

IBM **Watson**

Natural Language Classifier

Links, Best Practices, Source Code, and Tools



Welcome to the Natural Language Classifier! This document will help you get started.

Page	Topic
	Introduction
3	Documentation, Community, and Demos with Source Code
	Tools
4	Ground Truth Management Tool
	Demos
5	Build a Harry Potter sorting hat
	Best Practices
6	Obtain end user questions ASAP
7	Train users to ask natural language questions
8	What's the best naming convention for intent names?
9	Bootstrapping your next training set
10	NLC Confidences Scores Sum to 100%
	Design Patterns
11	Classify Complex Text with a Multi-Intent Classifier
12	Chaining multiple classifiers

Page	Topic
	Design Patterns Continued
13	Answer FAQs with NLC and let R&R answer the "long tail"
14	Combine Dialog + NLC + R&R
15	Answer Detailed Questions Using NLC + Entity Extraction
16	Using NLC on large pieces of text through decomposition
	Use Cases
17	Perform actions for users using NLC+Dialog
18	Topic Modeling Using the NLC

More Handbooks

Intro to Watson Development

<https://ibm.box.com/s/nav52vt6q2xwib5zqwupwjf78mxtgems>

Retrieve and Rank (R&R) Handbook

<https://ibm.box.com/s/n0lqowt0v97nxb5mtei6qrbkktbdt2dm>

Personality (PI) Handbook

<https://ibm.box.com/s/6h8dxsc3pq5idtgehjb6fwh7vbejsvtc>



Docs, Tools, and Community

NLC on the WDC: <https://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/nl-classifier.html>
Read the NLC section of the Watson Developer Cloud to learn all about the service.

NLC on the Stack overflow: <http://stackoverflow.com/questions/tagged/ibm-watson>
Read questions and and get answers from other Watson NLC developers.

Technical Webinars: <http://www.pages03.net/ibmwatson/building-with-watson-web-series>
Sign up for the upcoming webinars on how to build conversational apps using Watson

Webinar: Interpreting Language Using the NLC: <https://goo.gl/5SFLmn>
A recent webinar on by Rahul A. Garg, Watson Offering Manager for the NLC

Code Libraries & SDKs

NodeJS Watson Library: <https://goo.gl/2nhR1n>
Kick start your Watson NodeJS development.

Java Watson Library: <https://github.com/watson-developer-cloud/java-sdk>
A library of Java utilities to jump start your Watson development.

iOS/Swift Watson Library: <https://github.com/watson-developer-cloud/ios-sdk>
Checkout this NodeJS library to kick start your Watson development.

Demos w/source code

<https://goo.gl/YEBfG0>

<https://goo.gl/OgfsN2>

<https://goo.gl/ARnUwJ>

Questions on the Natural Language Classifier

<https://goo.gl/IUIO9G>
Architecture: <http://goo.gl/BLNtbM>



NLC Ground Truth Management Tool

You can now let non-technical people train classifiers and manage their ground truth. This NLC tooling allows you to quickly upload and preview your ground truth while quickly viewing existing classifiers. Go here to get started:

http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/nl-classifier/tool_overview.shtml

The screenshot shows the IBM Watson Natural Language Classifier Ground Truth Management Tool interface. The top navigation bar includes 'Training data', 'Classifiers', and user icons. A message bar at the top states 'Import Complete nlc_factoid_training.csv import complete.' The main interface is divided into two panels: 'Classes' and 'Texts'.

Classes Panel: Shows '1 of 13 selected'. It includes a 'Delete' button, a search icon, and a 'Newest first' sort option. Below these is an 'Add class' button. A list of classes is displayed:

Class Name	Count
health-condition_cause	12
person-birthdate	8
person-birthplace	7
person-children	4

Texts Panel: Shows '5 of 78 selected'. It includes a 'Create classifier' button, upload/download/delete icons, and a 'Selected Classes' dropdown showing 'action-sms-create 398'. It also has an 'Assign classes' button, a 'Delete' button, a search icon, and a 'Newest first' sort option. Below these is an 'Add text' button. A list of texts is displayed:

Text	Class
*'s husband?	person-spouse
*'s spouse?	person-spouse

Build a Harry Potter Sorting Hat

Impress your kids and your colleagues with the power of the NLC

This fun and creative demo shows how a very simple training set can roughly discern your “personality type” enough to determine which Hogwarts house you belong too. The training dataset plus code in R are available for download



Blog: https://dreamtolearn.com/ryan/data_analytics_viz/98

Video: <https://www.youtube.com/watch?v=7SoUSpxONcq>

Best Practice

Obtain End User Text/Questions ASAP**What's the first mistake most R&R implementations make?**

That's right. They spend too much time working with pseudo-questions generated internally to bootstrap R&R. Don't let yourself spend too much time before getting your application in-front of real end-users to validate your assumptions about how your users will ask questions.

Anecdotally, 99% of implementations waste 75% of their time generating ground truth that doesn't properly match end user needs. Validate soon and validate often!

“Remember that all models are wrong.
The practical question is how wrong do
they have to be to be useful?”



George Box, 1987

Don't wait for perfection. Present your users with “Good Enough”.

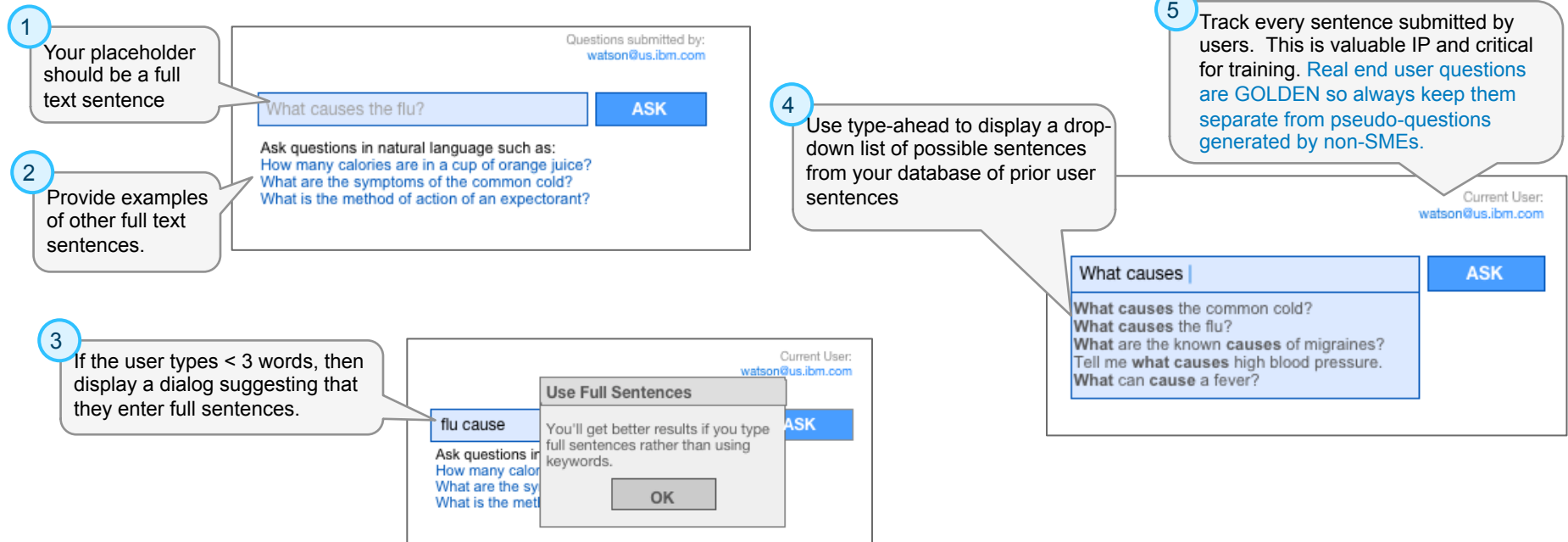
Pay close attention to George Box's quote and get in-front of your end users ASAP. Often a basic system is still useful enough to start asking meaningful questions. I.e. you need to know quickly understand your user's workflow, what questions/text they'll submit, and how it's worded. Normally a partial system will still be enough to validate and extend your initial assumptions.

Best Practice

Train users to ask natural language questions

Modify your user's default behaviors. No more keywords. Use full sentence.

If your UI has a text box, your users will likely default to entering keywords as they do with Google. You'll need to modify their behavior so they use full sentences. Here are a few recommended ways to achieve this.



Best Practice

What's the best naming convention for intent names?

Take time to establish a consistent naming convention for your intents. First, determine a hierarchy that groups similar intents. We recommend a naming convention with the broader category at the left moving to subcategories on the right:

<u>Text</u>		<u>Intent</u>	
Where is the fitness center?	→	fitness_center-location	Separate categories by dashes and replaces spaces between words with underscores
Where is the business center	→	business_center-location	
When does the fitness center open?	→	fitness_center-open_time	
When does the fitness center close?	→	fitness_center-close_time	
What are the business center hours?	→	business_center-hours	

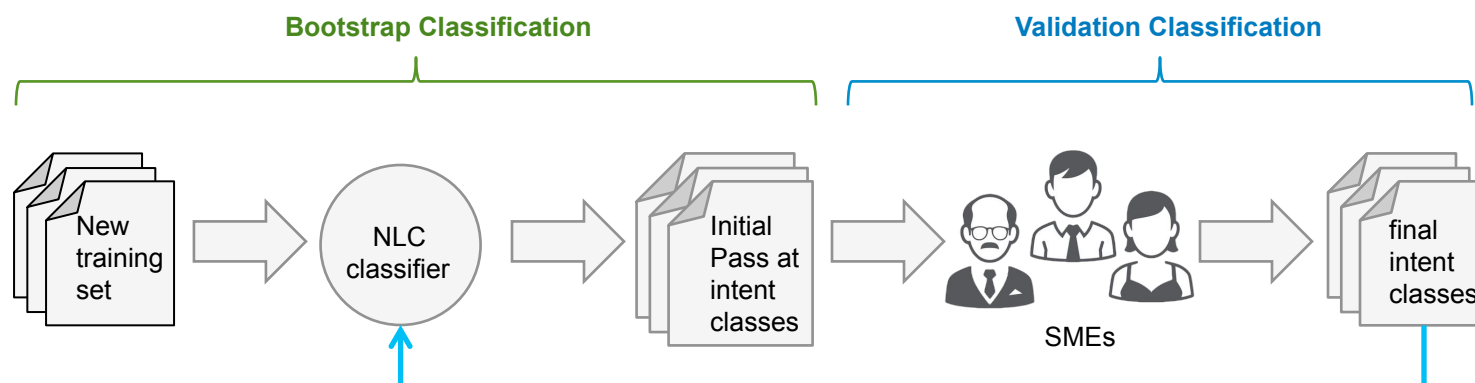
Next look for subtypes that are common between intents. E.g. “location” and “hours” in the examples above. These subtypes could span multiple levels (as below), but we recommend not going beyond 4 levels in your intent hierarchy. Having good intent names will allow you to rapidly sort/search for all intents of a given type.

Can you send 2 pillows to my room?	→	housekeeping-linens-pillows
Just spilled coffee on my bed sheets. Help!	→	housekeeping-linens-sheets

Best Practice

Bootstrap your NLC Training

Are you tired of spending so much time training your NLC classifiers?
Wouldn't it be great if prior training could speed-up the next round of training?

**Try bootstrapping your next round of NLC training**



Before you perform the next round of NLC training. Run your next set of training data against your current classifier and let it take a first pass at classification. Now go through this pre-classified training set and you only need to edit the classes that were mislabeled. Be extra careful not to simply accept the classes created by this bootstrapped classification. We don't want the classifier to skew your training! Also pay attention to where your classifications are failing/succeeding as that can guide your future training efforts!

Best Practice

NLC Confidences Scores Sum to 100%

? What confidence is returned by a classifier trained with only 1 intent class?

A single class NLC classifier would always return that intent class with confidence = 100% regardless of the text submitted. Let's see this with a simplified example where we train the NLC to recognize A and then submit X for classification.

If **A**   then given unrelated **X**, NLC will always return 100% confident the answer is **A**.

? What about training with 2 intent classes then submitting input text that's absolutely not related to either?

So this time, let's train with A and B.

If **A** and **B**   then given unrelated **X**, NLC will return 50% confident the answer is **A** and 50% it is **B**.

Similarly for more intent classes.

An NLC trained with 100 different intents classes would return confidence = 1% for each class when given text unrelated to the trained classes. This may sound like an odd usage of the idea of confidence, but it's incredibly useful. See the Design Pattern on how to "[Classify Complex Text with a Multi-Intent Classifier](#)" to learn more.

★ Trick Question: What if you trained the NLC on 10 classes and NLC was 50% the answer was either of two of them

It means the NLC is highly confident your text belongs equally to these classes. So you should consider two actions: (1) review the semantic overlap of the classes to determine if they are really distinct and (2) if they are distinct then use Watson Dialog service to ask the user which of the two options is actually the correct one as the user's text may ambiguously refer to both intents.

Design Pattern

Classify Complex Text with a Multi-Intent Classifier

How can we analyze text like: “Show me pink Audi convertibles”

We want to extract three intents from this text: color, model, and vehicle type. And we can do this by assigning three classes to our NLC training set. For example something like this:

Text	Class 1	Class 2	Class 3
Show me pink Audi convertibles	vehicle_color_pink	vehicle_model_audi	vehicle_type_convertible
Do you have blue BMW motorcycles?	vehicle_color_blue	vehicle_model_bmw	vehicle_type_motorcycle
Let me see pink BMW convertibles	vehicle_color_pink	vehicle_model_bmw	vehicle_type_convertible
...



Now use the fact that all NLC confidences always sum to 100%!

If the heading above doesn't make sense, look at the Best Practices slide on “NLC Confidences Scores Sum to 100%.” That will help you understand this next conceptual example where going to train the classifier

If **A1, A2, B1, B2, C1 & C2**   then given **A1+B2+C1**, NLC will return 33% confident the answer is **A1**, 33% it's **B2** and 33% it's **C1**!

★ **And the trick? Let A = vehicle colors, B = vehicle models, and C = vehicle types!**

Yes. You've just trained your classifier to distinguish three intent classes with one call! You can now create a virtual vehicle sales agent that classifies user text by color, model, and vehicle type in a single NLC call. For our query above of “Show me pink Audi convertibles”, first separate the classes in your NLC response by each type (color, model, type) and normalize each intent category to 100%. This would allow you to determine vehicle_color=“pink” at 100%, vehicle_model=“Audi” at 100%, and vehicle_type=“convertible” at 100%.

Design Pattern

Chaining Multiple NLC Classifiers

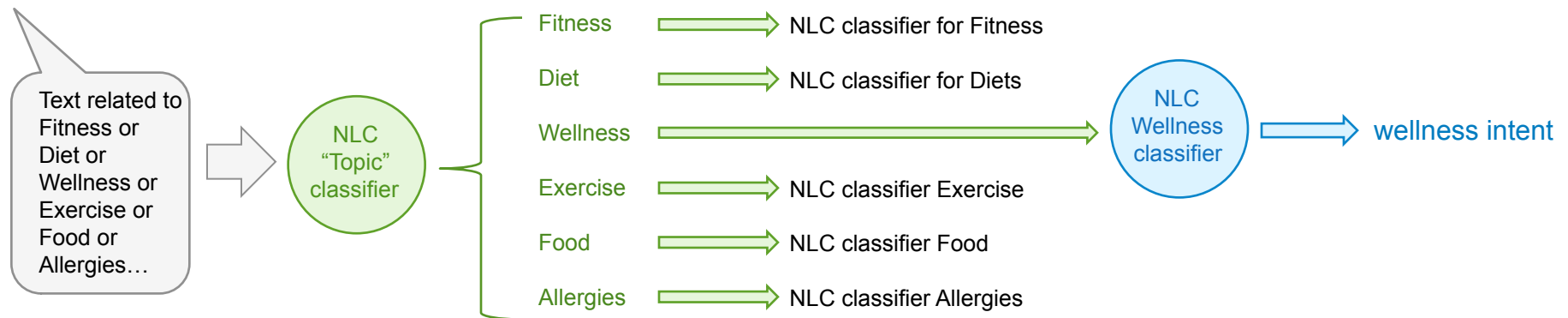
Do you have more classes than allowed by a single classifier?

Do your text classes have higher level categories that are easily separated?

Is training a single classifier becoming increasingly difficult to manage?

Then perhaps a multi-classifier solution is a solution for you:

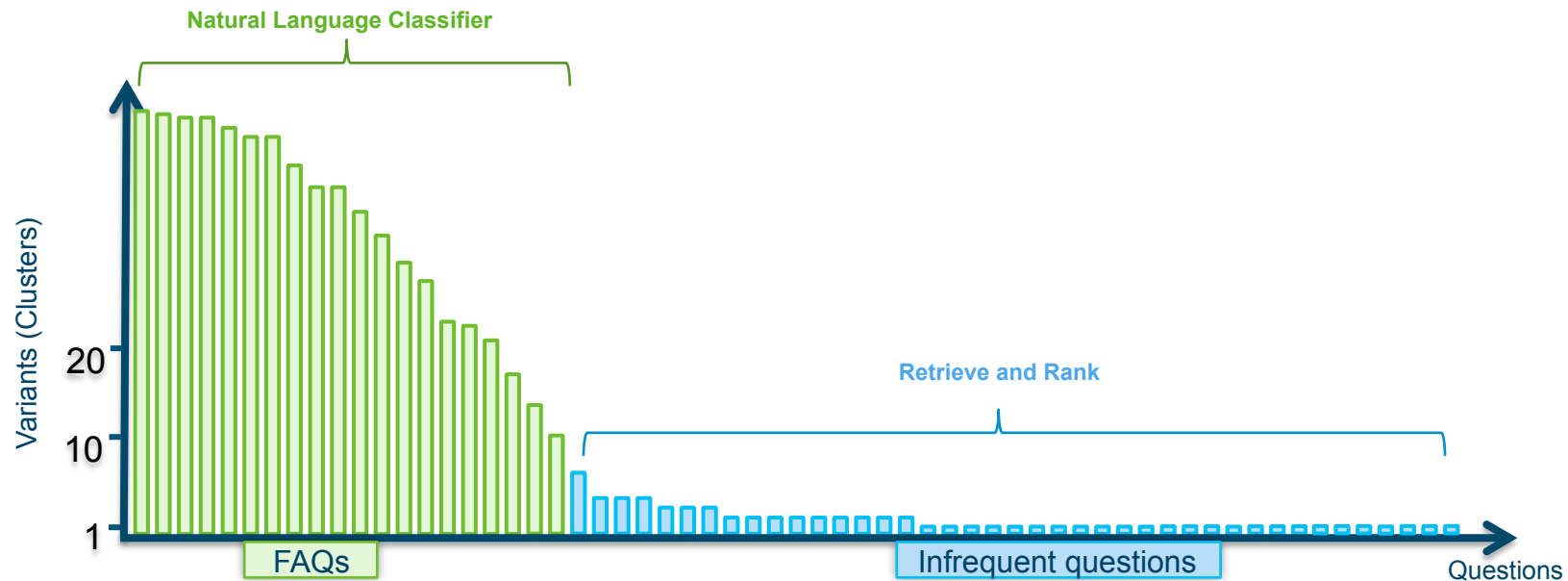
The initial classifier would perform a high-level separation of your text so that classifiers at the next level can separate your classes with higher confidence.



Design Pattern

Answer FAQs with NLC and let R&R answer the “long tail”

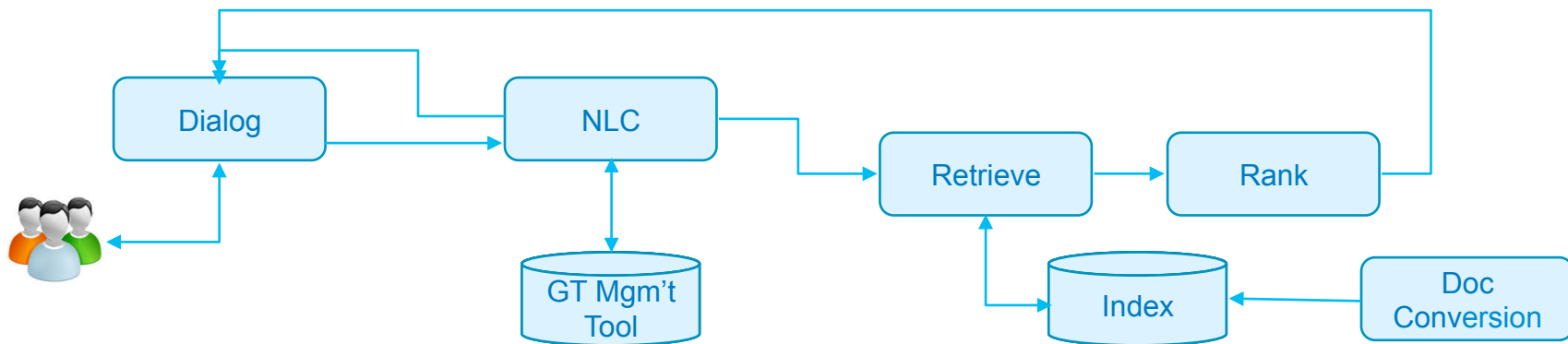
The Natural Language Classifier is well suited for Frequently Asked Questions (FAQs) where the effort to associate a single static answer to a question is rapidly rewarded. Retrieve to Rank is then used for infrequently asked questions or those for which multiple passages or frequently changing content must be searched.



Design Pattern

Combine Dialog + NLC + R&R

Combine Retrieve and Rank w/Dialog and NLC. Dialog provides the ability for a multi-turn experience where you asking clarifying questions to customers and track state across queries. NLC can be used to (1) detect specific domains of user interest so R&R can search only a subset of documents, (2) detect overlap between possible user intents so Dialog can request clarification by the user, or perhaps (3) NLC can be used to inject valuable run-time features into R&R for more targeted ranking of answers



Design Pattern

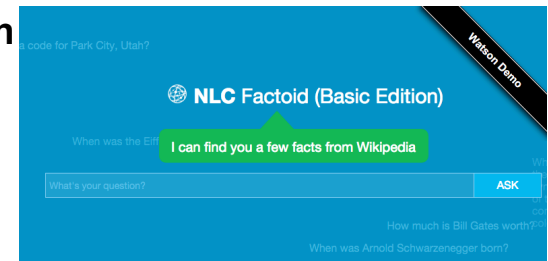
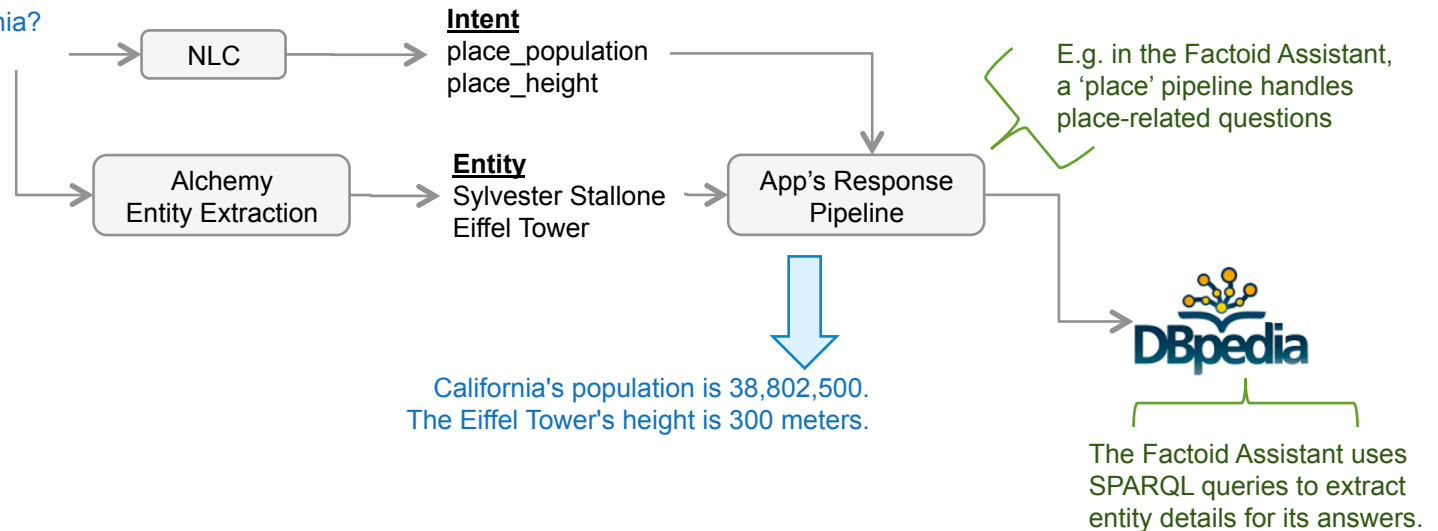
Answer Detailed Questions Using NLC + Entity Extraction

So you have the user's intent from NLC, what's next? Use entity extraction to obtain other useful details from the user's text. The NLC Factoid Assistant demo does this for entities contained in Dbpedia.

Demo: <http://goo.gl/u4jXiT>. Source code: <https://goo.gl/ARnUwJ>.

First identify the intent and entities being asked about:

What's the population of California?
How tall is the Eiffel Tower?

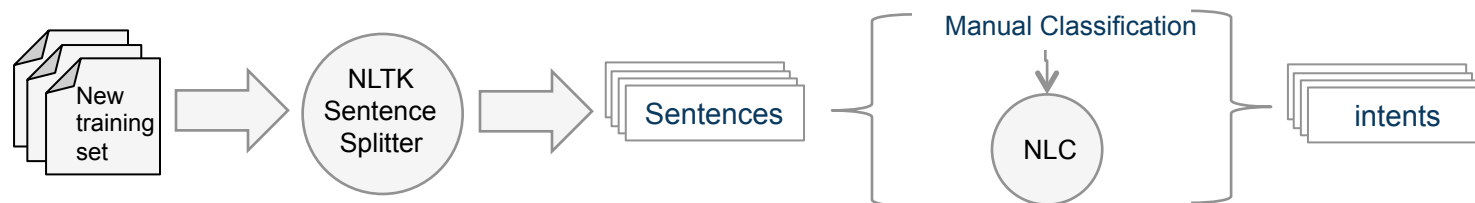


Design Pattern

Using NLC on large pieces of text through decomposition

NLC works with sentences < 1,024 characters so it won't directly work with long pieces of text such as emails. Long text can be broken into sentences - classifying each sentence and then aggregating the result into a summary of the message. There are a number of open source sentence segmentation tools. [NLTK works very well for this purpose](#)

The training phase requires manual annotation of a set of test documents (start with a couple of hundred - it should take a couple of hours to plow through them and assign a class for each sentence).



The production flow will use the trained classifier to automatically do the sentence level classification and then spit out a summary - you can use business rules to trigger actions at this point.

For an example use case where we detect the severity of complaint emails and rank them from minor to severe:

If the email contains 10% or higher "severe" sentences then prioritise triage.

If the email contains 90% "minor" sentences then leave at bottom of queue

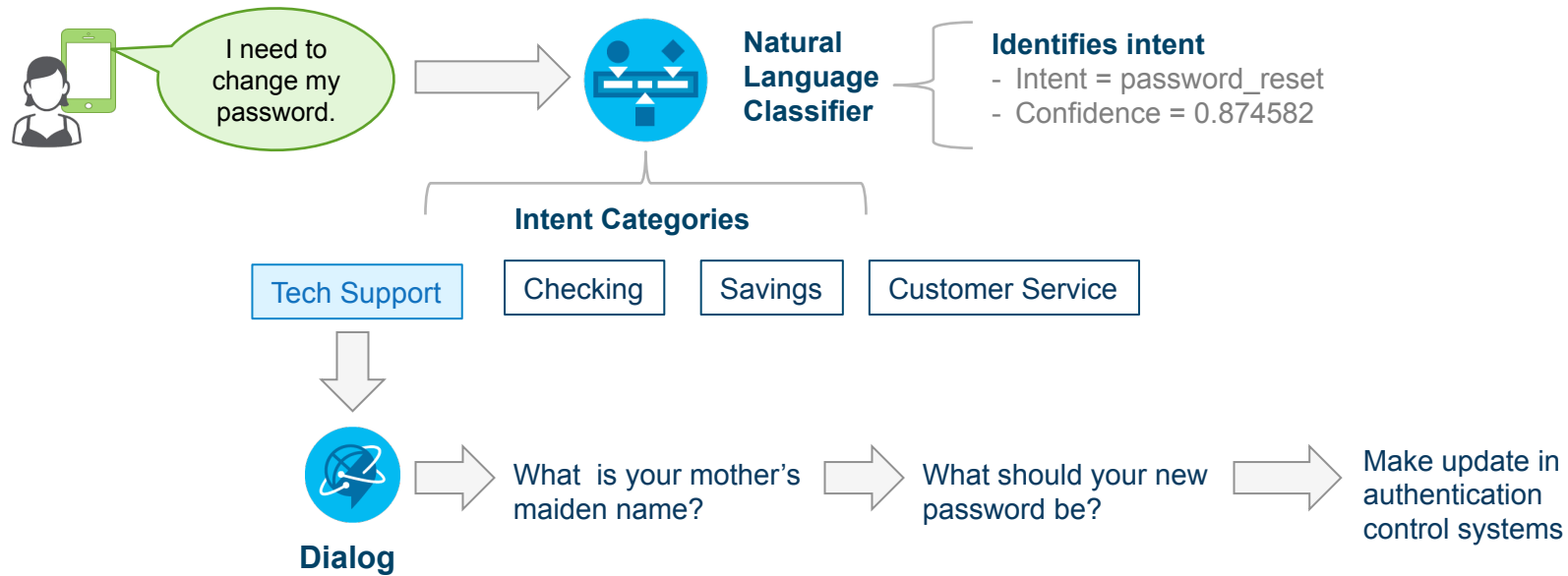
If the email contains 90% "severe" sentences then email the CEO and assemble the press team

Or train a meta-classifier (Decision tree or SVM) to make a classification based on the results from the sentence level annotations.

Use Case

Perform actions for users using NLC+Dialog

Your task? Build a voice-enabled online banking solution. For example, how would you allow a customer to request a password change? First train the NLC to identify the various intent categories such as Tech Support, Checking, Savings, and Customer Service. When a password change request arrives, the NLC would identify this and your system would recognize the need to first authenticate the user. The Watson Dialog service could be configured to perform the back-forth to obtain the user's authentication details and request the password reset from your bank's authentication system.



Use Case

Topic Modeling Using the NLC

Lucky you!

You have lots of natural language being generated by your customer/technical support teams. Or maybe it's your customers writing reviews, sending you emails, or tweeting about your products. Your challenge is to quickly categorize, analyze, and route the highest priority ones to the best person. Not a problem! You can do that and more using the NLC and Alchemy Language.

If your content is < 1024 chars, then send it directly to the NLC. If it's longer, reduce the size by parsing at the sentence or paragraph level. You could also test running your text through Alchemy Keywords to remove the lower relevance words.

