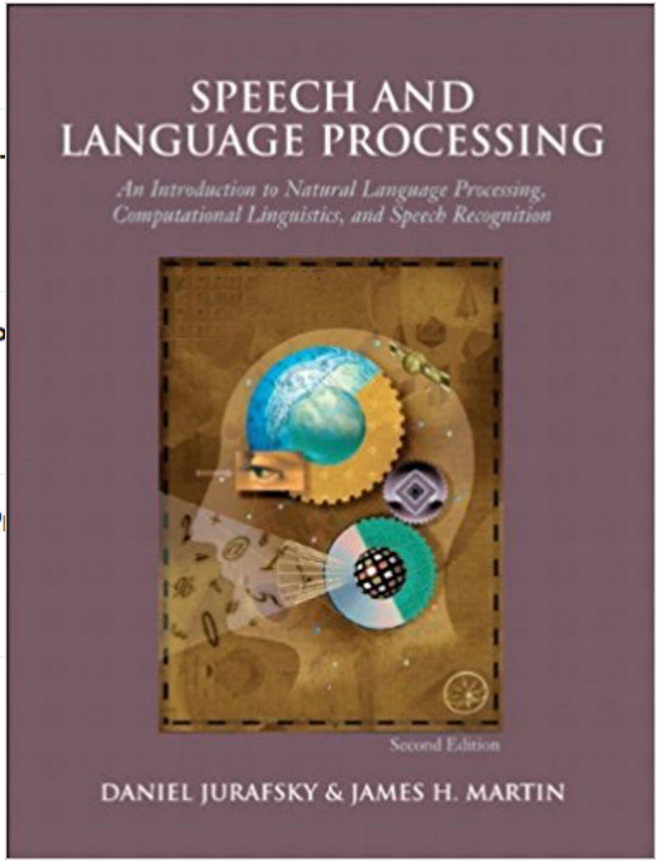
1. Проверяйте наличие облачного API решающего вашу задачу
    1. Проще всего использовать
    2. Цена зависит от количества запросов
    3. Нужно оправлять данные за пределы организации
    4. Чаще всего модели не настраиваются под ваши данные
2. Прототипируйте / проверяйте идеи со Stanford NLP
    1. Качественные модели
    2. Pipeline обработки данных расширяем, достаточно много примеров
    3. Лицензия GLP v3 (платно для коммерческого использования)
3. **Используйте OpenNLP в остальных случаях**
    1. Лицензия Apache (всегда бесплатно)
    2. Есть реализация под .NET Core

Матчасть

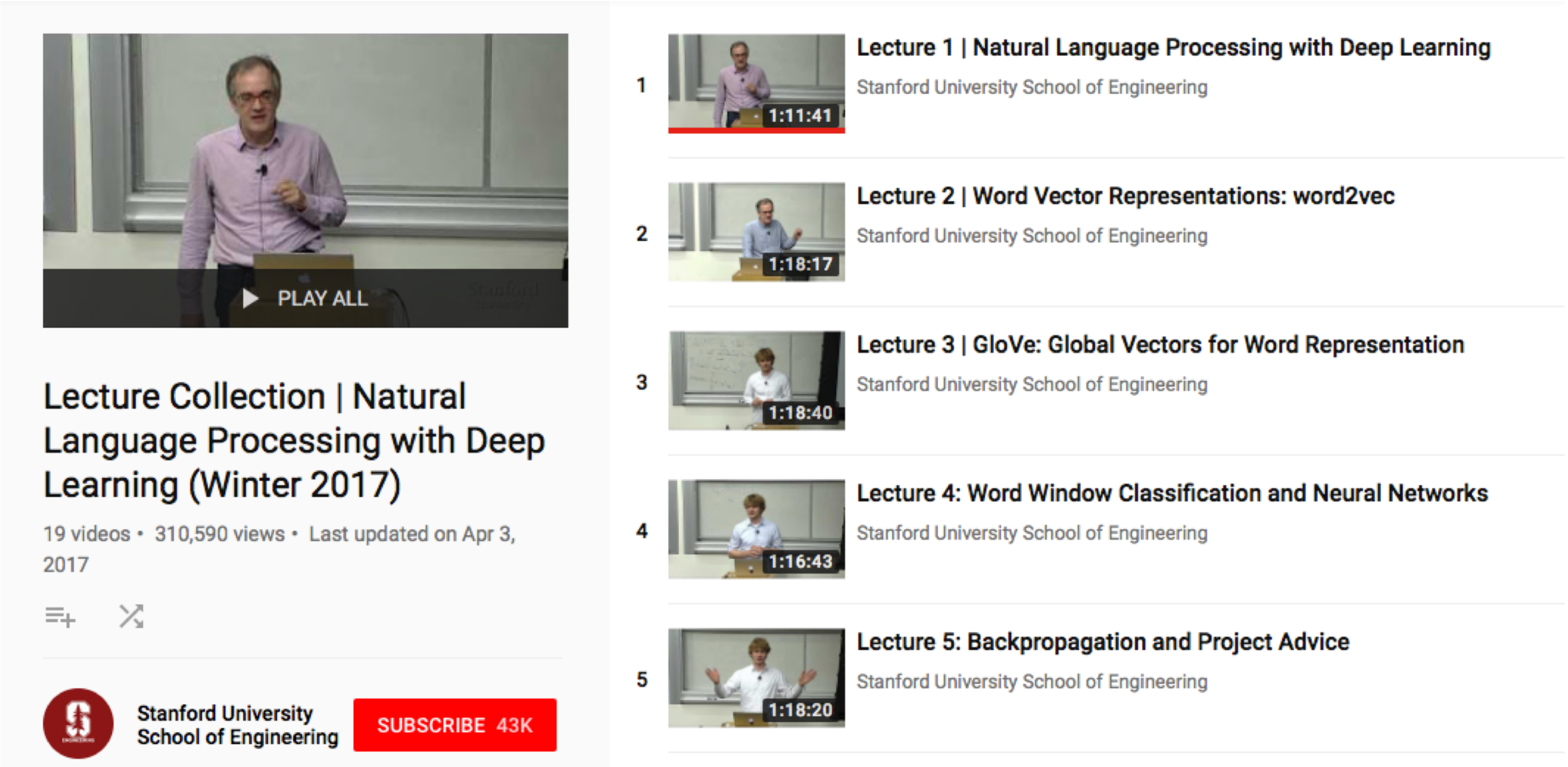
[Dan Jurafsky & Chris Manning: Natural Language]







[Natural Language Processing with Deep Learning (Winter 2017)](http://web.stanford.edu/class/cs224n/)



http://web.stanford.edu/class/cs224n/

Вопросы?

Спасибо!

@sergey_tihon
sergey.tihon@gmail.com